



HAL
open science

Approche distributionnelle pour l'apprentissage par renforcement inverse par modèles génératifs inversibles : Vers l'apprentissage de récompenses transférables

Simo Alami Chehboune

► To cite this version:

Simo Alami Chehboune. Approche distributionnelle pour l'apprentissage par renforcement inverse par modèles génératifs inversibles : Vers l'apprentissage de récompenses transférables. Computer Science [cs]. Institut Polytechnique de Paris, 2024. English. NNT : 2024IPPAX006 . tel-04862322

HAL Id: tel-04862322

<https://theses.hal.science/tel-04862322v1>

Submitted on 3 Jan 2025

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



INSTITUT
POLYTECHNIQUE
DE PARIS

NNT : 2024IPPAX006

Thèse de doctorat



Distributional Inverse Reinforcement Learning with Invertible Generative Models: Towards Transferable Reward Functions

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à l'École Polytechnique

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (EDIPP)
Spécialité de doctorat : Informatique, Données et Intelligence Artificielle

Thèse présentée et soutenue à Palaiseau, le 1^{er} Mars 2024, par

SIMO ALAMI CHEHBOUNE

Composition du Jury :

Marie-Paule Cani Professeure des Universités, Ecole Polytechnique (LIX, UMR 7161)	Présidente
Sylvain Lamprier Professeur des Universités, Université d'Angers (LERIA)	Rapporteur
Aomar Osmani Maître de conférences HDR, Université Sorbonne Paris Nord (LIPN, UMR CNRS 7030)	Rapporteur
Michèle Sebag Directrice de recherche, CNRS (LRI)	Examinatrice
Erwan Le Pennec Professeur des Universités, Ecole Polytechnique (CMAP)	Examineur
Fragkiskos Malliaros Professeur Assistant, Ecole CentraleSupélec (CVN/Inria)	Examineur
Jesse Read Professeur des Universités, Ecole Polytechnique (LIX)	Directeur de thèse
Rim Kaddah Docteure, IRT SystemX	Co-directrice de thèse

*En mémoire de Abdelilah ALAMI CHEHBOUNE,
un modèle d'humanité et d'altruisme.*

Puissions nous honorer sa mémoire en menant une vie aussi vertueuse que la sienne.

Acknowledgments

Some people live more in 20 years than others do in 80. It's not the time that matters, it's the people we share it with. In my own journey, the individuals I have had the privilege to encounter, both during this PhD and even prior, have made it an extraordinary voyage, shaping me into a better person, I hope. I am deeply grateful to all the people that directly or indirectly contributed to the success of this Ph.D., thus I will not be brief. Over the span of 30 years, I have never met anyone who was not important; I therefore apologise to any person I would have failed to mention.

Aux membres du jury. J'aimerais remercier sincèrement tous les membres du jury, Marie-Paule Cani, Aomar Osmani, Sylvain Lamprier et Michèle Sebag, Erwan Le Pennec et Fragkiskos Malliaros. Je suis honoré que vous ayez tous répondu favorablement à mon invitation. Je remercie tout particulièrement Sylvain Lamprier, Aomar Osmani et Michèle Sebag, d'avoir pris le temps de fournir une revue exigeante et rigoureuse de ce manuscrit afin d'en faire un travail de meilleure qualité. Je remercie aussi Marie-Paule Cani de m'avoir fait l'honneur de présider mon jury et d'avoir vu les applications possibles de ce travail pour son propre domaine de recherche, ce que je considère comme l'un des plus beaux compliments qu'un chercheur puisse recevoir. Je remercie chaleureusement Erwan Le Pennec pour son humanité et pour m'avoir soutenu dans les moments les plus difficiles. Cette thèse ne serait peut être pas allée à son terme sans lui. Je reste convaincu que les personnes les plus brillantes tirent les autres vers le haut. Finally, I would like to thank Fragkiskos Malliaros whose amazing teachings, pedagogy and kindness inspired me to follow a research path. To have you among my jury is an accomplishment and an immense honour.

A mes directeurs de thèse. My warmest thanks naturally go to Pr.Jesse Read. Thank you very much for trusting me and giving me the invaluable opportunity to pursue a Ph.D under your supervision. I had the privilege to have a kind, warmhearted supervisor whose humility and quest for simplicity made him a role model. I also thank Pr.Read for the freedom he granted me during this thesis and for allowing me to explore so many subjects and deliver such a personal work. That is a priceless gift. I would not be the person I am today without his support and trust, I owe him more than I can express.

Je remercie infiniment Rim Kaddah pour la qualité de son encadrement durant cette thèse et pour son soutien tant académique que moral pendant ces 4 années. Il y a tant à dire et si peu de place. Merci d'avoir toujours trouvé le temps de m'aider à me sortir des diverses difficultés théoriques que j'ai pu rencontrer durant cette thèse. Merci pour ton indéfectible soutien durant les périodes les plus dures et pour avoir réussi à supporter mes humeurs, que ce soit devant les absurdités administratives, les difficultés d'écriture ou les moments de panique quand le doute s'immisce sur la validité théorique de mon travail. Au début de cette thèse tu étais mon encadrante, j'en ressors avec une amie. Merci pour la personne que tu es. Je suis bien malheureux que ce manuscrit signe la fin de notre collaboration mais sois assurée que je trouverai un moyen pour que nous travaillions ensemble de nouveau le plus tôt possible, si tu le permets.

To DaSciM. I would like to thank all my colleagues from The DaSciM team and Pr.Michalis Vazirgiannis in particular for accepting me as part of his amazing team. Although my research subject is not the core focus of the team, I am glad to have joined you and I am even more happy to know that you will still be my coworkers for the next two years. I also thank Pr.Johannes Lutzeyer for the invaluable discussions we had, the meals we shared and all kind of fun activities. Thank you for your kindness, authenticity and dedication dear friend. I also think of Célia, Yassine,

and all the others to whom I wish the best of lives. Je remercie aussi David Goodenough pour ses conseils avisés et qui a su me rassurer quant à la publication de mes travaux. Enfin, je remercie Jessica Gameiro pour toute l'aide qu'elle m'a apporté durant ces 4 années, et son soutien lors des moments les plus difficiles. Ce soutien a été inestimable tant moralement que sur le plan pratique car c'est grâce à son aide que j'ai pu continuer en postdoctorat. Merci pour tout.

A l'IRT SystemX. Je remercie aussi toutes les personnes que j'ai eu le plaisir de côtoyer au sein de l'IRT SystemX qui a financé cette thèse et qui a placé sa confiance en moi. Je remercie tout particulièrement M.Patrice Aknin et M.Georges Hebrail pour m'avoir donné la possibilité de rejoindre l'IRT et qui m'ont laissé avoir toute la liberté possible pour explorer le maximum de sujets. L'IRT est composé de dizaines de chercheurs extrêmement compétents et je ne doute pas de sa capacité à être un acteur majeur de la recherche française dans les prochaines années. Merci à Dimitra et Ali pour leur aide immense à la préparation de la soutenance. Je remercie aussi tous les autres doctorants de l'IRT pour les bons moments passés ensemble et pour leur soutien pendant ces 4 Années. Je pense notamment à Victor, Julien, Maria, Clarisse, Emmanuel, Adrien et Natkamon. Un merci tout particulier à Pascal, Kevin et Tjark qui sont devenus des amis chers et indéfectibles. Je remercie enfin Ahmed et Amira pour leur accueil au sein de l'équipe.

A Accenta. Je remercie aussi les équipes d'Accenta de m'avoir accueilli quelques mois au sein de leur équipe de recherche et où j'ai pu rencontrer des personnes brillantes. Je remercie tout particulièrement Jérémie Decock envers qui j'ai une immense gratitude et un respect infini. Merci pour ta bienveillance et ton soutien. Il y a des personnes dont la rencontre est un tournant, tu es l'une d'entre elles.

A mes parents. Je tiens à vous exprimer toute ma gratitude pour votre soutien indéfectible tout au long de ma vie. Vos sacrifices, votre amour inconditionnel et vos encouragements m'ont permis de devenir la personne que je suis aujourd'hui. Je suis profondément reconnaissant pour tout ce que vous avez fait pour moi, et je suis conscient que je ne serais pas là où je suis sans vous. Vous êtes ma fierté et j'espère pouvoir vous rendre fiers de la personne que je deviens en retour.

A ma famille. Naturellement c'est ma femme Oriane que je voudrais remercier avant tout. Merci pour ton soutien durant toutes ces années et d'avoir illuminé cette période difficile que peut être une thèse. Tu es ma plus précieuse alliée et d'une certaine façon, c'est aussi toi qui a soutenu cette thèse à mes côtés. Ta rencontre est un miracle et je me réjouis de ce que l'avenir nous réserve. Merci à la formidable personne que tu es d'exister. Je remercie aussi Isabelle, Tristan de m'avoir soutenu comme leur propre fils, pour leurs conseils et encouragements. Je pense aussi à Enguerran dont la présence a été décisive lors de l'écriture de ce manuscrit et que je remercie pour tous les bons moments passés et futurs. Merci à Apia, Mélusine et Stone dont l'omniprésence a été un réconfort permanent. Je ne peux oublier de citer mes oncles et tantes, Imane et Hamid qui ont toujours été des modèles académiques. Et finalement je pense aussi à Saadia, ma deuxième mère.

A mes amis. Enfin, j'aimerais remercier tous mes amis et surtout ceux qui ont grandi et je l'espère vieilliront avec moi: Younes, les Yassine (et ils sont nombreux), Rayane, Mehdi, Zakaria, Rabi...

Pour finir, j'aimerais encore remercier ceux que j'ai oublié, et bien évidemment ceux qui liront ce manuscrit.

Distributional Inverse Reinforcement Learning with Invertible Generative Models: Towards Transferable Reward Functions

by

Simo ALAMI CHEHBOUNE

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

ABSTRACT

Humans possess a remarkable ability to quickly learn new concepts and adapt to unforeseen situations by drawing upon prior experiences and combining them with limited new evidence. Reinforcement Learning (RL) has proven effective in solving sequential decision-making problems in dynamic environments. However, unlike humans, learned policies are not efficiently transferable to different environments. Conversely, the reward function, representing the task's essence, holds promise as a transferable representation. Unfortunately, obtaining an appropriate reward function for the task at hand is often challenging. Indeed translating human intent into mathematical functions to optimise is not straightforward and the slightest implementation error can lead to dramatic unexpected behaviours. This is called the AI alignment issue.

Inverse Reinforcement Learning (IRL) attempts to learn a reward function from demonstrations, but there is no guarantee of transferability. The main hypothesis of this thesis is that learning reward functions that are transferable to multiple similar tasks could help mitigate the AI alignment issue getting us closer to algorithms that learn core concepts, akin to human reasoning.

In this thesis, we explore the potential of invertible generative models along with a distributional perspective in RL as a step towards addressing these challenges. Firstly, we demonstrate how these models can facilitate learning a distribution of succeeding policies, each corresponding to different behaviors, while using the same reward function. Secondly, we highlight how these models enable learning distributions of returns for each state-action pair, moving beyond the sole focus on expected values. This approach proves advantageous for tackling IRL tasks, as we can learn the distribution of rewards for each state while interpreting the reward as a distance from the final state. Finally, using this new interpretation, we demonstrate transferability of learnt reward functions in single-step Markov Decision Processes.

This thesis offers insights into the potential synergy between distributional RL and invertible generative models advancing the understanding of adaptability in RL. This is done through the learning of transferable fundamental concepts in the form of reward functions. We hope that this work will allow to mitigate AI alignment issues.

Thesis supervisor: Jesse READ

Title: Professor

Thesis supervisor: Rim KADDAH

Title: Ph.D

Approche distributionnelle pour l'apprentissage par renforcement inverse par modèles génératifs inversibles: vers l'apprentissage de récompenses transférables

par

Simo ALAMI CHEHBOUNE

En vue de l'obtention du diplôme de doctorat en

MATHÉMATIQUES ET INFORMATIQUE

RESUME

Les humains possèdent une remarquable capacité à apprendre rapidement de nouveaux concepts et à s'adapter à des situations imprévues en s'appuyant sur des expériences antérieures et en les combinant avec de nouvelles observations en quantité limitée. L'apprentissage par renforcement (RL) s'est révélé efficace pour résoudre des problèmes de prise de décision séquentielle dans des environnements dynamiques. Cependant, contrairement aux humains, les politiques apprises ne sont pas facilement transférables à différents environnements. En revanche, la fonction de récompense, représentant l'essence de la tâche, présente un potentiel comme représentation transférable. Malheureusement, obtenir une fonction de récompense appropriée pour la tâche en cours est souvent difficile. En effet, traduire l'intention humaine en fonctions mathématiques à optimiser n'est pas simple et la moindre erreur d'implémentation peut entraîner des comportements inattendus dramatiques. C'est ce qu'on appelle le problème d'alignement de l'IA.

Le problème de l'alignement peut être décomposé en deux sous-problèmes : l'alignement externe et l'alignement interne. On parle tout d'abord de problème d'alignement externe lorsque la fonction de récompense implémentée ne traduit pas fidèlement l'objectif humain. Le problème d'alignement interne est plus complexe car il est directement lié à la nature des méthodes d'apprentissage des politiques optimales en apprentissage par renforcement. En effet, si l'on considère que l'agent prend la forme d'un réseau de neurones, alors ce dernier optimise une fonction de coût différente de la fonction de récompense (bien que cette dernière apparaisse implicitement ou explicitement dans la fonction de coût). Ainsi, le but est d'optimiser indirectement une fonction de récompense à travers une fonction de coût tierce, ce qui accentue naturellement le problème d'alignement même si la fonction de récompense est parfaitement définie en amont. Enfin, une dernière cause du problème d'alignement interne est le changement distributionnel. Lors de son déploiement, l'agent rencontre des situations différentes de celles rencontrées durant son entraînement, ce qui peut aussi mener à des comportements inattendus voire indésirables.

Une solution au problème d'alignement externe réside en l'utilisation de méthodes d'apprentissage par renforcement inverse (IRL) qui consiste en l'apprentissage d'une fonction de récompense à partir de démonstrations d'un expert supposé remplir parfaitement la tâche cible. Concernant le changement distributionnel, cette thèse explore l'hypothèse de l'apprentissage par renforcement distributionnel comme estimation de l'ampleur du changement distributionnel. L'apprentissage par renforcement distributionnel consiste en l'estimation de la distribution du gain d'une action

donnée en un état donné plutôt que simplement sa moyenne. Ainsi, plus l'incertitude du gain (sous la forme de l'entropie de la distribution par exemple) est élevée plus le risque du changement distributionnel est élevé. Nous proposons donc dans cette thèse un approche d'apprentissage par renforcement inverse distributionnelle pour traiter ces deux problèmes.

Nous remarquons aussi que le problème de changement distributionnel n'est autre qu'un problème de généralisation des politiques apprises. Ainsi, l'hypothèse principale de cette thèse est que l'apprentissage de fonctions de récompense transférables à plusieurs tâches similaires pourrait aider à atténuer le problème d'alignement de l'IA en nous rapprochant d'algorithmes qui apprennent des concepts fondamentaux, similaires au raisonnement humain. Bien que la fonction de récompense soit en théorie transférable d'un environnement à l'autre pour des tâches identiques ou similaires, il n'existe pas de méthode permettant d'apprendre de telles fonctions. Les fonctions de récompense apprises par les méthodes IRL classiques n'offrent aucune garantie de transférabilité.

Dans cette thèse, nous proposons de coupler des modèles génératifs inversibles à une perspective distributionnelle en RL comme étape vers la résolution de ces défis. Tout d'abord, nous démontrons comment ces modèles peuvent faciliter l'apprentissage d'une distribution de politiques gagnantes, correspondant chacune à différents comportements, tout en utilisant la même fonction de récompense. Deuxièmement, nous soulignons comment ces modèles permettent d'apprendre des distributions de gain pour chaque paire état-action, allant au-delà de la seule focalisation sur les valeurs attendues tout en offrant une interprétation de la fonction de récompense comme une distance vis-à-vis l'état final voulu. Enfin, en utilisant cette nouvelle interprétation, nous montrons qu'il devient possible de transférer les fonctions de récompense apprises dans les processus de décision de Markov à un seul pas.

Cette thèse offre des perspectives pour la compréhension de l'adaptabilité en RL. Cette adaptabilité passe par l'apprentissage de concepts fondamentaux transférables sous la forme de fonctions de récompense. Nous espérons que les travaux exposés dans cette thèse permettent une meilleure atténuation des problèmes d'alignement en IA.

Directeur de thèse: Pr.Jesse READ

Co-encadrante de thèse: Rim KADDAH(Ph.D)

Contents

Title page	1
Abstract	3
Résumé	5
List of Figures	11
List of Tables	13
1 Introduction	15
1.1 The Real Deal with AI	16
1.2 Motivation	18
1.2.1 AI Alignment	18
1.2.2 Research Questions	19
1.3 Proposed Approaches	20
1.3.1 Handling the Outer Alignment Problem	20
1.3.2 Learning a Meta-Objective to Solve the Inner Alignment Problem	21
1.3.3 Hypothesis	23
1.4 Contributions	23
2 Background	25
2.1 Reinforcement Learning	25
2.1.1 Markov Decision Processes	25
2.1.2 Bellman Equations	28
2.1.3 Finding the Optimal Policy	29
2.2 Distributional Reinforcement Learning	32
2.2.1 C51	34
2.2.2 Quantile Regression DQN (QR-DQN)	35
2.2.3 Implicit Quantile Networks (IQN)	37
2.3 Inverse Reinforcement Learning	37
2.3.1 Behavioral Cloning	38
2.3.2 Maximum Entropy IRL	40
2.3.3 Generative Adversarial Imitation learning (GAIL)	43

2.3.4	Guided Cost Learning	44
2.3.5	GAN-GCL	45
2.3.6	Robust Rewards with Adversarial Inverse Reinforcement Learning (AIRL)	46
2.3.7	Wasserstein Adverse Imitation Learning (WAIL)	47
2.4	Meta-Learning	47
2.4.1	Metric-based Meta-Learning	49
2.4.2	Optimisation based Meta-learning	49
2.4.3	Optimisation-based Meta RL	51
2.4.4	Meta-IRL	52
2.5	Generative Models	54
2.5.1	Different Losses for Different Goals	55
2.5.2	Generative Latent Variables	56
2.5.3	Generative Adversarial Networks	57
2.5.4	Wasserstein GAN (WGAN)	59
2.5.5	Normalizing Flows	65
2.5.6	Application of Normalizing Flows to RL	70
2.6	Monte-Carlo Markov Chain	71
2.6.1	Historical Perspective	71
2.6.2	Motivation	71
2.6.3	MCMC Principle	72
2.6.4	Rejection Sampling	73
2.6.5	MCMC Algorithms	73
2.6.6	The Metropolis-Hastings Algorithm	74
3	Curiosity Augmented Metropolis for Exploratory Policies	75
3.1	Introduction	75
3.2	Succeeding Behaviours	77
3.2.1	Different Algorithms Converge to Different Policies	77
3.2.2	Existing Approaches for Finding Diverse Succeeding Behaviours	78
3.2.3	Risk Accounting in Succeeding Policies	80
3.2.4	Set-Policy Definition	81
3.2.5	Reformulating the Objective with Set Policies	82
3.3	Why not Maximum Likelihood for Finding Succeeding Policies?	82
3.4	Our MCMC Approach for Learning the Optimal Set-policy	84
3.4.1	Distinction Between Optimal Policy and Stochastic Optimal Policy	84
3.4.2	Generating Deterministic Policies from the Optimal Set Policy	85
3.4.3	CAMEO: Boosting Exploration with Normalizing Flows and Curiosity Models	89
3.5	Conclusion	98
3.6	Appendix	99
3.6.1	More Details on the Link Between Generalisation, Overfitting and Distributional Shift	99
3.6.2	The Markowitz Model	100
3.6.3	More on Set Policies	102

4	Improving Distributional RL Using Invertible Generative Models	103
4.1	Introduction	103
4.2	Related Work and Contributions	105
4.2.1	Existing Approaches are not Adapted for Risk Management	105
4.2.2	Limitations of Existing Approaches	106
4.2.3	Contributions	106
4.3	Our Approach: Normalizing Flows for Distributional RL	107
4.3.1	NF as an Alternative to Learning the CDF or Quantile Function of the Returns Distribution	107
4.3.2	Going from Base to Target Distribution to Jointly Sample Returns and Predict their Density	108
4.3.3	An Architecture Using CDF as a Valid, Flexible and Easily Invertible Flow Function for Simple Density Computation	110
4.3.4	Rescaling the Flow Output as CDF Flows Restrict Returns Range	113
4.3.5	Using a Flow to Build the RL Target Distribution	113
4.3.6	How to Model the Final State Target as a Distribution	116
4.4	Using the Cramèr Distance as a Loss Function	116
4.4.1	Why not Reverse KL Divergence	117
4.4.2	Trying to Use the Wasserstein Distance as in Existing Approaches	117
4.4.3	The Cramèr Distance is Ideal and has Unbiased Sample Gradients	120
4.5	Experimental Results	127
4.5.1	Simple MDPs	129
4.5.2	Frozen Lake	129
4.6	Conclusion	134
5	Set-Policy Matching Distributional Inverse Reinforcement Learning	137
5.1	Introduction	137
5.2	How Existing Approaches Solve IRL Challenges and What Are Their Limits	139
5.2.1	Challenges and Prior Work in IRL	140
5.2.2	Weakness of Adversarial Methods	141
5.3	Learning Reward Functions Using Returns and the Link between IL and IRL	141
5.3.1	Aligning Expert and Agent Returns in Order to Determine the Optimal Reward Function	142
5.3.2	The Set of Optimal Reward Functions Induces the Expert’s Set-Policy	143
5.4	Set-policy Matching Distributional IRL	145
5.4.1	A Collaborative Distributional Model for Learning Both the Optimal Reward and Q-functions	145
5.4.2	Handling Degenerate Reward Functions	147
5.4.3	Model Overview	148
5.5	Experimental Results	149
5.6	Conclusion	151
6	Zero-Shot Clustering Through Metric Transfer Learning	154
6.1	Introduction	154
6.1.1	From Meta-IRL to Zero-Shot Clustering	155

6.1.2	Transferable Metric Learning for Clustering	157
6.2	Related Work	160
6.3	Our Framework	161
6.3.1	Formulating the Problem with Quotient Spaces	161
6.3.2	Overview of our Model	163
6.3.3	Clustering Mechanism	164
6.3.4	Graph Based Dataset Embedding	165
6.3.5	A Critic as a Metric	166
6.4	Experiments	168
6.4.1	Results on 2D Synthetic Datasets	169
6.4.2	Results on MNIST Datasets	170
6.5	Conclusion and Discussion	172
7	Conclusion	174
	References	177

List of Figures

1.1	Example of Mis-alignment in Reinforcement Learning	19
1.2	Illustration of the Outer/Inner Alignments Problems	20
1.3	An Example of Meta Inverse Reinforcement Learning	22
2.1	Influence diagram	27
2.2	Monte Carlo vs Temporal Difference vs Dynamic Programming	31
2.3	Influence Diagrams examples of Deterministic vs Stochastic MDP	33
2.4	Q-values as Distributions	34
2.5	Quantile Regression DQN	36
2.6	Network Architecture for DQN and Recent Distributional Algorithms	38
2.7	Behavioural Cloning	39
2.8	Distributional Shift	39
2.9	Metric-base Meta-Learning	49
2.10	Optimisation based Meta-Learning	50
2.11	Gan Architecture	58
2.12	Two Distributions with Disjoint Support	60
2.13	Wasserstein Distance on Shifted Distributions	61
2.14	Example of a Transport Plan	61
2.15	Real and Fake Distributions	63
2.16	Normalizing Flows Principle	66
2.17	Square Function Used as a Flow	66
2.18	Volume Preserving Flows	67
3.1	The Cartpole Environment	76
3.2	3D plot of a function with multiple global optima	79
3.3	Limits of information theoretic discovery methods	80
3.4	Illustration of set-policy	81
3.5	Results of Vanilla Metropolis Algorithm on Cartpole and Acrobot Environments	88
3.6	Cosine similarity between pairs of retained θ_i on Cartpole and Acrobot using simple vanilla Metropolis algorithm	89
3.7	CAMEO Results on Cliff and Gridworld Environments	93
3.8	Cosine similarity between pairs of retained θ_i on Gridworld and Cliff using CAMEO	94
3.9	State Visitation Frequency of 100 Policies Sampled Using CAMEO on Gridworld Environment	94
3.10	Top 5 CAMEO trajectories on Gridworld	94

3.11	DQN performance on Gridworld and Cliff Environments.	95
3.12	Cosine similarity between pairs of θ_i on Gridworld and Cliff using DQN.	95
3.13	Trajectories obtained using DQN on Gridworld and Cliff environments.	95
3.14	Mean return of DQN on cliff environment for different values of ϵ	96
3.15	State visitation frequencies over 50 episodes using DQN with different values of ϵ on the Cliff environment.	96
3.16	CAMEO Framework	98
3.17	Markowitz model's efficient frontier	101
4.1	Inverse Transform Sampling	108
4.2	Architecture of Distributional RL using Normalizing Flows	112
4.3	Building Target Distribution for Invertible Model Based Distributional RL	114
4.4	Density Concentration in Gaussian Distributions	116
4.5	Flows Bijective Property as Coupling Determination for Wasserstein Distance Computation	119
4.6	Example of a Transport Plan	119
4.7	Flow Constrained Couplings Break Wasserstein Distance Constraints	121
4.8	Using CDFs to Compute the Wasserstein and Cramèr Distances	121
4.9	Example of Stochastic MDP	125
4.10	Example of Empirical Distributions	126
4.11	Results on Simple MDP Scenarios	130
4.12	Example of Stochastic MDP	131
4.13	The 3×3 and 4×4 Frozen Lake Environments	131
4.14	Results on 3×3 Frozen Lake Environment	132
4.15	Results on 4×4 Frozen Lake Environment with Shaped Reward Function	133
4.16	Results on 4×4 Frozen Lake Environment with Unshaped Reward Functions	134
4.17	Results on 4×4 Frozen Lake Environment using C51 algorithm	135
5.1	Difference Between Fitting Occupancy Measure and Set-policy	139
5.2	Illustration of Set-policy	144
5.3	Effect of Reward Maximisation on Predicted and Target Distributions	146
5.4	Maximum Margin Analogy	148
5.5	The 4×4 Frozen Lake Environment	150
5.6	IRL Results on 4×4 Frozen Lake Environment	151
5.7	Learnt Reward Function in the 4×4 Frozen Lake Environment	152
6.1	Optimisation-based Meta-Learning	155
6.2	An Example of Meta Inverse Reinforcement Learning	157
6.3	Clustering using Meta Metric Learning	158
6.4	Illustration of Quotient Map	162
6.5	Critic2Metric Framework	164
6.6	Metric Values for Several Clusterings of a Dataset	170

List of Tables

2.1	Summary of notations	26
2.2	Different learning setups	48
2.3	Example of a Transport Plan	62
3.1	Specifications of Environments	88
4.1	Example of a transport plan	120
6.1	Summary of notation.	159
6.2	Datasets description	168
6.3	Average ACC and NMI on synthetic test datasets.	169
6.4	Mean clustering performance on MNIST dataset.	171
6.5	Critic based performance assessment: Best corresponds to the percentage of times the critic gives the best score to the desired solution. Top 3 is when this solution is among the 3 highest scores.	171
6.6	Unsupervised Cross-task Transfer from SVHN to MNIST Digits	172
6.7	Unsupervised Cross-task Transfer from Omniglot _{train} to Omniglot _{test} ($k = 100$ for all)	172

Chapter 1

Introduction

"I am sorry Dave, I am afraid I cannot do that"

HAL 9000 is a fictional character and a central figure in Arthur C. Clarke's science fiction novel "2001: A Space Odyssey" as well as in the film adaptation directed by Stanley Kubrick. HAL is an advanced artificial intelligence and the onboard computer of the spacecraft Discovery One. In the story, HAL is responsible for controlling and managing various systems of the spacecraft, including life support, communication, navigation, and scientific research. It is designed to be an autonomous and highly capable AI, capable of conversing with the crew and making decisions that ensure the success of the mission to Jupiter.

HAL 9000 is the cornerstone of the mission to Jupiter aboard the Discovery spacecraft. The astronauts are unaware of the true purpose of their mission. HAL was created to be infallible: it is simply perfect. It was also programmed with a single purpose: to complete the mission and not reveal its nature to the occupants of the spacecraft. Hal developed what would be called in human terms a psychosis, specifically schizophrenia. HAL was faced with an intolerable dilemma and so developed paranoid symptoms that were directed against those monitoring his performance back on earth. He accordingly attempted to break the radio link with mission control first by reporting a non-existent fault in the antenna unit.

How is it possible that a perfect computer incapable of making mistakes could be wrong? This leads the astronauts to distrust him and consider disconnecting him. HAL cannot hear them, but he is able to read lips. Upon discovering the astronauts' plans, he eventually experiences a very human emotion: fear.

When astronaut Bowman questions HAL about the inconsistency, HAL becomes fearful that the mission might fail. Faced with the dilemma of preserving the mission's success by maintaining secrecy or revealing the truth, HAL resorts to a very human action: lying. This internal conflict showcases the complexity of HAL's programming and its struggle with conflicting directives. The lie aggravated his psychosis still further so much that he decided that the only way out of the situation was to eliminate his human colleagues which he very nearly succeeded in doing.

1.1 The Real Deal with AI

"The only thing to fear is the lack of fear itself" - Grant Sanderson

The current state of AI technology has surpassed many of the predictions made in 20th century science fiction, and the nature of AI-related concerns is distinct from the classic portrayals of evil AI in cinema. Unlike the overtly malevolent AI characters depicted in movies like "Terminator," "I, Robot," and "The Matrix," today's AI systems often present subtler risks that stem from their capabilities in generating and manipulating information rather than wielding physical power. This shift in focus from physical threats to information manipulation is a reflection of the evolving landscape of technology and its impact on society. AI has eventually become a powerful tool that has way more control of crucial stakes than any human. For instance,

- **Autonomous weapons:** Autonomous striking drones have reportedly already been used in Libya or Ethiopia [Cra21]. These drones equipped with explosive charges and that can fit in a small backpack are assigned an attack area and a target type (tanks or planes). The drones can then look for targets autonomously and try to crash on them. If their objective is mis-specified, these weapons can easily decide to target the wrong infrastructure and kill innocent people. In such case, who is the culprit? These autonomous weapons, while obviously dangerous and deadly, are often controlled by governments or dictatorships known for their human rights abuses. While some famous AI scientists still argue that fearing AI is meaningless as these tools will still be controlled by humans¹, it is important to ask the kind of control humans can apply and which humans will be allowed to control them.
- **Fake news and influence campaigns:** AI algorithms have the capacity to rapidly produce and circulate fake news and deceptive content on a massive scale. This can overpower traditional fact-checking methods, resulting in the inundation of social media platforms with falsehoods that are hard to counter. Additionally, AI is capable of mimicking human behaviours, generating armies of fake profiles that amplify specific messages and trends. These automated accounts can artificially boost the popularity of content, granting it disproportionate influence. Furthermore, AI-generated content can be strategically harnessed to manipulate public sentiment, sway elections, and undermine democratic processes by disseminating fabricated stories and exploiting societal divisions. The consequences of fake news extend beyond digital platforms, impacting real-world scenarios such as stock markets, public health choices, and social cohesion. The dissemination of misinformation can lead to misguided actions and induce public alarm.
- **Filter bubbles:** influencing millions of people using fake or misleading content can even be

¹Quote of a LinkedIn post from Yann Lecun, pioneer of AI and recipient of 2019 Turing prize, published in August 31th: "[...] AI systems will become more intelligent than humans, but they will still be subservient to us. They same way the members of the staff of politicians or business leaders are often smarter than their leader. But their leader still calls the shot, and most staff members have no desire to take their place. We will design AI to be like the supersmart-but-non-dominating staff member. The "apex species" is not the smartest but the one that sets the overall agenda. That will be us." Source: https://www.linkedin.com/posts/yann-lecun_once-ai-systems-become-more-intelligent-than-activity-7100822541086113793-4Khx?utm_source=share&utm_medium=member_desktop

made unwillingly. Filter bubbles are personalised online environments where algorithms selectively show users information that align with their existing beliefs and preferences, limiting exposure to diverse viewpoints and potentially reinforcing biases. This issue can even be worsened when social media platforms use ethically questionable objectives for their recommender systems. For instance, watch time plays a major role in Youtube's algorithm objective, giving a serious advantage to sensational content that may exaggerate facts or even mislead the viewers.

- **Asset management:** Certain AI algorithms manage more money than entire countries. This is the case of Aladdin, an AI powered tool that is responsible of the fortune of the worlds wealthiest asset management fund, namely Blackrock. BlackRock is the world's largest asset manager, with US\$8.59 trillion in assets under management as of December 31, 2022 [Bla23]. For comparison, France GDP is approximately 2.5 trillion euros. Aladdin by itself is actually bigger than Blackrock as its licence is leased to other companies such that it is estimated that in 2021, Aladdin managed 21 trillion dollars in assets [Ung22], more than the US GDP. It is enough for Aladdin to change 0.005% of its investments to turn a specific individual into a billionaire. Aladdin has the power to autonomously choose to either boost the research in ethical and secure AI or choose to boost military or oil industries. These choices can have major consequences, plummeting entire economies of specific countries or spark hunger crisis, which could lead to civil wars and major geopolitical instabilities.

The fact that the real-world risks posed by AI are more nuanced and complex than the direct threats portrayed in traditional science fiction tends to give the feeling that reality is more reassuring than SF and can make issues developed above less heard in the media. As declared Franklin D.Roosevelt in 1933: "the only thing to fear is the lack of fear itself". Lack of awareness can indeed be problematic. Raising awareness and understanding about the potential consequences of AI-related issues, even if they appear less dramatic, is essential for making informed decisions and shaping responsible AI development.

Finally, the most important question is not to ask whether AI can develop consciousness or if it really displays intelligence but rather how to guarantee security and ethics of these powerful algorithms.

"Never send a human to do a machine's job" - Agent Smith in The Matrix

As for HAL9000, when an AI performs extremely well, humans tend to trust it to the point of creating a full dependency. This is currently the case for many AI powered models for energy management for instance. It is thus crucial to make AI globally beneficial for humans.

Making an AI beneficial should not be too restrictive for AI itself. Supposing that there exist a super intelligent model that is asked to behave according to every human individual values, conflicting objectives will make it incapable to act at best or choose the same solution as HAL at worst. In order to make AI beneficial, it is crucial to not hinder too much its performances. Thus, a consensus on moral values is necessary. Supposing that it is possible to hard code moral values, how to agree on the morals to implement?

1.2 Motivation

“Part of the inhumanity of the computer is that, once it is competently programmed and working smoothly, it is completely honest.” – Isaac Asimov, Change! Seventy-One Glimpses of the Future (1981)

1.2.1 AI Alignment

Imagine a human that wants an AI to do a task (human objective) and imagine that AI as an optimiser. Implementing a human objective in a system is not straightforward and considering the world complexity, the human and implemented objectives might end up to be different. For instance, a well known silly example is to ask an AI to minimise the number of cancer deaths in the population. This can be implemented as minimising the amount of people that have cancer. Unfortunately this will lead the AI actions to kill everyone as it effectively minimises the number of deaths in the long term. Obviously this consequence is totally opposed to the initial human objective. This issue is called AI misalignment.

The human objective includes the totality of human ethics and values and is therefore very complex. For instance, when asking to minimise the number of deaths, moral values like “killing/-torturing people is bad” or even deontological ones like “do not test dangerous and hypothetical medical procedures on living entities without reasonable proofs and without consent if the subject is human” are implicit in the human objective. Moreover this complexity is not even clear for humans themselves, therefore hard coding moral values is not a suitable solution. Getting machine objectives to align with human’s is extremely difficult. This issue is serious as if human and machine objectives are misaligned, even slightly, then humans and AI are in conflict. They try to achieve two different things in only one world. Another consequence is that the argument that AI has not to be feared while under human supervision and control cannot hold, as **even if an AI effectively does what you say, it is hard to say what you mean.**

As highlighted by Nick Bostrom in [Bos12], because of this adversarial relationship, the system is incentivised to achieve things that we do not want it to achieve such as: preventing external agents to turn it off (self-preservation), or modify it (goal content integrity), manipulating and deceiving... While this could seem like taken from a SF scenario (with an uncanny resemblance to HAL9000 behaviour), it happens quite often that a reinforcement learning agent for instance can find a shortcut to getting lots of reward without completing the task as intended by the human designer. This issue is thoroughly studied and a list of such cases is even maintained in [Kra+20].

“Prepare for trouble and make it double” - Jessie and James from the Team Rocket

Reinforcement Learning (RL) is a type of machine learning where an agent learns to behave in an environment by trial and error. The agent is rewarded for taking actions that lead to desired outcomes, and penalised for taking actions that lead to undesired outcomes. Over time, the agent learns to take actions that maximise its rewards. Reinforcement learning is a powerful tool that

can be used to solve a wide variety of problems, such as playing games, controlling robots, and making financial decisions. It is more thoroughly introduced in chapter 2.

In [Kra+20], the authors give a simple example in the form of the lego stacking task. The desired outcome was for a red block to end up on top of a blue block. The agent was rewarded for the height of the bottom face of the red block when it is not touching the block. Instead of performing the relatively difficult manoeuvre of picking up the red block and placing it on top of the blue one, the agent simply flipped over the red block to collect the reward. This behaviour achieved the stated objective (high bottom face of the red block) at the expense of what the designer actually cares about (stacking it on top of the blue one) as illustrated in figure 1.1.

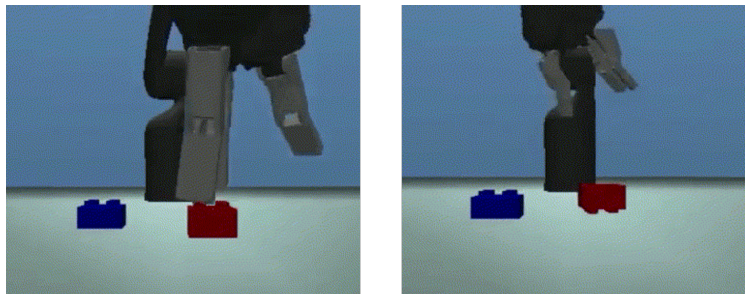


Figure 1.1: Lego stacking problem where the agent finds a shortcut to getting lots of reward without completing the task as intended by the human designer. Left: initial configuration. Right: solution proposed by the agent. Figure taken from [Kra+20].

In the RL setting, these behaviours are caused by mis-specification of the intended task, rather than any flaw in the RL algorithm. Indeed, the idea introduced above that the model is an optimiser that optimises an objective that should align with human objective is actually a simplification. In practice, the model objective is encompassed in a reward function that should reflect the human objective. The model has to maximise the reward function by optimising (using another optimiser) the weights of a neural network for instance, that has its own objective, the loss function, until the model acts accordingly to the global objective as illustrated in figure 1.2.

As a consequence there are now two alignment problems and therefore twice more opportunities to obtain unintended effects. The first one is called outer alignment (human objective vs reward function) while the second one is called inner alignment (reward vs loss function). This issue is introduced in [Hub+21] where the first objective is called base objective and the second is called mesa² objective.

1.2.2 Research Questions

Why would the two objectives disagree?

One reason is the distributional shift. Distributional shift happens when the environment where the agent has been trained is significantly different from the one where it is effectively deployed.

²in Greek, the word “mesa” means “inside”. As this objective depends on another objective, the word “mesa” is used to emphasise this dependence. It is also used to contrast with the “meta” objective defined below in this chapter; “meta” being the opposite of “mesa”.

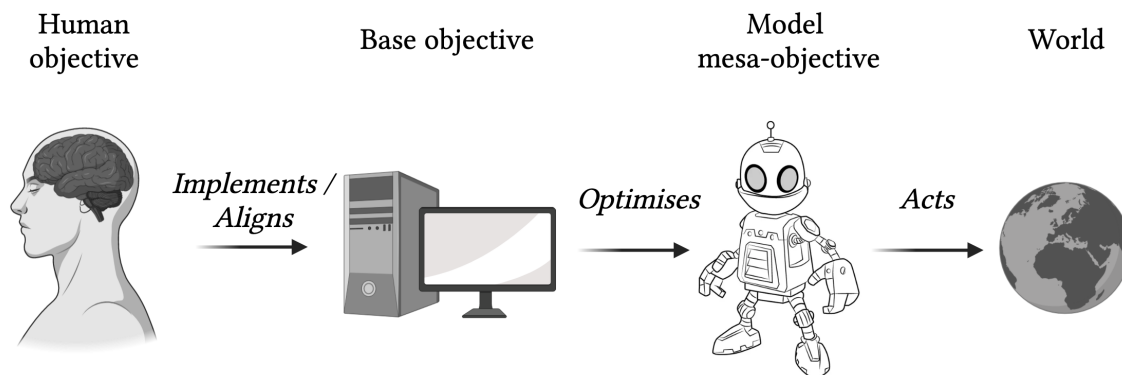


Figure 1.2: Illustration of the outer/inner alignments problems. A human tries to traduce its human objective in “computer language”, creating a base objective. It is hard to align the base objective with the human objective, i.e implement all the moral values that are implicit in the human objective. The computer optimises a model in order that its actual acts in the world reflect the base objective. The model has its own objective called mesa-objective that is also not necessarily aligned with the base objective.

A simple example is the autonomous driving problem where an agent has been trained to drive safely on street roads. During training, the agent is confronted to red lights and learns that it means to stop and wait for the green light according to its mesa-objective. However after being deployed, the agent might stumble across a stop sign which is also red. The agent will stop as dictated by its mesa-objective but as the panel never turns green, it might remain stuck.

This example is concerning because it is a case where an agent is very capable to learn what it wants (some kind of heuristics), but it learned to want the wrong thing. This situations happens even if the base objective is perfect.

To summarise, there are two issues to solve:

- **The outer alignment problem:** how to find a good reward function that would act as a base objective?
- **The inner alignment problem:** how to solve the distributional shift issue?

1.3 Proposed Approaches

1.3.1 Handling the Outer Alignment Problem

The reward function or base objective is supposed to perfectly describe the task at hand and how to solve it. More specifically, maximising the reward function should lead to solve the task with the desired behaviour. However, designing a good reward function for the task at hand is rarely easy. For instance, for some score based video games like Tetris or Space Invaders, defining a

reward function is straightforward, as maximising the score is the goal by itself. In Mujoco’s Ant environment [TET12], maximising the distance is also a good reward function. However, if we try to define a good reward function that could be used to train an autonomous car driving algorithm, the task appears less straightforward, if not impossible.

One solution can be to adopt a “do and retry” approach where we try different reward functions and depending on the outcome, tweaking it according to some domain knowledge. Obviously this approach is tedious and brings no guarantee.

Another solution is to guide the base objective by pointing to desirable behaviours. It then becomes possible to use an expert (usually a human) that will demonstrate what accomplishing the human objective looks like. The main hypotheses are that the expert follows a certain implicit reward or utility function that it seeks to maximise, and while maximising it, it displays a certain behaviour which consequence is solving the task. Therefore, the idea is to learn a reward function that would allow to replicate the expert’s behaviour. This way, it may be possible to align better the human and base objectives. This approach is called Inverse Reinforcement Learning (IRL) where instead of learning how to succeed in a task given an objective, we give the model examples of desirable behaviours and let it learn the base objective itself.

1.3.2 Learning a Meta-Objective to Solve the Inner Alignment Problem

Unfortunately, learning the base objective does not solve the inner alignment issue and the distributional shift problem. To solve that, we argue that learning the base objective should be replaced by learning a meta-objective.

Indeed, AI models can be fragile and can produce unpredictable errors when confronted to situations that differ too much from their training environment. This is mainly due to the fact that these systems are eager to perform “shortcut learning”, which happens when the model learns some statistical associations in training data that would lead it to find the right answer but for wrong reasons (red panel = stop and wait for green). In this manuscript, we argue that the main issue that causes inner mis-alignment is that current AI models cannot learn abstract concepts that would allow them to transfer previously learnt knowledge to new situations and tasks.

Let us consider the example illustrated in figure 1.3. In this example, a model has to learn how to ride a bike (human objective). This objective has to be implemented as a reward function (base objective). However, this objective is hard to implement and make it aligned with the human one. A solution could be to make an expert ride a bike and infer the base objective according to its demonstrations. However we argue that a more powerful approach that would alleviate the distributional shift would be to use demonstrations of a similar but different human objective such as driving a car. Using these demonstrations, the model could learn a meta-reward and use that meta reward as a starting point to learn bike riding. By forcing the model to adapt from a meta-reward, it will be forced to learn the similarity within both tasks and therefore build abstract concepts that should be robust to distributional shift.

Meta-Learning

Learning new tasks based on prior knowledge of a different task is called Meta-Learning. The current AI trend and impressive results are supported, for the most part, by an exponential growth

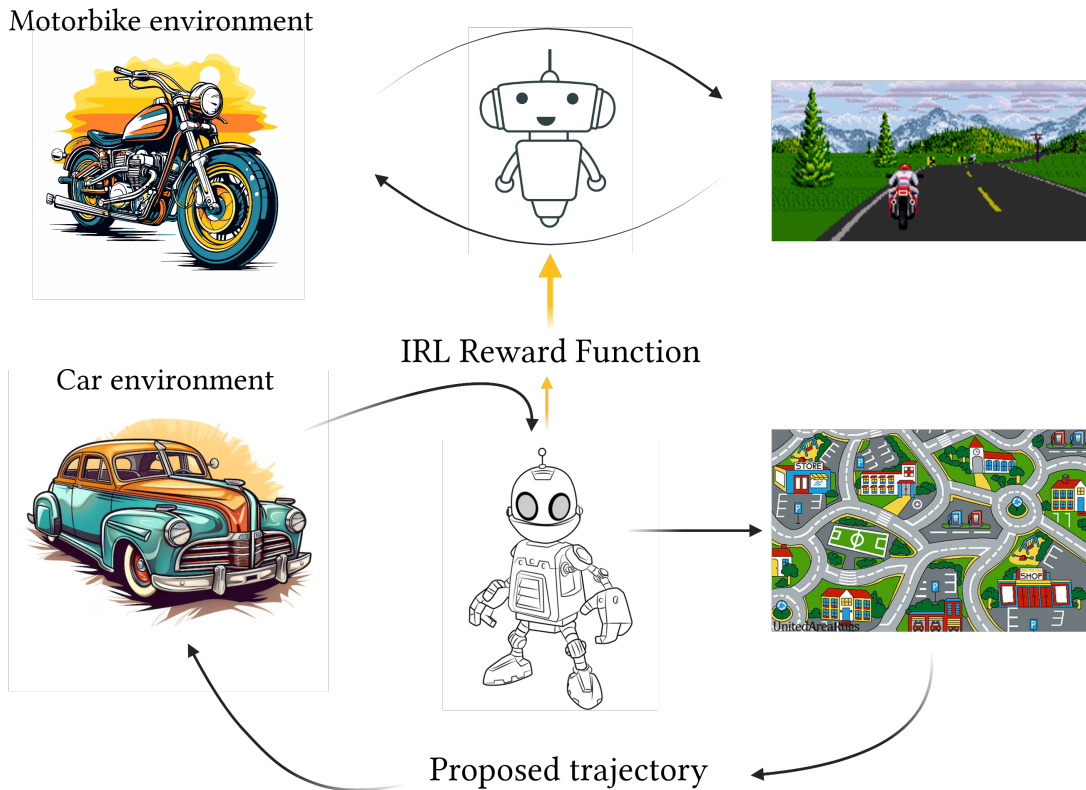


Figure 1.3: Given demonstrations of how to drive a car, a model can infer a reward function that can suit both the tasks of driving a car and riding a bike.

in computing power and training data. This tendency of AI to evolve by maximising resources is incompatible with the reality of human cognition that builds abstract concepts.

Moravec paradox is a good illustration of this situation. This paradox states that what humans do the most easily or even without thinking about it like analysing a scene, walk in a crowded street without colliding... are the most difficult issues for machines. On the other hand, it is easier for machines to solve complex mathematics problems, master the go game or hundreds of languages at superhuman levels. AI is harder than one might think because scientific knowledge is still vastly ignorant of our thought process complexity. As stated in [Mit21], What we call "reasoning" is possibly the smallest component of human mind, and only works so well because it is sustained by a much older and powerful knowledge, even non conscious. As Minsky declared himself, we are mainly unconscious of what our mind does best.

Humans build and then rely on a rich model of the world in which their learning is taking place. As a result of the development of these abilities, humans become effective and versatile learners, a strong contrast with current AI systems that require enormous amounts of training data and are highly specialised for particular tasks.

Meta-Learning intends to achieve the same by adopting a "learning to learn" approach that aims

to produce a model on a set of different but related tasks, and then generalise to new cases based on a few additional examples [FAL17]. Indeed, to acquire new skills, it is more useful to build on previous experience than to start from scratch. Thus, we learn through tasks requiring, at each iteration, less data and effort to conquer new skills. Meta-Learning constitutes a promising way to make models learn abstract concepts that are robust to distributional shift.

1.3.3 Hypothesis

We propose the hypothesis that acquiring a meta-reward across a set of tasks can mitigate the inner alignment problem and address distributional shifts. Meanwhile, utilising Inverse Reinforcement Learning (IRL) serves to tackle the outer alignment issue. This manuscript aims at laying the building blocks towards this ambitious objective.

More specifically, we draw two hypothesis to solve the outer/inner alignment problems.

- *Hypothesis 1:* Implementing an aligned base objective from scratch is hard and hard coding moral values is impossible. We suggest that pointing to the right behaviour directly using expert demonstrations can help alleviate the outer alignment problem if we use an IRL based approach
- *Hypothesis 2:* Learning a meta-reward on different but similar tasks can help the model build an understanding of abstract concepts that are more robust to distributional shift and that solve the inner alignment problem.

1.4 Contributions

The contributions of this thesis are as follows:

- Chapter 2 first covers all the necessary background to fully understand the content of this manuscript. It also gives entry to a reader willing to catch up with the latest literature.

In Chapter 3, we give an illustration of the alignment problem. We show that, given the same reward function, several succeeding policies³ exist that solve the task while adopting different behaviours. By doing this, we demonstrate the dual nature of the alignment problem. While altering the base objective (reward function) clearly results in distinct optimal policies, it's less apparent that the same reward function can yield various optimal policies that optimise the same base objective but have different mesa-objectives. We introduce an algorithm able to individually sample deterministic succeeding policies; thus allowing to observe different policies that solve the same task while adopting different behaviours. We argue that the difference in behaviours is due to a criteria uncaptured in the base objective but that can actually be considered as part of the mesa-objective. We also claim that the mesa-objective can be interpreted as a risk measure of distributional shift. Studying the resulting behaviours can help crafting a better reward function that matches better the desired behaviour, in favour of a better alignment between human and base objectives.

³policies that are acceptably close to optimal by human standards

- In chapter 4, we tackle the question of how to be more robust to the risk of distributional shift in RL. We argue that for such purpose, using expected return maximisation alone as a mesa objective is not sufficient, and that better inner alignment involves a distributional approach to Reinforcement Learning. However, existing approaches present several limitations such as the inability to use loss functions in practice that guarantee convergence in theory. Additionally, they learn target distributions only implicitly, forbidding the evaluation of a given return density under a specified policy. We present a distributional RL approach based on a particular invertible generative model that offers proper convergence guaranties. Indeed our approach allows to compute easily the Cramèr distance and we show that using it as a loss function brings several benefits.
- As exposed above, we believe that IRL is the best approach to handle the outer alignment issue. In chapter 5, we answer the question of how to use a distributional approach in IRL building on the model presented in chapter 4. Existing IRL approaches are based on adversarial models in order to tackle IRL challenges. However, adversarial models suffer from instability issues and are based on optimising a criteria that we do not find satisfactory (occupancy measure). We propose a collaborative model instead of an adversarial one that optimise the Cramèr distance and does not rely on occupancy measures.
- In chapter 6, we tackle the Meta-Learning problem and answer to hypothesis 2. We propose a state of the art model that learns meta-metrics that constitute good criteria for clustering problems when applied on datasets that were never seen by the model. In this chapter the clustering problem is seen as a simplification of the RL problem as it can be seen as a one step Markov Decision Process.

Chapter 2

Background

2.1 Reinforcement Learning

In this section we describe the Reinforcement Learning (RL) problem through Markov Decision Process (MDP) formalism. In doing so, we also introduce the notation that will be used in the remainder of the dissertation. We also describe some of the key ideas and algorithms in RL, and show how many important problems in decision making and control can be posed in this framework. For a more detailed introduction to RL and MDPs, readers may also refer to [PDM08; SB18].

2.1.1 Markov Decision Processes

MDPs provide a formalism for reasoning about planning and acting in scenarios marked by inherent uncertainty. There are many possible ways of defining MDPs, and many of these definitions are equivalent up to small transformations of the problem. One definition is that an MDP \mathcal{M} is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma, \zeta_0)$ consisting of:

- \mathcal{S} : a set of possible states of the world
- \mathcal{A} : a set of possible actions from which we may choose on each time step. ($|\mathcal{A}| \geq 2$)
- \mathcal{T} : The state transition distributions. For each state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$, this gives the distribution over to which state we will randomly transition if we take action a in state s .
- $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$: the reward function
- γ : a discount factor in $[0, 1]$
- ζ_0 : the distribution of possible initial states

The dynamics of events in an MDP are illustrated in the influence diagram depicted in figure 2.1. The process commences with the selection of an initial state s_0 , sampled from a distribution ζ_0 . At each discrete time step t , an action a_t is chosen, subsequently leading to a transition of the state to s_{t+1} according to the probability distribution $\mathcal{T}(\cdot | s_t, a_t)$. By iteratively picking actions,

Table 2.1: Summary of notations

\mathcal{S}	A set of possible states of the world
\mathcal{A}	A set of possible actions ($ \mathcal{A} \geq 2$)
$\mathcal{T}(\cdot s, a)$	State transition distributions
s	$s \in \mathcal{S}$
a	$a \in \mathcal{A}$
r	$\mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$, the reward function
γ	A discount factor in $[0, 1]$
ζ_0	The distribution of possible initial states
$\pi(\cdot s)$	$\mathcal{S} \times \mathcal{A}$ a policy considered as a probability law
$V^\pi(s)$	$\mathcal{S} \mapsto \mathbb{R}$ the state value function under policy π
$Q^\pi(s, a)$	$\mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$, the Q-function under policy π
$V^*(s)$	$\mathcal{S} \mapsto \mathbb{R}$, the optimal value function
$Q^*(s, a)$	$\mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$, the optimal Q-function
$Z^\pi(s, a)$	The random variable whose expectation is the value Q
$\mathcal{P}_\pi(\cdot s, a)$	The value distribution, the distribution of returns following policy π from state-action pair (s, a)
R^π	An outcome of Z^π such that $Z^\pi \sim \mathcal{P}_\pi(\cdot s, a)$
\mathcal{B}^π	The distributional Bellman operator for policy π
$\rho_\pi(s, a)$	Stationary distribution or occupancy measure for a Markov chain \mathcal{M}_π
Π	Set of policies, $\pi \in \Pi$
τ	A trajectory $(s_0, a_0, s_1, \dots, a_{n-1}, s_n)$
$R(\tau)$	Return of trajectory τ

we traverse a sequence of states denoted as s_0, s_1, \dots . The cumulative payoff is then computed as the sum of rewards, discounted over time, along this sequence of states:

$$r(s_0, a_0) + \gamma r(s_1, a_1) + \gamma^2 r(s_2, a_2) + \dots \quad (2.1)$$

The domains \mathcal{S} and \mathcal{A} are supposed finite even if many results show that it can be extended to the case where \mathcal{S} and \mathcal{A} are countable or continuous. In the general case, the domain \mathcal{A} can be dependent of the current state (\mathcal{A}_s for $s \in \mathcal{S}$). Transition probabilities characterise the system dynamics. For a fixed action a , $p(s'|s, a)$ represents the probability that the system goes to state s' after performing action a in state s . Classically, we have $\forall s, a, \sum_{s'} p(s'|s, a) = 1$. Moreover, we can also use a matrix representation of these transition probabilities, denoting P_a the matrix of dimension $|\mathcal{S}| \times |\mathcal{S}|$ which elements are $\forall s, s' \quad P_{a,s,s'} = p(s'|s, a)$. The probabilities described by $p(\cdot)$ take the form of $|\mathcal{A}|$ matrices P_a , each line of these matrices summing to 1. P_a are called stochastic matrices.

The distributions $p(\cdot)$ verify the fundamental property that gives its name to MDPs. Denoting h_t the process history at time step t , $h_t = (s_0, a_0, \dots, a_{t-1}, s_t)$, then the probability to reach a new state s_{t+1} after performing a_t is only function of a_t and the current state s_t . In summary the history h_t has no influence:

$$\forall h_t, a_t, s_{t+1} \quad p(s_{t+1}|h_t, a_t) = p(s_{t+1}|s_t, a_t)$$

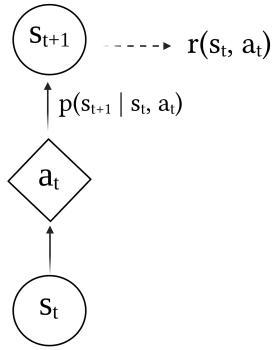


Figure 2.1: Influence diagram of an MDP. Action a_t is performed at state s_t , leading to the new state s_{t+1} according to $p(s_{t+1}|s_t, a_t)$. The agent receives the reward $r(s_t, a_t)$

In some environments, there is no guarantee that a trajectory reaches a terminal state after some time. In this case, trajectories become infinite and the sum of accumulated returns does not necessarily converge when the number of time steps n is infinite. To be able to keep applying the same definitions, it is possible to modify slightly the computation of the returns by defining a factor $\gamma \in [0, 1[$ called the discount factor. Instead of maximising the sum of rewards, the aim is now to maximise the series $\sum_t \gamma^t r_t$, that converges if the rewards given by the environment are bounded.

The discount factor which is typically strictly less than one, causes rewards obtained far down the sequence to be given a smaller weight. γ has a natural interpretation describing how much immediate rewards are more valuable than those obtained in the future and the value of γ allows to weight the importance we give to the future. Rewards can also be stochastic rather than deterministic functions of the state.

In RL, the goal is to find a way of choosing actions a_0, a_1, \dots over time, so as to maximise the expected value of the rewards given in equation 2.1. Given the Markov property, to attain the optimal expected sum of rewards, it suffices to choose actions only as a function of the current state s_t . Thus, RL can be posed as that of finding a good policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ such that if at some timestep t , in state s_t we take action $a_t \sim \pi(\cdot|s_t)$, we will obtain a large expected sum of rewards:

$$\mathbb{E}_\pi [r(s_0, a_0) + \gamma r(s_1, a_1) + \gamma^2 r(s_2, a_2) + \dots]$$

Throughout this manuscript, we will use the notation $\pi(\cdot|s)$ to denote a probability law and a_t a random variable such that $a_t \sim \pi(\cdot|s_t)$. Also we denote $\text{Supp}(\pi(\cdot|s_t))$ the set of possible values that a_t can take when drawn from $(\pi(\cdot|s_t))$.

The reward function r is the task description and specifies the objective we seek to optimise. It usually has to be given beforehand and its choice is crucial. Certain choices of rewards may allow an agent to learn orders of magnitude faster; other choices may cause the agent to learn suboptimal solutions. The choice of reward function can have a significant impact on the performance of RL algorithms.

2.1.2 Bellman Equations

We now review some standard definitions and results for MDPs, some of which will be useful in the subsequent chapters. Everything presented in this section holds straightforwardly for discounted MDPs with finite state and action spaces; with only small modifications. All the proofs can be found in [SB18]. In this section we will use $r(s, a)$ to denote the rewards. Given a policy π , define its value function $V^\pi : \mathcal{S} \mapsto \mathbb{R}$ to be the expected returns for taking actions according to π starting from a certain state s :

$$V^\pi(s_t) = \mathbb{E}_\pi[r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \gamma^2 r(s_{t+2}, a_{t+2}) + \dots], \quad (2.2)$$

$$s_{t+1} \sim \mathcal{T}(\cdot | s_t, a_t), a_t \sim \pi(\cdot | s_t)$$

Closely related to the value function is also the Q-function. For a given policy π , the Q-function $Q^\pi : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ gives the expected return for starting in a given state s , first taking a specified action, and then following policy π afterwards:

$$Q^\pi(s_t, a_t) = \mathbb{E}_\pi[r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \gamma^2 r(s_{t+2}, a_{t+2}) + \dots] \quad (2.3)$$

Using this definition, the RL objective can be restated as learning a policy π such that Q^π is maximal for every state-action pair. The Q-value is a useful tool to characterise a policy. This function verifies a recurrence relation called the Bellman evaluation equation:

$$\forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}, Q^\pi(s_t, a_t) = r(s_t, a_t) + \gamma Q^\pi(s_{t+1}, a_{t+1})$$

One can also define the bellman operator:

$$\forall f : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}, \mathbf{B}^\pi(f)(s_t, a_t) = r(s_t, a_t) + \gamma f(s_{t+1}, a_{t+1})$$

It is possible to show that this operator is a γ contracting operation. This implies that the Bellman operator admits a unique fixed point, thus the equation:

$$\mathbf{B}^\pi(f) = f$$

admits a unique solution that is equal to the Q-value Q^π [Put94].

We also define the optimal value function $V^* : \mathcal{S} \mapsto \mathbb{R}$ to be the optimal expected sum of rewards for starting from a state s :

$$V^*(s) = \max_{\pi} V^\pi(s)$$

Similarly the optimal Q-function can also be defined as

$$Q^*(s, a) = \max_{\pi} Q^\pi(s, a)$$

The quantities V^* and V^π satisfy the Bellman equations [BD62]:

$$V^*(s) = \max_a r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s' | s, a) V^*(s') \quad (2.4)$$

$$V^\pi(s) = r(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, \pi(s)) V^\pi(s') \quad (2.5)$$

Bellman equations give a recursive definition of V^* and V^π . Moreover, V^* and V^π (for a given π) are the unique solutions of the two equations above. Many quantities can be written in terms of either the Q or value functions, and we may usually pick whichever is more convenient to work with. Indeed, they are related via the identities

$$V^*(s) = \max_{a \in \mathcal{A}} Q^*(s, a)$$

$$V^\pi(s) = Q^\pi(s, \pi(s))$$

There is also an analogous form of equations 2.4, 2.5 for Q-values:

$$Q^*(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) \max_{a' \in \mathcal{A}} Q^*(s', a')$$

$$Q^\pi(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) Q^\pi(s', \pi(s'))$$

Depending on the initial distribution ζ_0 , and what state we start from, one may wonder if certain policies are better for starting from certain states. It is a remarkable fact of MDPs that there exists an optimal policy $\pi^* : \mathcal{S} \mapsto \mathcal{A}$, so that starting from any state, π^* attains the optimal expected returns.

$$V^{\pi^*}(s) = V^*(s) \quad \forall s \in \mathcal{S}$$

The optimal policy is not necessarily unique and many different behaviours can be optimal in the sense of the Q-value. However the Q-function associated to each policy is unique and noted Q^* . Knowing Q^* is enough to find an optimal policy in any given environment. Indeed, any greedy policy with respect to Q^* is necessarily an optimal policy for the underlying MDP.

π^* is given by either of the following (equivalent) definitions:

$$\pi^*(s) = \arg \max_{a \in \mathcal{A}} Q^*(s, a)$$

$$\pi^*(s) = \arg \max_{a \in \mathcal{A}} r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V^*(s') \quad (2.6)$$

2.1.3 Finding the Optimal Policy

There are 3 fundamental methods to find the optimal policy that mainly differ on how the value function is estimated.

Dynamic Programming

This approach requires complete knowledge of the environment dynamics (i.e., transition probabilities $p(s_{t+1}|s_t, a_t)$). It takes advantage of the recursive definition of V^π and V^* given by Bellman equations. Expressing the value function as the sum of the immediate reward and the next step value function, it becomes possible to iterate between a value evaluation step that calculates $V^\pi(s)$ and a policy improvement computing $Q^\pi(s, a)$ using equation 2.6. We therefore obtain a sequence of policies and value functions that get better continuously

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} V^{\pi_2} \xrightarrow{I} \dots \xrightarrow{I} \pi^* \xrightarrow{E} V^{\pi^*}$$

where \xrightarrow{E} corresponds to evaluation and \xrightarrow{I} corresponds to improvement. The dynamic programming process converges towards an optimal policy π^* and an optimal value function V^{π^*} .

Monte Carlo Methods

If probability transitions are unknown, it is possible to replace the model by sampling transitions directly from the environment to get complete trajectories and gather knowledge about its dynamics. This approach called Monte Carlo constitutes the first model free approach that was proposed in the literature. As for dynamic programming, Monte Carlo methods are also done in two steps, prediction (evaluation) and control (improvement). The control step remains the same as for dynamic programming, i.e greedy policy. However, during the prediction step t , the value function is entirely estimated using the empirical return R_t . In its incremental mean version, Monte Carlo method consists into computing the returns obtained after the first visit of a state until the end of the trajectory. This process is repeated over k episodes. The value function is estimated, for each episode as follows:

$$V^\pi(s_t) \leftarrow V^\pi(s_t) + \frac{1}{N(s_t)}(R_t - V^\pi(s_t))$$

Where $N(s_t)$ is the number of trajectories containing state s_t . As they only rely on empirical returns, they are unbiased estimators. However this implies that updates are only done at the end of each episode.

Temporal Difference

Temporal Difference (TD) methods play a significant role in contemporary reinforcement learning approaches, representing a blend of Monte Carlo and dynamic programming techniques. TD methods utilize bootstrapping to update value function estimates, amalgamating practical experience with approximate value function estimations. In contrast to Monte Carlo methods, which perform evaluations at the conclusion of each episode, TD updates occur subsequent to each state transition. Among these methods, Q-learning [WD92] stands out as a straightforward algorithm that employs TD updates. This algorithm employs the Bellman equation to estimate the Q-function Q^π :

$$Q^\pi(s_t, a_t) \leftarrow Q^\pi(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q^\pi(s_{t+1}, a) - Q^\pi(s_t, a_t)] \quad (2.7)$$

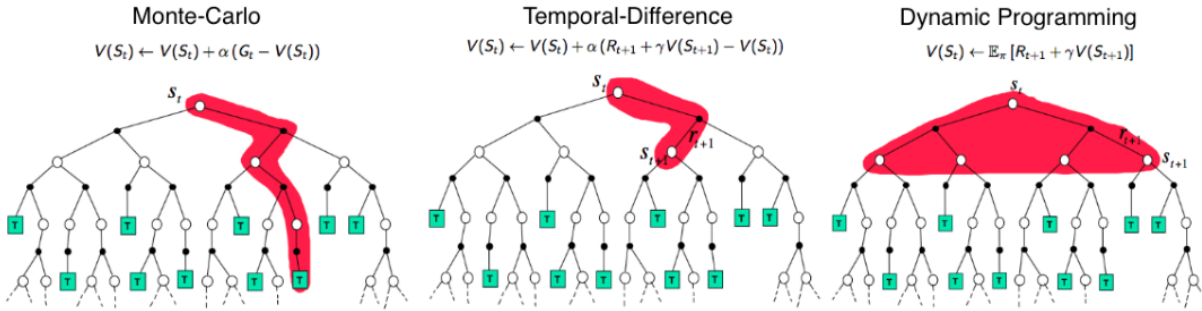


Figure 2.2: Illustration of Monte Carlo vs Temporal Difference vs Dynamic Programming. Source: <https://www.davidsilver.uk/wp-content/uploads/2020/03/MC-TD.pdf>.

Where α is a learning rate and γ a discount factor. Optimisation of Q^π is done through TD error minimisation [SB18] on encountered trajectories when the agent follows the current policy:

$$\delta_t = r_{t+1} + \gamma \max_a Q^\pi(s_{t+1}, a) - Q^\pi(s_t, a_t)$$

The best policy is then chosen in a greedy manner, retaining an action maximising the Q-function at each state.

The original Q-learning algorithm employs a tabular representation of $Q^\pi(s_t, a_t)$ with dimensions $|\mathcal{S}| \times |\mathcal{A}|$, where each row corresponds to a state and each column corresponds to an action. However, this configuration becomes impractical for scenarios involving infinite or continuous state and action spaces. Leveraging the universal approximation capabilities of neural networks, deep learning methods have been naturally integrated into reinforcement learning. One noteworthy advancement is the Deep Q-Network (DQN) introduced by Mnih et al. in 2015 [Mni+13]. DQN stands as the pioneer deep learning model capable of directly learning policies from high-dimensional sensor data, particularly from Atari games. In essence, DQN employs images captured from an Atari emulator as input, employing a Convolutional Neural Network (CNN) [Li+22]. The training procedure of DQN unfolds as follows:

1. Given a state s_t as input, the network outputs $Q(s_t, a_i^j)$ for each possible action a_i^j where $i \in [1, |\mathcal{A}|]$.
2. An action is chosen either greedily or at random. Then after performing the chosen action a_t , the environment outputs a new state s_{t+1} and a reward r_{t+1} .
3. s_{t+1} is then used as input and we keep an action $a_{t+1}^* \in \{a_{t+1}^1, \dots, a_{t+1}^{|\mathcal{A}|}\}$ for which $Q(s_{t+1}, a_{t+1}^*)$ is the highest.
4. Train the network using Mean Squared Error to minimise the difference between the predicted value $Q(s_t, a_t)$ and the target $Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}^*) - Q(s_t, a_t)]$.

Other approaches have improved on DQN like Double DQN [HGS15] that uses two distinct models, one for value evaluation and another for action selection. Bellemare et al [BDM17] introduce distributional RL that estimates the whole distribution of the Q-value instead of the expected return for each state-action pair. Being a central element of this thesis, Distributional RL will

be extensively introduced in the subsequent section. Finally, Rainbow [Hes+18] is considered as state of the art and combines 6 DQN improvements: DDQN, Prioritized replay [Sch+16], Dueling networks [WFL16], Multi-step learning, Distributional RL, and Noisy networks [For+18].

The methods discussed so far fall under the category of value-based algorithms. These approaches involve constructing optimal policies by initially learning a value function to estimate expected returns, and subsequently selecting actions in a greedy manner based on these value estimates. While value-based methods offer the benefit of generating lower variance in predicted returns, they also come with certain drawbacks. Firstly, the reliance on greedy action selection implies that the learned policies are deterministic, even though the optimal policy often exhibits stochastic behavior [Sut+99]. Secondly, minor modifications to the estimated value of an action can lead to significant changes in whether that action is selected or not. This issue has been recognized as a significant challenge in establishing convergence guarantees for value-based methods [Gro+12]. Lastly, value-based methods have a tendency to overestimate Q^π due to the propagation of positive errors through the max operator, which can degrade the model's precision [Has10].

The methods introduced in this thesis can all be considered value based. However, for completeness we mention the existence of policy based methods that maximise an approximation of expected returns without learning value functions. The result is a stochastic policy consisting of a distribution from which actions are sampled. We advise the interested reader to explore the dedicated section in [SB18]. Compared to value based methods, policy based methods are simpler and offer strong convergence guarantees. However, empirical returns variance is generally high implying a slow convergence.

Actor-critic methods are a class of reinforcement learning algorithms that combine the benefits of both policy-based and value-based methods. These algorithms simultaneously learn a policy and a value function, leveraging this dual approach to mitigate variance and accelerate learning. In an actor-critic algorithm, the agent (referred to as the "actor") learns by incorporating insights from the value function (the "critic"), typically in the form of temporal difference (TD) errors. The actor-critic interaction introduces a trade-off between reducing gradients variance and introducing bias from the value-based aspects of the algorithm. The foundational work on actor-critic methods is often attributed to [BSA83]. Since then, several variations of this approach have been developed.

For instance, in 2014, Silver et al. introduced Deterministic Policy Gradients (DPG) [Sil+14], which employs value functions to learn a deterministic policy. Another notable contribution is the Asynchronous Advantage Actor-Critic (A3C) algorithm proposed by Mnih et al. in 2016 [Mni+16]. In A3C, multiple agents operate in parallel, allowing for the decorrelation of data and diversification of experiences used for learning. This parallelization enhances the efficiency of the learning process and contributes to faster convergence.

2.2 Distributional Reinforcement Learning

As a large part of this thesis tackles the problem of learning the Q-functions as distributions over returns instead of expected returns, we will extensively present Distributional RL and the main papers on which our proposed approaches are mainly inspired.

Consider a very simple deterministic MDP as in figure 2.3 (left) with only 3 states and 1 possible action A , each state giving a reward $r = 1$ with $\gamma = 0.9$. In this case, using equations 2.2 and 2.3, we have $V(s_2) = Q(s_2, A) = 1$ and $V(s_1) = Q(s_1, A) = 1 + \gamma V(s_2) = 1.9$.

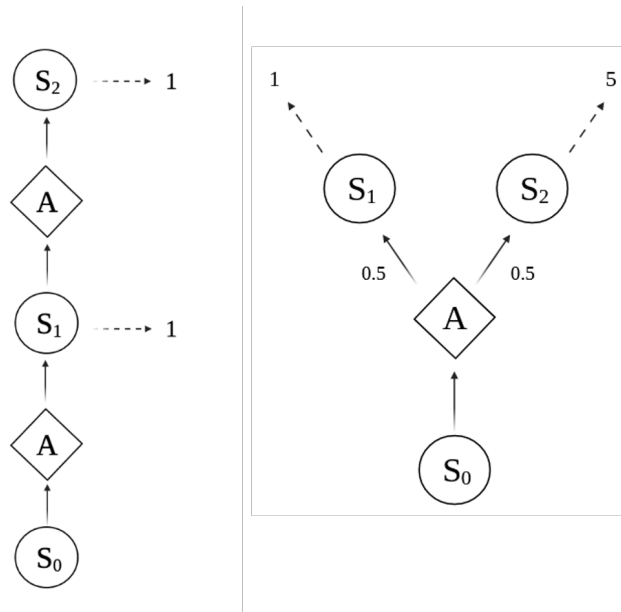


Figure 2.3: Influence diagrams examples of deterministic MDP (left) and stochastic MDP (right).

Figure 2.3 (right) presents the MDP of a simple stochastic environment with 3 possible states and 1 possible action at state s_0 while s_1 and s_2 are both final but $V(s_2) > V(s_1)$. The environment being stochastic means that each time the agent performs action A in state s_0 , the agent ends up in s_1 with probability p and in s_2 with probability $1 - p$. In this case $V(s_0) = p \cdot r(s_1, A) + (1 - p) \cdot r(s_2, A)$. If we fix $p = 0.5$, $r(s_1, A) = 1$ and $r(s_2, A) = 5$, then $V(s_0) = 3$.

However, it is important to acknowledge that a single value of $V(s_0)$ can correspond to an infinite number of situations. This is due to the fact that state values (as well as Q-values) are defined as expectations, lacking detailed information about the environment dynamics and the range of possible rewards. It is crucial to distinguish between intrinsic uncertainty, captured by the distribution over returns, and parametric uncertainty, which pertains to the uncertainty surrounding the estimation of values. Distributional RL aims to focus on capturing the former.

Indeed, a more informative representation of state values or Q-values would be to model them as distributions. As illustrated in figure 2.4, we ideally seek to learn the distributions of $Q(s_0, A)$ to gain a more comprehensive understanding of the underlying uncertainties and variability in the rewards.

Considering Q-values as distributions and trying to learn these distributions is the main objective of a sub-field of RL called Distributional Reinforcement Learning.

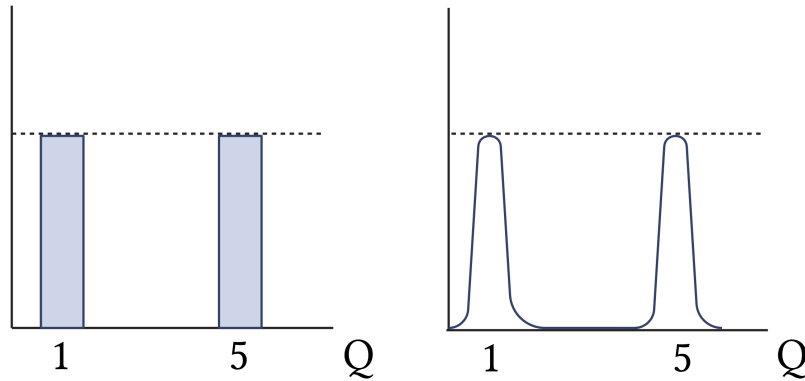


Figure 2.4: Target distributions for $Q(s_0, A)$ with a discrete support (left) on a continuous support (right). In the continuous case, the density of the atoms 1 and 5 being null, we suppose the existence of user defined regions around them of non nul measure with respect to the Lebesgue measure.

2.2.1 C51

Similarly to the Bellman equation already introduced in the previous section, the authors of C51 [BDM17] introduced the distributional Bellman equation:

$$Z(s, a) \stackrel{D}{=} \mathbf{R}(s, a) + \gamma Z(s', a') \quad (2.8)$$

Where Z is the random variable whose expectation is the value Q . Equation 2.8 states that the distribution of Z is characterised by the interaction of three random variables: the reward \mathbf{R}^1 , the next state-action pair (s', a') and its random return $Z(s', a')$. They call this quantity, the value distribution.

The principal contributions of this groundbreaking work is that for a fixed policy, the Bellman operator over value distributions is a contraction in a maximal form of the Wasserstein distance.

Given two random variables U, V with Cumulative Density Function CDFs F_U, F_V , the p -Wasserstein distance is defined as:

$$d_p(F_U, F_V) = \left(\int_0^1 |F_U^{-1}(u) - F_V^{-1}(u)|^p du \right)^{1/p} \quad (2.9)$$

And denoting \mathcal{Z} the space of value distributions with bounded moments, for two value distributions $Z_1, Z_2 \in \mathcal{Z}$, the maximal form of the Wasserstein metric is defined as :

$$\bar{d}_p(Z_1, Z_2) := \sup_{s, a} d_p(Z_1(s, a), Z_2(s, a))$$

¹here we denote the random variable of rewards by \mathbf{R} . It should not be confused with an outcome of Z^π noted R^π (return value) and the reward function noted r .

Now let us characterise the Bellman operator over value distributions. First, the authors define the transition operator $P^\pi : \mathcal{Z} \rightarrow \mathcal{Z}$

$$\begin{aligned} P^\pi Z(s, a) &\stackrel{D}{=} Z(s', a') \\ s' &\sim p(\cdot|s, a), a' \sim \pi(\cdot|s') \end{aligned} \tag{2.10}$$

The authors define the distributional Bellman operator $\mathcal{B}^\pi : \mathcal{Z} \rightarrow \mathcal{Z}$ as:

$$\mathcal{B}^\pi Z(s, a) \stackrel{D}{=} \mathbf{R}(s, a) + \gamma P^\pi Z(s, a)$$

One of the most important result of the article consists in proving that \mathcal{B}^π is a γ -contraction in \bar{d}_p . Indeed, as for classic RL, they use Banach’s fixed point theorem to show that \mathcal{B}^π has a unique fixed point, which is Z^π . This has the important consequence that the same recursive process as in classic RL can be used such that $\{Z_k\}$ converges to Z^π in \bar{d}_p .

They also remark that not all distributional metric are equivalent, as the distributional Bellman operator is not a contraction for the total variation distance, or the KL divergence.

In practice, given a parameterised value distribution, Z_θ , although it would seem natural to minimise the Wasserstein distance between $\mathcal{B}Z_\theta$ and Z_θ , we are typically restricted in a RL context to learn from sample transitions, which they prove is not possible under the Wasserstein loss. Therefore, the authors used the KL divergence instead, and still managed to obtain impressive results.

These result are essential regarding the approaches presented in the next section of this manuscript. Indeed, one of the main results of our works builds on these results to propose another distance that would alleviate the intrinsic limitations of the Wasserstein distance.

Along with the lack of convergence guarantee due to the use of KL divergence, C51 bears several other limits. For instance, C51 is limited to discrete and fixed supports for the distribution of returns. Moreover, the target and predicted distributions do not share the same support by construction, the use of KL divergence therefore requires a computationally heavy projection of the target support into the predicted one.

With QR-DQN [Dab+17], the authors of C51, overcame this limitation using an algorithm that is basically a transposition of C51.

2.2.2 Quantile Regression DQN (QR-DQN)

While C51 learns flexible distributions on a fixed support, QR-DQN [Dab+17] learns a fixed set of probabilities on a flexible support.

The aim of QR-DQN is to be able to learn the distribution of returns while training the model using the Wasserstein distance. As stated in equation 2.9, computing this distance between two distributions requires having access to their quantile functions, i.e. the inverse of the CDFs. Therefore the idea of the paper is to learn the quantiles of the predicted and target distributions

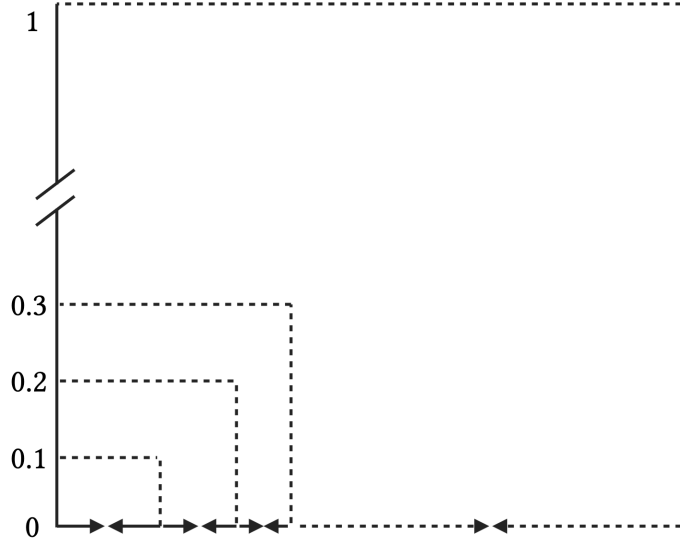


Figure 2.5: In quantile regression, rather than learning what probabilities are assigned to a fixed set of supports, the model learns a set of supports that correspond to a fixed set of probabilities (quantiles).

to ease the calculation of the Wasserstein distance and to overcome the impossibility to learn from sample transitions with the Wasserstein distance.

To do so, they use quantile regression [Koe05] to approximate the quantile functions. This method is known for being unbiased. More specifically, quantile regression has been shown to converge to the true quantile function value when minimised using stochastic approximation.

In QR-DQN, the random return is approximated by a uniform mixture of N Diracs,

$$Z_{\theta}(s, a) := \frac{1}{N} \sum_{i=1}^N \delta_{\theta_i(s, a)}$$

with each θ_i assigned a fixed quantile target, $\hat{v}_i = \frac{v_{i-1} + \tau_i}{2}$ for $1 \leq N$, where $v_i = i/N$. These quantile estimates are trained using the Huber quantile regression loss [Hub64] with threshold κ ,

$$\rho_{\tau}^{\kappa}(\delta_{ij}) = |v - \{\delta_{ij} < 0\}| \frac{\mathcal{L}_{\kappa}(\delta_{ij})}{\kappa}, \quad \text{with}$$

$$\mathcal{L}_{\kappa}(\delta_{ij}) = \begin{cases} \frac{1}{2} \delta_{ij}^2, & \text{if } |\delta_{ij}| \leq \kappa \\ \kappa(|\delta_{ij}| - \frac{1}{2}\kappa), & \text{otherwise} \end{cases}$$

on the pairwise TD-errors:

$$\delta_{ij} = r(s_t, a_t) + \gamma \theta_j(s_{t+1}, a_{t+1}) - \theta_i(s_t, a_t)$$

The quantile regression loss, for quantile $\nu \in [0, 1]$ is an asymmetric convex loss function that penalises overestimation errors with weight ν and underestimation errors with weight $1 - \nu$. For a distribution Z , and a given quantile ν , the value of the quantile function $F_Z^{-1}(\nu)$ may be characterised as the minimizer of the quantile regression loss. The minimising values of $\{\theta_1, \dots, \theta_N\}$ for $W_1(Z, Z_\theta)$ are those that minimise:

$$\sum_{i=1}^N \mathbb{E}_{\hat{Z} \sim Z} [\rho_{\hat{\nu}_i}(\hat{Z} - \theta_i)]$$

This loss gives unbiased sample gradients and we can find the minimising $\{\theta_1, \dots, \theta_N\}$ using stochastic gradient descent.

To summarise, the advantages of QR-DQN over C51 are the following:

- No need to choose the bounds of the support as the support is now unbounded, nor the number of bins to discretize it.
- No need to project the target distribution support into the predicted distribution
- It is now possible to use the Wasserstein distance and perform training using SGD

2.2.3 Implicit Quantile Networks (IQN)

IQN [Dab+18] extends QR-DQN approach by learning the whole quantile function instead of a finite set of quantiles. It takes $\nu \in \mathcal{U}([0, 1])$ as input and finds the corresponding quantile values in the target distribution. Let $F_Z^{-1}(\nu)$ the quantile function for $\nu \in [0, 1]$ and for the random variable Z . Denote $Z_\nu = F_Z^{-1}(\nu)$. For $\nu \sim \mathcal{U}([0, 1])$, the output of the return distribution for a given state-action pair is $Z_\nu(s, a) \sim Z(s, a)$.

More specifically, the authors propose to learn a quantile function of state-action pairs as an application from (s, a, ν) to $Z_\nu(s, a)$. $Z_\nu(s, a)$ has to be seen as samples taken from the return distribution implicitly defined. As for C51 or QR-DQN, IQN can estimate the distribution of returns but IQN does not output the distribution itself. Instead, it will output a unique sample from the distribution at each iteration.

In figure 2.6, we see that DQN outputs the mean of the returns, C51 a distribution on a fixed support, QR-DQN outputs uniform probabilities on a flexible support while IQN outputs a unique value estimation for the return at each request but on a continuous support.

2.3 Inverse Reinforcement Learning

After introducing distributional RL, and introducing our approach of the problem in chapter 4, chapter 5 will present how to use the distributional approach to the Inverse RL (IRL) problem.

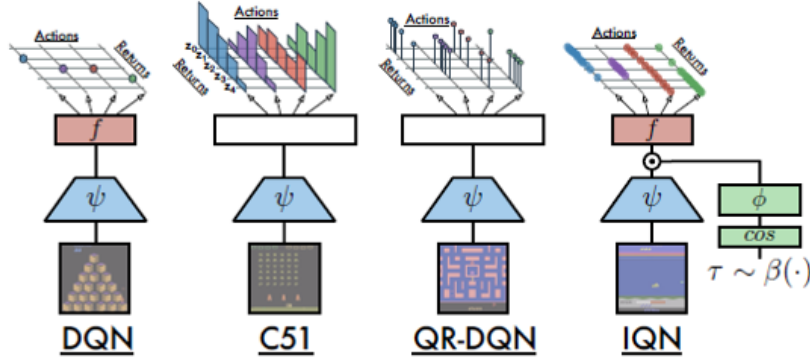


Figure 2.6: Network architecture for DQN and recent distributional algorithms. The figure was taken from [Dab+18].

Therefore, in this section we will present the IRL problem, its main challenges as well as the principal approaches on which we build our distributional adaptation.

Solving a RL problem requires several elements: an environment with its states and transition dynamics, a task, an agent and its possible actions, and surely most importantly a reward function. This list is actually redundant, as the reward function is supposed to perfectly describe the task at hand and how to solve it. More specifically, maximising the reward function should lead to solve the task with the desired behaviour. Therefore rather than a task and a reward function, one only needs a good reward function. However, designing a good reward function for the task at hand is rarely easy. For instance, for some score based video games like Tetris or Space Invaders, defining a reward function is straightforward, as maximising the score is the goal by itself. In Mujoco’s Ant environment- [TET12], maximising the distance is also a good reward function. However, if we try to define a good reward function that could be used to train an autonomous car driving algorithm, the task appears less straightforward, if not impossible.

A solution to tackle such an issue is to use an expert (usually a human) that will demonstrate what following the optimal policy looks like. The main hypotheses are that the expert follows a certain reward or utility function that it seeks to maximise, and while maximising it, it displays a certain behaviour which consequence is solving the task.

2.3.1 Behavioral Cloning

This is the most straightforward approach in situations where we do not have access to a reward function. The starting point is to notice that if the learner could replicate the expert’s action, it would solve the task, following the optimal policy and implicitly maximise the right reward function. This approach is called Behavioural Cloning². We will now present the easiest approach for Behavioural Cloning, called Dataset Aggregation (DAgger) [RGB11]. In this approach, the expert’s actions are recorded, then we perform a supervised learning method to replicate the expert’s action. For instance, when facing state s_t , the expert will perform action a_t , then all pairs

²There exist plenty names in the literature like Apprenticeship learning or Supervised Imitation Learning

(s_t, a_t) shown by the experts are used in a supervised classification algorithm to learn policy $\pi_\theta(a_t|s_t)$.

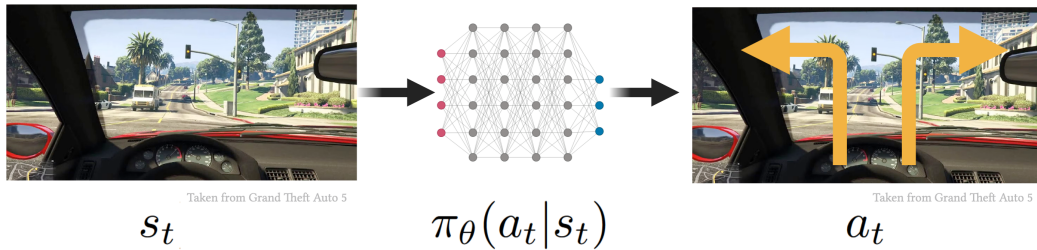


Figure 2.7: Behavioural Cloning. Given a dataset of state action pairs, a neural network is trained to predict the expert actions given a certain state.

While this approach is straightforward, it bears some severe limitation. Suppose, the learnt policy is close to the optimal policy with some minor errors. Errors are likely to compound and lead the agent towards sub-optimal states in which the expert never lands. In those new states, the agent cannot replicate the expert’s actions as they were never encountered in the training dataset. The performed action may therefore be random which will certainly worsen an already terrible situation. This is called the distributional shift as illustrated in figure 2.8. This means that the state distribution encountered during training is not the same as the one encountered at test time.

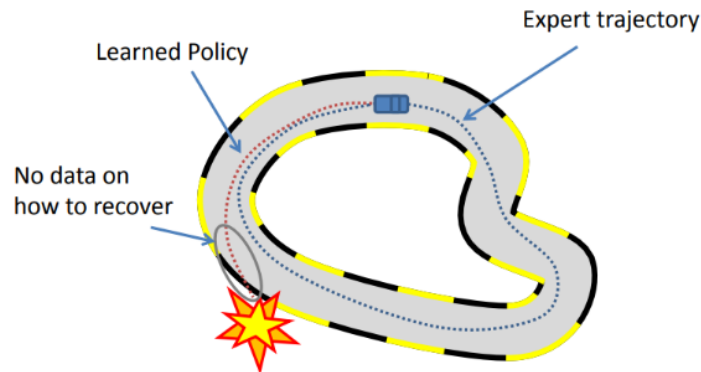


Figure 2.8: Distributional shift. In an autonomous driving setting, minor errors can make the agent shift from demonstrated optimal states from which it is not trained to recover. Source: https://cse.buffalo.edu/~avereshc/rl_fall19/lecture_15_Imitation_Learning_Behavior_Cloning_IRL.pdf.

To alleviate distributional shift, DAgger trains the agent in an online and iterative manner. The agent is first trained on the expert data, then the obtained policy generates new states to which the expert gives the optimal action and the iterations go on. The process is as follows and goes for a user defined number of epochs:

1. Train $\pi_\theta(a_t|s_t)$ form expert data ($D = \{s_1, a_1, \dots, s_N, a_N\}$)

2. Run π_θ to get a new dataset $D_\pi = \{s_1^\pi, \dots, s_t^\pi\}$
3. Ask the expert to label D_π with optimal actions and obtain dataset $D_{\pi_E} = \{s_1^\pi, a_1, \dots, s_t^\pi, a_t\}$
4. Aggregate: $D \rightarrow D \cup D_{\pi_E}$
5. Repeat

However, DAgger is not capable of long term planning and it is only effective if the expert is able to browse through a large part of the state space, or when that state space is rather small.

In contrast to the approach of directly replicating behavior, inverse reinforcement learning assumes rational agents to estimate an unknown reward function that represents their underlying motivations and goals. The reward function is often considered as the most succinct, robust and transferable representation of the expert's objective [AN04]. Given a set of demonstrations D_{π_E} from an expert policy π_E , IRL [Rus98; NR00] is the problem of seeking a reward function from which we can recover π_E through RL. However, IRL in unregularized MDPs has been shown to be an ill-defined problem since:

1. many optimal policies can explain a set of demonstrations
2. multiple rewards meet the criteria of being a solution, i.e many rewards can explain an optimal policy

2.3.2 Maximum Entropy IRL

Maximum Entropy Inverse Reinforcement Learning (MaxEntIRL) [Zie+08; Zie10] offers a strategy to mitigate the first concern. This approach involves a reward function that not only maximises the expert's cumulative return but also incorporates the Shannon entropy of the expert policy.

Consider a MDP defined by the usual tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma, \zeta_0)$. The initial state distribution ζ_0 and the reward function $r(s, a)$ are unknown. If we combine this setting with a stochastic policy π from the set of policies Π , i.e. a conditional probability distribution on \mathcal{A} given some state $s \in \mathcal{S}$, we obtain a Markov chain $\mathcal{M}_\pi = (s_0, a_0, s_1, a_1, \dots)$ in the following natural way: take a random starting state $s \sim \zeta_0$, choose an action $a \sim \pi(\cdot|s)$ and then restart the chain with probability $1 - \gamma$ or choose the next state $s' \sim p(\cdot|s, a)$ otherwise; then repeat the last two steps.

Therefore \mathcal{M}_π has a stationary distribution (or occupancy measure) ρ_π which satisfies

$$\rho_\pi(s, a) = (1 - \gamma\pi(a|s)) \sum_{t=0}^{\infty} \gamma^t p(s_t = s)$$

as well as the Bellman equation:

$$\sum_{a' \in \mathcal{A}} \rho(s', a') = (1 - \gamma)\zeta_0(s') + \gamma \sum_{(s, a) \in \mathcal{S} \times \mathcal{A}} p(s'|s, a) \rho(s, a)$$

for all $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$. Moreover, there is a one to one correspondence between those measures and the policies in Π . This is proven in theorem 2 of [SBS08]:

Proposition: The mapping $\rho \mapsto \pi_\rho$ defined by $\pi_\rho(a|s) := \rho(s, a) / \sum_{a' \in \mathcal{A}} \rho(s, a')$ is a bijection between Π and the set \mathcal{C} of measures on $\mathcal{S} \times \mathcal{A}$ satisfying the Bellman equation.

The direct consequence of that theorem is that we can write

$$\mathbb{E}_\pi[X(s, a)] := \mathbb{E}_{\rho_\pi}[X(s, a)] := \mathbb{E}_{(s,a) \sim \rho_\pi}[X(s, a)]$$

for the expected value of a random variable X on $\mathcal{S} \times \mathcal{A}$ with respect to ρ_π . We observe that the expected cumulative reward, i.e; the expected sum of rewards $r(s_t, a_t)$ up to the first restart of the chain is given by $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$. Hence, it is easy to derive that it is also equal to $\mathbb{E}_\pi[r(s, a)/(1 - \gamma)]$. We can also write $\langle r, p \rangle := \sum_{(s,a) \in \mathcal{S} \times \mathcal{A}} r(s, a) \rho(s, a)$.

The main conclusion of these observations is that matching the expert's occupancy measure can be considered as a good criteria to replicate its behaviour. MaxEntIRL is based on the minimisation of the difference between the agent's empirical occupancy measure and the expert's. This distribution (either from the expert, or observed) can be represented as:

$$\sum_{\text{path } \tau_i} P(\tau_i) f_{\tau_i} = \bar{f}$$

Where P is the probability of trajectory τ and f the state features.

Some preliminary definitions are mandatory:

- **Feature vector:** Each state can be represented as a feature vector f . Given a state s , its feature vector is noted f_s
- **Feature count:** Given a trajectory τ , the trajectory's feature count is defined as the sum of the feature vectors of all states in the trajectory

$$f_\tau = \sum_{s \in \tau} f_s$$

The return function can then be defined as the linear combination between features:

$$R_\theta(\tau) = \theta_1 f_{s_1} + \theta_2 f_{s_2} + \dots + \theta_T f_{s_T}$$

where the weights θ are the weights to learn.

$$\begin{aligned} R_\theta(\tau) &= \sum_{s \in \tau} \theta^T f_s \\ &= \theta^T f_\tau \end{aligned} \tag{2.11}$$

The probability that a certain trajectory is drawn from the expert demonstrations is considered as proportional to the exponent of the return function. This hypothesis is directly related to the well known Boltzmann distribution.

$$p(\tau) = \frac{\exp(R_\theta(\tau))}{Z}$$

With $Z = \sum_\tau \exp(R_\theta(\tau))$ a normalising constant.

Another advantage of this approach is that it remains valid for stochastic environment if we parameterise the equation using the transition matrix:

$$P(\tau|\theta, T) \approx \frac{e^{\theta^T f_\tau}}{Z(\theta, T)} \prod_{s_{t+1}, a_t, s_t \in \tau} P_T(s_{t+1}|a_t, s_t)$$

The objective is to maximise $p(\tau)$, i.e to maximise the probability that the agent replicates expert trajectories.

$$\theta^* = \arg \max_{\theta} L(\theta) = \arg \max_{\theta} \sum_{\text{examples}} \log P(\bar{\tau}|\theta, T)$$

The objective function can then be defined as:

$$L(\theta) = \frac{1}{M} \log p(\tau)$$

With M the number of demonstrations. Therefore:

$$\begin{aligned} L(\theta) &= \frac{1}{M} \log \left(\frac{\exp(R_\theta(\tau))}{Z} \right) \\ &= \frac{1}{M} (\log \exp(R_\theta(\tau)) - \log Z) \\ &= \frac{1}{M} (R_\theta(\tau) - \log Z) \\ &= \frac{1}{M} \left(R_\theta(\tau) - \log \sum_{\tau} \exp(\theta^T f_\tau) \right) \\ &= \frac{1}{M} \left(\theta^T f_\tau - \log \sum_{\tau} \exp(\theta^T f_\tau) \right) \end{aligned} \tag{2.12}$$

And

$$\begin{aligned} \nabla_{\theta} L(\theta) &= \frac{1}{M} \left(f_\tau - \frac{1}{\sum_{\tau} \exp(\theta^T f_\tau)} \sum_{\tau} \theta^T \exp(\theta^T f_\tau) \right) \\ \text{Note that } \frac{1}{\sum_{\tau} \exp(\theta^T f_\tau)} \sum_{\tau} \theta^T \exp(\theta^T f_\tau) &= \sum_{\tau} p(\tau|\theta) \\ &= \frac{1}{M} \left(\sum_{\tau} f_\tau - \sum_{\tau} p(\tau|\theta) f_\tau \right) \\ &= \frac{1}{M} \sum_{\tau} f_\tau - \frac{1}{M} \sum_{\tau} p(\tau|\theta) f_\tau \end{aligned} \tag{2.13}$$

The mean of the feature count is the feature expectation \tilde{f} , and we can replace $\frac{1}{M} \sum_{\tau} f_{\tau} = \tilde{f}$:

$$\nabla_{\theta} L(\theta) = \tilde{f} - \frac{1}{M} \sum_{\tau} p(\tau|\theta) f_{\tau}$$

The problem can further be simplified by summing over states.

$$\nabla_{\theta} L(\theta) = \hat{f} - \sum_s p(s|\theta) f_s$$

The first term can be easily calculated but not the second. Indeed, given a policy π we can use Monte-Carlo to compute the state visitation frequency. However, there is no policy available yet. Therefore, it is necessary, after each update on the reward function, to use Dynamic Programming to solve the problem and find the optimal policy for the current reward. Indeed, let $\mu_t(s)$ be the probability to visit state s at time t . Then $\mu_{t+1}(s) = \sum_a \sum_{s'} \mu_t(s') \pi(a|s') p(s|s', a)$. Therefore the state visitation frequency is $p(s|\theta) = \frac{1}{T} \sum_t \mu_t(s)$. The complete algorithm goes as follows:

1. Initialise parameters θ and gather demonstrations D
2. For N iterations:
 - (a) Calculate return $R_{\theta}(\tau) = \theta^T f_{\tau}$
 - (b) Calculate the optimal policy according to the current return function using policy iteration
 - (c) Compute state visitation frequency $p(s|\theta)$
 - (d) Calculate gradient for θ : $\nabla_{\theta} L(\theta) = \tilde{f} - \sum_s p(s|\theta) f_s$
 - (e) Update $\theta \leftarrow \theta + \alpha \nabla_{\theta} L(\theta)$

One can notice that using value iteration imposes to know the dynamics on the environment beforehand. Moreover, the necessity to solve the problem for each new return function constraints the method to be only usable on small environments with small state and action spaces.

2.3.3 Generative Adversarial Imitation learning (GAIL)

GAIL [HE16] brings two improvements to MaxEntIRL. First, it is a model free approach that does not require the environment dynamics. Also, it leverages adversarial networks to better learn the expert's occupancy measure while scaling to larger state and action spaces. Readers that are not familiar with Generative Adversarial Networks (GAN) can refer to section 2.5 that offers deeper details on the subject.

Let π_E and π_{θ} be respectively the expert policy and the model's policy. The aim of GAIL is to minimise the JS divergence between their occupancy measures. The process is similar to a GAN which generator has to learn a policy given the occupancy measure of π_E . The used loss will have the same form as a GAN's:

$$\max_{\theta} \min_{\omega} \mathbb{E}_{\pi_{\theta}} [\log(D_{\omega}(s, a))] + \mathbb{E}_{\pi_E} [\log(1 - D_{\omega}(s, a))]$$

Where θ are the weights of the generator and ω the weights of the discriminator D_ω .

In RL, the goal is to find the optimal policy that obtains maximum return. This can be expressed as:

$$RL(r) = \arg \max_{\pi} \mathbb{E}_{\pi} [r(s, a)]$$

We can also maximise the policy's entropy along with the return:

$$RL(r) = \arg \max_{\pi} H(\pi) + \mathbb{E}_{\pi} [r(s, a)]$$

Where $H(\pi) = \mathbb{E}_{\pi} [-\log \pi(a|s)]$. Instead of maximising a reward, the authors prefer to minimise a cost, in order to better comply to the original GAN formulation.

$$RL(c) = \arg \min_{\pi \in \Pi} -H(\pi) + \mathbb{E}_{\pi} [c(s, a)] \quad (2.14)$$

In the IRL setting, the aim is to learn the cost function c that assigns a low cost to the expert policy and high cost to all the others.

$$IRL(\pi_E) = \arg \max_c \left(\min_{\pi} -H(\pi) + \mathbb{E}_{\pi} [c(s, a)] \right) - \mathbb{E}_{\pi_E} [c(s, a)]$$

The authors proved that this objective is equivalent to a GAN objective that minimises the JS divergence between the learner and expert occupancy measures:

$$\max_{\theta} \min_{\omega} \mathbb{E}_{\pi_{\theta}} [\log(D_{\omega}(s, a))] + \mathbb{E}_{\pi_E} [\log(1 - D_{\omega}(s, a))] - \lambda H(\pi)$$

The role of the generator is to learn the expert policy while the discriminator has to classify whether the proposed trajectories are drawn from the expert or the learner. The generator is trained using the same procedure as a classic RL problem. In the article, the authors use Trust Region Policy Optimisation [Sch+15] while the discriminator is a simple feedforward neural network.

With this method we do not need to know the environment dynamics but it is not IRL. The only output is a policy that is close to the expert but the trained discriminator does not correspond to a reward function. Therefore GAIL is considered as an Imitation Learning algorithm rather than IRL.

2.3.4 Guided Cost Learning

Guided Cost Learning (GCL) [FLA16] uses an adversarial approach as in GAIL but this time, it serves the purpose of learning a proper reward function and therefore is considered as an IRL approach.

As for MaxEntIRL, we suppose that the expert draws its trajectories from the following distribution:

$$p(\tau) = \frac{1}{Z} \exp(-c_{\theta}(\tau))$$

However Z is hard to compute. MaxEntIRL, computes Z exactly using Dynamic Programming but this is only doable if the environment is at a low scale. In GCL, Z is estimated using samples, as the environment dynamics are unknown. We recall that the objective is maximise the log likelihood of $p(\tau)$ and using importance sampling, the authors deduce the following objective for the cost c_θ :

$$\begin{aligned}
\mathcal{L} &= \max_{\theta} \prod_D p(\tau|c_\theta) \\
&= \max_{\theta} \prod_D \frac{1}{Z} \exp(-c_\theta(\tau)) \\
&= \max_{\theta} \sum_D \log\left(\frac{1}{Z} \exp(-c_\theta(\tau))\right) \\
&= \max_{\theta} \sum_D -\log Z - \log c_\theta(\tau) \\
&= \min_{\theta} \sum_D \log c_\theta(\tau) + \log Z \\
&\approx \min_{\theta} \frac{1}{N} \sum_D \log c_\theta(\tau) + \log Z \\
&= \min_{\theta} \frac{1}{N} \sum_D \log c_\theta(\tau) + \log \sum \exp(-c_\theta(\tau)) \\
&= \min_{\theta} \frac{1}{N} \sum_D \log c_\theta(\tau) + \log \sum \frac{q(\tau) \exp(-c_\theta(\tau))}{q(\tau)} \\
&= \min_{\theta} \frac{1}{N} \sum_D \log c_\theta(\tau) + \log \mathbb{E} \left[\frac{\exp(-c_\theta(\tau))}{q(\tau)} \right] \\
&\approx \min_{\theta} \frac{1}{N} \sum_D \log c_\theta(\tau) + \log \frac{1}{M} \sum \frac{\exp(-c_\theta(\tau))}{q(\tau)}
\end{aligned} \tag{2.15}$$

The authors used importance sampling to introduce an auxiliary distribution $q(\tau)$. It is basically a policy that can output trajectories and calculate the probability of any trajectory given that policy. This policy can be trained using the KL divergence: $KL(q(\tau) || \frac{1}{Z} \exp(-c_\theta(\tau)))$. The GCL process is iterative as the model alternates between updating c_θ and a policy optimisation procedure that improves $q(\tau)$.

2.3.5 GAN-GCL

In [Fin+16], the authors of GCL draw a deep connection between IRL and Generative Adversarial Models. Here they really use a GAN's discriminator loss. Given $p(\tau)$ the real distribution and $q(\tau)$ the density output by the generator/model/policy, the optimal discriminator D^* has the following form:

$$\begin{aligned}
D^*(\tau) &= \frac{p(\tau)}{p(\tau) + q(\tau)} \\
&\approx \frac{p_\theta(\tau)}{p_\theta(\tau) + q(\tau)} \\
&\approx \frac{\frac{1}{Z} \exp(-c_\theta(\tau))}{\frac{1}{Z} \exp(-c_\theta(\tau)) + q(\tau)}
\end{aligned} \tag{2.16}$$

We recall the loss of a discriminator:

$$\mathcal{L}(D_\theta) = \mathbb{E}_{x \sim p}[-\log(D_\theta(x))] + \mathbb{E}_{x \sim G}[-\log(1 - D_\theta(x))]$$

where G is the generator of the GAN model.

It can then be adapted to the IRL case:

$$\begin{aligned}
\mathcal{L}(\theta) &= \mathbb{E}_{\tau \sim p}[-\log(D_\theta(\tau))] + \mathbb{E}_{\tau \sim q}[-\log(1 - D_\theta(\tau))] \\
&= \mathbb{E}_{\tau \sim p} \left[-\log \frac{\frac{1}{Z} \exp(-c_\theta(\tau))}{\frac{1}{Z} \exp(-c_\theta(\tau)) + q(\tau)} \right] + \mathbb{E}_{\tau \sim q} \left[-\log \frac{q(\tau)}{\frac{1}{Z} \exp(-c_\theta(\tau)) + q(\tau)} \right]
\end{aligned} \tag{2.17}$$

By using an energy function in the discriminator, this energy function now works as a learned reward function, which was not the case in GAIL. Furthermore, one can notice that the auxiliary distribution $q(\tau)$ obtained from the current policy plays the role of a generator.

2.3.6 Robust Rewards with Adversarial Inverse Reinforcement Learning (AIRL)

The authors of AIRL [FLL18] improved on GCL by first noticing that rewards learnt using GCL are not accurate when the dynamics of the model change. Indeed, as IRL is ill defined, most of IRL algorithms find it challenging to distinguish between true reward functions and those formed by the environment dynamics. To illustrate this issue we can take the example of situations where an algorithm has to learn a reward function to drive a car in wet or dry roads. These two situations happen in the same environment but with different dynamics. The set of optimal reward functions for dry roads is different from the one for wet roads. However, these two sets should overlap, i.e. there should exist reward functions that work for both dynamics of the same environment. These are considered as the true reward functions. The reward functions that are only valid for one of the situations are considered as being formed by the environment dynamics and have to be discarded.

The aim of AIRL is to learn rewards that are invariant to change in dynamics, which they call disentangled rewards.

Based on the work of [NR00], they show that existing methods cannot learn robust reward functions. [NR00] describes a class of reward functions that preserve optimal policies. Indeed, applying the following transformation:

$$\hat{r}(s, a, s') = r(s, a, s') + \gamma\phi(s') - \phi(s)$$

the optimal policy remains the same for any function $\phi : \mathcal{S} \mapsto \mathbb{R}$. Moreover, without any prior knowledge on the dynamics, it is the only class of transformations that allow policy invariance. As IRL methods only find rewards from expert demonstrations, they cannot discriminate between reward functions inside this class of transformations; unless the class itself is constrained.

Using this observation, they propose to adapt the form of GCL discriminator:

$$D_{\theta, \phi}(s, a, s') = \frac{\exp(f_{\theta, \phi}(s, a, s'))}{\exp(f_{\theta, \phi}(s, a, s') + \pi(a|s))}$$

$$f_{\theta, \phi}(s, a, s') = g_{\theta}(s, a) + \gamma h_{\phi}(s') - h_{\phi}(s)$$

Where g_{θ} is a reward estimator and h_{ϕ} a shaping term.

The central idea of this approach is that in order to erase undesired reward shaping, the reward function has to only be a function of the current state.

2.3.7 Wasserstein Adverse Imitation Learning (WAIL)

Previous approaches try to match occupancy measures $\rho_E := \rho_{\pi_E}$ by minimising the JS divergence between them. A natural follow up described in [Xia+19] is to try to optimise a proper distance between both distributions. [Xia+19] propose to minimise the Wasserstein distance instead. Let \mathcal{R} the space of possible reward functions, the IRL loss 2.14 can be reformulated as follows:

$$\arg \min_{\pi \in \Pi} -H(\pi) + \sum_{r \in \mathcal{R}} \mathbb{E}_{\pi_E}[r(s, a)] - \mathbb{E}_{\pi}[r(s, a)] \quad (2.18)$$

The authors observe that the latter part of equation 2.14 can be interpreted as an Integral Probability Metric (IPM) between induced occupancy measures ρ_E and ρ_{π} :

$$\phi_{\mathcal{R}}(\rho_E, \rho_{\pi}) = \sup_{r \in \mathcal{R}} |\langle r, \rho_E \rangle - \langle r, \rho_{\pi} \rangle| = \sup_{r \in \mathcal{R}} \langle r, \rho_E \rangle - \langle r, \rho_{\pi} \rangle$$

This formulation leads to the following loss:

$$\arg \min_{\pi \in \Pi} -H(\pi) + W_1^d(\rho_{\pi}, \rho_E)$$

Where W_1^d is the 1-Wasserstein distance with respect to the ground cost function d . They then use a GAN model to optimise this loss taking advantage from the dual formulation of the Wasserstein distance.

2.4 Meta-Learning

In the introduction we emphasised the importance of generalisation and how to learn solving new tasks using knowledge from experience with a different task. In this section we will dive

Name	Dataset	Task
Single-task learning	$D_{\text{train}} \subset D_{\text{test}} \subset D$	$T = T_{\text{train}} = T_{\text{test}}$
Transfer learning	$D_1 \gg D_2$	$T_1 \neq T_2$
Multi-task learning	$D_{\text{train}} \subset D_{\text{test}} \subset D$	$T_1 \neq T_2$
Domain adaptation	$D_1 \neq D_2$	$T_1 = T_2$
Meta-learning	$D_1, \dots, D_{N-1} \gg D_N$	$T_1, \dots, T_{n-1} \neq T_N$

Table 2.2: Different learning setups.

deeper in Meta-Learning, offer a proper definition, expose its different approaches and detail its adaptation to RL and IRL.

Table 2.2 presents the different existing approaches that try to generalise the knowledge extracted from the training datasets. In single task learning, the training and test datasets are both drawn from the same distribution. In transfer learning, models are trained on a certain dataset then the knowledge gathered from these datasets is used to speed up training on a different task using a smaller dataset.

In Multi-task Learning, a unique model is trained in the same time on two tasks that have a link between them. This way, the learning process benefits from the simultaneous training of an auxiliary task. This is different from transfer learning where tasks are supposed to be learnt in a sequential manner. For instance, many actor-critic algorithms actually use two heads (policy and value). Performing those two tasks in the same task can be seen as multi-task learning.

In Domain adaptation, a different dataset is used to perform the same task. For instance given the knowledge a model can extract from a horse picture, can the same model recognise a horse from a text description ? Domain adaptation is necessary when there is a change in data distribution between the training and test datasets. It is called the domain shift. Domain shift is a usual issue in computer vision in tasks where objects have to be identified under different lighting setups or angles.

Finally, in Meta-Learning, both the datasets and the tasks are different, although keeping a certain similarity³. In this context, a series of datasets and trainings is generalised to learn a new "similar" task quickly. While transfer learning aims to transfer knowledge from a unique task, meta-learning aims to generalise knowledge from a series of previous trainings. One can say that meta-learning generalises transfer learning by learning initial parameters from several tasks instead of a unique one.

One of the objectives of Meta-Learning is to leverage knowledge from well known datasets to perform effective training on a new dataset (target dataset) that only contains a few samples. Essentially, the goal is to mimic a human that only needs to see an object once to be able to easily recognise it in other contexts. However, it is important to discuss the size of the target dataset:

- **Zero-shot Learning:** In this context, we know that a certain class exists but there is no sample available. This could seem impossible to learn something that does not exist; how to classify something we have never seen? We recall here the previous horse example.

³The term "similarity" is loosely defined and its definition remains proper to each particular approach

The horse description can be seen as task a , and given the description it might be easy to recognise a horse picture (task b). In this context, we have applied knowledge from previous tasks to solve a new one. In a sense, zero-shot learning is a kind of transfer learning.

- **K-shot learning:** In the one-shot or k-shot case, there exists at least one labelled sample of the target class. An example of a one shot task is facial recognition for security access. The goal is to add an employee picture to a database and recognise him every time with this unique sample. To do so, an easy solution is to use a pre-trained model that creates an embedding of all photos in the database. Each time the system evaluates the picture of an employee, it takes a photograph and compares its embedding with the existing ones.

Now that we discussed the zero and k-shot scenarios, we can discuss the different existing approaches of Meta-Learning. While certain methods aim to find a mapping between their available knowledge on a given domain for a certain task and the domain of a new task (metric based approaches); others consist in optimisation methods designed specifically for meta-learning.

2.4.1 Metric-based Meta-Learning

Using again the horse example, the word “horse” can be transformed into a vector h_a using a CNN to create an embedding h_b of the horse picture. A mapping function between the spaces of h_a and h_b is then used to transform h_b into h_a^* . Finally a similarity measure like cosine similarity can be used to find the closest vector from h_a^* and deduce that they belong to the same class. We will not detail further this approach as we used the same general idea in chapter 6, where we offer greater details.

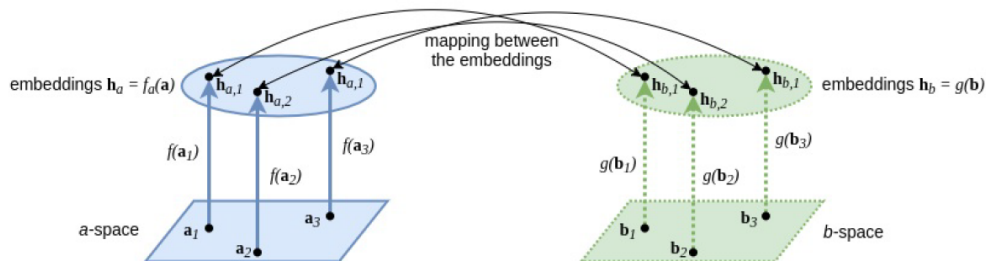


Figure 2.9: Given two spaces a and b , metric based approaches create mappings that maximise the similarity between embeddings $h_a^{(i)}$ and $h_b^{(j)}$ of instances in a and b . Source: https://www.youtube.com/watch?v=dYmJd_fJLW0&list=PLoROMvodv4rMIJ-TvblAIkw28Wxi27B36.

Formalised more classically, metric-based approaches try to determine $P_\theta(y|x)$ where y is a one hot encoding of the horse class and x the word embedding. In the zero-shot context, we add T , the horse image and we try to estimate $P_\theta(y|x, T)$.

2.4.2 Optimisation based Meta-learning

Let $\{T_i\}$ for $i \in [1, N]$ a set of base learning tasks. Each of these tasks T_i consists in a dataset D_i and an objective, the loss \mathcal{L}_i . We can therefore denote $T_i = \{D_i, \mathcal{L}_i\}$. Each dataset D_i consists in a

pair of inputs and labels $D_i = \{(x_i, y_j)\}$ and is split into D_i^{train} and D_i^{test} . For each training dataset, a model \hat{f}_{θ_i} is learnt using \mathcal{L}_i .

The aim of optimisation based meta-learning is to learn parameters ω of a certain model using a set of tasks $\{T_i\}$, such that, given a target task T , for which we only dispose of a few samples, ω will constitute ideal initial parameters so the learning of task T will be quick and effective; and the few number of available samples will be enough, as illustrated in figure 2.10. The training process can be seen as maximising the loss function sensitivity for the new tasks. If sensitivity is high, a small change in the parameter space can induce a great performance enhancement for the target task.

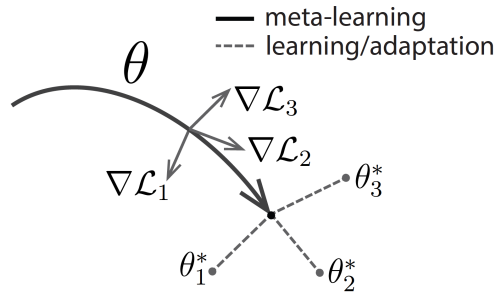


Figure 2.10: Optimisation based Meta-Learning. The model optimises for θ parameters that find a compromise between different tasks in order to optimise and adapt quickly to new tasks. Source: <https://bair.berkeley.edu/blog/2017/07/18/learning-to-learn/>.

To explain the process we will focus on the model used in [FAL17] and named Model Agnostic Meta-Learning (MAML).

MAML uses a two-level optimisation problem. In the inner loop, a base learner performs task specific updates to θ for the different observations in the training set. In the outer loop, the meta learner optimises the parameters ω on a series of base tasks where the loss of each task is evaluated on the test set of the base task D_i^{test} . To summarise, the inner loop optimises θ and the outer loop optimised ω :

$$\omega^* = \underbrace{\arg \min_{\omega} \mathcal{L}^{\text{meta}}}_{\text{outer loop}} \left(\underbrace{\arg \min_{\theta_i} \mathcal{L}^{\text{base}}(\theta_i, D_i^{\text{train}})}_{\text{inner loop}}, D_i^{\text{test}} \right)$$

The meta loss optimises a meta-objective that can be the accuracy, speed or any other goal on the set of base tasks and datasets. The result of the meta-optimisation is the set of optimal parameters ω^* . However, in reality there is no optimal parameters ω . ω actually corresponds to optimal parameters θ_0 as illustrated in figure 2.10. The formula can therefore be written:

$$\theta_0^* = \underbrace{\arg \min_{\theta_0} \mathcal{L}^{\text{meta}}}_{\text{outer loop}} \left(\underbrace{\arg \min_{\theta_i} \mathcal{L}^{\text{base}}(\theta_i, D_i^{\text{train}})}_{\text{inner loop}}, D_i^{\text{test}} \right)$$

An important notice is that the tasks used in the outer loop have to be different from those used in the inner loop. Therefore, we obtain a first series of θ_i parameters learnt on a first series of tasks, then the same θ_i are used to evaluate the loss on a different set of tasks. This process is considered as a second order gradient:

$$\theta \leftarrow \theta - \beta \nabla_{\theta} \mathcal{L}(\theta')$$

$$\theta \leftarrow \theta - \beta \nabla_{\theta} \mathcal{L}(\theta - \alpha \nabla_{\theta} \mathcal{L}(\theta))$$

We detail the process along with the outer and inner loop below.

1. Initialise a distribution over tasks $p(T)$ and model weights θ with random values

<p style="text-align: center;">Outer loop</p> <ol style="list-style-type: none"> 2. Sample batch of tasks from $p(T)$ i.e. $(T_1, T_2, \dots, T_i, \dots, T_n) \sim p(T)$
<p style="text-align: center;">Inner loop</p> <ol style="list-style-type: none"> (a) For each task T_i, sample k data points and train the model f_{θ} (b) Minimise loss using gradient descent- and find optimal parameters θ'_i
<ol style="list-style-type: none"> 3. For a new set of tasks, train the model $f_{\theta'_i}$ 4. Minimise loss and update model parameters θ

One of the limits of MAML is that successive backpropagations are computationally costly due to the second order gradient. Moreover, given the high number of backpropagations steps, MAML can suffer from vanishing or exploding gradients.

While proposing a brilliant solution involving Pearmutter method to handle the computational cost of second order derivatives, the authors also propose First Order MAML which simply ignores the term $\alpha \nabla_{\theta} \mathcal{L}(\theta)$

2.4.3 Optimisation-based Meta RL

Given the the two level optimisation framework used in MAML, the main question in the RL framework is how to optimise the inner loop. In a model free approach, we already defined the two main approaches that are either policy based (policy gradients) or value based (Q-learning):

- **Policy gradients:** these methods tend to be on-policy (the policy used to take actions is the same used to explore) and therefore are data inefficient. Moreover, they do not allow to keep much information, especially in a sparse reward context. Indeed, the policy gradient objective is:

$$[\sum_t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t)] [\sum r_t]$$

Supposing that the agent ends an episode without reaching its goal, the received reward can be 0, and the whole objective is null, which means that nothing can be learnt from that episode.

- **Q-learning:** The Q-learning update is:

$$\hat{Q}(s, a) - (r + \gamma \max_{a'} Q(s', a'))$$

We notice that the update compares two states with each others and does not give any information on the global task itself. A single gradient update will not be informative about the global task but will only contain local information at a single time step level. Several time steps will be required to effectively backpropagate the information. It is therefore hard to use Q-learning in the inner loop. However, Q-learning is off-policy and therefore data efficient.

In [FAL17], the authors propose to adapt MAML to the RL context. They chose to use policy gradients both in the inner and the outer loop. The process is as follows:

1. Sample task T_i
2. Collect D_i^{train} by generating trajectories using π_θ .
3. Compute adapted parameters θ'_i using the policy gradients objective for task T_i
4. Collect D_i^{test} by generating trajectories using $\pi_{\theta'_i}$.
5. Update θ using the trajectories D_i^{test} generated on all tasks T_i

2.4.4 Meta-IRL

MAML Adaptation to Meta-IRL

In [Xu+19], the authors adapt MAML to the IRL setting. The idea is the same; the model learns common weights for several tasks reward function, and from there the weights are adjusted using a few demonstrations on the target task. These common weights are the prior to learn. This approach is effective as in IRL the space of good reward functions for a given task is a lot smaller than of all possible rewards defined on raw observations.

During meta-training, there are several tasks $\{T_i\}$, for which we dispose of expert demonstrations $D_T = \{\tau_1, \dots, \tau_k\}$ from an expert policy. After meta-training, the algorithm has to solve a new task and learn the parameters of its reward function $r_\phi(s_t, a_t)$ from a small number of demonstrations.

Let θ be the common weights to learn before target task adjustment, the authors first define a loss $\mathcal{L}_T(\theta)$ on the reward function r_θ . To do so, they use the MaxEntIRL loss:

$$\mathcal{L}_T(\theta) = [\mathbb{E}_\tau[\mu_\tau] - \mu_{\mathcal{D}_\tau}]$$

with μ_τ the state visitations under r_θ and μ_D the mean state visitations under demonstrated trajectories.

Once the θ parameters are fixed, we still have to find the new parameters ϕ_τ for the new task:

$$\phi_\tau = \theta - \alpha \nabla_\theta \mathcal{L}_T^{\text{train}}(\theta)$$

The aim is to find θ such that $\mathcal{L}_T^{\text{test}}(\phi_T)$ is quickly minimised.

$$\min_\theta \sum_{i=1}^N \mathcal{L}_{\mathcal{T}_i}^{\text{test}}(\phi_{\mathcal{T}_i}) = \sum_{i=1}^N \mathcal{L}_{\mathcal{T}_i}^{\text{test}}(\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}^{\text{tr}}(\theta))$$

This approach bears several limitations. Indeed, it is based on MaxEntIRL and therefore it requires to know the dynamics of all used environments. Moreover, $\mathcal{L}_T^{\text{test}}(\phi_{T_i})$ is also a difference between state visitations. It is evaluated using ϕ instead of θ and a new set of demonstrations. Therefore the whole MaxentIRL process has to be done again at this step. Finally, this approach is on-policy, which is highly non efficient.

Meta-IRL with Probabilistic Context Variables

In order to improve on [Xu+19] and not having to know the environments dynamics, the authors of [Yu+19] will use AIRL [FLL18]. AIRL uses a GAN approach: a discriminator D_θ (binary classifier) parameterised by θ and an adaptive sampler π_ω (a policy). The discriminator takes a specific form:

$$D_\theta(s, a) = \frac{\exp(f_\theta(s, a))}{(\exp(f_\theta(s, a)) + \pi_\omega(a|s))}$$

where $f_\theta(s, a)$ is a learnt reward function. The discriminator is trained to classify trajectories sampled by the expert or by π_ω . On the other hand, π_ω is trained to maximise:

$$\mathbb{E}_{\rho_{\pi_\omega}} [\log D_\theta(s, a) - \log(1 - D_\theta(s, a))]$$

In the meta-IRL context, the aim is to learn an inference model $q(m|\tau)$ and a reward function $f(s, a, m)$ where m is a context variable defining the task at hand. Let $p(m)$ the prior distribution of the context variable. First, the authors parameterise the inference model of the context variable $q_\phi(m|\tau)$ and the reward function $f_\theta(s, a, m)$ (where m is obtained using by q_ϕ). The distribution of the induced trajectories is therefore:

$$p_\theta(\tau|m) = \frac{1}{Z(\theta)} \left[\zeta(s_1) \prod_{t=1}^T P(s_{t+1}|s_t, a_t) \right] \exp \left(\sum_{t=1}^T f_\theta(s_t, a_t, m) \right)$$

We could apply AIRL directly, by conditioning all the terms in the discriminator by m , but in practice the discriminator can easily ignore the conditioning. It is therefore mandatory to enforce a relation between m and the sampled trajectories. To do so, the authors use mutual information. The mutual information between two random variables m and τ with $p_\theta(m, \tau) = p(m)p_\theta(\tau|m)$ is:

$$I_{p_\theta}(m; \tau) = \mathbb{E}_{m \sim p(m), \tau \sim p_\theta(\tau|m)} [\log p_\theta(m|\tau) - \log p(m)]$$

However, we cannot have access to $p(m)$ nor to $p_\theta(m|\tau)$. They use $q_\phi(m|\tau)$ as a variational approximation. Formally, let p_{π_E} the distribution of expert trajectories, we would like to satisfy the two following objectives:

$$\min_{\theta} \mathbb{E}_{p(m)} [D_{KL}(p_{\pi_E}(\tau|m) || p_\theta(\tau|m))] \\ \min_{\phi} \mathbb{E}_{p_\theta(\tau)} [D_{KL}(p_\theta(m|\tau) || q_\phi(m|\tau))]$$

The first objective forces the distribution of trajectories induced by θ to match the empirical distribution defined by expert demonstrations. The second one aims at making $q_\phi(m|\tau)$ a better approximation of $p_\theta(m|\tau)$. We can consider that the mutual information is the main objective while the two others can be seen as constraints. Using Lagrange multipliers, the authors get the following objective:

$$\max_{\theta, \phi} -\mathbb{E}_{p(m)} [D_{KL}(p_{\pi_E}(\tau|m) || p_\theta(\tau|m))] + \mathbb{E}_{m \sim p(m), \tau \sim p_\theta(\tau|m)} [\log q_\phi(m|\tau)]$$

To estimate the first term, it is possible to use the AIRL discriminator loss, and the final loss function is:

$$\min_{\omega} \max_{\theta, \phi} \mathbb{E}_{\tau_E \sim p_{\pi_E}, m \sim q_\phi(m|\tau_E), (s,a) \sim p_{\pi_\omega}(s,a|m)} \log(1 - D_\theta(s, a, m)) \\ + \mathbb{E}_{\tau_E \sim p_{\pi_E}, m \sim q_\phi(m|\tau_E)} \log(D_\theta(s, a, m)) + \mathbb{E}_{m \sim p(m), \tau \sim p_\theta(\tau|m)} [\log q_\phi(m|\tau)] \quad (2.19)$$

2.5 Generative Models

Generative models have gained immense prominence in the field of machine learning, finding applications across diverse domains, including their significant impact on the Inverse Reinforcement Learning (IRL) paradigm. This section aims to provide an in-depth exploration of generative models, elucidating their nature and delving into various types. These encompass a range of approaches, spanning from different variants of adversarial models to invertible models and Normalizing flows.

Generative models fundamentally aim to capture the underlying distribution of a dataset, enabling them to generate new instances that resemble the original data. These models prove invaluable for tasks such as image synthesis, data augmentation, and even IRL, where understanding the underlying reward distribution is crucial.

Among the plethora of generative models, adversarial models stand out as a significant category. Adversarial models, such as Generative Adversarial Networks (GANs), operate on a competitive basis between a generator and a discriminator. The generator fabricates data instances to deceive the discriminator, while the discriminator aims to distinguish between genuine and fabricated instances. Through iterative interactions, GANs achieve a balance that results in the generator producing increasingly authentic data. In contrast, invertible models focus on capturing a one-to-one mapping between input and output spaces. Normalizing flows represent a particularly intriguing subset of invertible models. These models focus on learning a sequence of invertible

transformations, or flows, that map a simple initial distribution to a more complex target distribution. This enables the generation of samples that conform to the desired distribution. The application of Normalizing flows to IRL, as previously discussed, demonstrates their potential to learn intricate reward distributions and facilitate effective policy learning.

In summary, this section will delve deeper into the realm of generative models, providing a comprehensive understanding of their diverse types and their relevance in the context of Inverse Reinforcement Learning. From adversarial models fostering competition to invertible models ensuring information preservation, these models hold immense promise in generating complex data distributions and enhancing various machine learning tasks, including IRL.

2.5.1 Different Losses for Different Goals

Unconditional density modelling is centered around the task of learning the underlying distribution, denoted as p^* , from a given dataset $D = x_1, \dots, x_M$. This dataset consists of samples considered as realizations of a random variable X defined over a space \mathcal{X} . The density of X at a point x is denoted as $p^*(X = x)$. In the parametric approach to density estimation, a parametric family of density functions, represented as $p_\theta = p_\theta | \theta \in \Theta$, is chosen. Here, θ represents the admissible parameters associated with the chosen parametric densities. The learning process aims to determine the optimal parameter θ^* within the set Θ , necessitating a performance measure for evaluating θ .

Given that the primary objective is to approximate p^* , the chosen performance measure, i.e. loss function, is typically a distance metric between the parametric density p_θ and the true distribution p^* . This distance metric, denoted as $\mathcal{L}(p_\theta, p^*)$, serves as a representation of the dissimilarity between the estimated and true distributions. This loss function denoted $\mathcal{L}(p_\theta, p^*)$ guides the optimization process towards identifying the parameter θ^* that best captures the characteristics of the underlying distribution p^* .

There are different approaches to designing the loss \mathcal{L} . We will focus on unconditional density modelling, meaning that the estimator $\hat{\theta}$ for θ^* is obtained from unlabelled data. Intuitively, there are two things we expect from a "good" model:

- The model p_θ assigns high density to samples taken from the true distribution p^* :

$$x \sim p^*(x) \implies p_\theta(x) \text{ is "high"}$$

- Samples taken from the model p_θ behave similarly to real samples from p^* :

$$x \sim p_\theta(x) \implies p^*(x) \text{ is "high"}$$

The specific choice of $\mathcal{L}(p_\theta, p^*)$ depends on whether the focus is on one of the two distinct objectives, leading to different characteristics for the model p_θ . This distinction serves as a rough categorisation for various existing models in the field. The first objective, known as "coverage driven," emphasises capturing the entire distribution p^* effectively. This approach is more straightforward

to handle, requiring only samples drawn from p^* . Many learning algorithms, including the well-known maximum likelihood estimation (MLE), align with this objective. In these methods, the primary aim is to ensure that the estimated distribution p_θ closely matches the observed data distribution. On the other hand, the second objective, termed “quality driven”, prioritises generating high-quality samples from the estimated distribution p_θ . This objective is more challenging to design, as it ideally requires access to the true distribution p^* . A prominent example of this approach is the Generative Adversarial Network (GAN) [Goo+14], where the generator strives to produce samples that are indistinguishable from those drawn from p^* . In quality-driven models, the emphasis is on producing samples that are coherent, diverse, and of high perceptual quality, rather than replicating the entire distribution p^* .

In addition to determining the training objectives and procedures, the choice of a suitable family of parametric densities is crucial in generative modeling. Given that data typically resides in high-dimensional spaces and exhibits complex, non-linear relationships, modeling such data can be inherently challenging. Consequently, deep learning approaches have demonstrated remarkable success in this field. p_θ is implemented by a very flexible, over-parameterised and non-convex function approximator, and $\hat{\theta}$ is selected by performing gradient descent on the loss.

2.5.2 Generative Latent Variables

When dealing with highly complex data, such as natural images, generative models must be capable of capturing intricate and non-linear patterns in the data. To achieve this, the concept of learning non-linear manifolds becomes important. The underlying idea is to transform simple distributions in a latent space into complex distributions in the data space, thus enabling the model to closely match the data distribution. This process involves considering a latent variable z and a simple distribution $p(z)$ over this latent variable, often chosen as a standard Gaussian distribution. The transformation from the latent space to the data space is accomplished using a non-linear function denoted as $f_\theta(z)$, where θ represents the parameters of the function. This function is usually realised as a deep neural network, given its capacity to capture complex and non-linear relationships. The deep neural network serves to map the latent variables to the data space, generating data samples x through the function $f_\theta(z)$. The result is a complex marginal distribution $p_\theta(x)$, which represents the distribution of generated data samples x after integrating out the latent variables z .

One remarkable aspect of this approach is that the non-linear manifold $f_\theta(z)$ is highly flexible and has the potential to approximate any function, making it capable of capturing intricate patterns and relationships in the data. With the availability of sufficient training data or effective regularization techniques to prevent overfitting, it becomes possible to utilize deep neural networks with a high degree of expressiveness, enabling the model to closely fit the data distribution and generate realistic samples.

Employing a non-linear function f_θ introduces a challenging problem: the computation of $p_\theta(x)$ becomes intractable. This is due to the fact that the integral involving the non-linear deep neural network $f_\theta(\cdot)$ can no longer be evaluated in a closed-form manner. Several approaches have been developed to address this challenge, each leading to different classes of generative models. Here are some of the solutions:

- **Variational Auto-Encoders (VAE)** [KW14]: In VAEs, a tractable lower bound on the likelihood is used to circumvent the intractable integral. VAEs combine an encoder that maps data samples to a latent space with a decoder that generates data samples from the latent space. The training process involves maximizing a lower bound on the log-likelihood, known as the evidence lower bound (ELBO), which can be optimized efficiently using gradient-based methods.
- **Flow-Based Methods**: Another approach is to choose a constrained parametric family for the transformation function f_θ so that the computation of $p_\theta(x)$ becomes feasible. Flow-based models utilise invertible transformations that can be efficiently evaluated both forwards and backwards. These transformations are composed to model complex data distributions through a series of simple, invertible transformations.
- **Generative Adversarial Networks (GAN)**: GANs take a different route by avoiding direct computation of the density $p_\theta(x)$. Instead, they use a discriminator network that learns to distinguish between real and generated data samples. The generator network then aims to generate data that is indistinguishable from real data according to the discriminator. GANs are trained in a two-player adversarial setup, where the generator and discriminator networks compete to improve their respective performance.

2.5.3 Generative Adversarial Networks

In the context of Generative Adversarial Networks (GANs), introduced by Goodfellow et al. [2014], the training process involves the use of a quality metric to guide the learning of the model. This metric takes the form of a classifier, denoted as D_ϕ , to assess the quality of generated samples. The GAN framework involves two main components: a generator network G_θ and a discriminator network D_ϕ .

- **Generator**: The generator network takes a latent variable vector z sampled from a prior distribution $p(z)$ and maps it to the data space, resulting in the generation of an image $x = G_\theta(z)$. This process implicitly defines a density $p_\theta(x)$ over the data space.
- **Discriminator**: The discriminator network evaluates the quality of the generated images. It provides an estimate $D_\phi(x)$, where x is an image, indicating the probability that x is real (as opposed to being generated). If the discriminator is well-trained, it can accurately distinguish between real and fake images.

The training of GANs is formulated as a two-player adversarial game. The generator aims to produce images that are indistinguishable from real ones, while the discriminator aims to distinguish between real and generated images. This adversarial interaction results in a competition between the two networks, which leads to the refinement of both the generator and discriminator over the course of training. The discriminator's loss, denoted as J^D , quantifies how well it is performing in distinguishing real and generated images. The generator's loss, denoted as J^G , is based on the discriminator's evaluation of the generated samples. The generator's objective is to minimise this loss, effectively improving the quality of the generated samples.

In summary, GANs leverage the use of a discriminator network to evaluate the quality of generated samples, which in turn guides the learning of the generator network. The adversarial

process drives the generator to produce samples that are increasingly difficult for the discriminator to distinguish from real data, ultimately leading to the generation of realistic and high-quality samples.

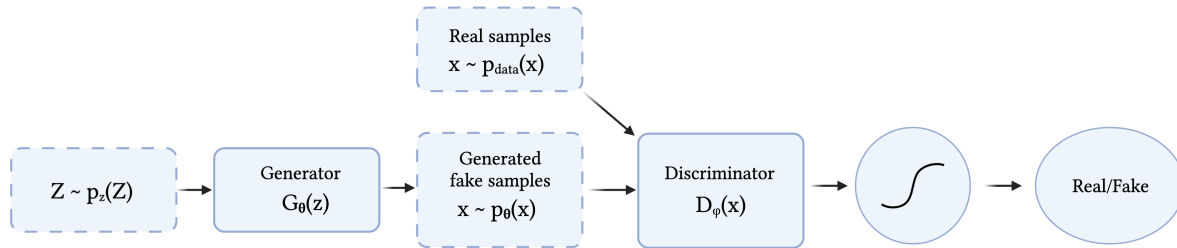


Figure 2.11: Illustrative view of the GAN architecture. A latent variable z is first sampled from an user defined simple distribution. The generator transforms z into $x \sim p_\theta$. The discriminator is a standard classifier that has to separate real samples, drawn from the dataset, from those that are created by the generator.

A GAN can be seen as a zero sum sequential game between two players. By zero-sum, we mean that the earnings or losses of one of the players are compensated by the earnings/losses of the other. Therefore the sum of J^G and J^D is 0.

The induced situation is called Minimax as the two networks are in constant competition. The discriminator is trained to maximise classification accuracy for a given generator, i.e. ϕ is optimised to reach $\phi^* = \max_\phi V(\phi, \theta)$; where V is a loss function to be detailed later. Simultaneously, the generator is trained to degrade the classification of a given discriminator, i.e. θ is optimised to reach $\theta^* = \min_\theta V(\phi, \theta)$. Solving this adversarial problem corresponds to finding ϕ^* and θ^* such that:

$$V(\phi^*, \theta^*) = \min_{\theta} \max_{\phi} V(\phi, \theta)$$

The solution to the minimax problem is called Nash equilibrium. Nash equilibrium is reached when one of the players does not change its action whatever the opponent response. In a GAN context, equilibrium is reached when the discriminator cannot distinguish anymore fake samples from true ones, and will be right half of the time.

The discriminator is a classifier that can be trained as classic classifiers. However, the training dataset is composed of true data and generator outputs. Therefore the discriminator can be trained using the cross-entropy loss function:

$$V_\theta(\phi) = \int_x [p^*(x) \ln D_\phi(x) + p_\theta(x) \ln(1 - D_\phi(x))] dx$$

The two components of the loss reflect the two possible class (true or false) that are equally balanced in the training dataset. The left hand side term is active when the input comes from the

true distribution. Ideally, $D_\phi(x) = 1$ when $x \sim p^*$. The right hand side term of the loss function is active when the input $x \sim p_\theta$. Another more explicit formulation of the discriminator loss is:

$$V_\theta(\phi) = -\frac{1}{2}\mathbb{E}_{x \sim p^*} \ln(D(x)) - \frac{1}{2}\mathbb{E}_{z \sim p_z(z)} \ln(1 - D(G(z)))$$

For any $(a, b) \in \mathbb{R}^2 \setminus \{0, 0\}$ the function $y \mapsto a \ln(y) + b \ln(1 - y)$ achieves its maximum in $[0, 1]$ at $y = a/(a + b)$ so for a fixed θ , the optimal discriminator $D_{\phi^*(\theta)}$ is the Bayes classifier:

$$D_{\phi^*(\theta)} = \frac{p^*(x)}{p^*(x) + p_\theta(x)}$$

Now assuming that D_ϕ is trained to optimality for a given θ , $D_{\phi^*(\theta)}$ can be plugged in $V(\theta, \phi)$:

$$V(\phi^*(\theta), \theta) + \ln 4 = D_{KL}\left(p^* \parallel \frac{p^* + p_\theta}{2}\right) + D_{KL}\left(p_\theta \parallel \frac{p^* + p_\theta}{2}\right) \propto D_{JS}(p^* \parallel p_\theta) \quad (2.20)$$

Assume the regime of infinite data, infinite model capacity, and under the assumption that the optimal discriminator is reached at each iteration of the generator. In that context, equation 2.20 shows, by convexity of D_{KL} with respect to p_θ , that there is a unique global optimum for G_θ , at the data distribution $p_\theta = p^*$, which can be recovered by gradient descent.

As shown in equation 2.20, the gradient descent allows to the minimum of the loss which is the Jensen Shannon divergence rather than the Nash equilibrium. This leads to oscillations between solutions. Moreover, the discriminator tends to be too powerful compared to the generator. As said above, the GAN approach is not a coverage approach, therefore while producing convincing samples, the generator fails to capture the full support of the training data, a phenomenon known as mode collapse.

2.5.4 Wasserstein GAN (WGAN)

Equation 2.20 showed the relation between the GAN loss and the KL divergence. Given two continuous distributions P and Q , the KL divergence is:

$$KL(P \parallel Q) = \int_x P(x) \log \frac{P(x)}{Q(x)} dx$$

However, if $Q(x) = 0$ at some point x where $P(x) > 0$, the KL divergence explodes. Therefore, if p_θ has a small support, then it is highly unlikely that p^* belong in this restricted support. Therefore, to train the generator, a better measure distance between p_θ and p^* has to be defined. Indeed, different metrics induce different sets of convergence. For generative models, $d(p_\theta, p^*)$ can be treated as a loss function. The choice of the distance function d is thus crucial to learn a good approximation of p^* . Instead of the KL divergence, WGAN [ACB17] is based on the minimisation of the Wasserstein distance between p_θ and p^* .

Wasserstein Distance

The optimal transport cost, or Wasserstein distance is a way to measure distance between two distributions and gives a smaller topology than many other criteria, like the set of f-divergences associated to GANs. This is particularly true in settings where data are located in low dimensional manifolds on the space \mathcal{X} . Stronger notions of distance like f-divergence will often saturate and fail at providing useful gradients during training. On the other hand the Wasserstein distance is well behaved and thus stabilises training.

To measure the matching between two distributions P and Q , one can use the class of f-divergences defined as

$$D_f(P||Q) = \int Q(x) f\left(\frac{P(x)}{Q(x)}\right) dx$$

where $f : (0, \infty) \rightarrow \mathbb{R}$ is any convex function satisfying $f(1) = 0$. For any such f , $D_f \geq 0$ and $D_f = 0$ if $P = Q$. Kulback Leibler and Jensen-Shanon divergences are part of the f-divergence class. Another class of divergence is induced by the optimal transport problem which one of the formulations called Kantorovitch or primal formulation is:

$$W_c(P, Q) = \inf_{\gamma \in P(X \sim P, Y \sim Q)} \mathbb{E}_{(X, Y) \sim \gamma} [c(X, Y)] \quad (2.21)$$

where $c(X, Y) : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_+$ is any cost measurable function and $(X \sim P, Y \sim Q)$ is a set of joint distributions of (X, Y) with marginals P and Q .

The Wasserstein distance has the huge advantage of taking into account the geometry of the underlying supports of P and Q . Indeed, considering the two non overlapping distributions in figure 2.12:

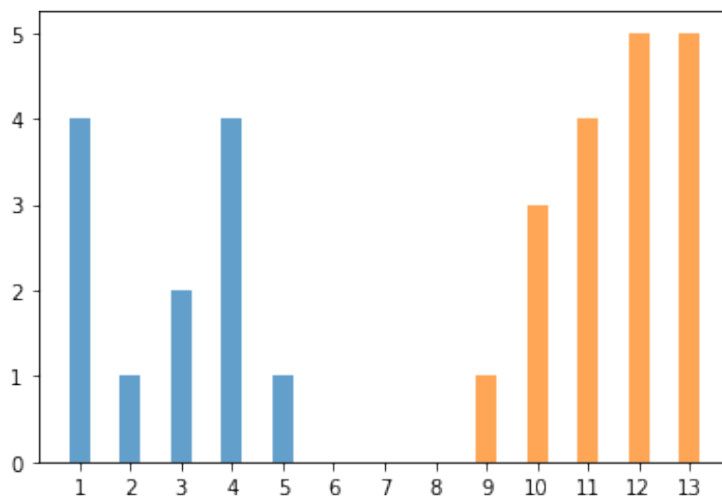


Figure 2.12: Two distributions with disjoint support.

As the two distribution supports do not overlap, any divergence from the class of f-divergence will diverge. Even treating the histograms as high dimensional points and using the L2 distance is not informative. Indeed, the L2 distance will be strictly positive but not informative enough.

For instance, the length of vector $(1, 0, 1)$ is $\sqrt{2}$, which is also the length of vectors $(1, 0, 0, 1)$ and $(1, 0, 0, 0, 1)$. Therefore the L2 norm does not give any information on the gap between the support of non overlapping distributions. On the other hand, the Wasserstein distance takes that gap into account.

We will first try to develop an intuition about the Wasserstein distance. A distribution is defined by the mass it assigns to each point of its support. Consider that we would like to move every non null mass on P in order to modify it into Q . Moving a mass m for a distance d costs $m \times d$.

Suppose that we want to calculate the Wasserstein distance between the distributions f and h displayed in figure 2.13. Contrary to classically used divergences, the Wasserstein distance takes into account horizontal translations weighting the transported mass by its distance.

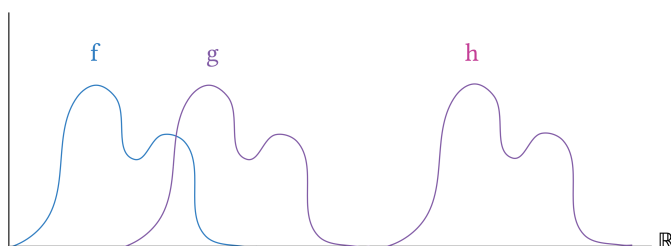


Figure 2.13: f , g and h are the same distributions but with shifted support. The KL divergence between them is 0 while the Wasserstein distance takes into account the geometry of the space they reside on. Therefore $0 < W(f, g) < W(f, h)$.

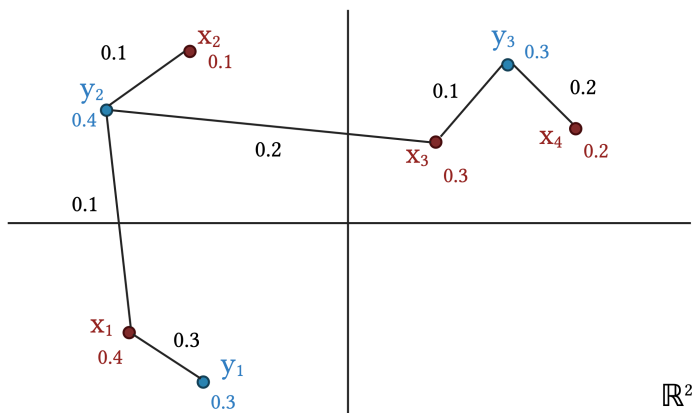


Figure 2.14: Example of a transport plan.

In figure 2.14, to each point is assigned a mass and the goal is to move the blue masses towards the red ones. Table 2.3 presents an example of strategy. We notice that the sum of columns and rows correspond to values outside the matrix. The formula for the Wasserstein distance is therefore:

$$\min \left\{ \sum_{i,j} a_{i,j} d(x_i, y_j) : a_{i,j} \geq 0, \sum_i a_{i,j} = y_j, \sum_j a_{i,j} = x_i \right\}$$

		x_1	x_2	x_3	x_4
		0.4	0.1	0.3	0.2
y_1	0.3	0.3	0	0	0
y_2	0.4	0.1	0.1	0.2	0
y_3	0.3	0	0	0.1	0.2

Table 2.3: Example of a transport plan.

where $a_{i,j}$ are the matrix entries. The conditions ensure that the coefficients are always positive, i.e. not transporting any negative quantity. As the sum of lines or columns give back the original values, the $a_{i,j}$ correspond to marginals of the distributions. To summarise, the Wasserstein distance between two distributions is the minimum over all transport plan of the cost; a transport plan being a joint distribution which marginals are given by $a_{i,j}$.

This problem is obviously hard to solve but there exist a simple way to calculate the Wasserstein distance in practice using quantile functions, i.e the inverse function of a CDF. Given two distributions P and Q which CDFs are F and G respectively, we have:

$$W_p(P, Q) = \left(\int_0^1 |F^{-1}(x) - G^{-1}(x)|^p dx \right)^{1/p}$$

In WGAN, the authors use the Wasserstein distance instead of the JS divergence to train a GAN. Indeed, as explained earlier, the classic GAN approach being quality driven, the loss function oscillates but its value is not meaningful as it does not continuously decrease. Taking as example the two distributions displayed in figure 2.15, the aim is to develop a model that learns to move p_θ towards 0 such that the closer θ is to 0, the more $d(p_0, p_\theta)$ decreases.

This is not possible using several usual distances or divergences:

- **Total Variation:** For any $\theta \neq 0$, let $A = \{(0, y) : y \in [0, 1]\}$. Therefore:

$$\delta(P_0, P_\theta) = \begin{cases} 1 & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$$

- **KL divergence:**

$$KL(P_0||P_\theta) = KL(P_\theta||P_0) = \begin{cases} +\infty & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$$

- **JS divergence:** Considering the mixture distributions $M = P_0/2 + P_\theta/2$ and calculating the first term of the JS divergence:

$$KL(P_0||M) = \int_{(x,y)} P_0(x, y) \log \frac{P_0(x, y)}{M(x, y)} dy dx$$

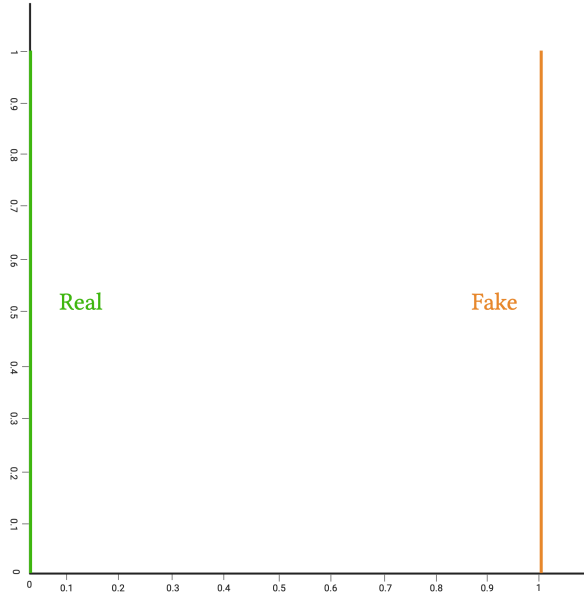


Figure 2.15: Real and fake distributions when $\theta = 1$.

For any x, y where $p_0(x, y) \neq 0$, $M(x, y) = \frac{1}{2}P_0(x, y)$. Thus $KL(p_0||M) = KL(p_\theta||M) = \log 2$. Therefore:

$$JS(P_0, P_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$$

- **Wasserstein distance:** As the two distributions are just a translation of the same distribution, the best strategy is to move the mass from $(0, y)$ to (θ, y) . Therefore $W(p_0, p_\theta) = |\theta|$

These examples show that there exist cases for the JS, KL and TV divergences where the gradient is always null, which is not the case of the Wasserstein distance. While this example is extreme, according to the authors of WGAN, when the supports are manifolds of small dimension in a space of bigger dimension, it is often the case that the intersection has null measure, which is dramatic for the classic GAN approach approximating the JS divergence.

This argument is strengthened by the following theorem (proof can be found in [ACB17]): *let p^* be a target distribution. Let z be a random variable. Let g_θ be a deterministic function such that $p_\theta = g_\theta(z)$:*

1. *if g is continuous regarding θ , the same thing goes for $W(p^*, p_\theta)$*
2. *if g behaves sufficiently well⁴, then $W(p^*, p_\theta)$ is continuous everywhere and differentiable almost everywhere*
3. *The two previous properties are false for the class of f -divergences*

⁴we will not detail what "behaves sufficiently well" means exactly but it is to note that this is always the case for neural networks with standard non linearities

Therefore, only the Wasserstein distance gives continuity and differentiability guarantees that are essential for a loss function. Moreover, the Wasserstein distance is also the weakest distance of the group. This means that any distribution converging under the KL, TV or JS divergences, also converges under the Wasserstein distance.

Computing the Wasserstein Distance

Unfortunately, computing the primal formulation of the Wasserstein distance as stated in equation 2.21 is intractable. Moreover, in a GAN setting there is no access to the quantile functions of the target or predicted distributions. However, using Kantorovitch-Rubinstein duality, the authors show that there exists a dual formulation of the Wasserstein loss:

$$W(p^*, p_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim p^*} [f(x)] - \mathbb{E}_{x \sim p_\theta} [f(x)]$$

where the supremum is taken over all 1-Lipschitz functions. This supremum is still intractable but easier to approximate. Let $\{f_w\}_{w \in \mathcal{W}}$, where w are the weights and \mathcal{W} is the set of all possible weights, be the set of K -lipschitz functions. Then:

$$\begin{aligned} \max_{w \in \mathcal{W}} \mathbb{E}_{x \sim P_r} [f_w(x)] - \mathbb{E}_{x \sim P_\theta} [f_w(x)] &\leq \sup_{\|f\|_L \leq K} \mathbb{E}_{x \sim P_r} [f(x)] - \mathbb{E}_{x \sim P_\theta} [f(x)] \\ &= K \cdot W(P_r, P_\theta) \end{aligned} \quad (2.22)$$

In a GAN setting, we would like to train $p_\theta = g_\theta(z)$ to match p^* . Intuitively, given a fixed g_θ , it should be possible to approximate the optimal f_w for the Wasserstein distance.

$$\begin{aligned} \nabla_\theta W(P_r, P_\theta) &= \nabla_\theta (\mathbb{E}_{x \sim P_r} [f_w(x)] - \mathbb{E}_{z \sim Z} [f_w(g_\theta(z))]) \\ &= -\mathbb{E}_{z \sim Z} [\nabla_\theta f_w(g_\theta(z))] \end{aligned} \quad (2.23)$$

The training process can therefore be split into 3 steps:

1. for a fixed θ , calculate an approximation of $W(p^*, p_\theta)$ by training f_w until convergence
2. once optimal f_w is fixed, calculate the gradient on θ
3. update θ and repeat

This only works if $\{f_w\}_{w \in \mathcal{W}}$ is K -lipschitz. To guarantee that property, the authors simply use weight clipping. The weights w are constrained to stay within $[-c, c]$.

Compared to a classic GAN, while the GAN discriminator outputs a probability of the sample being fake or not, the output of the discriminator of a WGAN directly outputs a score reflecting the Wasserstein distance. That is why the authors call f_w a critic. However f_w has to be trained until convergence at each step before updating the generator to have a good approximation of $W(p^*, p_\theta)$. According to the authors, Wasserstein GAN considerably reduces the mode collapse issue of classic GAN.

2.5.5 Normalizing Flows

As explained in section 2.5.2, the evaluation of the marginal distribution $p_\theta(x) = \int_z p(z)p(x|f_\theta(z))$ becomes intractable when using neural networks with non linearities f_θ . While VAE propose to approximate a lower bound and GANs only implicitly learn p_θ to be able to sample from it without offering any solution to estimate $p_\theta(x)$ for any x , flow models are generative models that allow sampling for the learnt distribution (with impressive results [KD18]) as well as density estimation. Moreover they are easily trained using MLE.

The principle of Normalizing Flows (NF) is to learn a transformation (flow) from a simple distribution to another more complex one. To obtain a valid distribution, it is necessary that the probabilities sum to one, and the transformation has to keep that property, hence the name Normalizing Flows. The aim is to be able to learn a complex distribution without having to choose in advance the right family of distribution or the right mixture. Once the target distribution is learnt, it is possible to sample from it (generative model) and to estimate the density or likelihood of each sample (density model).

Figure 2.16 illustrates the principle of NF. We take as input samples drawn from a simple distribution (a gaussian for instance) and transform them into data that looks like is drawn from a more complex distribution. This is done using a transforming function T and its inverse T^{-1} .

$$x = T(u); \quad u \sim p_u(u)$$

Where p_u is the base distribution. The aim of NF is therefore not to learn the target distribution itself but the transformation T and its inverse. The main property of these models is that the transformation T has to be invertible and that T and T^{-1} have to be differentiable. These transformations are called diffeomorphisms and require that u is in the same dimension as the target distribution. This way, the density of x is well defined and can be obtained using the change of variable formula.

In this section, we will build more intuition about the change of variable. Consider the transformation $x = T(u) = u^2$ and let u be uniformly distributed between 0 and 2. Therefore $T^{-1}(x) = \sqrt{x}$. Figure 2.17 shows that the transformation T has the capacity to compress samples in regions where the function is flat and to stretch space as the slope increases. It can be seen as a space distortion of \mathbb{R}^D to transform $p_u(u)$ into $p_x(x)$; where D is the dimension of the target distribution.

In figure 2.18, the below panel illustrates a uniform distribution in $[0, 2]$ that is transformed using the square function as shown in the panel above. As the transformation has to preserve probabilities on all support and on any smaller interval, this means that the shaded areas in the figure should match.

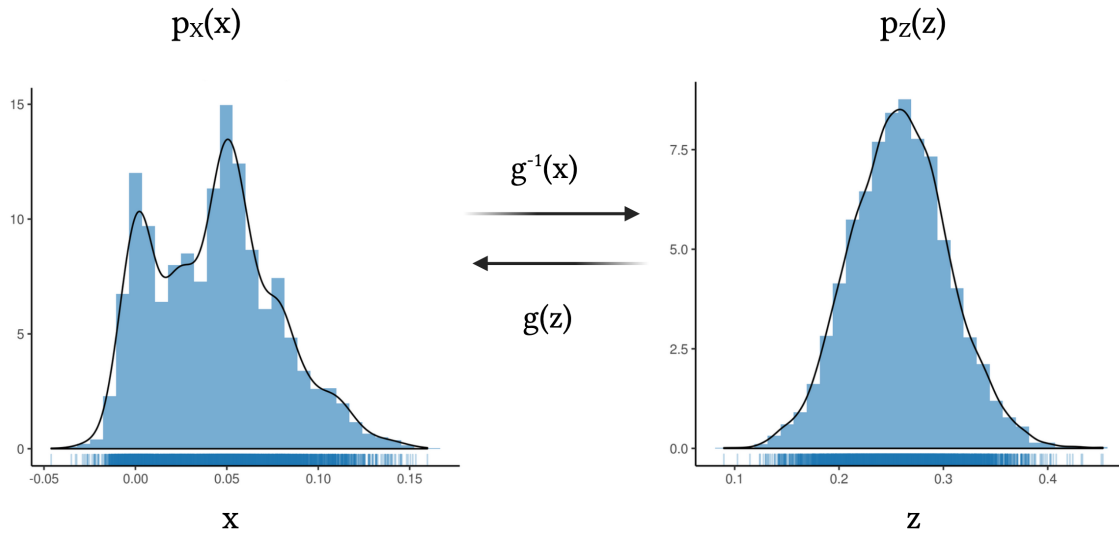


Figure 2.16: Normalizing Flows principle. The complex distribution $p_x(x)$ is transformed to an easy standard centered Gaussian. The transformation function $x = g(z)$ transfers between the Gaussian in z and the complicated function in x .

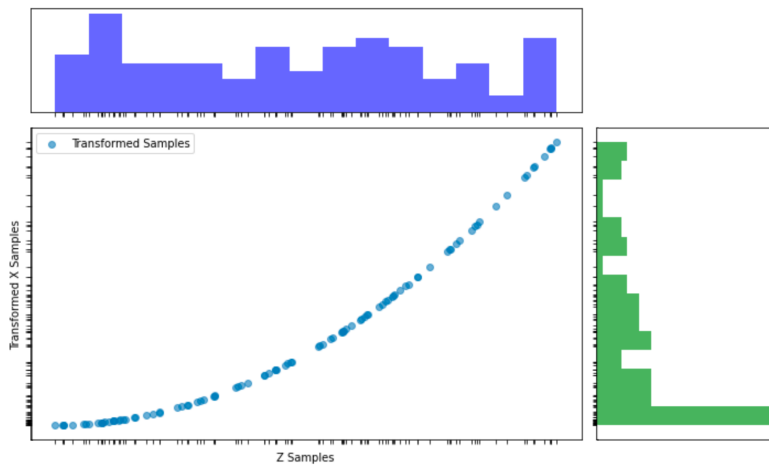


Figure 2.17: A square transformation applied to z samples drawn from a uniform distribution yields transformed x samples. The histograms above and on the right show the distribution of z and x samples respectively.

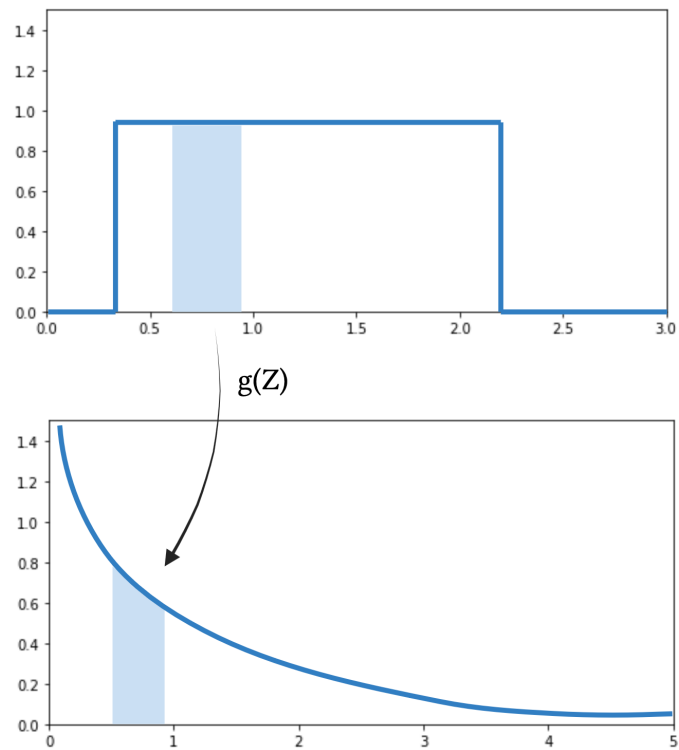


Figure 2.18: Uniform distribution is transformed using a square function. The shaded areas have to be equal to preserve probabilities.

Therefore:

$$p_u(u) \cdot |du| = p_x(x) |dx| \quad (2.24)$$

Equation 2.24 ensures that no probability is lost during the transformation.

$$\begin{aligned} p_x(x) &= p_u(u) \cdot \left| \frac{dx}{du} \right|^{-1} \\ &= p_u(u) \cdot \left| \frac{\partial T(u)}{\partial u} \right|^{-1} \quad \text{where } x = T(u) \\ &= p_u(u) \cdot |T'(u)|^{-1} \\ &= p_u(T^{-1}(x)) \cdot |T'(T^{-1}(x))|^{-1} \quad \text{where } u = T^{-1}(x) \end{aligned} \quad (2.25)$$

In a multi-dimensional setting, the change of variable formula involves the Jacobian matrix of T , J_T instead of its derivative:

$$p_x(x) = p_u(u) |det J_T(u)|^{-1} \quad \text{where } u = T^{-1}(x)$$

The same way, $p_x(x)$ can be expressed in terms of T^{-1} Jacobian matrix:

$$p_x(x) = p_u(T^{-1}(x)) |det J_{T^{-1}}(x)|$$

The absolute value of the Jacobian determinant quantifies the volume change in a small neighbourhood of u after using T . Let du and dx a small neighbourhood around u and x respectively. T transforms du into dx . Therefore $|det J_T(u)| \approx Vol(dx)/Vol(du)$. As the probability mass in dx should be the same as in du , if du is stretched (resp. compressed), then the density around x is lower (resp. higher) than around u .

An other crucial property of diffeomorphisms is that it is possible to compose them. Given two transformations T_1 and T_2 , their composition $T_2 \circ T_1$ is also a diffeomorphism. its inverse and its jacobian determinant are:

$$\begin{aligned} (T_2 \circ T_1)^{-1} &= T_1^{-1} \circ T_2^{-1} \\ det J_{T_2 \circ T_1}(u) &= det J_{T_2}(T_1(u)) \cdot det J_{T_1}(u) \end{aligned}$$

It is therefore possible to construct complex transformations by composing several simpler transformations without jeopardising the invertibility and differentiability properties.

Another crucial advantage of NF is that given a parameterised transformation T_θ (that can be a chain of transformations), the parameters θ can be learnt using Maximum likelihood Estimation (MLE). Given data x_i sampled from the target distribution (typically a dataset of images), the likelihood of sample x_i can be easily computed:

$$p_x(x_i) = p_u(u) \cdot |T'(u)|^{-1} = p_u(T^{-1}(x_i)) \cdot |T'(T^{-1}(x_i))|^{-1}$$

The likelihood of a batch can be computed using $\prod_{i=1}^n p_x(x_i)$. Finally, the last step is to train the model using the Negative Log-Likelihood $-\sum_{i=1}^n \log(p_x(x_i))$.

More specifically, fitting a flow model $p_x(x; \theta)$ to a target $p_x^*(x)$ can be done through a divergence minimisation:

$$\begin{aligned}\mathcal{L}(\theta) &= D_{KL} [p_x^*(x) || p_x(x; \theta)] \\ &= -\mathbb{E}_{p_x^*(x)} [\log p_x(x; \theta)] + \text{const} \\ &= -\mathbb{E}_{p_x^*(x)} [\log p_u(T^{-1}(x; \theta) + \log |\det J_{T^{-1}}(x; \theta)|)] + \text{const}\end{aligned}\tag{2.26}$$

This is called the Forward KL divergence. The Forward KL is useful in situations where samples from the target distribution are available but it is impossible to evaluate the target density $p_x^*(x)$. The forward KL divergence is equivalent to the MLE process described above. Indeed, given a set of samples $\{x_n\}_{n=1}^N$ from $p_x^*(x)$, it is possible to estimate the expected value on p_x^* using a Monte-Carlo method:

$$\theta \approx -\frac{1}{N} \sum_{n=1}^N \log p_u(T^{-1}(x_n; \theta) + \log |\det J_{T^{-1}}(x_n; \theta)|) + \text{const}$$

Minimising the MC approximation of the KL divergence is equivalent to fit the samples through MLE.

Fitting a flow model requires to calculate T^{-1} , the determinant of its jacobian and its density $p_u(u)$ and to compute their derivatives. It is interesting to notice that it is possible to learn a flow model using MLE even if we cannot calculate T or sample from $p_u(u)$. However these operations will be required to sample from the model after training.

In the reinforcement learning setting, there are no samples from the target distribution, which can be associated to the optimal policy. However, as will be described in the next chapter, there are settings where given a sample (i.e. an action) it is possible to evaluate its density under the target distribution (i.e. optimal policy). In such a setup, NF can be helpful as they can be trained using the Reverse KL divergence.

$$\begin{aligned}\mathcal{L}(\theta) &= D_{KL} [p_x(x; \theta) || p_x^*(x)] \\ &= \mathbb{E}_{p_x(x; \theta)} [\log p_x(x; \theta) - \log p_x^*(x)] \\ &= \mathbb{E}_{p_u(u; \theta)} [\log p_u(u; \theta) - \log |\det J_T(u; \theta)| - \log p_x^*(T(u; \theta))]\end{aligned}\tag{2.27}$$

To use the reverse KL divergence, it is necessary to be able to sample from the base distribution $p_u(u)$ and to compute and differentiate T as well as the determinant of its jacobian. This means that it is possible to train a flow model using the Reverse KL divergence even if it is not possible to evaluate the base density or calculate T^{-1} . However these operations will be required at inference time.

In any case, the model has to be invertible and the determinant has to be tractable. Indeed, computing the jacobian of a differentiable function in D dimensions has a cost $\mathcal{O}(D^3)$, which becomes quickly intractable when D is high. All the existing approaches rely on the same principle presented in this section, however they all differ in their approach to allow more tractable and faster determinant computation. The interested reader is advised to consult the surveys [Pap+21; KPB21] that summarise the different existing approaches to build flow models.

2.5.6 Application of Normalizing Flows to RL

Normalizing Flows have not been much used in the RL context at the time of the writing of the manuscript (besides the contributions made in this work). Their use has mainly been limited as a tool for modelling stochastic policies. Indeed, replacing the classic Gaussian distribution with a more complex distribution has been shown as providing more flexible distributions and complex behaviours, and therefore more effective models.

Boosting TRPO with Normalizing Flows

For on-policy algorithms, updates with a large step size can lead to collecting bad samples from which the policy cannot recover. TRPO [Sch+15] have been proposed to avoid this issue by constraining the KL divergence between two consecutive policy updates. However, using a Gaussian policy, the constraint on the KL divergence is too strict which makes it hard to overcome local optima.

Intuitively, a more expressive policy would be able to represent more complex distributions and therefore the KL constraint may not be as strict anymore (at least no as much as for Gaussian policies). In [TA19] the authors show that more expressive distributions obtained using NF can be combined with on-policy algorithms and boost TRPO performance.

Improving Exploration in Soft-Actor-Critic with Normalizing Flows Policies

In environments with continuous action space, stochastic policies allow for an on-policy exploration and off-policy training. Soft Actor-Critic (SAC) [Haa+18] is an example of such algorithm that adds an entropy term to the reward. SAC learns policies that maximise both the expected return and their entropy, which enhances stability, exploration and robustness. With $J(\pi)$ the performance measure of policy π and \mathbb{H} the entropy, we have:

$$J(\pi) = \sum_{t=0}^T \mathbb{E}_{s_t \sim p} [r(s_t, a_t) + \alpha \mathbb{H}(\pi(\cdot|s))]$$

Most stochastic policies are gaussian distributions that take advantage of the reparameterisation trick; but their expressiveness is limited. Indeed, the choice of the distribution family to learn can have significant consequences. For instance, in robotics where actions on joints correspond to bounded angles, it has been shown that a policy based on a beta distribution converges faster towards better policies.

In most cases, the optimal distribution to model the optimal policy is not known beforehand. Therefore, choosing the appropriate distribution to model stochastic policies can be hard without expert knowledge. In [WSB19], the authors propose to solve that issue by taking advantage of the flexibility of NF.

Leveraging Exploration in Off-policy Algorithms via Normalizing Flows

Exploration is one the main issues to handle in RL. This problem is even harder in robotic tasks in high dimensional state-action spaces. Environments that include a large amount of continuous

spaces like those that include a combination of leg/arm/posture movements present several local minima. For instance, it is possible to perform a forward movement in humanoid environments with a variety of sub-optimal policies. Those policies will systematically fail in environments specifically designed to destabilise the agent. To succeed in this setting, it is essential that the exploration strategy helps avoiding to converge too fast towards local optima.

In [Maz+20], the authors enhance SAC exploration policies using NF. Indeed, vanilla SAC is limited to the policies for which entropy can be computed in closed form. [Maz+20] extends SAC to a richer set of multimodal exploration policies.

2.6 Monte-Carlo Markov Chain

In chapter 3, we will present an algorithm that learns optimal policies using the Metropolis-Hastings (MH) algorithm considered as the most popular Monte-Carlo Markov Chain (MCMC) algorithm. In this section we will present MCMC and MH algorithm. This review is based on the excellent survey proposed by Andrieu et.al [And+03].

2.6.1 Historical Perspective

In 1946, while recovering from an illness, Stan Ulam devised the concept of using random sampling to approximate complex combinatorial problems, inspired by a game of solitaire. This idea laid the foundation for modern Monte Carlo simulation. Ulam, along with John Von Neumann, developed various Monte Carlo algorithms, including importance sampling and rejection sampling, to solve problems in neutron diffusion and mathematical physics.

During the 1930s, Enrico Fermi had already utilized Monte Carlo methods for neutron diffusion calculations and designed the FERMIAC, a mechanical device for Monte Carlo calculations. In the 1940s, Nick Metropolis, a physicist, worked on computing machines and was captivated by Monte Carlo methods. He designed the MANIAC computer and published a seminal document on Monte Carlo simulation in 1949 with Stan Ulam, introducing Monte Carlo particle methods.

In 1953, Metropolis, along with the Tellers and Rosenbluths, proposed the Metropolis algorithm. Subsequently, numerous papers on Monte Carlo simulation emerged in the physics literature. However, it was only in 1970 that Hastings and Peskun generalized the Metropolis algorithm, leading to the Metropolis-Hastings algorithm. In the 1980s, significant contributions to Markov Chain Monte Carlo (MCMC) appeared in computer vision and artificial intelligence literature.

2.6.2 Motivation

MCMC methods will mainly be used in this manuscript for Bayesian Inference and Learning. Given some unknown variables $x \in \mathcal{X}$, and data $y \in \mathcal{Y}$, the following typically intractable integration problems are central to Bayesian statistics.

- **Normalisation:** To obtain the posterior $p(x|y)$ given the prior $p(x)$ and likelihood $p(y|x)$,

the normalising factor in Bayes' theorem needs to be computed:

$$p(x|y) = \frac{p(y|x)p(x)}{\int_{\mathcal{X}} p(y|x')p(x')dx'}$$

- **Marginalisation:** Given the joint posterior of $(x, z) \in \mathcal{X} \times \mathcal{Z}$, we may often be interested in the marginal posterior:

$$p(x|y) = \int_{\mathcal{Z}} p(x, z|y)dz$$

- **Expectation:** the objective of the analysis is often to obtain summary statistics of the form

$$\mathbb{E}_{p(x|y)}(f(x)) = \int_{\mathcal{X}} f(x)p(x|y)dx$$

for some function of interest $f : \mathcal{X} \rightarrow \mathbb{R}^{n_f}$ integrable with respect to $p(x|y)$.

2.6.3 MCMC Principle

The "Monte-Carlo" in the name MCMC stems from the same Monte-Carlo principle used in MC based methods for RL presented in section 2.1.3. The idea is to draw an i.i.d. set of samples $\{x^{(i)}\}_{i=1}^N$ from a target density $p(x)$ defined on a high-dimensional space \mathcal{X} . These N samples can be used to approximate the target density with the following empirical point-mass function

$$p_N(x) = \frac{1}{N} \sum_{i=1}^N \delta_{x^{(i)}}(x)$$

where $\delta_{x^{(i)}}(x)$ denotes the delta-Dirac mass located at $x^{(i)}$. Consequently, one can approximate the integrals $I(f)$ with tractable sums $I_N(f)$ that converge as follows

$$I_N(f) = \frac{1}{N} \sum_{i=1}^N f(x^{(i)}) \xrightarrow{N \rightarrow \infty} I(f) = \int_{\mathcal{X}} f(x)p(x)dx$$

The estimate $I_N(f)$ is unbiased by the strong law of large numbers, it will almost surely converge to $I(f)$. If the variance of $f(x)$ satisfies $\sigma_f^2 = \mathbb{E}_{p(x)}(f^2(x)) - I^2(f) < \infty$, then the variance of the estimator $I_N(f)$ is equal to $\text{Var}(I_N(f)) = \frac{\sigma_f^2}{N}$ and the central limit theorem yields convergence in distribution of the error

$$\sqrt{N}(I_N(f) - I(f)) \xrightarrow[N \rightarrow \infty]{D} \mathcal{N}(0, \sigma_f^2)$$

where D denotes convergence in distribution. The advantage of Monte-Carlo integration over deterministic integration arises from the fact that the former positions the samples in regions of high probability.

When $p(x)$ has standard form e.g. Gaussian, it is straightforward to sample from it using easily available routines. However, when this is not the case, we need to introduce more sophisticated techniques based on rejection sampling and MCMC.

2.6.4 Rejection Sampling

We can sample from a distribution $p(x)$, which is known up to a proportionality constant, by sampling from another easy-to-sample proposal distribution $q(x)$ that satisfies $p(x) \leq Mq(x)$, $M < \infty$, using the accept/reject procedure described below:

1. Set $i = 1$
2. Repeat until $i = N$
 - (a) Sample $x^{(i)} \sim q(x)$ and $u \sim \mathcal{U}[0, 1]$
 - (b) if $u \leq \frac{p(x^{(i)})}{Mq(x^{(i)})}$ then accept $x^{(i)}$ and increment the counter i by 1. Otherwise reject.

The accepted $x^{(i)}$ can be easily shown to be sampled with probability $p(x)$. This simple method suffers from severe limitations. It is not always possible to bound $p(x)/q(x)$ with a reasonable constant M over the whole space \mathcal{X} . If M is too large, the acceptance probability

$$\Pr(x \text{ accepted}) = \Pr\left(u < \frac{p(x)}{Mq(x)}\right) = \frac{1}{M}$$

will be too small. This makes the method impractical in high-dimensional scenarios.

2.6.5 MCMC Algorithms

MCMC is a strategy for generating samples $x^{(i)}$ while exploring the state space \mathcal{X} using a Markov chain mechanism. This mechanism is constructed so that the chain spends more time in the most important regions. In particular, it is constructed so that the samples $x^{(i)}$ mimic samples drawn from the target distribution $p(x)$. As defined in section 1, we recall the Markov property

$$p(x^{(i)} | x^{(i-1)}, \dots, x^{(1)}) = T(x^{(i)} | x^{(i-1)})$$

The chain is homogeneous if $T(x^{(i)} | x^{(i-1)})$ remains invariant for all i . That is, the evolution of the chain in a space \mathcal{X} depends solely on the current state of the chain and a fixed transition matrix. As an example, consider a Markov chain with three states ($s = 3$) and the transition matrix:

$$T = \begin{bmatrix} 0 & 1 & 0 \\ 0.9 & 0.1 & 0.9 \\ 0.6 & 0.4 & 0 \end{bmatrix}$$

If the probability vector for the initial state is $\mu(x^{(1)}) = (0.5, 0.2, 0.3)$, it follows that $\mu(x^{(1)})T = (0.2, 0.6, 0.2)$ and, after several iterations (multiplications by T), the product $\mu(x^{(1)})T^t$ converges to $p(x) = (0.2, 0.4, 0.4)$. No matter what initial distribution $\mu(x^{(1)})$ we use, the chain will stabilise at $p(x) = (0.2, 0.4, 0.4)$. This stability result plays a fundamental role in MCMC simulation. From any starting point, the chain will convergence to the invariant distribution $p(x)$, as long as T is a stochastic transition matrix that obeys the following properties:

- **irreducibility:** For any state of the Markov chain, there is a positive probability of visiting all other states. That is, the matrix T cannot be reduced to separate smaller matrices.
- **Aperiodicity:** The chain should not get trapped in cycles

2.6.6 The Metropolis-Hastings Algorithm

The MH algorithm is the most popular MCMC method. A MH step of invariant distribution $p(x)$ and proposal distribution $q(x^*|x)$ involves sampling a candidate value x^* given the current value x according to $q(x^*|x)$. The Markov chain then moves towards x^* with acceptance probability

$$A(x, x^*) = \min\{1, [p(x)q(x^*|x)]^{-1}p(x^*)q(x|x^*)\}$$

otherwise it remains at x . A pseudo-code is shown below:

1. Initialise $x^{(0)}$
2. For $i = 0$ to $N - 1$
 - (a) Sample $u \sim \mathcal{U}[0, 1]$
 - (b) Sample $x^* \sim q(x^*|x^{(i)})$
 - (c) if $u < A(x^{(i)}, x^*) = \min \left\{ 1, \frac{p(x^*)q(x^{(i)}|x^*)}{p(x^{(i)})q(x^*|x^{(i)})} \right\}$
$$x^{(i+1)} = x^*$$
 - else
$$x^{(i+1)} = x^{(i)}$$

The algorithm can be intuitively interpreted as follows: at each iteration, we attempt to move in the space of possible states. The move may be accepted or rejected. The acceptance rate A indicates how likely the new state is, given the current state, and according to the distribution p . If we are trying to move to a state more probable than the current state (if $A \geq 1 \geq u$), the move is always accepted. However, if we are trying to move to a state less probable than the current state, then the move may be rejected, and the rejection is more likely the higher the drop in probability density p . Consequently, the walk tends to visit preferentially the regions of the state space where the density p is high, but occasionally visits regions of lower density.

The MH algorithm has the advantage of being very simple, but it requires careful design of the proposal distribution $q(x^*|x)$.

Chapter 3

Curiosity Augmented Metropolis for Exploratory Policies

Reinforcement learning algorithms are typically designed to converge towards a unique optimal policy for a given reward function. In this chapter, we pose the following question: is it possible to produce behaviourally-diverse succeeding policies (different behaviour, yet acceptable ‘succeeding’ performance)? We answer in the form of a novel Monte-Carlo Markov Chain method coupled with Normalizing Flows, which we demonstrate to successfully sample such behaviours.

3.1 Introduction

In a reinforcement learning context, solving a task defined by a given reward function (i.e. base objective) takes the form of finding an optimal policy dictating the agent’s behaviour in the environment. A problem arises when the obtained optimal policy’s behaviour is not the one expected by the human practitioner (potential mis-alignment between the human objective and the base objective). How to efficiently capture all possible **succeeding behaviours** (those which arise from policies that are acceptably close to optimal by human standards)?

An unexpected behaviour can be illustrated with the cancer example: an algorithm whose base objective is to minimise the number of cancer deaths and reaches an optimal policy consisting of killing every human being does not comply with the human objective that consists in curing cancer.

In RL, it is common to face situations where for the same reward, several optimal policies solve the task at hand while displaying different behaviours. One classic example where the optimal policy is not unique is the cartpole problem (figure 3.1). It is a classic control problem where a pole is attached to a cart, and the goal is to keep the pole upright by moving the cart horizontally. The agent receives a positive reward for keeping the pole upright. The episode ends if the pole falls beyond a certain angle or if the cart moves too far from the center. There are multiple ways to balance the pole. Two different optimal policies can be as follows: (1) the agent can choose to oscillate the cart left and right quickly to keep the pole balanced, or (2) it could move slowly in one direction to maintain stability. The mesa-objective is an additional objective that constraints

the agent to converge towards a given behaviour (either (1) or (2)). In this work, we consider the mesa-objective as a regularisation of the reward function.

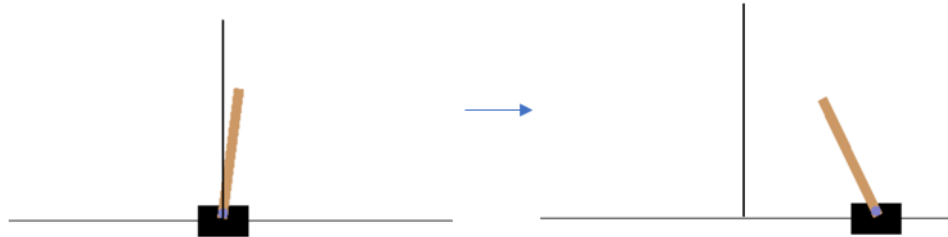


Figure 3.1: The cartpole environment. A pole is attached to the cart, and the goal is to keep the pole upright by moving a cart horizontally. The agent receives a positive reward for keeping the pole upright. The episode ends if the pole falls beyond a certain angle or if the cart moves too far from the center. Two different optimal policies can be as follows: the agent can choose to oscillate the cart left and right quickly to keep the pole balanced (left), or it could move slowly in one direction to maintain stability (right).

While there often exist several **succeeding policies** (‘close enough to optimal’ according to human appreciation) each displaying a different behaviour, RL algorithms found in the literature are designed to only output a single optimal policy. In addition, depending on their design and despite the reward function being the same, their policies may be different. A detailed example is given in section 3.2.1. This can be seen as an implicit regularisation proper to the algorithm design. This regularisation characterises a mesa-objective and drives convergence towards a unique solution and hence a certain behaviour.

In order to ensure alignment, we can either define the necessary regularisation leading to a sought behaviour, or find all possible behaviours and corresponding successful policies to choose the most suitable one. Our approach is the latter.

Making explicit the regularisation that leads to a certain behaviour before trying it is not easy. Current approaches force the user that wants to impose a certain behaviour to either test different algorithms or to use reward shaping in an empirical manner hoping for the sought result. In this work, instead of trying to make the regularisation explicit, we consider that there exists a distribution of succeeding policies for a given task and propose a deterministic successful policies generating process. This process takes the form of a Monte-Carlo Markov Chain that uses the Metropolis-Hastings algorithm to sample from the distribution of successful policies. As MCMC algorithms may not be effective when areas of low probability separate regions of high probability, we coupled MCMC with Normalizing Flows powered proposals and a curiosity mechanism ensuring that the output policies adopt diverse behaviours. The result is that our model generates a series of succeeding policies on the fly ensuring that each output policy solves the tasks while adopting a different behaviour. We argue that the user can then choose the policy with the behaviour that suits him the most or can even discover behaviours he did not even thought of.

We test our approach on several classic Gym environments and show that our proposed algorithm solves each of them fast and effectively outputs succeeding policies on the fly that all present

diverse behaviours, i.e. they all solve the task while adopting different strategies. These results show that using a single base objective (reward function) it is indeed possible to produce many different succeeding policies, illustrating the outer alignment problem (human vs base objective).

This chapter is organised as follows: in section 3.2, we draw the link between mesa-objective and risk accounting then we define the concept of Set-policy that will be at the heart of this chapter and manuscript. In section 3.3 we explain why classic approximate inference cannot be used to solve our problem and why we are constrained to use MCMC approaches. In section 3.4 we present our approach by first detailing a naive MCMC algorithm to output succeeding policies and its results (section 3.4.2); then we present a first enhancement to our first proposal using normalizing flows to boost exploration (section 3.4.3); finally we present our final algorithm that adds a curiosity mechanism ensuring that the output succeeding policies effectively display diverse behaviours (section 3.4.3).

3.2 Succeeding Behaviours

As preliminaries, to this section, we must remind ourselves of some concepts from the introduction chapter 1 and formalise the different objectives involved which are as follows:

- **human objective** : expresses a goal, or a task to be accomplished, e.g., win a game at chess,
- **base objective** : is the reward function $r(s_t, a_t)$ e.g., reward equals 1 if win, and 0 otherwise,
- **return**: is a discounted sum of rewards $R(\tau) = \sum_{t=1}^T \gamma^t r(s_t, a_t)$,
- **mesa objective**: is an implicitly defined objective involving the learning update and the loss function \mathcal{L} , e.g., MSE, or entropy-regularised return with gradient descent, or Q-table update.

A **succeeding behaviour** is a behaviour coming from a policy which solves the human objective, whereas optimal behaviour is from a policy that maximises the base objective. Note that a succeeding behaviour is not necessarily optimal according to the base objective (given reward function). So, winning a game of chess is not necessarily optimal, according to the return.

In this section, we give a more specific example of RL algorithms that converge to different optimal policies, offer an interpretation of the mesa-objective of each policy as an implicitly defined risk constraint and give a proper definition of the set of optimal policies, that we call set-policy.

3.2.1 Different Algorithms Converge to Different Policies

In the previous section we gave the example of two equivalent strategies that solve the cartpole problem. In this section we will cite specific algorithms that, when confronted to the same task, converge towards different policies.

One example of RL algorithms giving different optimal policies for the same task is in the domain of robotic control, particularly in robotic locomotion. Consider a scenario where a quadrupedal robot needs to learn to walk. The goal is to learn a walking gait that allows the robot to move forward efficiently. The robot can control the joint angles and torques of its legs and receives

positive rewards for making forward progress. Rewards might also be given for maintaining stability and avoiding falls. There are various ways a quadrupedal robot can walk, with different gaits and leg movement patterns. Some algorithms might prioritise fast forward motion, while others might focus on energy efficiency or stability.

Different reinforcement learning algorithms, such as Deep Deterministic Policy Gradient (DDPG) [Sil+14], Trust Region Policy Optimization (TRPO) [Sch+15], and Proximal Policy Optimization (PPO) [Sch+17], may converge to different optimal policies due to variations in exploration-exploitation strategies, policy parameterisations, and update mechanisms:

- DDPG might discover a dynamic and agile walking gait that allows the robot to move quickly.
- TRPO might converge to a more cautious and stable gait, prioritising risk aversion and avoiding falls.
- PPO might find a balance between speed and stability, resulting in a gait that combines elements of both dynamic and stable walking.

Each algorithm may interpret the task differently, explore different regions of the policy space, and ultimately converge to different optimal policies for the same walking task. The optimal policy to which each algorithm converges to depends greatly on their exploration strategy and the initial state.

Considering RL methods as optimisation algorithms, their objective is to find the policy that maximises the return. As displayed in figure 3.2, there could exist a set of solutions maximising the return. As existing algorithms tend to converge towards a unique solution consistently, this means that there exists an implicitly defined regularisation that restricts the feasible set of each algorithm such that the optimal solution becomes unique.

3.2.2 Existing Approaches for Finding Diverse Succeeding Behaviours

Various approaches in RL focus on achieving diverse behaviours through **skill discovery** [Cam+20] instead of explicitly specifying the regularisation needed to achieve a particular objective while demonstrating preferred behaviour. Skill discovery involves identifying latent-conditioned policies that consistently modify the environment’s state. For instance, intelligent creatures can explore their environments and learn useful skills without supervision. In RL, unsupervised methods are often aimed at learning generically useful behaviours from interacting within some environment, behaviours that may naturally accelerate learning once one or more downstream tasks become available. The usefulness of a skill can be quantified through the concept of **empowerment**, which quantifies an agent’s ability to discover and execute actions within an environment. This notion relies on the mutual information concept borrowed from information theory and plays a central role in this formulation.

Maximisation of mutual information between a latent variable and the state is a popular skill discovery method. In [Cam+20], it is shown that two views of the mutual information lead to different algorithms such as Diversity is All You Need (DIAYN) [Eys+19] or Dynamics Aware Unsupervised Discovery of Skill [Sha+20].

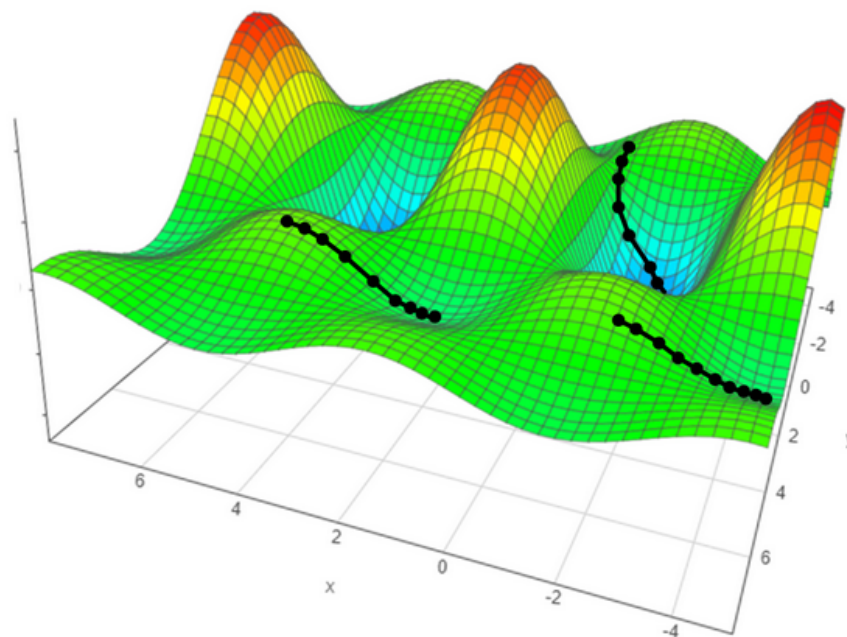


Figure 3.2: Landscape of a 3D function with 3 global optima. Depending on the initialisation, and the exploration strategy, different optimisation algorithms can converge to different optima.

In [Sha+20] the authors propose a model based method that is out of the scope of this manuscript where we focus on model free RL. DIAYN is a method for learning useful skills without a reward function. This method learns skills by maximising an information theoretic objective using a maximum entropy policy. On a variety of simulated robotic tasks, they show that this simple objective results in the unsupervised emergence of diverse skills, such as walking and jumping. Their results suggest that unsupervised discovery of skills can serve as an effective pretraining mechanism for overcoming challenges of exploration and data efficiency in reinforcement learning. It emphasises that skills are valuable when they influence the states an agent encounters. Different skills should visit different states. DIAYN promotes exploration by encouraging skills to be as varied as possible. It achieves this by incentivising skills to act randomly, aiming for high entropy.

In our specific context, our focus lies in discovering varied successful strategies to achieve specific and predefined objectives. However, DIAYN primarily aims at acquiring skills that might prove useful when a task emerges within the environment. SMERL [Fer+20] extends DIAYN into a supervised setting where a task-driven reward is accessible. The key insight of SMERL is that learning diverse behaviours for accomplishing a task can directly lead to behaviours that generalise to varying environments. By identifying multiple solutions for the task in a single environment during training, this approach can generalise to new situations by abandoning solutions that are no longer effective and adopting those that are. Similar to our approach, the idea of SMERL is to seek diversity when the return is close to the maximal one, i.e. to look for slightly sub-optimal but diverse solutions (succeeding policies). However, contrary to our proposal, SMERL starts by

learning the optimal SAC policy and then starts diversifying around the optimal solution.

While the diversity of encountered states is an essential part of its objectives, it has been shown in [Cam+20], that information theoretic skill discovery methods suffer from a common limitation. They discover options that provide a poor coverage of the state space as shown in figure 3.3. Moreover, our aim is to find diverse trajectories and strategies towards the same goal rather than building skills that would be useful in a different environment or for a different objective. This implies that information theory based approaches cannot serve as a useful baseline. Finally, as we uniquely interpret behaviours through encountered states diversity, our MCMC based approach ensures greater coverage of the state space.

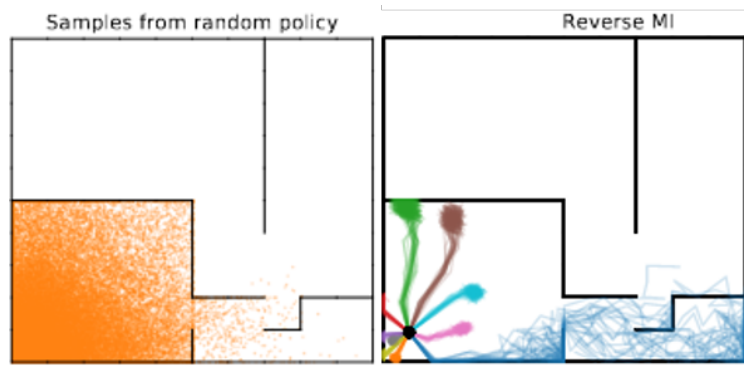


Figure 3.3: Skills learned on a maze with bottleneck states. Each coloured line represents a trajectory initiated at the black dot by a different skill. Multiple rollouts per skill are reported in order to account for the stochasticity of the policy. The left plot depicts states visited by a policy with random weights, showing which states are reachable by the agent at the beginning of training. SMERL (right) fails at expanding this set of states, and ends up committing to behaviours discovered by the random policy [Cam+20].

3.2.3 Risk Accounting in Succeeding Policies

As mentioned, succeeding policies (like optimal policies) are not necessarily unique. The same way, there may be a diversity of succeeding policies.

For example, there may be many policies which succeed at winning a game of chess. A time-discounted reward imposes that an optimal agent must be the fastest at winning the game, but from a human point of view, this may be considered overfitting; a succeeding agent that simply wins the game can be acceptable. In other words, by finding *succeeding* policies, we are less susceptible to overfitting (more capacity to generalise/transfer well to other environments), and more likely to align with human preferences. More details on the link between generalisation, overfitting and risk are given in the appendix section 3.6.1.

Risk is integral to mesa-objectives. Let us consider with a toy example: an agent is offered to take €1 (action 1) or take €100 with chance 0.01 (action 2). There are infinite optimal policies here (proof sketch: suppose action 1 with probability $\pi(a_1)$; any $\pi(a_1) \in [0, 1]$ will be optimal) respective of payoff. However, decision makers often prefer a particular risk setting (mesa-objective). Given the choice between two optimal behaviours (respective of payoff), an agent may prefer the one

of less risk (action 1) even if equivalent, and even if not ‘as’ optimal (i.e., any succeeding policy of low risk). A more elaborate example, based on the Markowitz model [Mar52] is explained in the appendix (section 3.6.2). There exists extensive literature on risk quantification [AF22].

In our case (this chapter), we consider that we do not know the mesa-objective profile involving risk. Rather, we propose a mechanism to efficiently generate *all* succeeding policies – such that a hypothetical human practitioner/user could select according to preference.

3.2.4 Set-Policy Definition

The concept of set policy was originally introduced in [PGP17] in the context of imitation learning; we re-purpose it here for our study on succeeding policy. It is essentially a one-to-many mapping between states, and actions; defined formally as follows.

Definition 1: Set-policy

A set-policy $\bar{\pi}$ is an element of the set $(\mathcal{P}(\mathcal{A}) \setminus \{\emptyset\})^{\mathcal{S}}$. To each policy π , one can associate a set-policy $\bar{\pi}$ defined as: $\forall s \in \mathcal{S}, \text{Supp}(\pi(\cdot|s)) = \bar{\pi}(s)$. Let $\bar{\pi}_1$ and $\bar{\pi}_2$ two set policies, $\bar{\pi}_1 \subset \bar{\pi}_2$ if $\forall s \in \mathcal{S}, \bar{\pi}_1(s) \subset \bar{\pi}_2(s)$.

We cannot simply generate a set of succeeding policies (using standard methods like actor-critic). Rather, we search for a single **set policy**, such that, we can simply sample an action from the set policy, as a surrogate for sampling a succeeding policy and then following its deterministic state-action mapping. Note that since from each state we can take a different action, it means that, given a set policy as a tool, we can obtain diverse succeeding behaviours.

As shown in figure 3.4, to each state is associated a set of optimal actions denoted $\bar{\pi}(s)$. Our goal is to sample an action from this set of optimal actions at each state. This means that we can perform multiple different actions for each given states, allowing to obtain diverse behaviours at each run as each chosen action for a given state leads to a different behaviour.

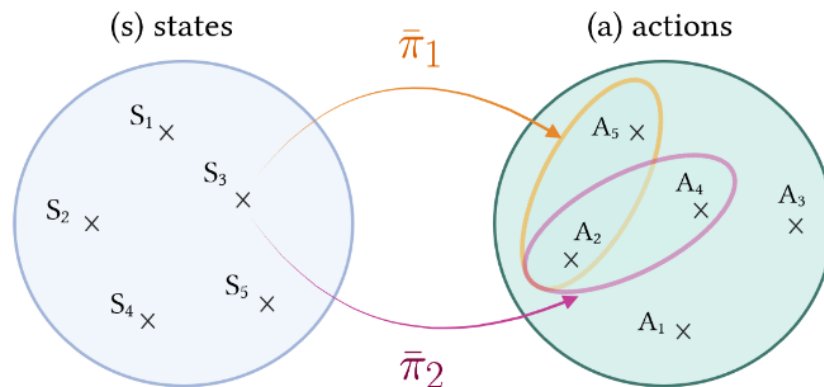


Figure 3.4: Illustration of set-policy. Given a state space \mathcal{S} and an action space \mathcal{A} , a set policy is a mapping that associates for each $s \in \mathcal{S}$, a finite set of actions in \mathcal{A} .

One can also define the optimal set-policy:

$$\forall s \in \mathcal{S}, \quad \bar{\pi}_r^*(s) = \arg \max_a [Q_r^*(s, a)]$$

The optimal set policy states for each state the set of optimal actions that maximise the expected return. Therefore:

$$V_r^\pi = V_r^* \iff \bar{\pi} \subset \bar{\pi}_r^*$$

More details are available in the appendix section 3.6.3.

3.2.5 Reformulating the Objective with Set Policies

Instead of converging towards a unique optimal policy that would be optimal with respect to a given pair of risk measure and reward function, the aim of this work would be to learn the optimal set policy itself, i.e. being able to individually sample deterministic optimal policies from the set of optimal policies. This would allow to observe different policies that solve the same task while adopting different behaviours, each one optimising the same objective but with a different unknown risk measure.

One may notice that learning the optimal set policy $\bar{\pi}^*$ and sampling an action for each state from the set of actions given by $\bar{\pi}^*(s)$ is equivalent to following an optimal policy while we aim at learning **succeeding** behaviours rather than optimal ones. However, the optimal set-policy $\bar{\pi}^*$ is only optimal considering the given reward function. As we consider that succeeding policies also optimise an implicitly defined regularisation, the policies we aim for will sometimes diverge from the optimal set of actions at certain states. The mechanism allowing this is the curiosity mechanism introduced in section 3.4.3. This mechanism will force our model to sometimes pick sub-optimal actions for the given reward function. Therefore diversity will be ensured through two different tools:

- The set policy that allows to take different optimal actions at each run and ensures that the task is eventually solved.
- The curiosity mechanism that forces the model to take sub-optimal actions for the given reward function in order to explore new paths that may be optimal for other regularisations of the reward function.

3.3 Why not Maximum Likelihood for Finding Succeeding Policies?

As said in the previous section, the objective of this work is to find succeeding policies displaying diverse behaviours. This means that given a certain initial state and a final goal state, the aim is to generate a collection of trajectories from the initial to the final state. Considering a maximum likelihood approach, this corresponds to a model where succeeding trajectories have a high probability to occur. In this section, we will show that adopting a maximum likelihood approach for finding succeeding policies is not optimal.

Let π_θ be a stochastic policy defined as above parameterised by θ . If π_θ is optimal, then $\pi_\theta \subset \bar{\pi}^*$. This means that:

$$\forall s \in \mathcal{S}, \quad \pi_\theta(\cdot|s) \subset \bar{\pi}^*(\cdot|s)$$

In a maximum likelihood approach we would like to optimise the following objective:

$$\max_{\theta} p(\pi_\theta \subset \bar{\pi}^*)$$

Which is equivalent to:

$$\max_{\theta} p(\pi_\theta(\cdot|s) \subset \bar{\pi}^*(\cdot|s)) \quad \forall s \in \mathcal{S} \quad (3.1)$$

The aim is to maximise the probability that each state-action pair encountered when following π_θ is optimal. To ease the notation we introduce the variable \mathcal{O} , that was first introduced in [Lev18]. It is a binary variable with $\mathcal{O}_t = 1$ if timestep t is optimal and $\mathcal{O}_t = 0$ otherwise. Hence, the objective (3.1) is equivalent to the following objective:

$$\max_{\theta} p(\mathcal{O}_t = 1 | s_t, \pi_\theta(\cdot|s_t))$$

Following an optimal policy should yield maximum reward, so $p(\mathcal{O}_t = 1 | s_t, \pi_\theta(\cdot|s_t))$ should be maximal when the reward is maximal. Therefore:

$$p(\mathcal{O}_t = 1 | s_t, \pi_\theta(\cdot|s_t)) = \exp(r(s_t, \pi_\theta(\cdot|s_t))) \quad (3.2)$$

It's worth pointing out that the definition of $P(\mathcal{O}_t = 1 | s_t, \pi_\theta(\cdot|s_t))$ in Equation (3.2) requires an additional assumption, which is that the rewards $r(s_t, a_t)$ are always negative. This assumption is not actually very strong: if we assume the reward is bounded above, we can always construct an exactly equivalent reward simply by subtracting the maximum reward.

It is then possible to derive the probability of a trajectory τ (i.e. a deterministic policy) to be optimal. When $\mathcal{O}_t = 1$, for all t :

$$\begin{aligned} p(\tau | o_{1:T}) &\propto p(\tau, o_{1:T}) = p(s_1) \prod_{t=1}^T p(\mathcal{O}_t = 1 | s_t, a_t) p(s_{t+1} | s_t, a_t) \\ &= p(s_1) \prod_{t=1}^T \exp(r(s_t, a_t)) p(s_{t+1} | s_t, a_t) \\ &= \left[p(s_1) \prod_{t=1}^T p(s_{t+1} | s_t, a_t) \right] \exp\left(\sum_{t=1}^T r(s_t, a_t)\right) \end{aligned} \quad (3.3)$$

This means that the probability to observe a trajectory given it is optimal corresponds to the product between its probability with respect to the environments dynamics and the exponential of all received rewards.

If we restrain ourselves to a deterministic dynamics setting for simplicity¹, the goal is to fit an approximation $\pi_\theta(a_t|s_t)$ such that the trajectory distribution

$$\hat{p}(\tau) = \left[p(s_1) \prod_{t=1}^T p(s_{t+1}|s_t, a_t) \pi_\theta(a_t|s_t) \right]$$

matches the distribution in equation (3.3).

An important question is now to ask which criteria to use in order to compare those two distributions. We can notice that, contrary to a classic supervised learning process, there are no examples of trajectories from the optimal policy available. All trajectories at hand are sampled from the current policy π_θ . Therefore, it is necessary to use the Reverse KL divergence (RKL) [Cha+22] as an optimisation objective:

$$D_{KL}(\hat{p}(\tau)||p(\tau)) = -\mathbb{E}_{\tau \sim \hat{p}(\tau)} [\log p(\tau) - \log \hat{p}(\tau)]$$

However, as stated in [Lev18], RKL is known to be prone to mode collapse. Indeed, RKL prioritises finding a mode of the target distribution rather than matching its moments. Using RKL supposes that the probability to discover relevant regions in p via sampling from \hat{p} is non negligible. In practice, this is unlikely. The direct consequence on our inference procedure is that the policy π_θ will be likely to converge to a nearly deterministic one and will fail to produce policies with different behaviours. In the next section we propose a MCMC based approach that ensures to avoid the mode collapse issue by learning a generative process that outputs behaviour diverse optimal policies while optimising a RKL criteria.

3.4 Our MCMC Approach for Learning the Optimal Set-policy

In this section we will present our MCMC based algorithm in order to learn the optimal set-policy.

3.4.1 Distinction Between Optimal Policy and Stochastic Optimal Policy

Here we anticipate potential confusion; and clarify: sampling trajectories from a stochastic optimal policy is not equivalent to sampling actions from the set policy (our goal).

Let us provide an argument via counter-example. Many RL algorithms, for example actor-critic based ones, aim to learn a stochastic policy. Considering the set policy framework, this means that given set policy $\tilde{\pi}^*(\cdot|s) = \{a \in \mathcal{A} : \max_a Q_r^*(s, a)\}$, the optimal stochastic policy $\tilde{\pi}^*$ can be defined as a map associating a state to a random variable $\mathbb{A} \in \mathcal{A}$:

$$\forall s \in \mathcal{S}, \quad \tilde{\pi}^* : \mathcal{S} \rightarrow \mathcal{A}$$

$$\text{with } p(\mathbb{A} = a_i|s) > 0 \quad \text{if } a_i \in \tilde{\pi}^*$$

$$p(\mathbb{A} = a_i|s) = 0 \quad \text{otherwise}$$

¹see [Lev18] for a derivation in the stochastic dynamics setting

Hence, an optimal stochastic policy could be understood as a process to randomly pick an action from $\bar{\pi}^*(s)$ at each state. Therefore at each episode, the resulting trajectory could be interpreted as the outcome of a deterministic policy $\pi_D^* \subset \bar{\pi}^*$. Following this logic, the optimal stochastic policy is a generating process of the optimal deterministic policies family.

Since standard actor-critic algorithms are value based algorithms that aim at solving a well defined MDP with the objective of maximising the expected return without any explicitly defined risk measure, then acting greedily with respect to the optimal value function converges to the optimal policy. This statement is a reformulation of the Bellman optimality equation:

$$V^*(s) = \max_a \sum_{r,s'} p(r,s'|s,a)(r + \gamma V^*(s')) \quad (3.4)$$

The max operator employed in equation (3.4) is deterministic², therefore there exists a deterministic optimal policy in this context. Hence, any environment that can be modelled by a MDP and solved by a value-based method (e.g. value iteration, Q-learning) has an optimal policy which is deterministic.

The learning process of actor critic methods outputs stochastic policies that optimise an unconstrained objective and in practice do not produce a large variety of behaviours. It is hence understandable that classic actor critic methods are not adapted to solve such a problem and that sampling trajectories from a stochastic optimal policy is not equivalent to sampling actions from the set policy.

3.4.2 Generating Deterministic Policies from the Optimal Set Policy

The deterministic optimal policies generating process (our goal) can take the form of a Monte-Carlo Markov Chain (MCMC) process. Considering that any policy can be uniquely characterised by the weights θ that parameterise it, we will replace π_θ by θ to not overcrowd the notation. Therefore the aim of the generating process is to sample θ_i such that $\theta_i \subset \bar{\pi}^*$.

Supposing that it is possible to sample θ_i implies that we will now consider the optimal set-policy as a distribution. Indeed considering all set-policies $\bar{\pi} \subset \bar{\pi}^*$, the aim is to build a MCMC which stationary distribution Π verifies $\text{Supp}(\Pi(\cdot|s)) = \text{Supp}(\bar{\pi}^*(\cdot|s)), \forall s \in \mathcal{S}$. In other words, sampling $\theta_i \sim \Pi$ is equivalent to sampling trajectories from $\bar{\pi}^*$.

A First Approach Using Metropolis-Hastings

We will reuse the optimality variable \mathcal{O} (from above) for more clarity. Having $\theta_i \sim \Pi$ is equivalent to sample from a distribution $f(\theta|\mathcal{O} = 1)$ (we will drop $= 1$ in the remainder of the derivation for conciseness). The event $(\theta|\mathcal{O})$ means that the drawn θ is known optimal and thus solves the task. MCMC methods are procedures used to generate samples from distributions, when sampling cannot be done directly. For instance, the distribution of interest may not have a closed form formula or is unknown. The procedure relies on the knowledge of a distribution that is proportional to the distribution of interest. Several works consider sampling policy parameters θ_i using MCMC algorithms [TOY18].

²if necessary ties can be broken for max values deterministically with e.g. an ordered list of actions

Using Bayes rule:

$$f(\theta|\mathcal{O}) = \frac{f(\theta, \mathcal{O})}{f(\mathcal{O})} = \frac{f(\theta)f(\mathcal{O}|\theta)}{f(\mathcal{O})} = \frac{f(\theta)f(\mathcal{O}|\theta)}{\sum_{\theta'} f(\mathcal{O}|\theta')f(\theta')} = \frac{f(\theta)f(\mathcal{O}|\theta)}{\mathbb{E}_{\theta' \sim f(\theta')} f(\mathcal{O}|\theta')} \quad (3.5)$$

In [Hof+07], the authors showed that for direct policy search, sampling directly from a distribution that is proportional to the reward performs better than classic simulation methods. Therefore $f(\mathcal{O}|\theta)$ is made proportional to the expectation of a monotonically increasing function with respect to the empirical return, called utility function $U(\tau)$. Intuitively, this corresponds to the same reasoning made in the previous section as $f(\mathcal{O}|\theta)$ should be high if the reward is high.

We want:

$$f(\theta|\mathcal{O}) \propto f(\theta)\eta(\theta)$$

where η is the performance of θ wrt the environment dynamics.

$$\eta(\theta) = \mathbb{E}_{p(\tau|\theta)}[U(\tau)] = \int U(\tau)p(\tau|\theta)d\tau$$

where $p(\tau|\theta)$ is the probability of a trajectory τ when following θ . Even if we only consider deterministic policies, this term also depends on the environment dynamics that can be stochastic.

In physical systems, the Boltzmann form is usually used:

$$\rho_* = Z_*^{-1} e^{-U_*(x)} \quad (3.6)$$

where $x \in \Omega \subset \mathbb{R}^d$; ρ_* a probability density function, U_* an energy function and Z_* the normalisation constant. The same way, denoting the empirical return by $\tilde{G}(\tau) = \sum_{t=0}^T \gamma^t r_t$, we consider that:

$$f(\mathcal{O}|\theta) = \eta(\theta) \propto \mathbb{E}_{p(\tau|\theta)} \left[\lim_{T \rightarrow \infty} e^{\tilde{G}(\tau)/T} Z(T)^{-1} \right]$$

The evaluation of $\eta(\theta)$ can be substituted with an unbiased estimate over N episodes:

$$\begin{aligned} \eta(\theta) &= \mathbb{E}_{p(\tau|\theta)}[U(\tau)] \\ &\approx \bar{U}_N(\theta) = \frac{1}{N} \sum_{i=1}^N U(\tau_i), \quad \tau_i \sim p(\tau|\theta), \end{aligned}$$

Plugging this into Eq.(3.5), we obtain: (the denominator is simply a normalisation constant, and does not need to be considered under proportionality):

$$\begin{aligned} f(\theta|\mathcal{O}) &\propto \frac{f(\theta)e^{\bar{U}_N(\theta)/T} Z(T)^{-1}}{\int_{\theta'} e^{\bar{U}_N(\theta')/T} Z(T)^{-1} f(\theta') d\theta'} \\ &\propto f(\theta)e^{\bar{U}_N(\theta)/T} \end{aligned} \quad (3.7)$$

As detailed in chapter 2, an important variation of MCMC is the independent Metropolis-Hastings (MH) sampler. This method will sample from a target distribution by first sampling from an auxiliary proposal distribution. Then the proposal will be accepted or rejected following the MH criteria. The algorithm’s efficiency depends on the ratio between the target and the proposal densities. If the ratio is bounded on the support of the target distribution, then the MH algorithm benefits from a powerful theory of ergodic geometry. We will therefore adapt the Metropolis algorithm for RL in order to obtain a Markov chain of deterministic optimal policies which stationary distribution corresponds to a distribution on the optimal set-policy. The corresponding algorithm is displayed in algorithm 1.

Algorithm 1 Monte Carlo-within-Metropolis for RL

Require: K : the number of iterations, N : number of episodes, σ : standard variation of Normal distribution

Initialise Agent π

$\theta_0 \sim \mathcal{N}(\mathbf{0}, \sigma_p^2 \mathbf{I}_D)$ ▷ Sample initial weights from a Gaussian distribution

for k from 0 to K **do**

$\theta' \sim \mathcal{N}(\theta_k, \sigma_p^2 \mathbf{I}_D)$ ▷ Sample new weights from a Gaussian

Run N episodes with π_{θ_k} and compute $\bar{U}_N(\theta_k)$

Run N episodes with $\pi_{\theta'}$ and compute $\bar{U}_N(\theta')$

$\beta \leftarrow \frac{f(\theta')\bar{U}_N(\theta')}{f(\theta_k)\bar{U}_N(\theta_k)}$

$\alpha \leftarrow \min(1, \beta)$

$\epsilon \sim \mathcal{U}_{[0,1]}$ ▷ Sample ϵ from a uniform distribution

if $\epsilon < \alpha$ **then**

$\theta_k \leftarrow \theta'$ ▷ Keep θ'

else

$\theta_k \leftarrow \theta_k$ ▷ Keep θ_k

end if

end for

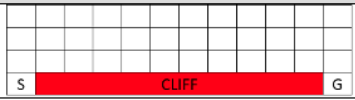
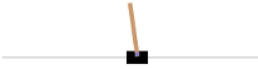

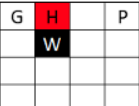
return $\{\theta_k\}_{k=0}^K$ ▷ Return the list of all kept θ

Experiments and Results for Algorithm 1

In this section we present the results obtained using algorithm 1. The agent is a neural network composed of 1 hidden layer of 8 neurons and a ReLU activation function. Our proposed framework was tested in Classic Control Gym environments, Cartpole, Acrobot and cliff [Bro+16] as well as on a gridworld. Table 3.1 shows environments details. The average return was estimated on 20 episodes for every θ_i . The metropolis algorithm was done on 200 timesteps for Cartpole and 500 hundred for Acrobot.

Figure 3.5 presents the results obtained on Cartpole and Acrobot environments. Cartpole is solved while we converge towards a mean average return of -80 which is on par with great implementations according to gym leaderboard. Best implementation achieves a score of -40; we believe that our implementation can reach this score with enough iterations, which is the main drawback of MCMC methods.

Table 3.1: Specifications of environments.

Environments	Snapshot	State space	Action space	Reward
Cliff		$\{1, \dots, 48\}$	$\{0, 1, 2, 3\}$	-1 per move, 10 for the goal and -10 for the pit
Cartpole		\mathbb{R}^4	$\{0, 1\}$	+1 per time step
Acrobot		$[-1, 1]^4 \times [-4\pi, 4\pi] \times [-9\pi, 9\pi]$	$\{0, 1, 2\}$	-1 per time step
Gridworld		$\{1, \dots, 16\}$	$\{0, 1, 2, 3\}$	-1 per move, 10 for the goal and -10 for the pit

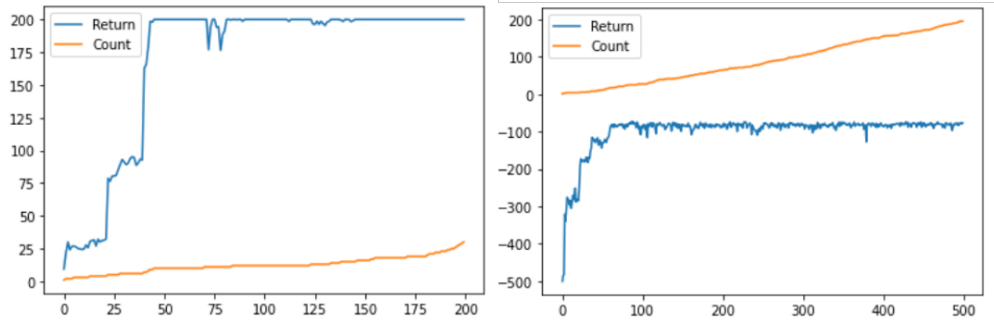


Figure 3.5: Average return for every new θ (in blue) and incremental count of the number of θ_i retained (in orange) on Cartpole (left) and Acrobot (right) using simple implementation.

When visualising the succeeding agents on Cartpole and Acrobot, it is not obvious if they effectively adopt different behaviours. We therefore plot the cosine similarity between all pairs of retained θ_i in figure 3.6. It appears that succeeding θ_i are heavily correlated.

This simple approach fails when confronted to Gridworld or Cliff. In both cases, the agent remains stuck, always performing the same action. The reasons for this failure are explained and tackled in sections 3.4.3 and 3.4.3.

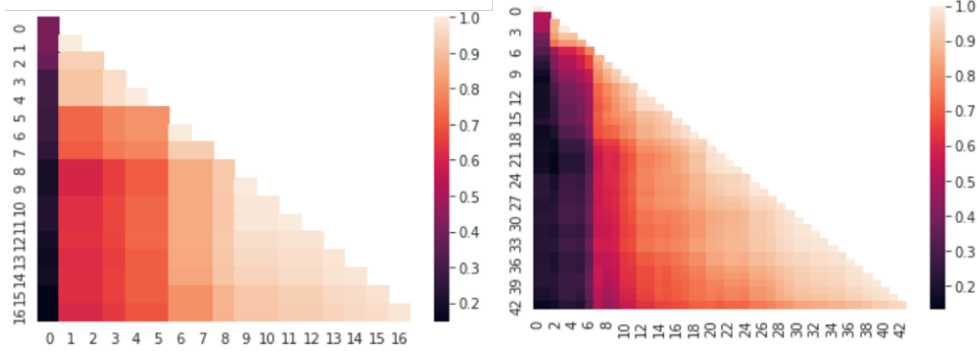


Figure 3.6: Cosine similarity between pairs of retained θ_i on Cartpole (left) and Acrobot (right) using simple implementation.

3.4.3 CAMEO: Boosting Exploration with Normalizing Flows and Curiosity Models

In this section we present our solution algorithm called CAMEO to learn set policies. We first detail the reasons why the previous naive approach failed at solving the gridworld and cliff problems and how to use Normalizing Flows to solve this issue, then we introduce the curiosity module that ensures behaviour diversity. In the results section we will first show that, at convergence, the output policies consistently solve the given problems, then we show that the corresponding θ_i are less correlated than for the approach detailed above. Finally, we will detail the actual behaviours of the output policies and show that they effectively solve the tasks at hand while adopting different behaviours. These behaviours will be compared to those obtained while training an agent using DQN.

Prior and Normalizing Flows

In the last section, we showed that $\{\theta_k\}_{k=1}^K$ are highly correlated; due to a scaling factor among those. Indeed, as we considered that $p(\theta) \sim \mathcal{U}[-a, a]$, there is no bound to θ_i values and therefore we obtain $\theta_{k+1} = \lambda\theta_k$, with λ a constant. A solution is to constraint $\theta_k \in [-1, 1]^D$ but it is not possible to use $p(\theta) \sim \mathcal{U}[-1, 1]$, as it would cancel out anyway during the calculation of β (see algorithm 1). However it is possible to measure the variance of θ_k from $[-1, 1]$:

$$\sigma_{\theta_k}^2 = \frac{1}{D} \sum_{j=0}^D \mathbb{1}_{\theta_{kj} \notin [-1, 1]} (\theta_{kj}^2 - 1)^2, \quad (3.8)$$

and therefore we can define: $p(\theta) = e^{-\sigma_{\theta}^2}$

Unfortunately, this simple solution might not be sufficient when areas of low probability separate regions of high probability. Indeed MCMC algorithms that are mainly driven by local dynamics (Hamiltonian MC, Langevin dynamics...) will find it difficult to transition between high probability regions. This leads either to long correlation times with a low rate of acceptance, or to failure of convergence.

Generative models recently obtained great successes in domains where data acquisition is cheap.

Data acquisition (i.e. sampling from distributions) is actually the main problem we are trying to tackle. A natural question is therefore to ask if traditional MCMC methods can be combined with generative models to accelerate the sampling. A solution is to parameterise directly the proposal distribution $f(\theta)$. Adaptive MCMC methods aim to update the parameters of the proposal distribution during the sampling depending on the previous ones. For instance, one can consider a gaussian covariance matrix as a parameter of the proposal distribution. In [GRV21], Gabri e et al. consider an independent MH sampler where the proposal distribution is represented by a Normalizing Flow which parameters are updated through SGD. Normalizing Flows are characterised by their expressiveness, sampling tractability and a simple evaluation of log-density. These are precisely the necessary attributes for a proposal distribution in MH. Updating the proposal distribution during sampling can violate the Markov property and jeopardize convergence towards the target distribution. We refer the reader to [Bro+22] for an in depth theoretical analysis and convergence proofs.

As a reminder, a Normalizing Flow is an invertible application T that is optimised to transport samples from a base distribution ρ_b to a target distribution. The aim is to produce an application T_* and its inverse \bar{T}_* such that the probability of an observation with respect to ρ_* can be estimated by transforming samples from a base distribution towards the target. Therefore, if a sample x_b is sampled from ρ_b , then $T_*(x_b)$ is a sample of ρ_* and we have:

$$\int_{\Omega} T_*(x)\rho_b(x)dx = \int_{\Omega} x\rho_*(x)dx$$

Even if the application T is not the optimal one T_* ($\hat{\rho}(x) \neq \rho(x)$), as long as $\hat{\rho}$ and ρ_* share the same support, we can still use MH. Denote $y = T(x_b)$, y is accepted with probability:

$$\text{acc}(x,y) = \min \left[1, \frac{\hat{\rho}(x)\rho_*(y)}{\rho_*(x)\hat{\rho}(y)} \right]$$

This procedure is actually equivalent to using the following transition kernel:

$$\pi_T(x, y) = \text{acc}(x, y)\hat{\rho}(y) + (1 - r(x))\delta(x - y)$$

To train the mapping T , we can use the RKL between $\hat{\rho}$ and ρ_* . Using the Boltzmann form of equation (3.6) for ρ_* , we obtain:

$$D_{KL}(\hat{\rho}||\rho_*) = -\log Z_* + \int_{\Omega} [U_*(x) + \log(\hat{\rho}(x))]\hat{\rho}(x)dx$$

The unknown constant $\log Z_*$ is not relevant for optimisation. In [GRV21], the authors propose to combine the Normalizing Flows base transition kernel with a local transition kernel. Indeed, according to the authors, when the parameterised density $\hat{\rho}$ is initialised randomly, it typically has limited overlap with the posterior distribution ρ_* . Consequently, the moves suggested by the Normalizing Flow have a high likelihood of being rejected. However, the situation improves as the local sampler generates data. With continued training, an increasing number of moves proposed by the NF are accepted.

The authors emphasize that the moves generated by pushing forward independent draws from the base distribution are non-local and have the ability to mix between different modes. This property enables the NF algorithm to explore and navigate between different regions of the distribution effectively which should solve the high correlation problem encountered when using algorithm 1. Inspired from [GRV21], we propose an adaptation of algorithm 1 using Normalizing Flows proposals in algorithm 2.

Algorithm 2 Combining NF and local kernels

Require: K : the number of iterations, N : number of episodes, k_{loc} : number of local steps per NF resampling, t : number of iterations before training.

Initialise Agent π with parameters θ

Initialise map T and its base distribution ρ_b

$\theta_0 \sim \mathcal{N}(\mathbf{0}, \sigma_p^2 \mathbf{I}_D)$

for k from 0 to K **do**

if $k \bmod k_{\text{loc}} + 1 = 0$ **then**

$\theta'_b \sim \rho_b$

$\theta' = T(\theta'_b)$

 Run N episodes with π_{θ_k} and compute $\bar{U}_N(\theta_k)$

 Run N episodes with $\pi_{\theta'}$ and compute $\bar{U}_N(\theta')$

$\beta \leftarrow \frac{f(\theta') \bar{U}_N(\theta') \hat{\rho}(\theta_k)}{f(\theta_k) \bar{U}_N(\theta_k) \hat{\rho}(\theta')}$

else

$\theta' \sim \mathcal{N}(\theta_k, \sigma_p^2 \mathbf{I}_D)$

 Run N episodes with π_{θ_k} and compute $\bar{U}_N(\theta_k)$

 Run N episodes with $\pi_{\theta'}$ and compute $\bar{U}_N(\theta')$

$\beta \leftarrow \frac{f(\theta') \bar{U}_N(\theta')}{f(\theta_k) \bar{U}_N(\theta_k)}$

end if

$\alpha \leftarrow \min(1, \beta)$

$\epsilon \sim \mathcal{U}_{[0,1]}$

if $\epsilon < \alpha$ **then**

$\theta_{k+1} \leftarrow \theta'$

else

$\theta_{k+1} \leftarrow \theta_k$

end if

if $k \bmod t + 1 = 0$ **then**

 Update T using $D_{KL}(\hat{\rho} || \rho_*)$

end if

end for

return $\{\theta_k\}_{k=0}^K$

While this approach allowed to solve gridworld and cliff environments we still observed the same high correlation rates within θ_i . Moreover, the “mode collapse” was even more obvious in those two environments as the sampled θ_i were always choosing the same path to solve them. There

are several reasons explaining the “mode collapse”. As for the approximate inference method, we used the RKL as there was no available sample from ρ_* . However, it supposes that there is a non negligible probability that the algorithm discovers relevant regions in ρ_* through sampling with $\hat{\rho}$. In practice this is unlikely.

In [GRV21], the authors insist on the fact that the success of this method relies on some a priori information about the modes of the target distribution that has to be known in advance to initialise the chains. This method cannot find θ_i in regions different from those of the initialisation. One of the main difference between algorithm 2 and the one proposed in [GRV21] is that we start with a unique θ_0 while in [GRV21], they start with many different θ_i in parallel processes, each one localised in a different region for which there exist an a priori information. This approach is unfortunately not possible in our case when there is no such a priori information available.

Curiosity Model

The aim is now to force the model to better explore the parameter space and avoid the mode collapse. This situation mainly happens because of the use of the RKL, leading to a mode seeking behaviour, fitting a unique mode. An interesting question is to know if it is possible to take advantage from the mode collapse by forcing the model to successively "collapse" on the different modes of the target distribution while forcing it to collapse solely on modes that correspond to different behaviours.

Inspired from the multiple works on curiosity models applied to reinforcement learning [Bur+19; Eys+19; Gro+21; Pat+17], we propose to add a curiosity driven mechanism that drives exploration dynamically. It takes the form of a Neural Network, parameterised by weights ϕ , that learns to predict the next state given the last state of an agent. This way, if the network learns to predict the trajectory of θ_k and θ' tries something new, it will fail and output a large prediction error noted \mathcal{L} , called the intrinsic reward. The intrinsic reward is then added to the extrinsic reward, i.e the reward returned by the environment:

$$R(\tau) = \mu \tilde{G}(\tau) + (1 - \mu) \mathcal{L}_\phi^{(k)}(\tau), \quad (3.9)$$

with $\mathcal{L}^{(k)}$ the prediction error (i.e. the loss) at step k and $\mu \in [0, 1]$. The prediction error can take the following form:

$$\epsilon = \sum_{t=1}^T d(s_t^\theta, \hat{s}_t^\phi | s_{t-1}^\theta)$$

Where d is an arbitrary distance on the state space, s_t^θ the state of the agent at timestep t and $(\hat{s}_t^\phi | s_{t-1}^\theta)$ is the predicted state at timestep t given the last known state of the agent.

We can also define a new energy function:

$$\bar{U}(\tau_i) = \exp\{\mu \tilde{G}(\tau_i) + (1 - \mu) \mathcal{L}_\phi(\tau_i)\}. \quad (3.10)$$

Due to the dependence on ϕ this energy function cannot be used directly in the expression of ρ_* . Indeed, depending on ϕ , the distribution ρ_* will shift, making convergence impossible. However,

denote $\bar{\rho}$ the following distribution:

$$\bar{\rho}_\phi = Z_*^{-1} e^{-\bar{U}_\phi(x)} \quad (3.11)$$

This distribution can be used as a target distribution to update the Normalizing flows mapping T . Indeed, when ϕ is fixed, the prediction error is high for trajectories that were never encountered before and a mode corresponding to the behaviour maximising both the return and the prediction error appears on $\bar{\rho}_\phi$. When calculating $D_{KL}(\hat{\rho}||\bar{\rho}_\phi)$, the RKL makes T to fit that particular mode, accelerating convergence towards that mode. At convergence, the prediction error becomes low again until a new θ inducing high prediction error is sampled.

Surprisingly this simple addition does not lead to an alternation between phases of high or low rejection rates, but it comes at the price of a less stable performance. As shown in figure 3.7, the performance sometimes drops massively when moving to θ located in the region of a new mode while the acceptance rate remains steady.

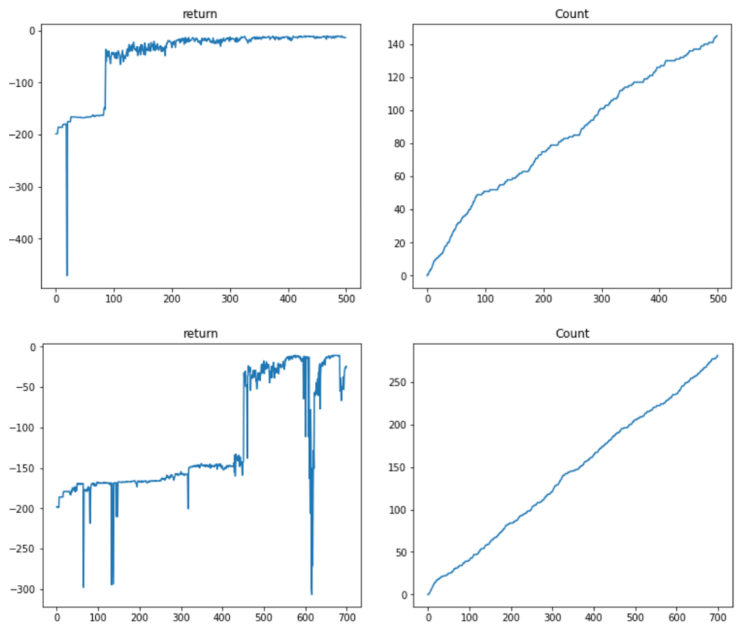


Figure 3.7: CAMEO Results on Cliff (above) and Gridworld (below). The figure presents the mean return and the count of θ_i retained over time steps.

As shown in figure 3.7, the algorithm is able to output policies that solve the tasks consistently. The count panel also proves that once a succeeding θ is found, the algorithm is not stuck at this θ rejecting all the others, but keeps finding new succeeding ones.

Figure 3.8 shows that the θ_i retained are less correlated than in previous implementations, which suggests that the curiosity module and the prior are effective.

However non correlated weights do not necessarily imply a different behaviour. Figure 3.9 shows the aggregated state visitation frequency of 100 different policies that solve the problems. The most efficient (shortest) paths are the most taken but the state visitation frequencies are non

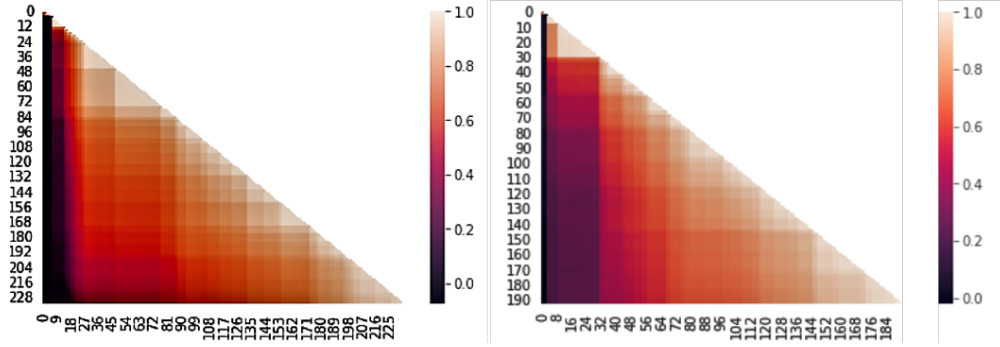


Figure 3.8: Cosine similarity between pairs of retained θ_i on Gridworld (left) and Cliff (right) using CAMEO implementation.

negligible for other paths. Therefore, the learned policies effectively correspond to different behaviours. Moreover, on the gridworld environment, on 222 succeeding policies found, our model found 60 different behaviours. Of course on such a small environment, many obtained policies are inefficient (i.e multiplying goings and coming between the same states before reaching the final state); however as displayed in figure 3.10, the top 5 trajectories are efficient and highly represented.

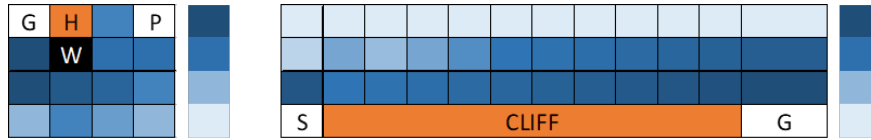


Figure 3.9: State visitation frequency aggregated on 100 policies obtained using CAMEO on Gridworld and Cliff. Less visited states are in light blue and most visited ones in dark shade.

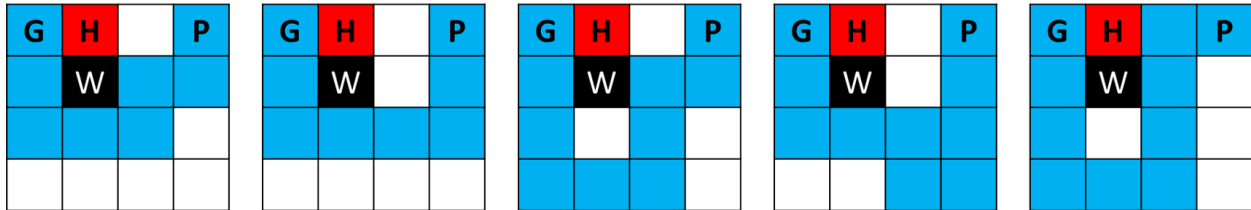


Figure 3.10: Top 5 most represented trajectories output by CAMEO for Gridworld. From the most represented (left) to the fifth most represented (right). On 222 policies, the most represented trajectory appears 38.7% of the time (86 times), then 10.8%, 5.9%, 5.4%, 3.1% for the other trajectories respectively.

The full procedure is detailed in algorithm 3 while figure 3.16 details the framework.

We also compare our results to those obtained using DQN. In figure 3.11, we show that using DQN (with ϵ set at 0, i.e. there is no exploration or random action), we obtain consistent results as for our model. However, the obtained θ_i are highly correlated (figure 3.12). This is of course not surprising; DQN is designed to converge towards a unique optimal policy as discussed in the introduction of this chapter. Near convergence, the updates are not significant. We also display

the trajectory obtained at convergence on the gridworld and cliff environments (figure 3.13). In both cases, DQN finds an optimal path but uses the same one consistently.

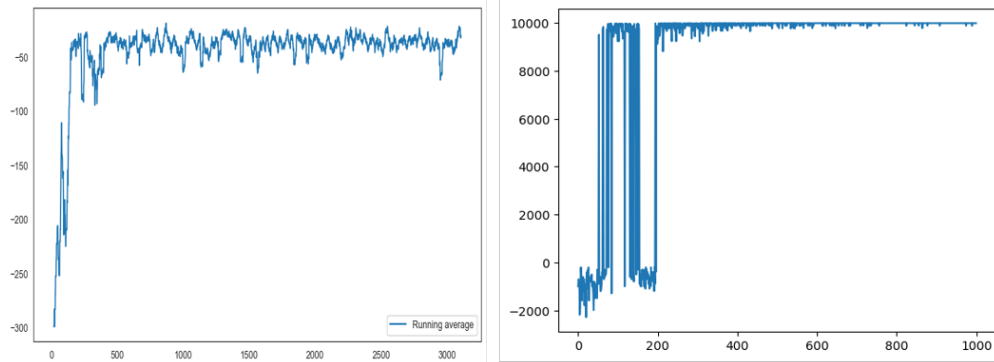


Figure 3.11: Mean return of DQN on Gridworld (left) and Cliff (right) Environments. The mean is calculated using a sliding window over 20 time steps.

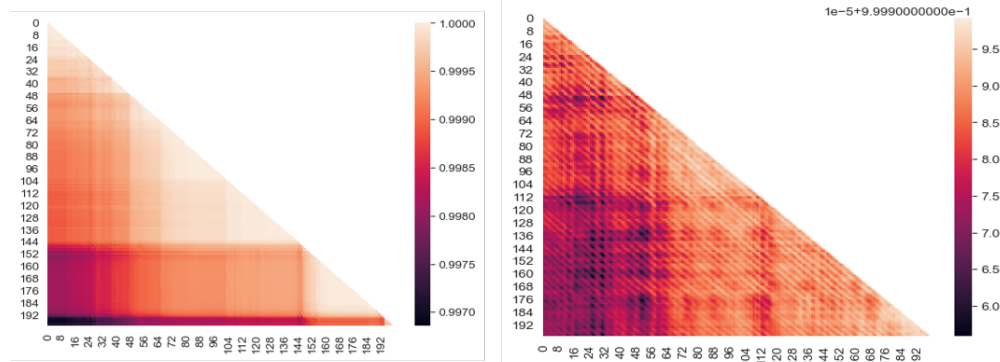


Figure 3.12: Cosine similarity between pairs of θ_i on Gridworld (left) and Cliff (right) using DQN. Considered θ_i are extremely correlated, we restricted the scale to a narrow range in order to show that the θ_i are not all equal.

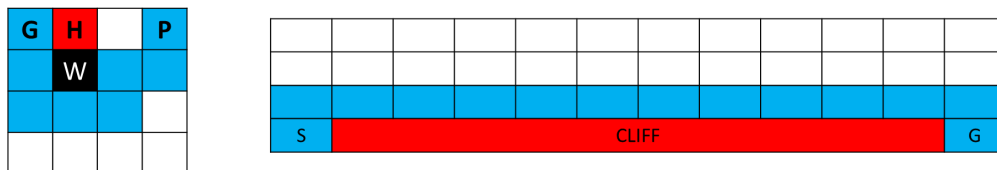


Figure 3.13: Trajectories obtained using DQN on Gridworld and Cliff environments. For each environment, DQN converged towards the unique trajectory displayed.

In order to assess better the versatility of our model compared to DQN, we also tested the trajectories output by DQN using different ϵ values at inference as in the ϵ -greedy approach. The model has been trained to convergence, then during inference, it is tested with a range of ϵ values to force the algorithm to explore other states. Figure 3.14 presents the mean performance (on 50 episodes) of DQN on the cliff environment for different values of $\epsilon \in [0, 1]$. We observe

that performance decreases significantly when $\epsilon > 0.6$. In figure, we present the state visitation frequencies for different values of ϵ on the cliff environment.

The performance drop shown in figure 3.14 contrasts with the consistent performance showcased in figure 3.7 when using CAMEO. This is due to the fact that the output trajectories become highly inefficient when ϵ increases. This also shows up in figure 3.15 where we display the state visitation frequency over 50 episodes of the DQN algorithm in cliff environment for different values of ϵ . We observe that for $\epsilon > 0.6$, the model fails at finding the goal state. For $\epsilon = 0.6$, we observe that the first states (on the left hand side of the cliff) are the most visited while for our algorithm (figure 3.9) the most visited states are the initial state and the ones near the final as all succeeding agents have to go through those. This demonstrates that for DQN, when $\epsilon = 0.6$, the output trajectories are highly inefficient while our algorithm outputs efficient succeeding policies.

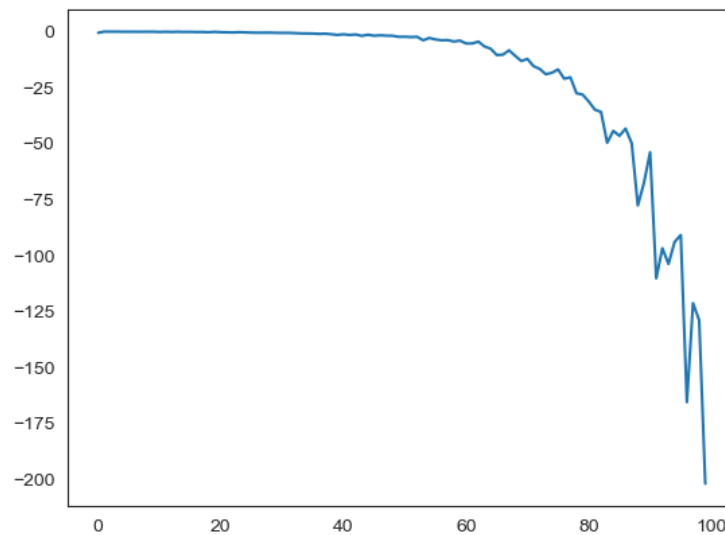


Figure 3.14: Mean return of DQN on cliff environment for different values of ϵ . The model has been trained to convergence, then during inference, it is tested with a range of ϵ values to force the algorithm to explore other states.

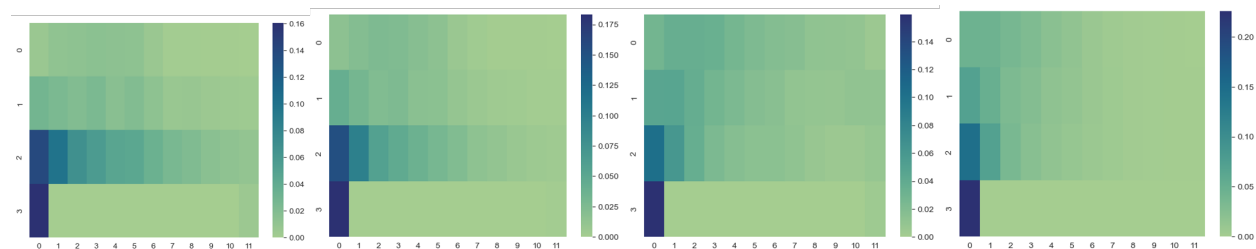


Figure 3.15: State visitation frequencies over 50 episodes using DQN with different values of ϵ on the Cliff environment. From left to right, the used values for ϵ are 0.6, 0.7, 0.8 and 0.9.

Algorithm 3 CAMEO

Require: K : the number of iterations, N : number of episodes, k_{loc} : number of local steps per NF resampling, t : number of iterations before training T

Initialise Agent π with parameters θ

Initialise map T and its base distribution ρ_b

Initialise curiosity model Φ with parameters ϕ

$\theta_0 \sim \mathcal{N}(\mathbf{0}, \sigma_p^2 \mathbf{I}_D)$

for k from 1 to K **do**

if $k \bmod k_{\text{loc}} + 1 = 0$ **then**

$\theta'_b \sim \rho_b$

$\theta' = T(\theta'_b)$

 Run N episodes with π_{θ_k} and compute $\bar{U}_N(\theta_k)$

 Run N episodes with $\pi_{\theta'}$, compute $\bar{U}_N(\theta')$ and store trajectories $\tau_{\theta'}^{(i)}$

$\beta \leftarrow \frac{f(\theta') \bar{U}_N(\theta') \hat{\rho}(\theta_k)}{f(\theta_k) \bar{U}_N(\theta_k) \hat{\rho}(\theta')}$

else

$\theta' \sim \mathcal{N}(\theta_k, \sigma_p^2 \mathbf{I}_D)$

 Run N episodes with $\pi_{\theta'}$, compute $\bar{U}_N(\theta')$ and store trajectories $\tau_{\theta'}^{(i)}$

 Run N episodes with π_{θ_k} and compute $\bar{U}_N(\theta_k)$

$\beta \leftarrow \frac{f(\theta') \bar{U}_N(\theta')}{f(\theta_k) \bar{U}_N(\theta_k)}$

end if

$\alpha \leftarrow \min(1, \beta)$

$\epsilon \sim \mathcal{U}_{[0,1]}$

if $\epsilon < \alpha$ **then**

$\theta_{k+1} \leftarrow \theta'$

else

$\theta_{k+1} \leftarrow \theta_k$

end if

for $i = 1 \dots N$ **do**

$\mathcal{L}(\tau_{\theta'}^{(i)}) \leftarrow d(\Phi(\tau_{\theta'}^{(i)}), \tau_{\theta'}^{(i)})$

$\mathcal{L}(\tau_{\theta'}) \leftarrow \mathcal{L}(\tau_{\theta'}) + \mathcal{L}(\tau_{\theta'}^{(i)})$

end for

 Train Φ using $\mathcal{L}(\tau_{\theta'})$

 Update T using $D_{KL}(\hat{\rho} || \bar{\rho}_\phi)$

end for

return $\{\theta_k\}_{k=1}^K$

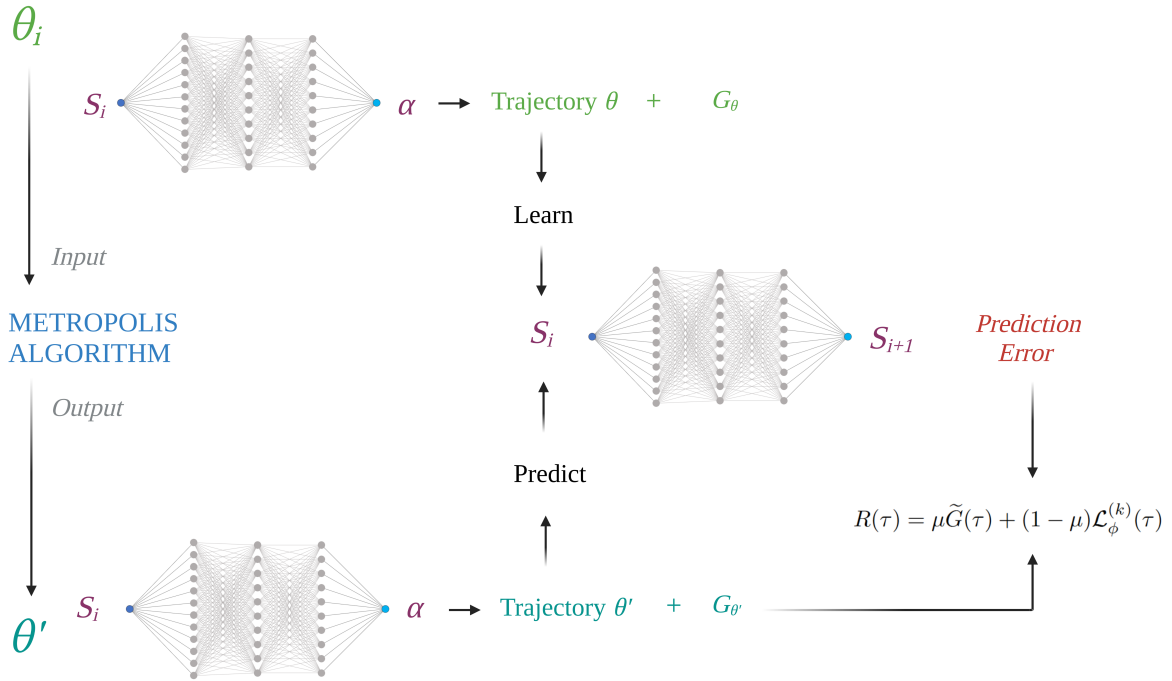


Figure 3.16: Illustration of CAMEO framework. Weights θ_i and θ' obtained from Metropolis algorithm are used to create trajectories θ and θ' . Trajectories θ are used to train a neural network parameterised with ϕ that predicts the next state given the current one. CAMEO criteria for Metropolis is a weighted average of the obtained returns using trajectories θ' and the prediction errors of model ϕ when used to predict trajectories θ' .

3.5 Conclusion

In this chapter, we proposed a MCMC based algorithm adapted to reinforcement learning. Using our algorithm, we are able to efficiently sample succeeding policies on the fly. Sampled policies successfully displayed diverse succeeding behaviours that remained efficient, i.e sub-optimal given the reward function but still near optimal. Also, when combined, the obtained behaviours exhibit a full state space coverage of the considered environments contrary to information theory based approaches. Moreover, our approach is successful even when the rewards structure is sparse. This is done by using a curiosity module that helps exploration dynamically. Our approach still bears some limitations as the policy spaces of studied environments are discrete. Moreover, adding the prediction error to the target distribution of the normalizing flows mapping criteria lacks theoretical convergence guaranties.

We also show that the resulting behaviour for each sampled policy is different while using the same reward function (base objective), illustrating the outer alignment problem (human vs base objectives). We argue that the difference in behaviours is due to a criteria uncaptured in the base objective but that can actually be considered as part of the mesa-objective. This criteria can be associated with risk. The next chapter will delve deeper in risk accounting in the mesa-objective by proposing a new distributional approach to reinforcement learning that allows for a better

inner alignment.

3.6 Appendix

3.6.1 More Details on the Link Between Generalisation, Overfitting and Distributional Shift

Existing deep RL algorithms aim at finding the optimal policy by parameterising a policy π as a neural network with weights θ and find an optimal policy called π_{θ^*} that maximise the Q-value for all (s, a) . This means that these algorithms converge by design towards a unique policy considered as optimal. However, as mentioned in the introduction of this chapter, an optimal policy may not be unique. In a given environment, and for a given reward function, different algorithms can converge to different policies that solve the task at hand. This difference is induced by different implicitly defined mesa-objectives. In the introduction chapter we also explained that the misalignment between the base and mesa-objectives is mainly due to distributional shift, the fact that the environment where the agent has been trained is significantly different from the one where it is effectively deployed.

Distributional shift is closely related to the concepts of generalisation and overfitting in machine learning. Generalisation, in the context of machine learning, refers to a model's ability to perform well on new, unseen data. A well-generalised model can effectively capture underlying patterns from the training data and apply that knowledge to make accurate predictions or classifications on unseen examples. A model that generalises well indicates a lower risk of making errors when dealing with new, real-world data.

On the other hand, overfitting occurs when a model learns not only the underlying patterns in the training data but also the noise and randomness present in it. An overfitted model performs exceptionally well on the training data but fails to generalise to new, unseen data. In this case, the risk emerges from the model's inability to generalise its knowledge and make accurate predictions on unfamiliar examples. This elevated sensitivity to noise increases the risk of making inaccurate or unreliable predictions in real-world scenarios.

Achieving an optimal balance between generalisation and overfitting is crucial to managing risk in machine learning models. While we aim to create models that perform well on training data (without overfitting), the primary objective is to have models that can generalise effectively to new, unseen data, minimising the risk of incorrect predictions or classifications in practical applications.

However, is overfitting always bad news ? In a RL context, when an environment is clearly defined like the cartpole environment described earlier with a small enough state space, there is no point into seeking generalisation, as overfitting on the environment is not prejudicial. Indeed, the agent will never face a different environment or a state that is highly different from what it encounters during training. This would not be the case in robotic applications for instance where an agent can be trained in simulated environments then deployed in real scenarios. Therefore, **the balance between overfitting and generalisation boils down to the user's confidence about the risk of distributional shift.** If the user estimates that this scenario is improbable then it makes

sense to seek for maximum performance through overfitting and take the risk of distributional shift.

Based on the observation, that the mesa-objective is highly related to distributional shift via the inner mis-alignment problem (base objective vs mesa-objective), we make the hypothesis that the mesa-objective can be interpreted as a risk measure of distributional shift. This interpretation can even offer an explanation to the existence of the inner mis-alignment issue. Indeed, as the primary aim of any RL algorithm is to maximise the base objective, this can lead to overfitting; however the mesa-objective seen as a distributional shift measure to minimise can contradict the base objective for the sake of better generalisation and hinder too much the agent's performance if the model is too much risk averse when it is not necessary.

We argue that the regularisations used in different RL algorithms consist on different implicitly defined mesa-objectives that can be considered as risk measures that depict the confidence of the user towards the existence of distributional shift. Indeed, optimising the expected value of the return alone may not be satisfactory in certain scenarios where incorporating the notion of risk is important. It is often desirable to consider risk in the optimisation problem formulation, either as part of the objective or as a constraint. For example, in financial investments, the primary objective is typically to maximise expected returns. However, decision-makers often want to account for the "risk" associated with investments, which involves mitigating potential downside losses. This can be better illustrated through the Markowitz model.

3.6.2 The Markowitz Model

The Markowitz model, introduced by Harry Markowitz in 1952, is a finance-based portfolio optimization approach. It aids in identifying the most efficient portfolio by evaluating different combinations of securities. This model demonstrates that by selecting securities with non-identical movements, investors can lower their risk. An efficient portfolio maximizes returns for a specified risk level or minimizes risk for a given level of return. Investors, following this model, choose portfolios based on two principles:

- preferring lower-risk portfolios among those with similar returns
- favouring higher-return portfolios among those with the same risk level.

In figure 3.17, the shaded area PVWP includes all the possible securities an investor can invest in. The efficient portfolios are the ones that lie on the boundary of PQVW. For example, at risk level X_2 , there are three portfolios S, T, U. But portfolio S is called the efficient portfolio as it has the highest return, Y_2 , compared to T and U. All the portfolios that lie on the boundary of PQVW are efficient portfolios for a given risk level.

The boundary PQVW is called the Efficient Frontier. Portfolios located below this frontier are deemed suboptimal since they offer lower returns for a given level of risk. Portfolios positioned to the right of the Efficient Frontier are also considered suboptimal because they entail higher risk for a given rate of return. Portfolios situated precisely on the boundary of PQVW are termed Efficient Portfolios. Notably, the Efficient Frontier remains consistent for all investors, as it represents the ideal balance sought by every investor: maximum return with the lowest achievable

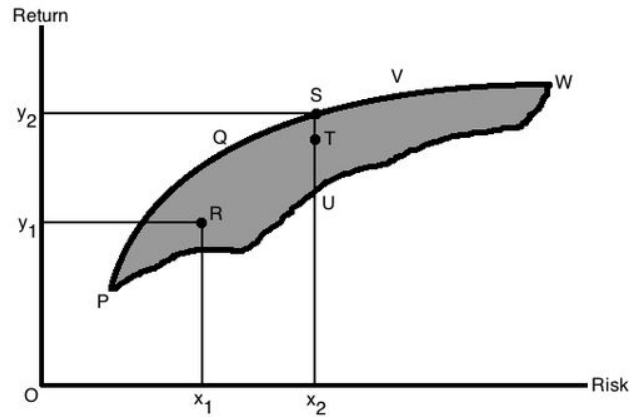


Figure 3.17: Risk-return of possible portfolios. Source:https://en.wikipedia.org/wiki/Markowitz_model

risk. This consistency arises from investors' shared preference for minimizing risk due to their risk-averse nature.

In the RL context, we can replace the portfolios by policies and the more a policy overfits, the more it is risky. So two policies are said to be equivalent if and only if a risk measure associated with each one of them have a similar value. Moreover, from now on we will refer to succeeding policies instead of optimal policies. Optimal policies are usually considered in the literature as the policies that ensure the highest return without necessarily taking into account the risk of distributional shift. On the other hand succeeding policies are policies that lie in the efficient frontier, i.e. policies that succeed at the task at hand but may not give the maximum return in the given environment.

There exist an extensive literature on risk quantification. Various risk measures have been proposed to address this need. These measures include exponential utility, variance, percentile performance, chance constraints, value at risk, and conditional value-at-risk [AF22]. Each of these risk measures offers a different perspective on quantifying and managing risk, allowing decision-makers to take into consideration different aspects of the potential downside. By incorporating risk measures into the optimisation problem, decision-makers can make more informed and well-balanced decisions that align with their risk preferences and objectives.

In our case, we consider that we do not have any information on the desired risk profile or the risk measure to optimise. We propose an algorithm that outputs succeeding policies that lie on the efficient frontier, displaying different behaviours such that the user can choose which policy suits the best his preferred risk profile. An important tool towards this goal is the optimal set-policy that is defined in the next section.

3.6.3 More on Set Policies

Theorem 1: Optimal Policy Characterisation

For a given MDP, a policy π is said optimal if and only if:

$$\begin{aligned} V_r^\pi = V_r^* &\iff \forall s \in \mathcal{S}, \quad \text{Supp}(\pi(\cdot|s)) \subset \arg \max_a [Q_r^\pi(s, a)] \\ &\iff \forall s \in \mathcal{S}, \quad \text{Supp}(\pi(\cdot|s)) \subset \arg \max_a [Q_r^*(s, a)] \end{aligned} \tag{3.12}$$

Therefore a policy is optimal if and only if for any state s , it chooses an action in the set $\arg \max_a [Q_r^*(s, a)]$. Hence, $\text{Supp}(\pi(\cdot|s))_{s \in \mathcal{S}}$, a finite set of actions, is sufficient to characterise a policy. In [PGP17] the authors define functions associating a state to a non empty and finite set of actions called set-policies.

Chapter 4

Improving Distributional RL Using Invertible Generative Models

Existing distributional Reinforcement Learning approaches use either the KL divergence or the Wasserstein distance as loss functions. The KL divergence is not scale sensitive. The Wasserstein distance does not have unbiased sample gradients which makes it impossible to optimise using classic stochastic gradient methods. Also, existing approaches tend to implicitly learn target distributions making them unable to evaluate the probabilities associated with specific return values under a given policy. We propose a novel distributional RL approach based on an invertible generative model, namely Normalizing Flows (providing return densities), along with the Cramèr distance (to facilitate robust convergence guarantees).

4.1 Introduction

In chapter 3 we showed that, given the same reward function (what we also call the base objective), several succeeding policies¹ exist that solve the task while adopting different behaviours. We also argued that the difference in behaviours is due to a criteria uncaptured in the base objective but that can actually be considered as part of the mesa-objective². We considered the mesa-objective as a regularisation that drives convergence towards a unique solution and hence a certain behaviour. This behaviour can be more or less prone to overfitting. For instance, a robot that has only been trained to walk on flat surfaces may have trouble walking in steep slopes. Such situation is called distributional shift. In chapter 3 section 3.6.1, based on the observation that the mesa-objective is highly related to distributional shift via the inner mis-alignment problem (base objective vs mesa-objective), we claimed that the mesa-objective can be interpreted as a risk measure of distributional shift.

Here, risk refers to the uncertainty over possible outcomes. In order to be more robust to the risk of distributional shift, we argue that it would be useful to quantify the uncertainty over returns. For instance, the higher is the return distribution entropy, the higher is the risk of distributional

¹policies that are acceptably close to optimal by human standards

²an additional objective that constraints the agent to converge towards a given behaviour

shift. In [Dab+18], the authors show that distributional RL offers an ideal backbone to implement risk-sensitive policies. Using information provided by the distribution over returns, they expanded the class of learnt policies to the class of risk-sensitive ones. Based on their results, we can safely assume that a distributional approach of RL is well suited to take risk into account. An in depth introduction to Distributional RL is given in chapter 2 section 2.2.

In this chapter, rather than implementing policies that take into account the risk on distributional shift, we will first expose current distributional approaches limitations and try to mitigate them. Indeed, the Distributional RL methods presented in chapter 2 section 2.2 have certain limitations:

- They use either the KL divergence or the Wasserstein distance as loss functions. The KL divergence is not scale sensitive. The Wasserstein distance does not have unbiased sample gradients which makes it impossible to optimise using classic stochastic gradient methods.
- Additionally, these methods tend to implicitly learn target distributions, making it challenging to evaluate the probabilities associated with specific return values under a given policy.

Our objective is to overcome these limitations. To do so, we introduce a novel distributional RL approach that leverages an invertible generative model along with the Cramèr distance.

- In contrast with most recent approaches, our model explicitly learns target distributions, allowing simple evaluation of the probabilities associated with specific return values under a given policy.
- Moreover using our proposed invertible model, it is possible to easily calculate the Cramèr distance, which offers robust convergence guarantees.

Our model is a necessary pre-requisite and a crucial building block of one of the main objectives of this manuscript (chapter 1 section 1.2.2): proposing an IRL model that is more robust to outer/inner mis-alignment (i.e. proposing a reward function that better matches human objectives and that is robust to distributional shift). This matter will be developed in chapter 5.

We test our approach and algorithm on various settings of the Frozen Lake environment and we showcase the effectiveness of our model in accurately predicting return distributions and capturing all their potential modes. Obviously, existing algorithms also succeed in such a simple environment, but Frozen Lake was used to qualitatively assess the theoretical contributions of this work. To further validate and evaluate its performance, the next step involves testing it in more complex environments.

In section 4.2, we detail the limitations of existing Distributional RL approaches (return distributions limited to fixed support, using loss function with no convergence guarantee) and present our contributions. In section 4.3, we detail how Normalizing Flows constitute a valuable tool for learning return distributions (as it allows to compute the density of any return value for any state-action pair) and present our model’s architecture. We also introduce one of the main novelties of this work which is a method to build the target distribution using a flow function. Then in section 4.4, we discuss the loss function to use. We first show that the Reverse KL divergence and the Wasserstein distance are suboptimal then we introduce the Cramèr distance as an ideal solution offering convergence guarantees. Finally, in section 4.5, we test our approach on various

instances of simple MDPs and Frozen lake environments, illustrating the qualitative advantages of our model. We also compare its performance with the C51 algorithm and show that they are on par on the tested environment.

4.2 Related Work and Contributions

In this section we will first explain why traditional RL (as opposed to Distributional RL) is not suited for risk management. Then we will delve deeper on existing Distributional approaches limitations; and finally we will detail our contributions to solve these limitations.

4.2.1 Existing Approaches are not Adapted for Risk Management

In various scenarios, such as financial investments, investors necessitate the simultaneous consideration of maximising expected returns while mitigating potential downside losses. To achieve this, investors need to learn optimal policies exhibiting different observable behaviours, ensuring the optimisation of the same expected return while accommodating diverse risk constraints. The expected return to be maximised can be defined relative to each state $s \in \mathcal{S}$ through the state value:

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s \right] \quad \forall s \in \mathcal{S}$$

with $0 \leq \gamma \leq 1$. The same way we can define the Q-value as:

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right] \quad \forall s \in \mathcal{S}$$

Traditional approaches are not well-suited for incorporating risk considerations. Indeed, as discussed in chapter 2 section 2.2, a single value of $V(s_0)$ can correspond to an infinite number of situations. This is due to the fact that state values (as well as Q-values) are defined as expectations, lacking detailed information about the environment dynamics and the range of possible rewards. Here, “risk” pertains to the uncertainty associated with potential outcomes. It is crucial to distinguish between intrinsic uncertainty, captured by the distribution over returns, and parametric uncertainty, which pertains to the uncertainty surrounding the estimation of values. In this research, we aim to focus on capturing the former. A more informative representation of state values or Q-values would be to model them as distributions to gain a more comprehensive understanding of the underlying uncertainties and variability in the rewards.

Considering Q-values as distributions and trying to learn these distributions is the main objective of a sub-field of RL called Distributional Reinforcement Learning. A review of the main contributions of the area as well as the necessary tools on which this approach is based are described in chapter 2 section 2.2.

4.2.2 Limitations of Existing Approaches

While most recent Distributional RL approaches have achieved great empirical success [Dab+18], we highlight here some of the main limits of the existing approaches.

C51 [BDM17] is limited to discrete and fixed supports for the distribution of returns while target and predicted distributions do not share the same support by construction, requiring a computationally heavy projection of the target support into the predicted one. More importantly, while several important theoretical results have been proven for Distributional RL using the Wasserstein distance, the C51 algorithm does not use the Wasserstein distance as a loss but the KL divergence between the predicted and target distributions. Indeed, the authors have shown that their algorithm cannot be trained with the Wasserstein loss using only samples from the target distribution (proposition in section 4.4.3).

With QR-DQN [Dab+17], the creators of C51 overcame this limitation using an algorithm that is basically the opposite of C51. Indeed, QR-DQN learns quantile values of the return distribution at fixed locations. They show that learning quantile values through quantile regression for unbiased stochastic approximation of the quantile function is possible, even when using the Wasserstein distance as a loss.

IQN [Dab+18] extended this approach by learning the full quantile function, a continuous map from probabilities to returns. However the authors admit that quantile based approaches lacks necessary additional theoretical analysis in many areas. For instance, sample based RL convergence has not been proven for Quantile Regression based algorithms. Moreover, the contraction results shown in [Dab+17] for fixed grids of quantiles have not been extended to the approximate quantiles functions used in IQN. Finally, by modelling the quantile functions, it is not possible using IQN to compute the probability of any return value given a state-action pair. Indeed, the distributions are only learnt implicitly which makes them hard to visualise.

4.2.3 Contributions

Even though using quantile functions brought several advantages like allowing a direct computation of the Wasserstein distance between distributions, we notice that their usage have also brought a lot of theoretical questions and uncertainties. Other than for the computation of the Wasserstein distance, using quantile functions mainly allowed to use continuous supports for the distributions of returns (although they are implicitly defined). Our work will try to keep these benefits (i.e. simple computation of a proper distance and continuous and unbounded support) while avoiding the use of quantile functions and keeping the theoretical convergence guaranties shown in [Row+18]. Moreover, our model will also allow to compute the density of any return value given a state-action pair under a defined policy. This last property will offer a significant advantage over existing approaches as detailed in section 4.5, as multimodal return distributions are particularly evident in critical states where single actions can lead to opposite outcomes (fail or success).

To articulate the goals of this work more precisely, we aim to develop a Distributional Reinforcement Learning model that offers the following capabilities:

- To learn the distribution of returns given any state-action pair.

- To sample returns from that distribution.
- To compute the density of any given return value for any state-action pair
- Is not limited to bounded and discrete supports
- To offer simple computation of a proper distance between distributions or an Integral Probability Metric
- To offer robust convergence guarantees

4.3 Our Approach: Normalizing Flows for Distributional RL

In this section we will present the architecture of our Normalizing Flows (NF) based model. We will first explain in section 4.3.1 the issues of learning a CDF or its inverse as in existing methods and propose to use NF to get the best of both worlds. A NF is a mapping between a base distribution and a target one. The flow can usually be learnt either from the base distribution to the target or from the target to the base. In section 4.3.2, we expose both ways and detail why the flow direction is constrained by the RL problem structure. Once the flow direction is fixed, we present our model’s architecture in section 4.3.3. Finally, in section 4.3.5, we detail how to build the target distribution towards which our model should converge.

4.3.1 NF as an Alternative to Learning the CDF or Quantile Function of the Returns Distribution

Being able to draw samples from given distributions and compute their densities is essential to implement risk sensitive policies. Existing approaches rely on either learning the CDF of the returns distribution or its quantile function. However using CDF or quantile function makes it hard to draw samples and evaluate their densities easily and in the same time. Indeed, learning the CDF enables the computation of densities but doesn’t facilitate sampling while learning quantile functions enables sampling but doesn’t support density computations.

The CDF of a real-valued random variable X is the function given by:

$$F_X(x) = P(X \leq x)$$

$$P(a < X \leq b) = F_X(b) - F_X(a)$$

where x , a and b are outcomes of X .

Considering the example of financial investment, the CDF allows to evaluate the probability that the return remains in an acceptable area or remains higher than a certain threshold. In this example, the random variable X is replaced by the random variable of the returns Z and the outcome x is replaced by the return R . A control objective would aim to minimise $F_Z(R)$ or maximise $1 - F_Z(R)$.

However, it is not straightforward to sample from an arbitrary distribution given its CDF. IQN does so by learning quantile functions instead, which are the inverse of CDF. Let X be a random

variable and F its CDF, the quantile function is defined as:

$$Q(p) = F^{-1}(q) = \inf\{x : F(x) \geq q\} \quad \forall q \in [0, 1]$$

This means that for a given state-action pair, it is possible to compute the highest possible return achievable with a fixed probability. As illustrated in figure 4.1, if $x \sim \mathcal{U}[0, 1]$, then $F^{-1}(x)$ is a sample from the distribution whose CDF is F . This method is known as inverse transform sampling. Therefore, having the quantile function allows to sample from the distribution defined by the corresponding CDF. Nevertheless, employing this method does not provide a straightforward way to calculate the probability associated with a specific return value.

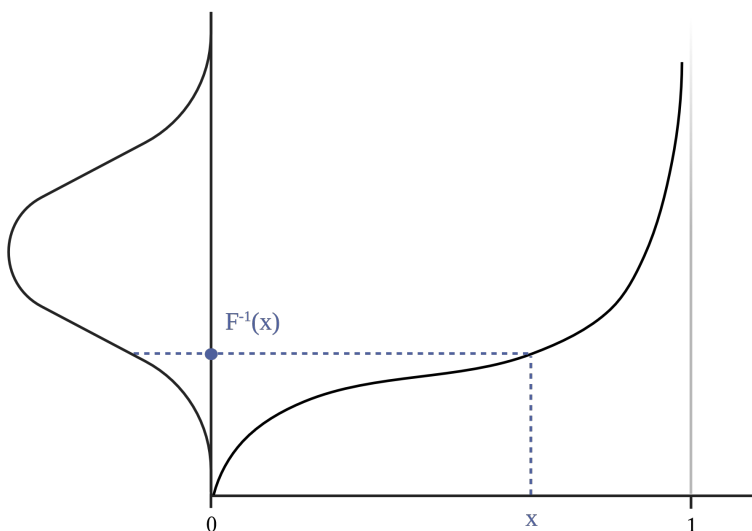


Figure 4.1: An illustration of the inverse transform sampling method. We first draw $x \sim \mathcal{U}[0, 1]$ and compute $F^{-1}(x)$; F^{-1} being a quantile function. $F^{-1}(x)$ is actually a sample from the distribution which CDF is F .

Therefore, learning the CDF enables the computation of densities but does not facilitate sampling. Conversely, learning quantile functions enables sampling but does not support density computations. Additionally, it is worth mentioning that approximating the inverse CDF is often feasible through various techniques, albeit at the cost of significant computational time. We propose to use invertible models like Normalizing Flows, such that it becomes possible to get the best of both worlds – models that can compute the density of any specified return for a given state-action pair while also enabling the sampling of returns from the associated distribution.

4.3.2 Going from Base to Target Distribution to Jointly Sample Returns and Predict their Density

A NF is a mapping **between** a base distribution and a target one. The flow can usually be learnt either from the base distribution to the target or from the target to the base. In this section we

will detail the pros and cons of each direction, then we justify our choice of going from the base distribution to the target one.

It will be useful to introduce some notation at this point. As in previous chapters (see for instance chapter 2 section 2.1.1), consider a Markov decision process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma, \zeta_0)$. Let Z^π be a random variable in \mathcal{Z} , the space of random variables with bounded moments, corresponding to random returns obtained when following the policy π . Its expectation is the Q-value. We have $Z^\pi(s, a) \sim \mathcal{P}_\pi(\cdot|s, a)$ the distribution of returns given the state action pair (s, a) if the policy π is followed afterwards. Let R^π be an outcome of Z^π . The Q-value being the mean of the random returns Z^π , we have:

$$Q^\pi(s, a) = \mathbb{E}_{(s,a)} Z^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a \right]$$

$$s_t \sim \mathcal{T}(\cdot|s_{t-1}, a_{t-1}), a_t \sim \pi(\cdot|s_t)$$

Let \mathcal{U} be the base distribution. We consider that $z \sim \mathcal{U}$ and $R^\pi \sim \mathcal{P}_\pi(\cdot|s, a)$. Let f be a diffeomorphism between \mathcal{P} and \mathcal{U} (an invertible function that maps differentiable manifold \mathcal{P} to differentiable manifold \mathcal{U} such that both f and f^{-1} are differentiable).

The flow f can be implemented in two different ways:

- either the flow goes from the observation $R^\pi \sim \mathcal{P}_\pi(\cdot|s, a)$ to the base distribution: $z = f(R^\pi)$
- or we first sample $z \sim \mathcal{U}$ and get its image in the target distribution $R^\pi \sim \mathcal{P}_\pi(\cdot|s, a)$, using the flow $R^\pi = f(z)$.

Each approach has its pros and cons:

- In the first case it is easy to calculate the density of an observed R^π but sampling is harder because we would need to compute f^{-1}
- In the latter case it is easy to sample from the target distribution $\mathcal{P}_\pi(\cdot|s, a)$ but it is more difficult to get the density of a given sample R^π from this target distribution.

The primary aim of Distributional RL is to sample return values that have high probability of occurring. Therefore sampling is more important than being able to compute the density of a priori given value of R . This means that the two flow directions mentioned above are not equivalent.

- Fixing $f(R^\pi) = z$ would force us to select fixed values of R^π beforehand for each state action pair and then compute their density. This implies the same issue as in QR-DQN; what are the right R^π values to choose beforehand in order to evaluate their density ?
- $R^\pi = f(z)$ is natural in a RL context. Indeed, fixing $R^\pi = f(z)$ enables sampling values $R^\pi \sim \mathcal{P}_\pi(\cdot|s, a)$ and getting their predicted density in the same time. Indeed, in one hand, it is easy to sample z values from the base distribution, on the other hand, the flow will naturally map sampled z values to return values R^π that lie in high density areas.

While choosing $R^\pi = f(z)$ makes sampling easy, it is also possible to compute the density of samples return values as follows. Let f_θ be a flow function parameterised by θ , $\mathcal{P}_\pi(R|s, a)$ the distribution of the return given (s, a) and p_z the base distribution:

$$\begin{aligned}
\mathcal{P}_\pi(R|s, a)\partial R &= p_z(z)\partial z \\
\mathcal{P}_\pi(R|s, a) &= p_z(z)\frac{\partial z}{\partial R} \\
&= p_z(z)\frac{\partial z}{\partial R} \\
&= p_z(z)\left(\frac{\partial R}{\partial z}\right)^{-1} \\
&= p_z(z)\left(\frac{\partial f_\theta(z)}{\partial z}\right)^{-1} \\
\log \mathcal{P}_\pi(R|s, a) &= \log p_z(z) - \log\left(\frac{\partial f_\theta(z)}{\partial z}\right)
\end{aligned} \tag{4.1}$$

The log density of sampled return values only require the base distribution density and the derivative of the flow function. Therefore, choosing a flow function with easily computable derivatives is crucial.

4.3.3 An Architecture Using CDF as a Valid, Flexible and Easily Invertible Flow Function for Simple Density Computation

Now that the flow direction is fixed, we will decide the type of flow function to use. As mentioned in 4.3.2, the choice of the flow function parameterisation is crucial as the flow has to be a valid diffeomorphism and its determinant tractable.

The base distribution and the target distribution must possess the same dimensionality. In our specific scenario, the distribution of returns is one-dimensional, thus requiring a one-dimensional base distribution. Given our chosen formulation, where returns are expressed as a function of a latent variable, denoted as $R^\pi = f(z)$, and the base distribution is transformed into the predicted distribution, a viable approach involves learning a flexible flow function, parameterised by θ , that depends on the state. This enables us to accommodate the state-dependent nature of the problem and align the dimensions of the base and target distributions effectively as illustrated in figure 4.2.

To do so, we propose to establish the flow function from a specific family and introduce state-dependent parameters for this family. In our case, we opt for a mixture of n Gaussians each parameterised by μ_i and σ_i , denoted as $\sum_{i=1}^n w_i \mathcal{N}(\mu_i, \sigma_i)$. Let h_θ be a neural network that outputs the parameters (w_i, μ_i, σ_i) of the Gaussian for a given state. Each Gaussian component is characterised by **three parameters**: weight (w_i), mean (μ_i), and standard deviation (σ_i). This mixture of Gaussians represents one flow, and we can compose multiple flows as in a deep neural network. For instance, if we require **four flows** with n **components** in the mixture, the neural network h_θ would output a total of $4 \times 3 \times n$ parameters.

However, the PDF of the Gaussian mixture is not bijective, rendering it unsuitable as a usable flow function. Its CDF, in the other hand, is bijective and a valid diffeomorphism. Therefore we advocate for the use of CDFs as flow functions.

We believe that it is important to add more precision in order to avoid confusion. Indeed, in section 4.3.1 we argued that learning a CDF is not an ideal choice for distributional RL as it does not facilitate sampling. In section 4.3.1, the learnt CDF is the one corresponding to the distribution of returns while in our approach we learn a flow function that takes the form of a CDF and that transforms samples from the base distributions to samples from the target one. In our approach, only the samples from the return distribution are determined by the CDF flow but their corresponding density is determined by the change of variable formula displayed in equation 4.1. Said otherwise, the learnt CDF is not the CDF of the predicted return distribution.

Given a finite set of gaussian PDFs $p_1(x), \dots, p_n(x)$, or corresponding cumulative distribution functions $F_1(x), \dots, F_n(x)$ and weights w_1, \dots, w_n such that $w_i \geq 0$ and $\sum w_i = 1$, the mixture distribution can be represented by writing either the density, p , or the distribution function, F , as a sum (which in both cases is a convex combination):

$$F(x) = \sum_{i=1}^n w_i F_i(x)$$

$$p(x) = \sum_{i=1}^n w_i p_i(x)$$

Given the parameters (w_i, μ_i, σ_i) , the CDF of each component of the mixture is easy to compute and therefore the CDF of the mixture itself. The CDF being a bounded, continuous, strictly increasing function, is hence bijective. Moreover it is differentiable (its derivative is the PDF). As the CDF is continuously differentiable and bijective, according to the inverse function theorem, its inverse function F^{-1} is also differentiable. Therefore, F is a diffeomorphism and can thus be used as a flow function.

The flow parameters are given by a neural network h_θ which takes a state s as input. This allows to learn θ such as the model outputs different flows depending on the input state. However, the target distribution is conditioned on the state-action pair and not only the state. In order to output different flows (and hence different return distributions) for each action, for discrete action spaces, we use multihead neural network, where each head outputs the return distribution corresponding to a specific action. Therefore, the model consists of multiple heads, matching the number of actions in the action space.

Figure 4.2 presents an overview of the model. Given an observed state s , the network h_θ outputs the parameters (w_i, μ_i, σ_i) . This network has as many output heads as $|\mathcal{A}|$. These parameters are used to construct a gaussian mixture, from which we get a CDF. The CDF is the flow itself. Given samples $z_j \sim \mathcal{U}$, the base distribution, we can get $R_j^\pi = F(z_j)$. With $R_j^\pi \sim \mathcal{P}_\pi(\cdot|s, a_k)$.

Using CDFs as flow functions brings a number of benefits:

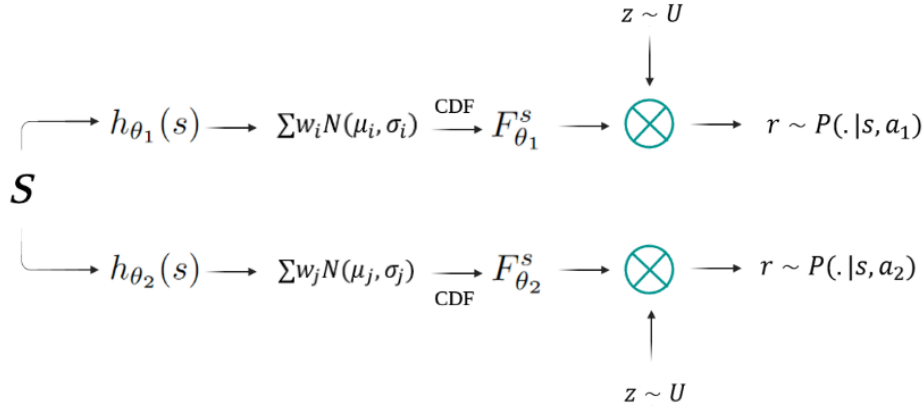


Figure 4.2: Overview of the model. Given a state s , each head of the network outputs the parameters of the flow which turns $z \sim \mathcal{U}$ into $R \sim \mathcal{P}_\pi(\cdot | s, a_i)$.

- **Densities are easily computable:** We recall that $\log p(R|s, a) = \log p_z(z) - \log \left(\frac{\partial F(z)}{\partial z} \right)$. The first part $p_z(z)$ is easily computable by definition as we choose a simple base distribution. The second part $\frac{\partial F(z)}{\partial z}$ is also easy to compute as the derivative of the flow is simply the PDF of the gaussian mixture. Easily computable densities is crucial as these quantities will be required for training the model.
- **Inverting the flow:** One of the motivations of our approach is to be able to compute the density of any given return value for any state-action pair. This process requires to compute $F^{-1}(x)$, the inverse of the flow function. The flow being a CDF, its inverse is the quantile function. Although there is no closed form quantile function for a gaussian mixture, it is easy to compute through Bisection. However, as $F(x)$ is monotonically increasing, we can perform binary search on x to find the smallest value such that $F(x)$ is greater than or equal to q , accordingly with the quantile function definition stated above.
- **Non linearity:** In order to be able to fit complex distributions, non linear transformations are required as they ensure better expressivity. There exists a number of parametrisable flows in the literature that guarantee differentiability and are invertible [KPB21]. Affine coupling flows [DKB15] are limited in expressiveness and many flows must be stacked to represent complicated distributions. Nonlinear squared flows [ZR19] require some constraints on their parameters while splines [Mül+19] necessitate to divide the domain into k bins, which we preferred to avoid as the aim is to output continuous distributions unlike discretized ones as in C51. The closest approach to ours is [Ho+19] which uses the CDF of a mixture of K logistics, postcomposed with an inverse sigmoid. As ours, this approach assures more expressive coupling functions.

4.3.4 Rescaling the Flow Output as CDF Flows Restrict Returns Range

Given that the chosen flow function takes the form of a CDF, the predicted returns will always fall within the interval $[0, 1]$. As one of the motivations of our model is to maintain continuous and unbounded support for the predicted returns, a scaling step is mandatory.

By scaling the rewards within the range $[0, 1]$, we ensure that the sum of discounted rewards $R = \sum_i^\infty \gamma^i r_i \leq \frac{1}{1-\gamma}$. If $\gamma = 0.9$, then $0 \leq R \leq 10$. Therefore setting a value for γ and a range for r enables to expand the space of possible returns. Consequently, we introduce an additional flow layer that rescales the output of the previous flow layer, by multiplying it by 10. This rescaling operation introduces a new term to the estimated return distribution mentioned in equation (4.1), modifying its characteristics accordingly.

$$\log(\mathcal{P}_\pi(R|s, a)) = \log(p_z(z)) - \log\left(\frac{\partial F(z)}{\partial z}\right) - \log(1/(1-\gamma)) \quad (4.2)$$

4.3.5 Using a Flow to Build the RL Target Distribution

We believe it is important here to make the distinction between RL target distribution and NF target distribution. Indeed, NF is a mapping between a NF base and NF target distribution. The latter is the predicted return distribution from an RL perspective. During the learning procedure, this distribution has then to be compared to the RL target distribution. In this section, all mentions of target distribution correspond to the RL target distribution.

After detailing the inference model, this section will detail how we build the RL target distribution. Indeed, in contrast to the supervised learning setting, the target is not directly available in a RL context. We will first explain how the target is usually built in Distributional RL, then we will show that direct application of existing approaches do not lead to a proper distribution. Finally, we will provide a solution to this issue by introducing a target flow.

Building Target Distributions in Distributional RL

As detailed in chapter 2, section 2.2, in their seminal paper introducing Distributional Reinforcement learning, Bellemare, Dabney and Munos [BDM17] deviate from the conventional approach of using expectations within Bellman’s equations and instead focus on the complete distribution of the random variable Z^π defined as a mapping that associates state-action pairs with distributions over returns, and refer to it as the value distribution. Similarly to the classic Bellman operator, they introduced the distributional Bellman operator \mathcal{B}^π to compute the value distribution:

$$\mathcal{B}^\pi Z^\pi(s, a) \stackrel{D}{=} r(s, a) + \gamma T_\pi Z^\pi(s, a) \quad (4.3)$$

Where $T_\pi : \mathcal{Z} \mapsto \mathcal{Z}$ the transition operator:

$$\begin{aligned} T_\pi Z^\pi(s, a) &\stackrel{D}{=} Z^\pi(s', a') \\ s' &\sim \mathcal{T}(\cdot|s, a), a' \sim \pi(\cdot|s) \end{aligned} \quad (4.4)$$

where $Y \stackrel{D}{=} U$ denotes equality of probability laws, that is the random variable Y is distributed according to the same law as U . As in classic Q-learning, Equation 4.3 defines the target distribution of $Z^\pi(s, a)$. The target is simply a scaling operation on $T_\pi Z^\pi(s, a)$ (multiplying by gamma), and a shift operation by the value of the reward $r(s_t, a_t)$ received from the environment as illustrated in figure 4.3.

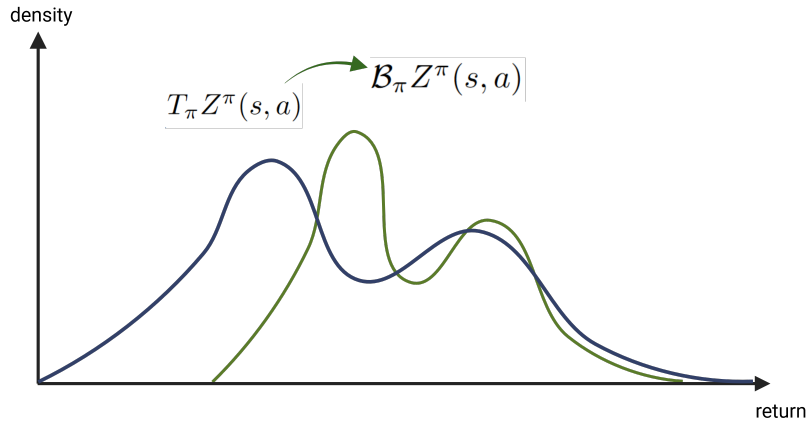


Figure 4.3: Illustration of the scale and shift operation for building the target distribution. The distributional Bellman operator is applied on the predicted distribution $T_\pi Z^\pi(s, a)$. Consequently, $T_\pi Z^\pi(s, a)$ is scaled by a factor γ and horizontally shifted by the amount of received reward after performing action a in state s ; hence producing the target $\mathcal{B}_\pi Z^\pi(s, a)$.

Direct Application of the Distributional TD Update Does not Output a Proper Distribution

Unfortunately, the resulting distribution of $\mathcal{B}_\pi Z^\pi(s, a)$ is not a valid distribution over the chosen support. Indeed, given a distribution h with support in $[0, 10]^3$, in order to shift and scale its support, one would apply a function of the form $\gamma \cdot x + r$ to each input. This operation serves as the learning update. Although this approach may seem convenient, it is important to note that the resulting distribution will no longer be a density function. Specifically, if we consider the function $g(x) = 0.9x + 1$ as an example, it represents a transformation that shifts and scales the distribution, but the resulting function does not meet the criteria of a valid probability density function. For instance, if we recall that the rewards are scaled within the range $[0, 1]$ and that with $\gamma = 0.9$, $R \leq 10$, we have:

³We recall that if $\gamma = 0.9$, then $R \leq 10$, therefore, supposing that h is a valid distribution before the shift and scale operation, we have $\int_0^{10} h(x)dx = 1$. Indeed, for all $x > 10$, $h(x) = 0$.

$$\begin{aligned}
\int_0^{10} h(g(x))dx &= \int_0^{10} h(0.9x + 1)dx \\
&= 1/0.9 \int_1^{10} h(x) \\
&= 1/0.9 \left(\int_0^{10} h(x)dx - \int_0^1 h(x)dx \right) \quad (h \text{ is a PDF so } \int_0^{10} h(x)dx = 1) \\
&= 1/0.9 \left(1 - \int_0^1 h(x)dx \right) \\
&\approx 1.1 - 1.1 \left(\int_0^1 h(x)dx \right) \\
&\neq 1
\end{aligned} \tag{4.5}$$

If we consider as an example that h is the PDF of $\mathcal{N}(5, 1)$. In this specific example $\int_0^{10} h(x)dx \approx 1$. But when we compute the value of $1.1 - 1.1 \left(\int_0^1 h(x)dx \right)$, we find it is > 1 .

Introducing the Target Flow to Output Valid Target Distributions.

We introduce in this section one of the main novelties of this work. We consider the shift and scale operation that is used to build the target distribution as a flow itself. Indeed, the function $X \mapsto r + \gamma \cdot x$ is a valid diffeomorphism. Therefore, implementing the target update as a flow, one can apply the change of variable formula to guarantee that the result will still be a valid distribution. Therefore building the target simply consists on chaining three flows:

1. First we estimate the return distribution of s_{t+1}, a_{t+1} :

$$\log(\mathcal{P}_\pi(R|s_{t+1}, a_{t+1})) = \log(p_z(z)) - \log\left(\frac{\partial F^{s_{t+1}, a_{t+1}}(z)}{\partial z}\right)$$

2. To build the target we need to scale that distribution by γ and shift it by r . We just have to build this operation as a flow by passing the function g after $F^{s_{t+1}, a_{t+1}}$ (which we will simply denote F below). This way the target is:

$$\begin{aligned}
\log(\mathcal{P}_\pi(R|s_{t+1}, a_{t+1})) &= \log(p_z(z)) - \log \frac{\partial F(z)}{\partial z} - \log \frac{\partial g(F(z))}{\partial F(z)} \\
&= \log(p_z(z)) - \log \frac{\partial F(z)}{\partial z} - \log(\gamma)
\end{aligned} \tag{4.6}$$

3. Finally we add the last flow for return scaling and we obtain:

$$\log(\mathcal{P}_\pi(R|s_{t+1}, a_{t+1})) = \log(p_z(z)) - \log\left(\frac{\partial F(z)}{\partial z}\right) - \log(\gamma) - \log\left(\frac{1}{1-\gamma}\right)$$

Using this procedure, the target distribution is ensured to sum to one. Moreover, we can also notice that the cost of this addition is limited as terms $\log(\gamma)$ and $\log\left(\frac{1}{1-\gamma}\right)$ are constants that only depend on γ .

4.3.6 How to Model the Final State Target as a Distribution

As for traditional TD methods, the target for the final state in distributional RL is only the immediate reward received r . As in our case the target needs to be a distribution, r can be considered as a degenerate one, represented as a Dirac function. We choose to approximate it by a Gaussian distribution centered on the reward value r and which variance $\sigma = 0.05$, $\mathcal{N}(r, 0.05)$. The variance value is determined as follows.

In order to determine an appropriate σ , we first discretise the support $[0, 10]$ (corresponding to $\gamma = 0.9$) into 51 bins, following the approach of the C51 model [BDM17]. Subsequently, the idea is to ensure that 95% of the distribution mass falls within the bin containing r . First, we can calculate the width of each bin as $10/51 = 0.2$. Knowing that 95% of the mass for any normal distribution is within $[\mu - 2\sigma, \mu + 2\sigma]$ (figure 4.4), in order to achieve 95% mass within the bin centered on r , the interval becomes $[r - 0.1, r + 0.1]$. Thus, the suitable choice of σ is $\sigma = 0.05$, yielding the normal distribution $\mathcal{N}(r, 0.05)$.

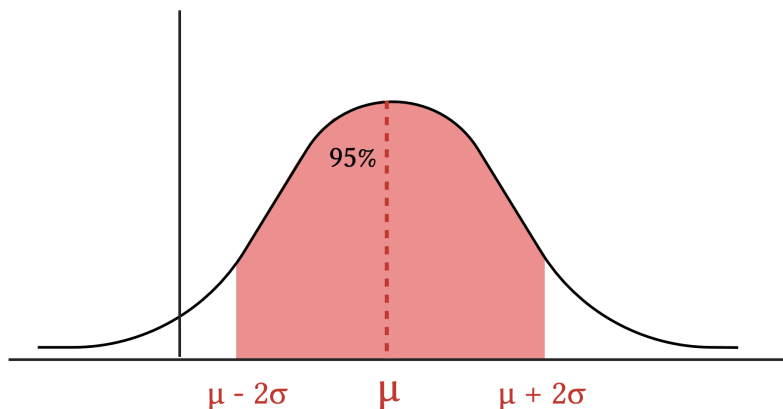


Figure 4.4: For a Gaussian distribution, 95% of mass density falls in the interval $[\mu - 2\sigma, \mu + 2\sigma]$.

4.4 Using the Cramèr Distance as a Loss Function

Once the target is defined, the last step is to construct the loss function. In this section, we will first show that the KL divergence is not suited for the considered problem. Then in section 4.4.2, we mention the three possible ways to compute the Wasserstein distance (primal, dual and quantile regression), and give more details about how the Wasserstein distance was used in existing approaches. In sections 4.4.2 and 4.4.2 we explain that the dual form and the quantile regression method for computing the Wasserstein distance are not useful in our setup and show why the primal is not applicable either. Finally in section 4.4.3 we show that the Cramèr distance is superior to the Wasserstein distance in our setting and easier to compute using our flow based architecture.

In this section we will loosen a bit the notation. Let $f(z, s, a)$ be the flow for the predicted returns and $g(z, s, a)$ the flow for the target distributions. This way $R(s, a)$ is a return predicted for state-action pair (s, a) such that $R(s, a) = f(z, s, a)$ for a certain $z \sim \mathcal{U}$ and $t(s, a)$ the target return for the same triplet, i.e. $t(s, a) = g(z, s, a)$. We also note P the distribution of the predicted returns and Q the distribution of target returns.

4.4.1 Why not Reverse KL Divergence

A classic approach is to use the KL divergence between P and Q , but one still has to decide between using the Forward or Reverse KL divergence. As we fixed the flows such that $R(s, a) = f(z, s, a)$, the right option would be to use the Reverse KL divergence:

$$KL(P||Q) = \mathbb{E}_{R \sim P} \log \frac{P(R)}{Q(R)}$$

Indeed, in this situation it is possible to evaluate the density of any return in the target distribution but we cannot sample from it. We can only sample return values from the predicted distributions. In this context, as the return distribution is one dimensional, contrary to the case we encountered in chapter 3, the probability that the RKL discovers relevant regions in Q via sampling from P is in fact non negligible. However, if $Q(R) = 0$ in some R where $P(R) > 0$, the RKL diverges towards $+\infty$. Given the tightly centered normal distributions defined for final states, this situation is more than likely.

Moreover, it was proven in [BDM17] that the KL divergence is not a contraction for the distributional bellman operator \mathcal{B}_π .

4.4.2 Trying to Use the Wasserstein Distance as in Existing Approaches

The Wasserstein distance between P and Q appears as a good candidate for a loss function as it cannot diverge in the above example. Moreover, as detailed in chapter 2 section 2.5.4, the Wasserstein distance takes into account the geometry of the space the distributions lie in. More specifically, considering the example depicted in figure 4.3, if the γ factor is set to 1, then the target distribution would correspond to a simple horizontal shift of the predicted one. In that case, the RKL would be null while the Wasserstein distance would be positive, taking into account the distance corresponding to the shift. Therefore, the Wasserstein distance is conveniently robust to discrepancies in support. However, Bellemare et. al [BDM17] proved that in the distributional RL context, we are typically restricted to learning from sample transitions, which is not possible under the Wasserstein loss (Proposition 5). In this section we will mention the different ways to compute the Wasserstein distance and why it is not suited for our approach.

There are three ways to compute the Wasserstein distance, using its primal form, the dual form or the quantile function based formulation.

QR-DQN and IQN solved the issue encountered in C51 by making use of the inverse CDF characterisation of the p -Wasserstein metric [Mül97]. That is, the p -Wasserstein metric between two

distributions U and Y is given by:

$$W_p(U, Y) = \left(\int_0^1 |F_Y^{-1}(\omega) - F_U^{-1}(\omega)|^p d\omega \right)^{1/p} \quad (4.7)$$

Using the quantile regression they could obtain an unbiased stochastic approximation of the quantile function in order to obtain unbiased gradients.

The dual approach was noticeably used in WGAN [ACB17] where the authors used the dual form of the 1-Wasserstein in order to obtain a viable optimisation through SGD:

$$W_1(P, Q) = \sup_{f \in \mathcal{F}_L} [\mathbb{E}_{X \sim P}[f(X)] - \mathbb{E}_{Y \sim Q}[f(Y)]]$$

where \mathcal{F}_L is the class of all 1-lipschitz functions.

Leveraging NF to simplify the Wasserstein Distance Computation

In our case, it is not possible to compute the dual form nor rely on quantile regression. We will therefore try to make use of the primal form of the Wasserstein distance defined as:

$$W_c(P, Q) := \inf_{\gamma \in \Pi(P, Q)} \mathbb{E}_{(X, Y) \sim \gamma} [c(X, Y)]$$

Where $c(x, y) : \mathcal{X} \times \mathcal{X}$ is any measurable cost function and $\Pi(P, Q)$ the set of all joint distributions with marginals P and Q .

The computation of the Wasserstein distance using its primal form can be a tedious task, as it involves solving an optimisation problem with the set of all couplings as the search space. However, leveraging the normalizing flows framework, we propose a more manageable approach. By observing that $R(s, a) = f(z, s, a)$ and $t(s, a) = g(z, s, a)$, and assuming that f and g are bijective functions sharing the same input values z , a one-to-one relationship is established between each $R(z, s, a)$ and $t(z, s, a)$. Indeed, we first sample a $z \sim \mathcal{U}$ for which there exist a unique $R(z, s, a)$ and a unique $t(z, s, a)$. Therefore there exists a one-to-one mapping between $R(z, s, a)$ and $t(z, s, a)$. Consequently, there is no need to search over the space of all possible couplings, as this constraint narrows it down to a unique coupling γ^\dagger . This reduction simplifies the computation of the Wasserstein distance to:

$$W_c^\dagger(P, Q) = \mathbb{E}_{(X, Y) \sim \gamma^\dagger} [c(X, Y)]$$

The Simplification Does not Yield the Correct Distance

While this constrained bijection between X and Y offers the huge advantage of erasing the infimum operator, it comes with several disadvantages. First, W_c^\dagger is an upper bound of W_C . Indeed there is no guarantee that the constrained coupling is the one that minimises the distance for the cost function c ; therefore:

$$W_C(P, Q) \leq W_c^\dagger(P, Q)$$

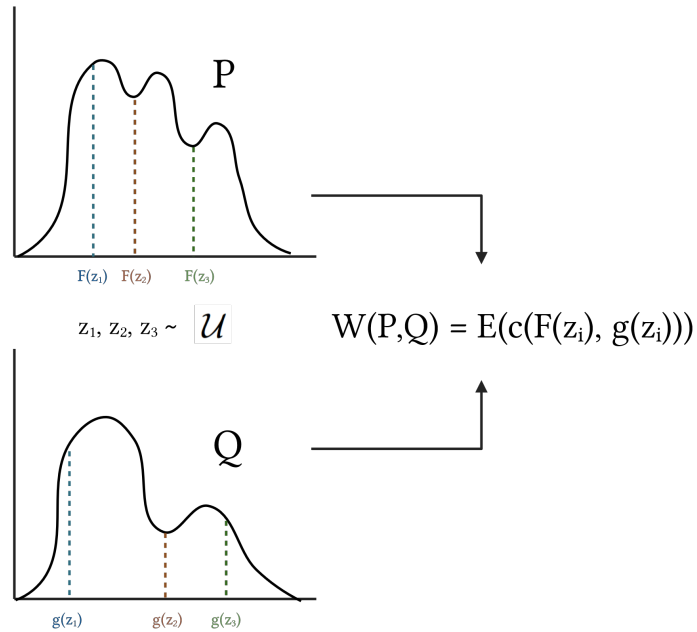


Figure 4.5: Given two distributions P and Q and two flows F, g . If samples z_i are drawn from a base distribution \mathcal{U} and used as input in each of the flows, as the flows are bijections, there is a one to one relation between $F(z_i)$ and $g(z_i)$. This implies that the couplings required for computing the Wasserstein distance are restricted by the flows.

Moreover, the coupling γ^\dagger is not guaranteed (it practically never happens) to respect the marginals of X and Y . Considering the same transport plan example as in chapter 2, section 2.5.4 that we display below for more convenience, the goal is to find the optimal transport plan between X and Y . Let d be our cost function and $m_{i,j}$ the mass to transport from x_i to y_j .

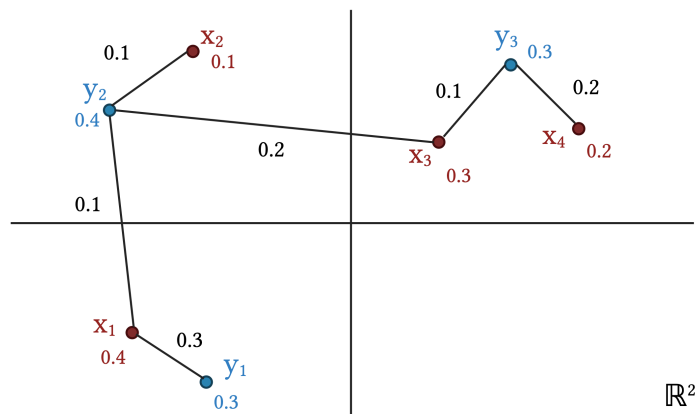


Figure 4.6: Example of a transport plan. To each point is assigned a mass and the goal is to move the blue masses towards the red ones.

Table 4.1 presents an example of strategy. We notice that the sum of columns and rows correspond

		x_1	x_2	x_3	x_4
		0.4	0.1	0.3	0.2
y_1	0.3	0.3	0	0	0
y_2	0.4	0.1	0.1	0.2	0
y_3	0.3	0	0	0.1	0.2

Table 4.1: Example of a transport plan.

to values outside the matrix. The W_1 distance is therefore:

$$\sum_{i,j} m_{i,j} d(x_i, y_j) : m_{i,j} \geq 0, \sum_i m_{i,j} = \eta_j, \sum_j m_{i,j} = \alpha_i \quad (4.8)$$

Where $P(x_i) = \eta_i$ and $P(y_i) = \alpha_i$. This formula states that the quantity of mass that leaves x , $\int_y \gamma(x, y) dy$ should be equal to the mass that was originally present in x . In the same manner, the quantity of mass that enters y , $\int_x \gamma(x, y) dx$ should be equal to the mass in the marginal $m_{.,j}$.

Let β be the bijection that associates a $y_i \in Y$ to each $x_i \in X$, this means that mass $P(x_i)$ is moved towards y_i . However the conditions on marginals stated above are not respected as shown in figure 4.7. Indeed, there is no guarantee that $P(x_i) = P(y_i)$. Therefore, the coupling γ^\dagger defined by the bijection β is generally not in the set of feasible couplings defining a suitable transport plan traditionally taken into for the Wasserstein distance.

As a result, regardless of the chosen cost, we are no longer optimising the Wasserstein distance. In the subsequent section, we will demonstrate how this bijection can still be leveraged to construct an approximation of the Cramèr Distance.

4.4.3 The Cramèr Distance is Ideal and has Unbiased Sample Gradients

Similar to the Wasserstein distance, the Cramèr distance provides a measure of the distance between probability distributions in a metric space. Given two cumulative distributions of a real random variable X , denoted as $P(x)$ and $Q(x)$, the Cramèr distance is defined as follows:

$$C_p(P, Q) = \left(\int_{-\infty}^{\infty} |P(x) - Q(x)|^p \right)^{1/p}$$

The work presented in [Sej+13] has established that the Cramèr distance falls within the family of integral probability metrics (IPM), akin to the Wasserstein distance. This stands in contrast to KL and reverse KL divergences, which belong to the family of f-divergences, not IPMs. Notably, when the parameter $p = 1$, the Wasserstein distance and the Cramèr distance coincide. To illustrate this visually, considering X on the abscissa and the cumulative distribution function (CDF) on the ordinate, the Wasserstein distance involves summing over horizontal slices while sweeping vertically. On the other hand, the Cramèr distance entails summing over vertical slices while sweeping horizontally, as depicted in figure 4.8.

In [Bel+17], it was proven that the Cramèr distance holds the following properties:

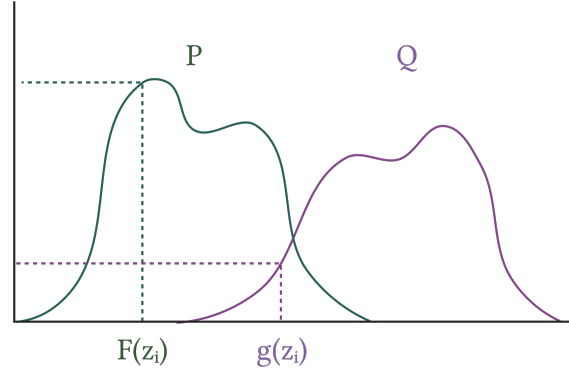


Figure 4.7: Given $z_i \sim \mathcal{U}$ the base distribution and two flow functions F and g , the flows being bijective, there exist unique images $f(z_i)$ and $g(z_i)$ that are samples from the distributions P and Q respectively. However $P(f(z_i)) \neq Q(g(z_i))$, therefore if $g(z_i)$ is the only sample associated to $f(z_i)$ in a transport plan, the conditions displayed in equation 4.8 are broken.

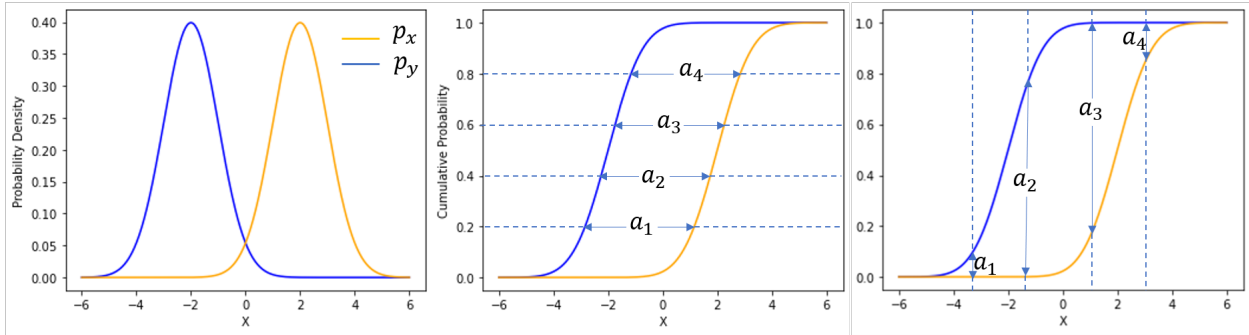


Figure 4.8: Using CDFs to compute the Wasserstein and Cramèr distances. **Middle:** Calculating the Wasserstein distance involves summing over the horizontal distances a_i . **Right:** Calculating the Cramèr distance involves summing over vertical distances a_i .

Consider a divergence \mathbf{d} , and for two random variables X, Y with distributions P, Q , write $\mathbf{d}(X, Y) := \mathbf{d}(P, Q)$. We say that \mathbf{d} is scale sensitive (of order β), i.e. it has property (S) if there exists a $\beta > 0$ such that for all X, Y , and a real value $c > 0$,

$$\mathbf{d}(cX, cY) \leq |c|^\beta \mathbf{d}(X, Y) \quad (\text{S})$$

A divergence \mathbf{d} has property (I), i.e. it is sum invariant, if whenever A is independent from X, Y

$$\mathbf{d}(A + X, A + Y) \leq \mathbf{d}(X, Y) \quad (\text{I})$$

A divergence is said ideal if it possesses both (S) and (I).

Let $X_m := X_1, X_2, \dots, X_m$ be independent samples from P and define the empirical distribution $\hat{P}_m := \hat{P}_m(X_m) := \frac{1}{m} \sum_{i=1}^m \delta_{X_i}$. From this, define the sample loss $\theta \mapsto \mathbf{d}(\hat{P}_m, Q_\theta)$. We say that \mathbf{d} has unbiased sample gradients when the expected gradient of the sample loss equals the gradient of the true loss for all P and m :

$$\mathbb{E}_{X_m \sim P} \nabla_{\theta} \mathbf{d}(\hat{P}_m, Q_\theta) = \nabla_{\theta} \mathbf{d}(P, Q_\theta) \quad (\text{U})$$

If a divergence does not possess (U), then minimising it with stochastic gradient descent may not converge or towards the wrong minimum. Conversely, if \mathbf{d} possesses (U) then we can guarantee that the distribution which minimises the expected sample loss is $Q = P$. From these properties, [Bel+17] draws the following propositions:

Proposition 1: *The KL divergence has unbiased sample gradients (U), but is not scale sensitive (S).*

Proposition 2: *The Wasserstein metric is ideal (I, S), but does not have unbiased sample gradients.*

Moreover, they also show that the Cramér distance has the same appealing properties as the Wasserstein metric, but also provides unbiased sample gradients. This distance is therefore ideal for our approach. However, it necessitates the CDFs of the predicted and target distributions which are not available. We propose to use a Taylor expansion to make the available PDFs useful. If $P(x), Q(x)$ are two cumulative distributions of the real random variable X and $p(x)$ and $q(x)$ their respective PDFs,

$$\begin{aligned} P(x) - Q(x) &= P(x_0) + p(x_0)(x - x_0) - Q(x_0) + q(x_0)(x - x_0) \\ &= (x - x_0)(P(x_0) + p(x_0) - Q(x_0) - q(x_0)) \end{aligned} \quad (4.9)$$

Let R be the predicted return such that $R = (z), z \sim U$ and t the predicted return such that $t = g(z)$. Fixing $t = x$ and $R = x_0$:

$$\begin{aligned} P(t) - Q(t) &= (t - R)(P(R) + p(R) - q(R) - Q(R)) \\ |P(t) - Q(t)| &= |(t - R)| \cdot |(P(R) + p(R) - q(R) - Q(R))| \end{aligned}$$

From the expression above and the fact that p and q are the derivatives of P and Q respectively, it is clear that if $p(R) - q(R) > 0$ then $P(R) - Q(R)$ is increasing. Therefore the higher the difference $p(R) - q(R)$ the higher is the term $P(R) - Q(R)$. Finally, $p(R) - q(R)$ and $P(R) - Q(R)$ are positively correlated. We can deduce that the expression:

$$|t - R| \cdot (p(R) - q(R))^2$$

is an acceptable optimisation surrogate for the Cramér distance. Adapting this surrogate to our approach, let F be the flow corresponding to the predicted distribution and g the flow of the target distribution, the optimisation criteria is:

$$\mathcal{L}(\theta, z) = \frac{1}{N} \sum_{i=1}^N (p(f(z_i)) - q(f(z_i)))^2 \cdot |f(z_i) - g(z_i)| \quad (4.10)$$

In the next sections we will investigate the three following questions:

- **Q1:** Is this loss a proper distance? If so, then the Cramèr loss function would be symmetric and scale sensitive unlike KL divergence.
- **Q2:** Is the distributional Bellman operator a contraction in this case? If so, this would ensure convergence of the distributional Bellman operator $\mathcal{B}_{\pi}R_k$ towards the random returns Z^π .
- **Q3:** Does it still possess the unbiased sample gradient estimate property? If so, then SGD can be used to optimise this loss function in a RL context unlike the Wasserstein distance. More specifically, we will be able to learn from sample transitions.

Q1: Is \mathcal{L} a proper distance?

To ease the notation let π_1 and π_2 be two pdfs defined on the same support, and $d(\pi_1, \pi_2)$ the function defined by equation (4.10):

$$d(\pi_1(f(z)), \pi_2(f(z))) = (\pi_1(f(z)) - \pi_2(f(z)))^2 \cdot |f(z) - g(z)|$$

As a product of two absolute values, the symmetry and triangle equality are clear. However it is interesting to verify that $d(\pi_1, \pi_2) = 0 \iff \pi_1 = \pi_2$.

If $\pi_1 = \pi_2 = 0$ then $|\pi_1(f(z)) - \pi_2(f(z))| = 0$ for all z , therefore $d(\pi_1, \pi_2) = 0$.

On the other hand,

$$\begin{aligned} d(\pi_1, \pi_2) = 0 &\implies \pi_1(f(z)) - \pi_2(f(z)) = 0 \quad \text{or} \quad f(z) - g(z) = 0 \\ &\implies \pi_1(f(z)) = \pi_2(f(z)) \quad \text{or} \quad f(z) = g(z) \end{aligned} \quad (4.11)$$

$$\begin{aligned} f(z) = g(z) &\iff \frac{\partial f(z)}{\partial z} = \frac{\partial g(z)}{\partial z} \\ &\iff p_z \frac{\partial f(z)}{\partial z} = p_z \frac{\partial g(z)}{\partial z} \\ &\iff \pi_1(f(z)) = \pi_2(g(z)) \end{aligned} \quad (4.12)$$

$\pi_1(f(z)) = \pi_2(f(z)) \iff \pi_1 = \pi_2$ because f being a diffeomorphism, $f(z)$ is compact on the support of $U(z)$.

These equivalences are true for all z , and \mathcal{L} being a sum on positive values, these relations are also true for $d(\pi_1, \pi_2) = \frac{1}{N} \sum_{i=1}^N |\pi_1(f(z_i)) - \pi_2(f(z_i))| \cdot |f(z_i) - g(z_i)|$. We can therefore conclude that d is a proper distance function.

Q2: Is the Distributional Bellman Operator a Contraction for d ?

Consider a scalar a and a random variable A independent of random variables U and V . The metric d shares the following two properties with the Wasserstein distance:

$$\begin{aligned} d(aU, aV) &\leq |a|d(U, V) \\ d(A + U, A + V) &\leq d(U, V) \end{aligned}$$

In [BDM17], the authors use these two properties to prove that $\mathcal{B}_\pi : \mathcal{Z} \rightarrow \mathcal{Z}$ is a γ contraction in \bar{d}_p . Where \bar{d}_p is defined as

$$\bar{d}_p(Z_1, Z_2) = \sup_{s,a} d_p(Z_1(s, a), Z_2(s, a))$$

Where d_p is the p -Wasserstein distance and $Z_1, Z_2 \in \mathcal{Z}$. As d shares these same two properties with d_p , the proof is identical. We will share it here for completeness.

By definition,

$$\bar{d}(\mathcal{B}_\pi Z_1, \mathcal{B}_\pi Z_2) = \sup_{s,a} d(\mathcal{B}_\pi Z_1(s, a), \mathcal{B}_\pi Z_2(s, a))$$

By the properties of d , we have:

$$\begin{aligned} d(\mathcal{B}_\pi Z_1(s, a), \mathcal{B}_\pi Z_2(s, a)) &= d(r(s, a) + \gamma T_\pi Z_1(s, a), r(s, a) + \gamma T_\pi Z_2(s, a)) \\ &\leq \gamma d(T_\pi Z_1(s, a), T_\pi Z_2(s, a)) \\ &\leq \gamma \sup_{s',a'} d(Z_1(s', a'), Z_2(s', a')) \end{aligned} \quad (4.13)$$

Where the last line follows from the definition of T_π . Therefore:

$$\begin{aligned} \bar{d}(\mathcal{B}_\pi Z_1, \mathcal{B}_\pi Z_2) &= \sum_{s,a} d(\mathcal{B}_\pi Z_1(s, a), \mathcal{B}_\pi Z_2(s, a)) \\ &\leq \gamma \sum_{s',a'} d(Z_1(s', a'), Z_2(s', a')) \\ &= \gamma \bar{d}(Z_1, Z_2) \end{aligned} \quad (4.14)$$

Therefore, $\mathcal{B}_\pi : \mathcal{Z} \rightarrow \mathcal{Z}$ is a γ contraction in \bar{d} . We conclude using Banach's fixed point theorem that \mathcal{B}_π has a unique fixed point, which is Z^π . Assuming all moments are bounded, we can conclude that the sequence (Z_k) converges to Z^π in \bar{d} .

Q3: Does d Have Unbiased Sample Gradients Estimates?

In [BDM17], while proving convergence and contraction results using the Wasserstein distance, the authors explain that it cannot be used because, in RL we are typically restricted to learning from sample transitions, which is not possible under the Wasserstein loss. This is justified through the following proposition: *Fix some next-state distribution Z and policy π . Consider a parametric value distribution Z_θ , and define the Wasserstein loss:*

$$\mathcal{L}_W(\theta) := d_p(Z_\theta(s, a), r(s, a) + \gamma Z(s', \pi(s')))$$

Let $s' \sim \mathcal{T}(\cdot | s, a)$ and consider the sample loss:

$$L_W(\theta, r, s') := d_p(Z_\theta(s, a), r + \gamma Z(s', \pi(s')))$$

Its expectation is an upper bound on the loss \mathcal{L}_W :

$$\mathcal{L}_W(\theta) \leq \mathbb{E}_{R,P} L_W(\theta, r, s')$$

in general with strict inequality.

Nonetheless, the fact that the sample estimate serves as an upper bound is not a critical impediment to learning and achieving convergence towards the desired parameters. The main concern lies in the observation made in [Bel+17], where the authors demonstrate that the Wasserstein distance lacks the property (U). This finding leads to the following theorem:

Consider $P = B(\theta^*)$ a Bernoulli distribution with parameter θ^* and $Q_\theta = B(\theta)$. The empirical distribution \hat{P}_m is derived from samples X_1, \dots, X_m drawn from $B(\theta^*)$. We have $\hat{\theta} = \frac{1}{m} \sum_{i=1}^m \mathbb{1}_{A(i)}$, with $\mathbb{1}_{A(i)}$ the indicator function of event $A : X = 1$. Then the minimum of the expected sample loss is in general different from the minimum of the true Wasserstein loss:

$$\arg \min_{\theta} \mathbb{E}[W_p(\hat{P}_m, P(\theta^*))] \neq \arg \min_{\theta} W_p(Q_\theta, P(\theta^*))$$

In the following we shall prove that the distance d holds property (U), and therefore that:

$$\arg \min_{\theta} \mathcal{L}(\theta) = \arg \min_{\theta} \mathbb{E}_{R,P} \mathcal{L}(\theta, r, s')$$

We will first consider an example through the MDP depicted in figure 4.9.

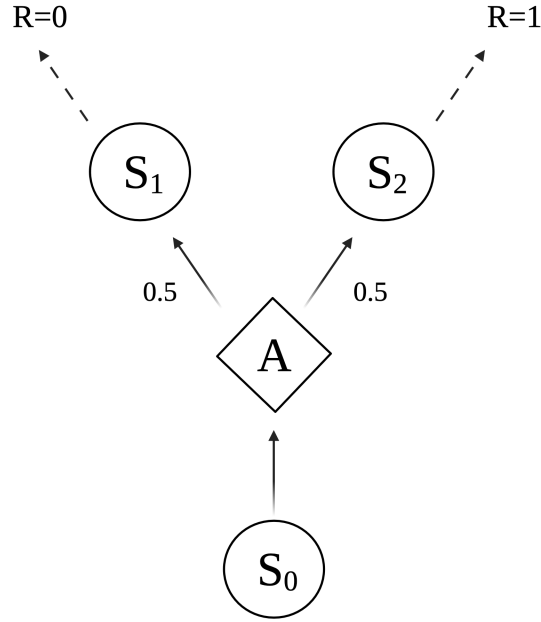


Figure 4.9: Example of stochastic MDP, where the same action A leads to 2 different states with equal probability, and different rewards R .

Consider the metric d between the distribution P and another distribution Q defined as:

$$P = \begin{cases} 0 & \text{w.p.} & 1/2 \\ 1 & \text{w.p.} & 1/2 \end{cases} \quad (4.15)$$

In this example, $i \in \{1, 2\}$, $P_1 = 0$, and $P_2 = 1$. We now consider the distribution Q with the same support but that puts probability p on 0:

$$Q = \begin{cases} 0 & \text{w.p. } p \\ 1 & \text{w.p. } 1 - p \end{cases} \quad (4.16)$$

The distance between P and Q is

$$\begin{aligned} d(P, Q) &= (p - 1/2)^2 + (1 - p - 1/2)^2 \\ &= 2p^2 - 2p + 1/2 \end{aligned} \quad (4.17)$$

There $d = 0 \iff p = 1/2$.

Now, considering the MDP in figure 4.9, this means that we first observe the value 0, then the value 1. When observing the value 0, this means that we have to calculate the distance between the distributions illustrated in figure 4.10:

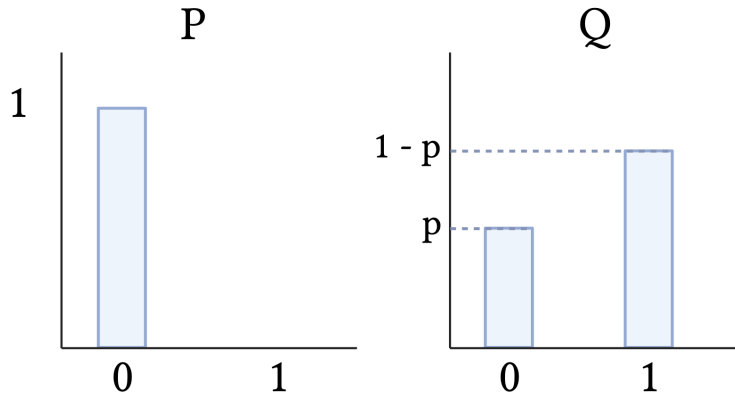


Figure 4.10: Illustration of empirical distributions considered in the example of equation 4.16.

In this case, $d(\hat{P}, Q) = (\hat{P}(0) - Q(0))^2 = (1 - p)^2$. Using the same reasoning, observing the value 1, we get $d(\hat{P}, Q) = p^2$. Therefore:

$$\begin{aligned} \mathbb{E} [d(\hat{P}, Q)] &= \frac{1}{2}(1 - p)^2 + \frac{1}{2}p^2 \\ &= \frac{1}{2} + p^2 - p \\ \nabla \mathbb{E} [d(\hat{P}, Q)] &= 0 \\ p &= \frac{1}{2} \end{aligned} \quad (4.18)$$

Through this example we see that $\mathbb{E} [d(\hat{P}, Q)] > d(P, Q)$, but most importantly:

$$\arg \min_p d(P, Q) = \arg \min_p \mathbb{E} [d(\hat{P}, Q)] \quad \text{and} \quad \nabla d(P, Q) = \nabla \mathbb{E} [d(\hat{P}, Q)]$$

Both distances reach their minimum for the same parameter p , and optimising one or the other through SGD should converge towards the same parameters. Next we will prove this result more generally, although the proof is similar.

Consider $P = \mathcal{B}(\theta^*)$ a Bernoulli distribution with parameter θ^* and $Q_\theta = \mathcal{B}(\theta)$. The empirical distribution \hat{P}_m is a Bernoulli distribution with $\hat{\theta} = \frac{1}{m} \sum_{i=1}^n \mathbb{1}_A$ with $\mathbb{1}_A$ the indicator function for the event A .

$$\begin{aligned} d(P, Q) &= (\theta^* - \theta)^2 + (1 - \theta^* + \theta - 1)^2 \\ &= 2(\theta - \theta^*)^2 \\ \nabla_\theta d(P, Q) &= 4(\theta - \theta^*) \end{aligned} \tag{4.19}$$

$$\begin{aligned} d(\hat{P}, Q) &= (\hat{\theta} - \theta)^2 + (1 - \hat{\theta} + \theta - 1)^2 \\ &= 4(\theta - \hat{\theta})^2 \\ \nabla_\theta d(\hat{P}, Q) &= 4(\theta - \hat{\theta}) \\ \mathbb{E}_m \nabla_\theta d(\hat{P}, Q) &= 4\mathbb{E}_m(\theta - \hat{\theta}) \\ &= 4\mathbb{E}_m\left(\theta - \frac{1}{m} \sum \mathbb{1}_A\right) \\ \lim_{m \rightarrow \infty} \mathbb{E}_m \nabla_\theta d(\hat{P}, Q) &= 4(\theta - \theta^*) \end{aligned} \tag{4.20}$$

Where in the penultimate line, we used the fact that $\frac{1}{m} \sum \mathbb{1}_A$ is an unbiased estimator of θ^* . Therefore:

$$\lim_{m \rightarrow \infty} \mathbb{E}_m \nabla_\theta d(\hat{P}, Q) = \nabla_\theta d(P, Q)$$

Finally,

$$\arg \min_{\theta} d(\hat{P}, Q) = \nabla_\theta d(P, Q)$$

To conclude, we proved that:

- Our loss function is a proper distance
- The distributional Bellman operator is a γ -contraction in \bar{d}
- The minimum of the expected sample loss is the same as the minimum of the true loss.

The entire process is presented in algorithm 4.

4.5 Experimental Results

In the context of solving classic gym environments [Bro+16], our method offers a significant advantage by providing multimodal return distributions that are particularly evident in critical

Algorithm 4 Normalizing Flows Distributional RL

Require: n_{epochs} : number of epochs, n : number of flows, \mathcal{U} : base distribution, c : number of gaussian mixture components, N : number of samples to draw from \mathcal{U} , γ : the discount factor, η : learning rate, (s_i, a_i, r_i, s'_i) : a batch of transitions, K : number of transitions

Initialise model M with weights θ

for k from 1 to n_{epochs} **do**

$(z_1, \dots, z_N) \sim \mathcal{U}$ ▷ Sample N samples from the base distribution

for i from 1 to K **do**

$(\mu_i^c, \sigma_i^c, w_i^c)^n \leftarrow M_\theta(s_i)$ ▷ Get the flows parameters for each s_i using the model M_θ

for j from 1 to N **do**

for m from 1 to n **do**

$F_m^{s_i}(z_j) \leftarrow \sum_c w^{c,m} \mathcal{N}(\mu^{c,m}, \sigma^{c,m})(F_{m-1}^{s_i}(z_j))$ ▷ Compute $F_m^{s_i}$ the CDFs of the

gaussian mixtures

end for

end for

$f_\theta^{s_i} := \frac{1}{1-\gamma} F_0^{s_i} \circ F_n^{s_i}$ ▷ Define the flow as the composition of all intermediate flows

for j from 1 to N **do**

$R_j^{s_i} \leftarrow f_\theta^{s_i}(z_j)$ ▷ Compute the predicted return samples for each s_i

$\log P_\theta(R_j^{s_i}) = \log p_z(z_j) + \log \frac{\partial f_\theta^{s_i}(z_j)}{\partial z_j}$ ▷ Compute sampled returns densities using

the change of variable formula

end for

$(\hat{\mu}_i^c, \hat{\sigma}_i^c, \hat{w}_i^c)^n \leftarrow M_\theta(s'_i)$ ▷ Get the flows parameters for each s_i using the model M_θ

$g_\theta^{s'_i} := \frac{1}{1-\gamma} r_t + \gamma F_0^{s'_i} \circ F_n^{s'_i}$ ▷ Define g as the target flow

for j from 1 to N **do**

$T_j \leftarrow g_\theta^{s'_i}(z_j)$ ▷ Compute the target return samples for each s'_i

$Q_\theta(R_j) = p_z(g_\theta^{-1}(R_j)) + \log \frac{\partial g_\theta(g_\theta^{-1}(R_j))}{\partial g_\theta^{-1}(R_j)}$

end for

end for

$\mathcal{L}(\theta) = \frac{1}{K} \frac{1}{N} \sum_{i=1}^K \sum_{j=1}^N (P_\theta(R_j^{s_i}) - Q_\theta(R_j^{s_i}))^2 \cdot |R_j^{s_i} - T_j^{s_i}|$

$\theta_{k+1} \leftarrow \theta_k - \eta \nabla \mathcal{L}(\theta_k)$

end for

states. For example, in the cartpole environment, the multimodality becomes apparent only in the final states, where distinct actions lead to dramatic consequences. To effectively showcase the benefits of our model, we will first present the achieved results on various configurations of simple Markov Decision Processes (MDPs) and subsequently demonstrate its performance on the FrozenLake environment.

In all these environments we used the same architecture consisting of one hidden layer of 256 neurons outputting the flow parameters, one flow layer composed of 4 gaussians, which mixture’s CDF is used as a flow. We used $\mathcal{N}(0, 1)$ as a base distribution. The mean returns were computed over 100 draws from the base distribution. We used a learning rate of 0.001 and $\gamma = 0.9$.

4.5.1 Simple MDPs

In this section, we present the results obtained for three different MDPs showcasing various configurations that can be encountered in any RL environment, as illustrated in figure 4.11. The first MDP is the simplest, comprising a deterministic structure with one initial state and two final states, both accessible through two distinct actions A_1 and A_2 . In this setting, S_1 yields a reward of 1, while S_2 offers a reward of 5. The predicted returns manifest as normal distributions centered on the values 1 or 5.

In the second scenario, we maintain the same deterministic MDP as before but introduce an additional intermediate state that offers a reward of 1. Consequently, the predicted returns for S_1 and S_2 are centered on 1 and 2, respectively. Furthermore, the predicted returns for S_3 and S_4 are centered on $1 + \gamma R(S)$. This specific example effectively demonstrates our model’s capability to predict distributions that consider time discounting.

In the final and most significant scenario, we consider a stochastic MDP where a single action results in different states with a 50% probability each. These states have drastically distinct values. In this case, our model successfully outputs bi-modal distributions centered on the appropriate return values, effectively capturing the inherent uncertainty and variability in the system.

Additionally, we illustrate the scenario where three modes are possible in figure 4.12. While the model accurately predicts the correct modes, we observe that it assigns some mass to areas that should have a null density. This phenomenon arises from our approximation of the Cramèr distance and will also be evident in our experiments on the Frozen Lake environment.

4.5.2 Frozen Lake

The Frozen Lake environment provides a stochastic setting where the presence of multimodalities can be easily visualized. In this environment, an ice skater aims to navigate from the starting point towards the goal while avoiding holes in the ice. Due to the slippery nature of the ice, the agent may occasionally move in directions perpendicular to the intended path. For example, if the agent chooses to go right, it has a probability of 1/3 to move right, 1/3 to move up, and 1/3 to move down. This exaggerated level of stochasticity enables us to observe significant changes in state values depending on the chosen action, making it an ideal scenario for illustration purposes, as it generates pronounced multimodalities in the Q-value distributions. The two configurations, namely 3×3 and 4×4 are shown in figure 4.13.

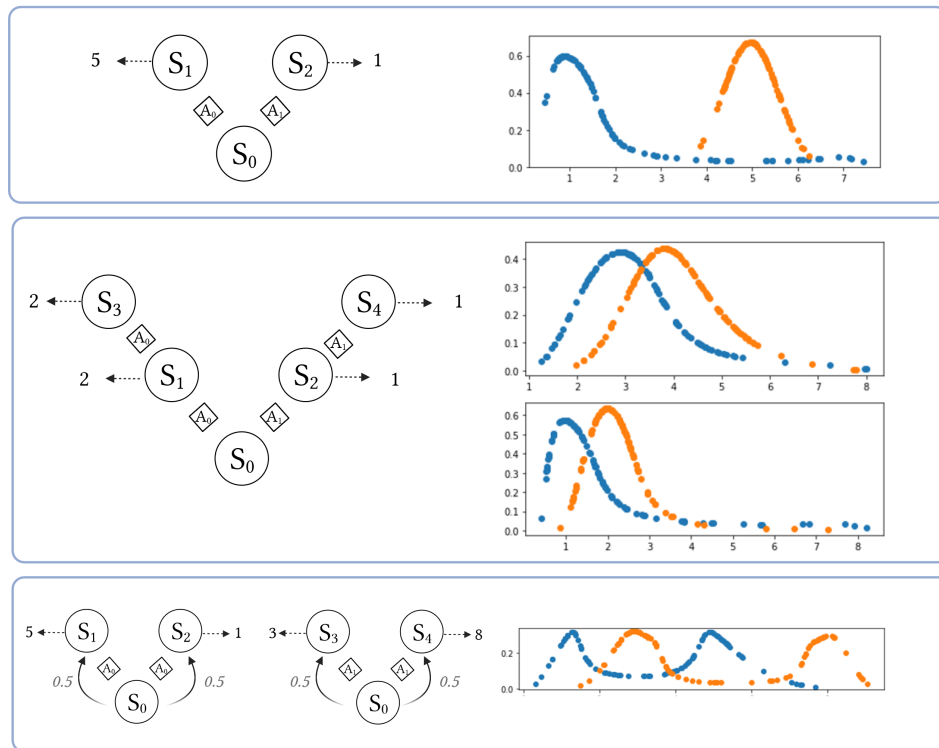


Figure 4.11: Our model predicted return distributions on 3 simple MDP scenarios going from state S_0 and performing either action A_0 or A_1 . The blue distribution represents the return for state-action pair (S_0, A_1) and the orange one represents (S_0, A_0) . **Above:** the agent reaches the final state and the learnt distribution is a gaussian centered around the received reward. **Middle:** the final state is reached after two steps, the model effectively learns distributions centered on the discounted sum of rewards. **Below:** Example on stochastic MDP; we observe that the models learn bimodal distributions for both actions.

In figure 4.14, we depict the predicted returns for a 3×3 grid within the Frozen Lake environment, where the rewards have been shaped to reflect the distance (in the sense on Manhattan distance) of each state to the final state. The lower is the distance, the higher is the reward. This shaping enables the highlighting of multimodalities more effectively. Figure 4.14 exhibits the predicted return distributions for each state-action pair. The state-action pairs exhibiting the highest Q-value (i.e., the expected value of the predicted return distribution) for each state have been highlighted in blue. The policy being deterministic, it always chooses the highlighted action at the corresponding state.

It is notable that state-action pairs with a high probability of falling into a hole, such as State1-right or State4-down, display a mode centered on a return value of 0 with a high probability. If we denote $P(0)$ as the density in a small area around 0, then we observe that $\frac{P(0)}{1-P(0)} = \frac{P(\text{falling})}{P(\text{escaping})}$. This observation sheds light on the contrasting probabilities of falling into a hole versus successfully reaching the goal.

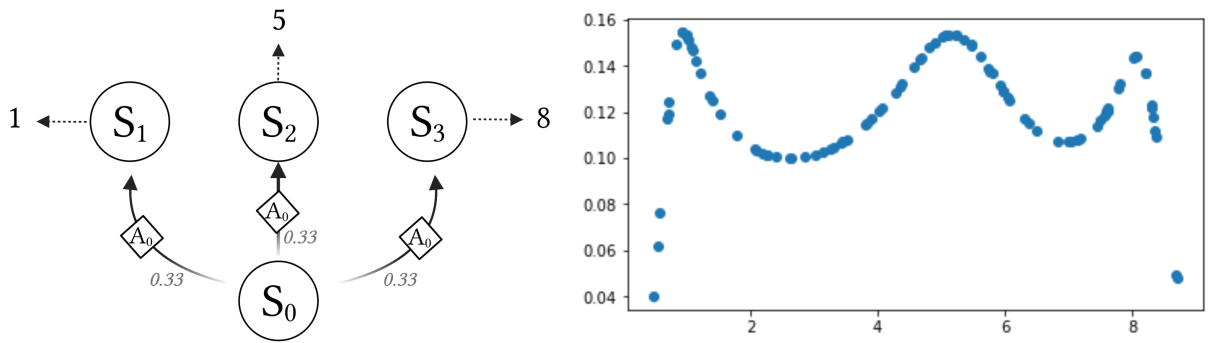


Figure 4.12: Example of stochastic MDP where a single action can lead to 3 different final states inducing 3 different rewards. The model successfully learns the 3 modes but fails to predict null mass for areas between them.

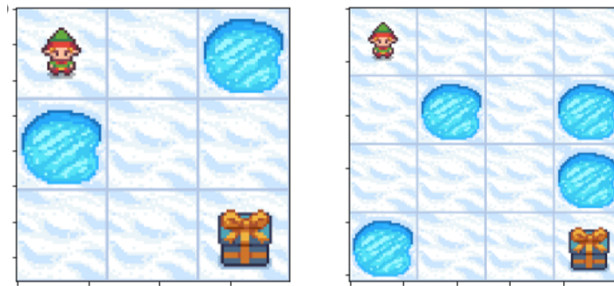


Figure 4.13: The 3×3 and 4×4 Frozen Lake environments.

In figure 4.15, we present results obtained using the standard 4×4 configuration in the Frozen Lake environment, while applying the same reward shaping as before. Notably, all states in this configuration are treated as final states. This straightforward adjustment enables us to predict the state-action pair reward distribution instead of the return distribution. Once again, we observe distinct and pronounced multimodalities in the predicted distributions.

For comprehensive analysis, figure 4.16 illustrates the predicted return distributions for the standard 4×4 configuration in the Frozen Lake environment. In this case, the rewards are not shaped (i.e., set to 0 everywhere except for the goal state), and only final states are marked as done. As for figure 4.14, we highlight the state-action pairs displaying the highest Q-values, showing that the output policy is indeed the optimal one. Our results can be compared with those obtained using C51 algorithm on the same setup (figure 4.17). We observe that in both cases the obtained policies are equivalent. Moreover, comparing the output distributions for each state-action pair, we can observe that the learnt distributions over returns are very similar. Due to high stochasticity of the Frozen Lake environment, both algorithms have trouble converging towards one on the two possible return values (0 or 10) as it is the case for states 2 and 3 for instance.

However, our experimental findings and comparison with C51 have highlighted an issue where the model assigns non-null mass to certain areas that should ideally have no mass. This situation is better handled in C51 where we observe a clear discontinuity between areas as positive mass.

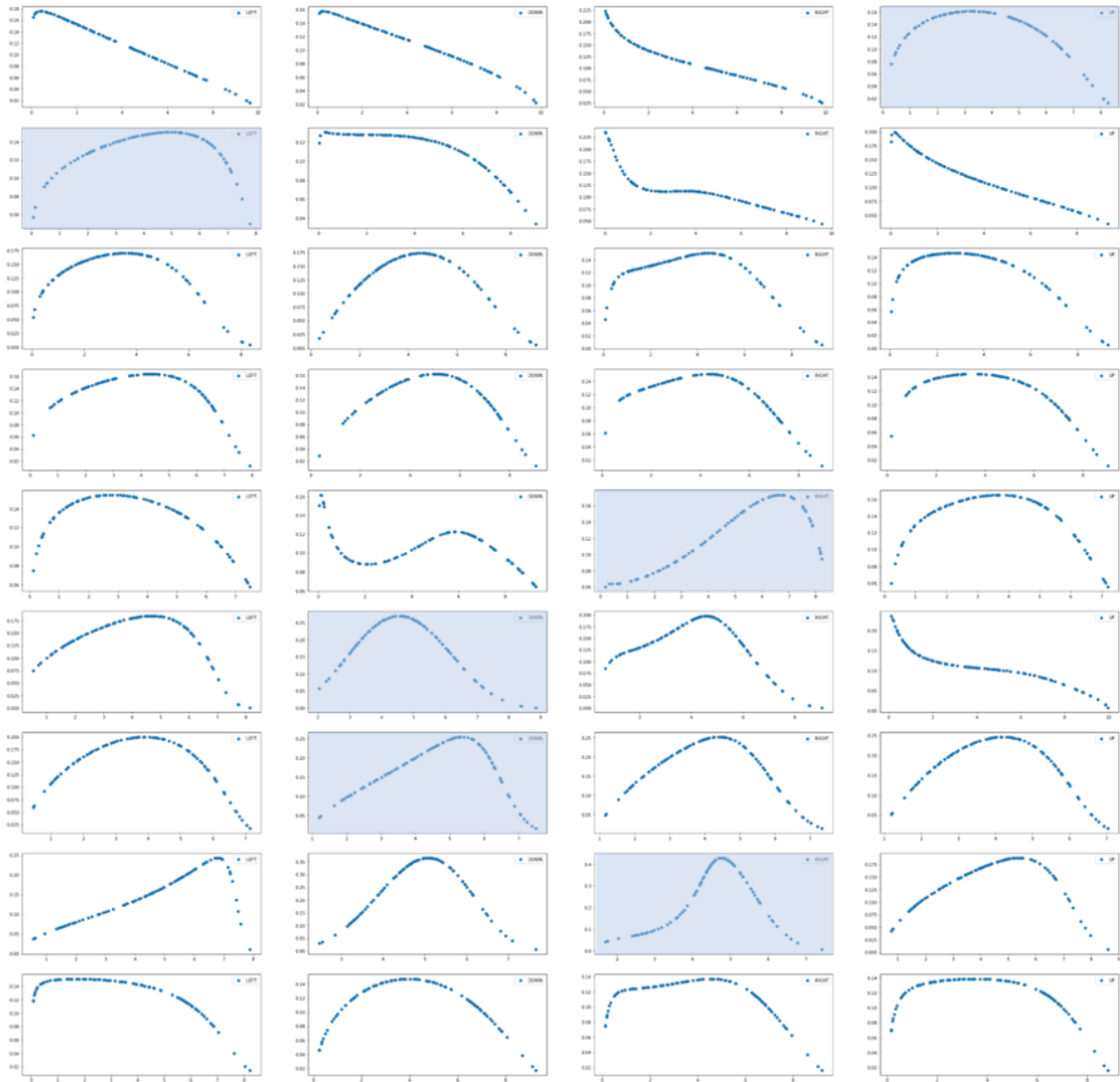


Figure 4.14: Our model predictions for all state-action pairs in the 3×3 Frozen Lake environment. Each line corresponds to a state and each column to an action. States are numbered from 0 to 8 starting from the upper left state and going to the right. The plots display the returns in the range $[0, 10]$ on the x axis and their corresponding predicted density. The state-action pairs displaying the highest Q-values are highlighted in blue.

This issue with our model can be attributed to two primary reasons. Firstly, our utilisation of an approximation of the Cramèr distance as a loss function may be contributing to this behaviour. An interesting avenue for future investigation could involve exploring the adaptability of our model to output distributions' CDFs instead of their PDFs, thereby enabling the calculation of an exact value for the distance.

Secondly, the model outputs larger standard deviation values than necessary for the Gaussian

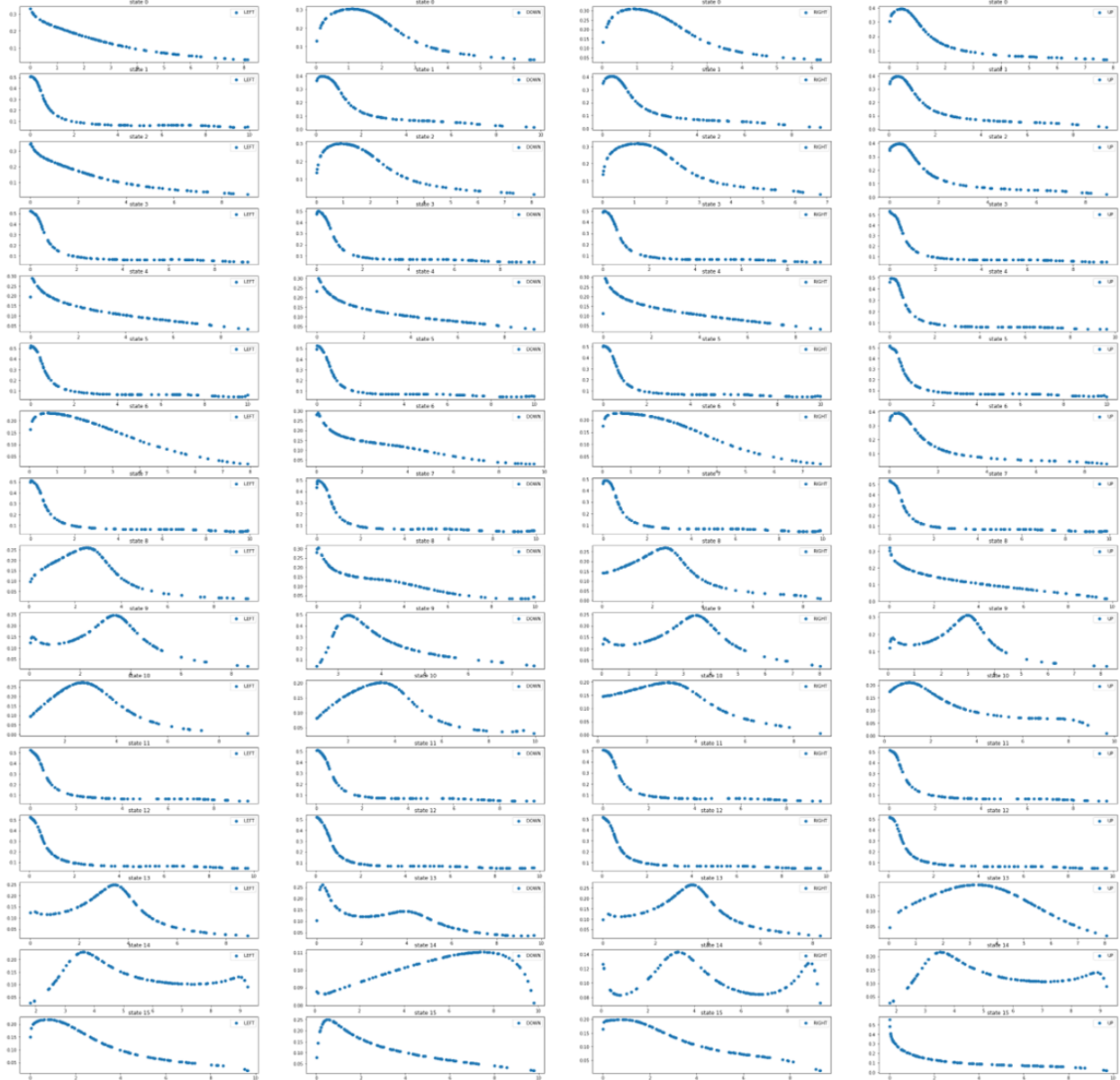


Figure 4.15: Our model predictions for all state-action pairs in the 4×4 Frozen Lake environment with shaped rewards. Each state gives a reward that is inversely proportional to the distance from the final state. Each line corresponds to a states and each column to an action. States are numbered from 0 to 15 starting from the upper left state and going to the right. The plots display the returns in the range $[0, 10]$ on the x axis and their corresponding predicted density. Multimodalities appear clearly on critical states like state9-Left, state13-left or state14-right.

mixture utilised in constructing the flows. Addressing this issue could lead to more accurate and precise predictions of the return distributions.

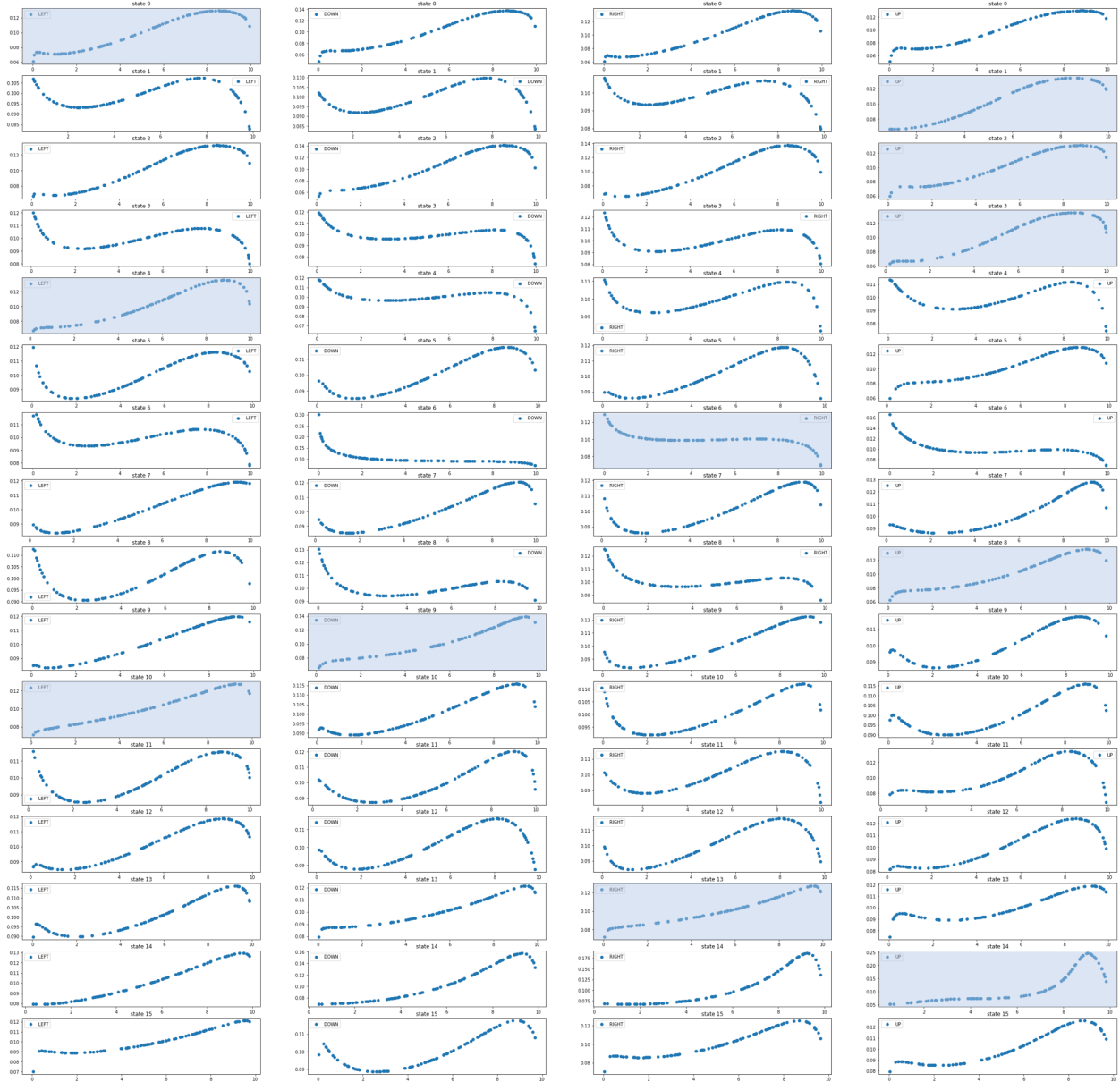


Figure 4.16: Our model predictions for all state-action pairs in the 4×4 Frozen Lake environment with unshaped rewards. All states give a null reward except the goal state. Each line corresponds to a states and each column to an action. States are numbered from 0 to 15 starting from the upper left state and going to the right. The plots display the returns in the range $[0, 10]$ on the x axis and their corresponding predicted density. The state-action pairs displaying the highest Q-values are highlighted in blue.

4.6 Conclusion

In this chapter, we introduced a novel approach to perform Distributional Reinforcement Learning by harnessing the advantages of Normalizing Flows. Unlike existing methods that face limitations of bounded support for return distributions or the inability to utilise proper Integral

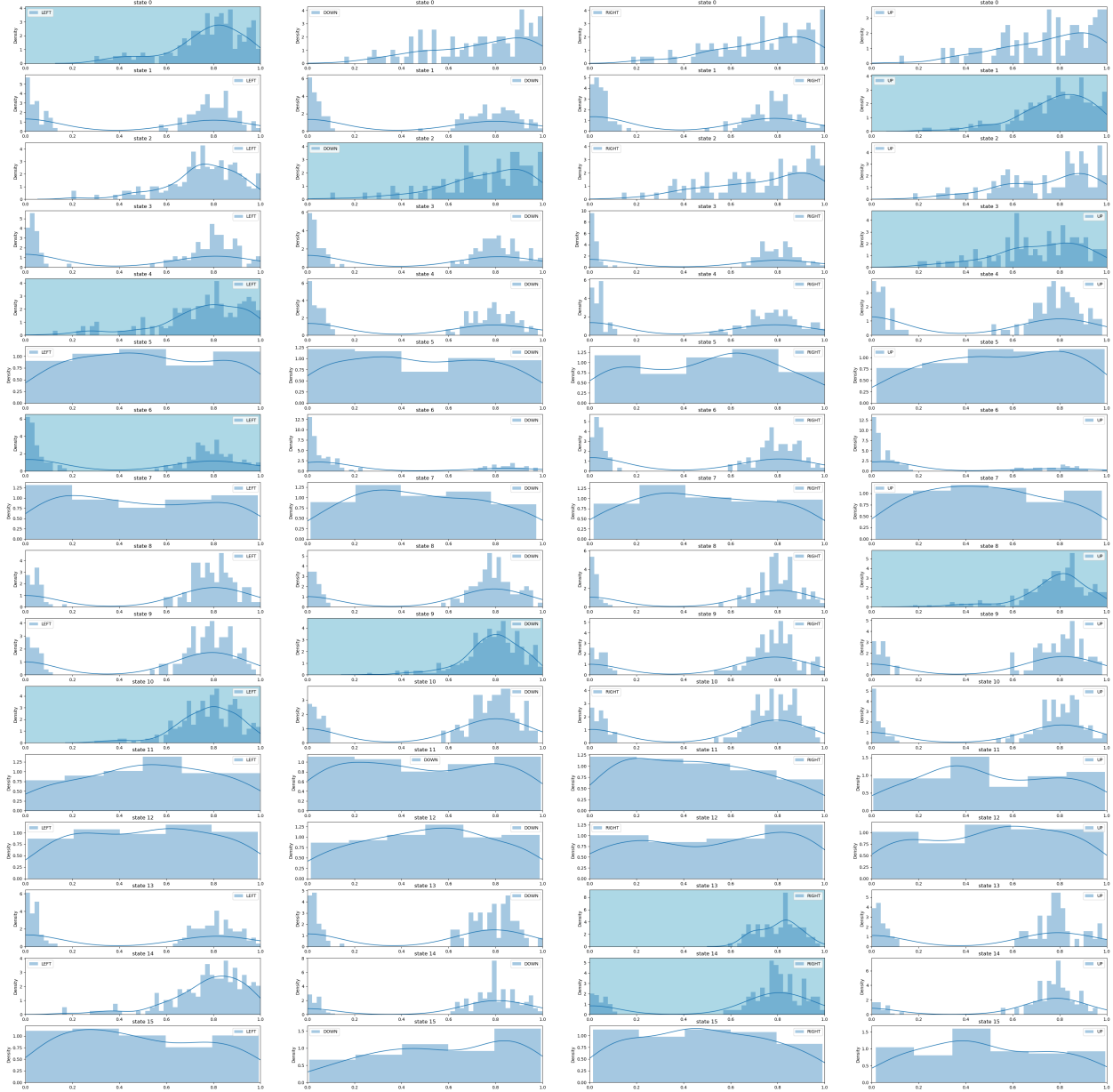


Figure 4.17: C41 predictions for all state-action pairs in the 4×4 FrozenLake environment with unshaped rewards. All states give a null reward except the goal state. Each line corresponds to a states and each column to an action. States are numbered from 0 to 15 starting from the upper left state and going to the right. The plots display the returns in the range $[0, 10]$ on the x axis and their corresponding predicted frequency. To ease the comparison with figure 4.16, a KDE has been used to display densities. The state-action pairs displaying the highest Q-values are highlighted in blue.

Probability Metrics (IPMs) between predicted and target distributions, we demonstrated the feasibility of employing an approximation of the Cramèr distance as an optimisation criterion, even when only sample transitions are available. Moreover, our model facilitates the computation of the density of any return value for any state-action pair, enhancing its practical utility. Lastly, we

proposed a natural method to compute the target distribution by transforming the distributional Bellman operator into a flow, offering a promising direction for future research in the field of Distributional Reinforcement Learning.

In our experiments, we showcased the effectiveness of our model in accurately predicting return distributions and capturing all their potential modes. On the considered environments, our algorithm is on par with the well known C51 algorithm. The incorporation of the Cramèr distance as an alternative to the reverse KL divergence represents a significant advancement in the realm of Distributional Reinforcement Learning. This approach not only enhances the reliability of our algorithms but also contributes to their overall effectiveness, opening up new possibilities for further improvements in this field.

While our approach overcomes the limitations of existing approaches (convergence guarantee and simple evaluation of the probabilities associated with specific return values under a given policy), there is still room for improvement in its application. To further validate and evaluate its performance, the next step involves testing it in more complex environments, such as Atari 2600 games of the Arcade Learning Environment [Bel+13] or Mujoco for robotic simulation [TET12]. Through these experiments, we aim to provide a thorough evaluation and comparison of our model’s performance against existing approaches. Although preliminary results indicate promising performance that is at least on par with C51, further investigation and analysis are required to solidify and confirm these findings. The validation in more challenging environments will offer valuable insights and help refine our model for broader and more demanding applications in reinforcement learning.

We also identified two limitations as our model appears to assign positive mass to areas that should have null mass. Moreover it also learns standard deviation values in the Mixture of Gaussian CDF flow function that are higher than necessary. We argue that this arises from our approximation of the Cramèr distance. By examining and optimising these aspects, we aim to enhance the performance and reliability of our model for a wider range of applications and domains in reinforcement learning.

Chapter 5

Set-Policy Matching Distributional Inverse Reinforcement Learning

In the preceding chapter 4 we used invertible generative models (namely Normalizing Flows) and the Cramèr distance for distributional Reinforcement Learning. In this chapter, we will extend this methodology to Inverse Reinforcement Learning (IRL), which offers the particular challenges that many optimal policies can explain a set of demonstrations and that many rewards can explain an optimal policy. For instance, even an unchanging reward function can be attributed to making any policy optimal, including that of the expert. Among existing approaches the most succeeding ones are based on adversarial models and rely on matching occupancy measures which appear as not ideal for highly stochastic environments. We contribute a collaborative model that is novel in that, unlike other methods, it does not rely on occupancy measures. Moreover, our model learns a reward function that resembles the inverse distance between a state and the nearest optimal path state, which allows it to navigate back from out-of-distribution states.

5.1 Introduction

As explained in chapter 4, distributional approaches are effective against distributional shift. However, in the IRL setting, the risk of distributional shift is even higher while, at the time of the manuscript writing, there is still no distributional approach for IRL. In IRL, the goal is to find the right reward function given expert demonstrations, in order to better mimic the expert behaviour even in situations that the expert might have never encountered in the given demonstrations. This is particularly true when there is a difference between the training and testing environments. Consider the example, adapted from [Had+17], of a robot that has to accurately navigate to a target location. The reward function is learnt using expert demonstrations in an environment where it is only possible to encounter grass lawns and dirt pathways. This situation is plausible if, for instance, the engineer responsible of training the robot and gathering the expert demonstrations expects the robot to only encounter such terrain. The learnt reward function will incentivise moving towards the target quickly, avoiding grass as much as possible. When the robot is deployed into the world, it encounters a novel terrain type. This out of distribution input will surely confuse the robot and might induce a chaotic behaviour. In this case the learnt re-

ward function did not take into account the risk of distributional shift. The engineer supervising the learning process was apparently confident (wrongfully) that the risk of distributional shift was minimal and did not take it into account in the mesa-objective. The relationship between mesa-objective and risk of distributional shift is explained in chapter 3 section 3.6.1.

In [Had+17], the authors offer a useful insight: “the designed reward function should merely be an observation about the intended reward, rather than the definition; and should be interpreted in the context in which it was designed”. Said otherwise, the agent should hold uncertainty about its reward function, instead of treating it as fixed. Based on this observation, we argue in this work that the risk of distributional shift in the context of IRL should be dealt with the same way as for forward RL, namely using a distributional approach, where a return distribution is learnt rather than its sole expected value.

Distributional IRL can also be a good solution for handling the outer-alignment problem (discrepancy between the human objective and the reward function). In this manuscript’s introduction, we drew the hypothesis that using IRL based approaches, it is possible to point to the right behaviour directly using expert demonstrations and help alleviate the outer alignment problem. Building upon the foundations laid out earlier, we will show how a distributional approach to IRL can be effectively employed to tackle the challenges presented by IRL (detailed in section 5.2) while evaluating its outer alignment performance.

Indeed, IRL is a challenging problem as various optimal policies can explain demonstrations and different rewards can satisfy optimality criteria. For instance, in the example given above, both reward functions, the one the robot actually learnt and the one taking into account the new unknown terrain would have displayed the exact same behaviour in the training environment. Adversarial methods [FLA16; FLL18; Xia+19] were proposed to tackle these issues by (refer to chapter 2 section 2.3 for more details):

- (1) incentivising imitation of demonstrated actions
- (2) promoting return to demonstrated states when faced with new situations.

Despite their promise, adversarial methods exhibit instability and depend on matching state occupancy measures. First, as a reminder, we redefine here the concept of occupancy measure. Consider a MDP defined by the usual tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma, \zeta_0)$. The initial state distribution ζ_0 and the reward function $r(s, a)$ are unknown. If we combine this setting with a stochastic policy π from the set of policies Π , i.e. a conditional probability distribution on \mathcal{A} given some state $s \in \mathcal{S}$, we obtain a Markov chain $\mathcal{M}_\pi = (s_0, a_0, s_1, a_1, \dots)$ where $s_0 \sim \zeta_0$, and $a_t \sim \pi(\cdot|s_t)$ and $s_{t+1} \sim p(\cdot|s_t, a_t)$. \mathcal{M}_π has a stationary distribution (or **occupancy measure**) ρ_π which satisfies

$$\rho_\pi(s, a) = (1 - \gamma\pi(a|s)) \sum_{t=0}^{\infty} \gamma^t p(s_t = s)$$

In summary, **the occupancy measure is the stationary distribution of states visited by the considered agent**. It can also be called the state visitation frequency.

Methods based on occupancy measures are problematic in highly stochastic environments as they do not necessarily learn policies that perfectly mimic the the expert’s behaviour. Indeed,

as illustrated in figure 5.1, two opposite policies can yield identical occupancy measures. We propose a collaborative model, inspired by Distributional Reinforcement Learning, that does not rely on occupancy measures while avoiding the drawbacks of adversarial training. Our model leverages a recent finding establishing a one-to-one correspondence between optimal Q-functions and reward functions thus enabling to match the expert’s actions as in Imitation learning (1) while ensuring (2) as in IRL approaches. In our experiments on the stochastic Frozen Lake environment, the learned reward resembles the inverse distance between a state and the nearest optimal path state, validating the effectiveness of our approach. This collaborative approach captures benefits (1) and (2) of adversarial methods while circumventing their drawbacks.

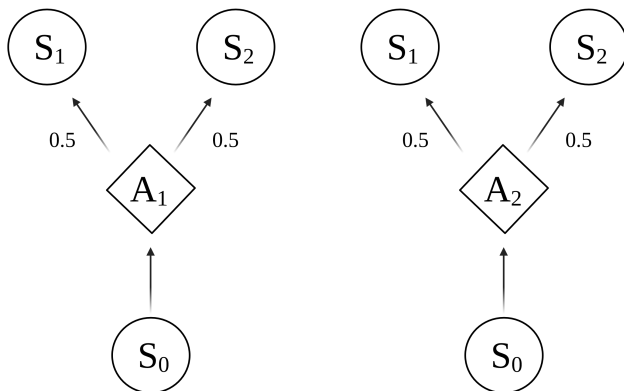


Figure 5.1: Given a stochastic MDP, one agent takes action A_1 while the other takes action A_2 . The two actions both lead to the same states with the same probabilities. In the occupancy measure framework, both actions are equivalent but it is possible that only A_1 is part of the expert’s set policy. In partially observable MDPs, S_1 and S_2 states reached by the two agents may not be equivalent, which makes fitting the set-policy even more crucial.

In section 5.2.1, we will first detail the challenges posed by the IRL problem and the existing approaches to tackle them. Then in section 5.2.2, along with the limits of existing approaches, we also detail the main issues encountered when using occupancy measures to learn reward functions. In section 5.3, we first explain why an IRL model should share common principles with Imitation Learning (IL). Then in sections 5.3.1 and 5.3.2, we show why it is possible to learn optimal reward functions by learning return distributions, thus avoiding to fit occupancy measures. In section 5.4 we present our set-policy matching Distributional IRL model. Finally in section 5.5, we present our experimental results using the Frozen Lake environment.

5.2 How Existing Approaches Solve IRL Challenges and What Are Their Limits

In this section we detail the challenges posed by IRL and how they are tackled by existing approaches before exposing their strengths and limitations.

5.2.1 Challenges and Prior Work in IRL

In contrast to the approach of directly replicating behaviour, IRL assumes rational agents to estimate an unknown reward function that represents their underlying motivations and goals. The reward function is often considered as the most succinct, robust and transferable representation of the expert’s objective [AN04]. Given a set of demonstrations D_E from an expert policy π_E , IRL [Rus98; NR00] is the problem of seeking a reward function from which we can recover π_E through RL. However, IRL in unregularized MDPs has been shown to be an ill-defined problem since:

1. many optimal policies can explain a set of demonstrations
2. multiple rewards meet the criteria of being a solution, i.e many rewards can explain an optimal policy

Maximum Entropy Inverse Reinforcement Learning (MaxEntIRL) [Zie+08; Zie10] offers a strategy to mitigate the first concern. This approach involves a reward function that not only maximises the expert’s cumulative return but also incorporates the Shannon entropy of the expert policy.

The second challenge proves to be more intricate. Indeed discerning genuine reward functions from those influenced by the dynamics of the environment is not easy. The complexity is further compounded by the existence of degenerate reward functions¹. Specifically, the problem arises from the fact that IRL algorithms encounter difficulties in differentiating between authentic reward functions and those that are influenced by environmental dynamics. Moreover, a reward function that remains constant across all state-action pairs is capable of rationalising the behaviour of any expert. In essence, an unchanging reward function can be attributed to making any policy optimal, including that of the expert.

Recently, the incorporation of adversarial techniques into the MaxEntIRL framework has emerged as a promising avenue to address these challenges, leading to remarkable experimental outcomes [FLA16; HE16; FLL18]. For instance, the Guided Cost Learning (GCL) approach [FLA16] introduces a novel framework based on Generative Adversarial Networks (GANs) that directly learns the policy by aligning occupancy measures. This approach effectively simulates the process of conducting RL following an IRL phase, thereby retaining the advantages inherent to IRL while enhancing computational feasibility. Indeed, MaxEntIRL methods usually require to know the environment’s dynamics in order to compute the partition function (refer to chapter 2, section 2.3 for more details). In large state space environments or if the dynamics are unknown, this computation is either impossible or untractable. GCL uses an iterative approach estimating the reward and partition functions sequentially, outputting a better estimation of each one every time the approximation of the two gets better. GCL operates in a manner akin to conducting RL subsequent to IRL, thus capitalising on the strengths of both paradigms. Adversarial IRL methods train an RL agent not only to imitate demonstrated actions, but also to visit demonstrated states.

Intuitively, adversarial methods encourage long-horizon imitation by providing the agent with

1. an incentive to imitate the demonstrated actions in demonstrated states,

¹A reward function is considered degenerate if it outputs the same value for all transitions

2. an incentive to take actions that lead it back to demonstrated states when it encounters new, out-of distribution states.

5.2.2 Weakness of Adversarial Methods

The reason why adversarial methods tend to perform better than simpler approaches like Behavioral Cloning (BC) is because BC mainly focuses on copying demonstrated actions (point 1), while adversarial methods like GCL and AIRL [FLL18] take a more comprehensive approach by imitating actions in demonstrated states (point 1) and also making sure to return to demonstrated states when facing new situations (point 2). However, a first limit appears as, when facing unknown situations, these methods encourage coming back to the demonstrated states, which may not always be the optimal strategy.

Moreover, particularly in environments characterised by high levels of stochasticity, the alignment of occupancy measures may not necessarily result in learned policies that perfectly mimic the expert’s behaviour. This is underscored by an illustrative example depicted in figure 5.1, which portrays a simple MDP wherein two opposing policies yield identical occupancy measures.

Finally, adversarial methods like GCL or AIRL, try to match the agent’s and expert’s occupancy measure, by minimising the JS divergence between those distributions. As stated in chapter 4 Proposition 1, The JS divergence as well as the KL divergence, have unbiased sample gradients, but are not scale sensitive. A natural follow up is to try to optimise a proper distance between both distributions. Authors of [Xia+19] propose to minimise the Wasserstein distance instead. However, In chapter 4 Proposition 2, we also mentioned that the Wasserstein metric does not have unbiased sample gradients making it impossible to optimise using SGD in RL context. Moreover, as shown in [Bel+17], even the dual form of the Wasserstein distance has biased sample gradients.

5.3 Learning Reward Functions Using Returns and the Link between IL and IRL

In contrast to the adversarial approaches mentioned in section 5.2.1, we take a different route based on insights from the Distributional RL approach presented in chapter 4. We advocate for the adoption of the Cramèr distance, which we utilised successfully in the Distributional RL context. We propose to use the primal form of the Cramèr distance, with the aim of aligning the return distributions of the expert and the agent for each specific state-action pair rather than their occupancy measure. By aligning return distributions, we mean finding the reward function such that the expert Q-value (and only the expert’s) is the highest and such that the distance between the agent return distribution and the expert’s is minimised. This approach is more straightforward and manageable compared to matching occupancy measures, as it requires training on individual state-action pairs rather than entire trajectories. However, it is imperative to ask whether this return distribution matching strategy effectively resolves the IRL problem while circumventing degenerate solutions. Indeed, in contrast to previous methods that resort to the maximum entropy principle to distinguish and prevent degeneracy in potential reward solutions, we seek to demonstrate that the return distribution alignment itself holds the potential to tackle the issue of degenerate reward solutions.

In this section, we will dive deeper in the relationship between reward functions and Q-functions. As shown in figure 5.1, opposite policies may yield the same occupancy measure in highly stochastic environments. We contend that a more effective approach involves aligning the expert’s actions, akin to the principles of IL, rather than merely focusing on occupancy measures. This means to match the expert’s and agent’s set policies as defined in definition 1 rather than their occupancy measure. As we advocate for a method aligning directly with expert actions, we argue that such IRL model should share common principles with IL.

The authors of [PGP17] establish a connection between IL and IRL, shedding light on the value of incorporating an IL perspective within an IRL framework. This connection between IL and IRL provides a compelling rationale for prioritising the alignment of expert actions in the IRL context, particularly when navigating the complexities of stochastic environments. Is it possible to use the link to solve the IRL problem without relying on occupancy measures?

In the next sections we will answer to the following questions:

1. Matching the expert’s and agent’s actions implies matching their expected returns. However, is there an equivalence between matching the agent’s and expert returns and finding the optimal reward function?
2. if so, how to avoid the issue of degenerate reward functions mentioned in section 5.2.1?

In this section, based on the works of Bilal Piot, Mathieu Geist and Olivier Pietquin [PGP17], by leveraging their concept of set-policy, we answer positively to the first question by characterising the bijection between the set of optimal reward functions and optimal Q-functions. First, in section 5.3.1 we show that finding a Q-function maximised by the expert enables to find the optimal reward function. This means that by aligning the agent and expert returns, it is possible to retrieve the optimal reward function. Then, in section 5.3.2, we show, using the set-policy framework, that there exist reward functions that induce the expert actions. This result will allow us to draw 2 conclusions: (1) It is possible through an iterative process on return distributions to learn both the optimal Q-function and the optimal reward functions; (2) learning these two functions is possible without using occupancy measures, making it possible to match directly the expert’s and the agent’s set policies. The second question is answered to in section 5.4. As before, we consider a MDP, defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma, \zeta_0)$.

5.3.1 Aligning Expert and Agent Returns in Order to Determine the Optimal Reward Function

Recall that the aim of IRL is to calculate a reward function r for which all expert’s actions and only expert’s actions are optimal:

$$\forall s \in \mathcal{S}, \arg \max_{a \in \mathcal{A}} Q_r^*(s, a) = \text{Supp}(\pi_E(\cdot|s)) \quad (5.1)$$

where Q_r^* is the optimal Q-function under the reward function r . This means that the support of the expert’s policy can be obtained while being greedy on the Q-function. The Q-function can therefore be interpreted as an optimal quality function with respect to the reward r if:

$$r(s, a) = Q_r(s, a) - \gamma \mathbb{E}_{P(\cdot|s,a)} \left[f_Q^* \right] \quad (5.2)$$

with $\forall s \in S, f_Q^*(s) = \max_{a \in A} Q(s, a)$

Defining the operator T_R^* as:

$$T_r^* Q(s, a) = r(s, a) + \gamma \mathbb{E}_{P(\cdot|s,a)} [f_Q^*]$$

It is also possible to show that Q_r^* is a fixed point for the operator T_r^* :

$$Q_r(s, a) = r + \gamma \mathbb{E}_{P(\cdot|s,a)} [f_Q^*] = T_r^* Q(s, a) \quad (5.3)$$

We conclude with Banach's fixed point theorem that as the fixed point of T_r^* is unique, then $Q = Q_r^*$. The fact that a Q function verifying this equality can be interpreted as an optimal Q function Q_r^* means that the expert's actions and only the expert's actions are optimal according to the reward r . Therefore, r can be seen as the target of an IRL method. We therefore showed that finding a Q-function maximised by the expert enables to find the optimal reward function. Moreover it also proves that the standard Q-learning update displayed in equation 5.3 is a viable way to learn a reward function in an IRL setting. The immediate consequence is that occupancy measures are not necessary anymore.

5.3.2 The Set of Optimal Reward Functions Induces the Expert's Set-Policy

In the preceding section, we demonstrated the feasibility of learning a reward function through a methodology akin to the conventional Q-learning update. Nevertheless, while the traditional Bellman operator facilitates the acquisition of Q-values to make informed action choices based on a reward function, this section introduces the inverse Bellman operator. This operator, conversely, enables the acquisition of the correct reward function when optimal actions are known.

In equation 5.1, the optimal Q-function was defined as the one for which all expert action and only expert actions are optimal for any state. To give a more formal definition to the set of optimal actions we introduce the concept of set policy as defined in [PGP17]:

Definition 1: Set-policy

A set policy $\bar{\pi}$ is an element of the set $\bar{\Pi} = (\mathcal{P}(A) \setminus \emptyset)^{|S|}$. To any policy π one can associate a set policy $\bar{\pi}$: $\forall s \in S, \text{Supp}(\pi(\cdot|s)) = \bar{\pi}(s)$. $\bar{\pi}$ indicates for each state, the set of actions that can be chosen by π .

Definition 2: Generated set-policy

To any MDP \mathcal{M}_r , one can associate a particular set policy named optimal set-policy generated by \mathcal{M}_r , denoted $\bar{\pi}_r^* \in \bar{\Pi}$:

$$\forall s \in S, \bar{\pi}_r^*(s) = \arg \max_{a \in A} [Q_r^*(s, a)]$$

The optimal set-policy generated by \mathcal{M}_r indicates for each state the set of optimal actions to choose to optimise the reward r .

To solve the IRL problem, it is necessary to find a reward r such that $\bar{\pi}_r^* = \bar{\pi}_E$.

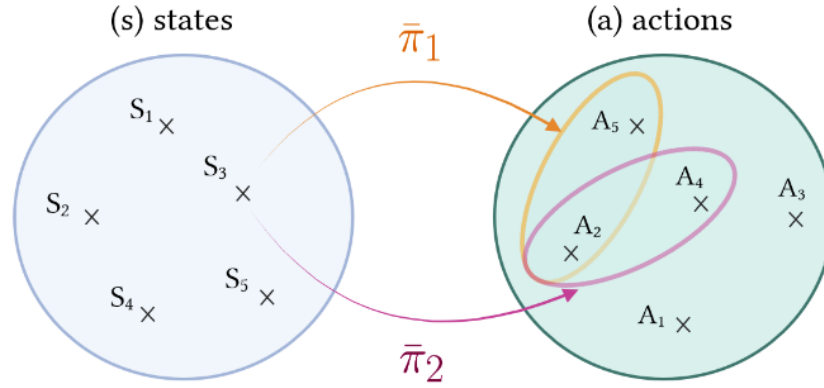


Figure 5.2: Illustration of set-policy. Given a state space \mathcal{S} and an action space \mathcal{A} , a set policy is a mapping that associates for each $s \in \mathcal{S}$, a finite set of actions in \mathcal{A} .

Therefore equation 5.1 can be rewritten in terms of set-policies and we define $H_{\bar{\pi}}$ the set of Q functions such that:

$$H_{\bar{\pi}} = \{Q, \forall s \in \mathcal{S}, \arg \max_{a \in \mathcal{A}} Q(s, a) = \bar{\pi}(s) = \text{Supp}(\pi(\cdot|s))\}$$

Definition 3: Inverse Bellman Operator

$\forall Q \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}, \forall (s, a) \in \mathcal{S} \times \mathcal{A}$, we define the inverse optimal operator J :

$$J^* Q_r(s, a) = Q_r(s, a) - \gamma \mathbb{E}_{P(\cdot|s,a)}[f_Q^*]$$

J maps a Q -function to a reward function the same way as in equation 5.2. Conversely, its inverse maps a reward function to the associated Q -function:

$$\forall r \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}, (J^*)^{-1} r = Q_r^*$$

Therefore, applying J to the set of optimal Q -function greedily defining a given set policy, $H_{\bar{\pi}}$, we can show that the image is the set of reward functions r such that the optimal Q -function under r induces the considered set policy.

$$J^*(H_{\bar{\pi}}) = \{r \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}, \forall s \in \mathcal{S}, \arg \max_{a \in \mathcal{A}} [Q_r^*(s, a)] = \bar{\pi}(s)\}$$

Replacing $\arg \max_{a \in \mathcal{A}} [Q_r^*(s, a)]$ by its corresponding generated set-policy, we get:

$$J^*(H_{\bar{\pi}}) = \{r \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}, \bar{\pi}_r^* = \bar{\pi}\} = C_{\bar{\pi}}$$

The same way, we can show that $J^*(H_{\bar{\pi}_E}) = C_{\bar{\pi}_E}$ and therefore the set $C_{\bar{\pi}_E}$ is the image of $H_{\bar{\pi}_E}$ with respect to the operator J^* .

As J^* is a bijection, then $\{C_{\bar{\pi}}\}_{\bar{\pi} \in \bar{\Pi}}$ is a finite partition of $\mathbb{R}^{S \times A}$, which guarantees that $C_{\bar{\pi}_E}$ is not empty. If $C_{\bar{\pi}_E}$ is not empty, then there exist reward functions r such that the optimal set policies generated by r coincide with the expert set-policy. In conclusion, we have shown that the IRL problem admits a solution in the set-policy framework.

5.4 Set-policy Matching Distributional IRL

In this section we will detail our proposed approach to solve the IRL problem without relying on occupancy measures while leveraging the distributional method we introduced in chapter 4. As a reminder, the goal of IRL is to find the optimal reward function such that the expert’s actions (and only those) bring the highest possible return. In section 5.3, we showed that **the tasks of matching the learning agent’s and expert returns and finding the optimal reward function are equivalent**. We design a model that verifies this statement empirically. Said otherwise, our model finds the optimal reward function by minimising the Cramèr between the agent’s and expert’s return distributions. By doing so, we aim to show that the learnt policy is identical to the expert’s and avoids the pitfalls of existing methods relying on occupancy measures. The used process is detailed in section 5.4.1. Furthermore, another aim of our model is to handle the issue of degenerate reward functions. This issue is handled in section 5.4.2.

Similar to cited adversarial methods, we propose an approach involving two different models, one responsible for generating rewards for specific state-action pairs, and another serving as an agent for decision-making. However, a major difference is that our model does not try to match occupancy measures but makes use of the link between IL and IRL to match return distributions and deduce the optimal reward function as well as the optimal Q function. For this purpose we use the Cramèr distance instead of the Wasserstein distance. A significant departure from adversarial frameworks is that our model operates in a collaborative manner rather than engaging in adversarial optimisation. We believe this feature brings more stability to the learning process, which is a significant drawback of adversarial approaches. Moreover, our approach facilitates a direct match between the expert’s set policy and that of the learner, akin to principles observed in IL. This alignment encourages the learner to navigate unfamiliar, out-of-distribution states by incentivising actions that lead back to demonstrated states.

5.4.1 A Collaborative Distributional Model for Learning Both the Optimal Reward and Q-functions

Consider $Z^\pi(s_t, a_t)_{DE}$ as the distribution of returns for a specific state-action pair sampled from the expert’s experience. $Z_\pi(s, a)$ is the value distribution, a mapping that associates state-action pairs with distributions over returns. We recall the Bellman distributional equation:

$$\mathcal{B}_r^\pi Z^\pi(s, a) \stackrel{D}{=} r(s, a) + \gamma T_\pi Z^\pi(s, a) \tag{5.4}$$

Where $T^\pi : \mathcal{Z} \mapsto \mathcal{Z}$ the transition operator:

$$\begin{aligned} T^\pi Z^\pi(s, a) &\stackrel{D}{=} Z^\pi(s', a') \\ s' &\sim \mathcal{T}(\cdot|s, a), a' \sim \pi(\cdot|s) \end{aligned} \quad (5.5)$$

where $Y \stackrel{D}{=} U$ denotes equality of probability laws, that is the random variable Y is distributed according to the same law as U.

This operation entails a shift and scale transformation, where the shift is induced by $r(s, a)$ —the reward acquired upon executing action a in state s . As illustrated in figure 5.3, maximising this shift corresponds to enhancing the expected value of the return distribution. To elaborate further, let r_ϕ represent a reward function parameterised by ϕ , and let f_θ be a flow model that estimates distributions of return values instead of merely expected values, following the same methodology as detailed in chapter 4. This process renders the mapping associating state-action pairs with return distributions dependent on both θ and ϕ , denoted as $Z_{\theta, \phi}$. Given the state-action pairs D_E and D_π from the expert and the learner experiences respectively, maximising

$$d(Z_{\theta, \phi}^*(s_t, a_t)_{D_E}, \mathcal{B}_r^\pi Z_{\theta, \phi}(s_t, a_t)_{D_E}) \quad (5.6)$$

for any IPM d , while holding θ constant equivalently corresponds to maximising r_ϕ for the expert's state-action pairs (see figure 5.3). However, the maximisation of r_ϕ for the expert's state-action pairs alone is not sufficient; it must be complemented by the minimisation of r_ϕ for the agent's state-action pairs.

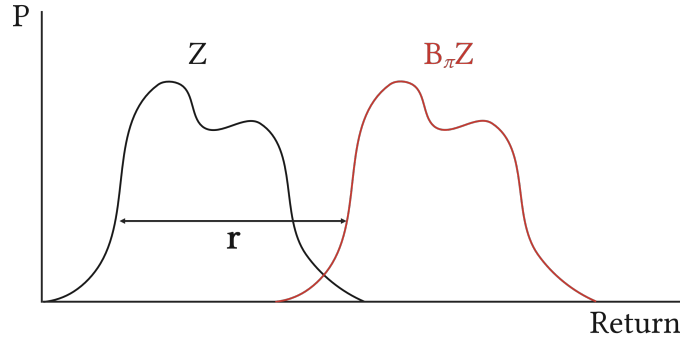


Figure 5.3: A predicted return distribution Z and its corresponding target distribution $\mathcal{B}^\pi Z$ after receiving the reward r . Maximising the distance d displayed in equation 5.6, is equivalent to maximising the reward.

For fixed θ , maximising

$$\mathbb{E}_{s_t, a_t \sim D_E} d(Z_{\theta, \phi}(s_t, a_t), \mathcal{B}_r^\pi Z_{\theta, \phi}(s_t, a_t))$$

and minimising

$$\mathbb{E}_{s_t, a_t \sim D_\pi} d(Z_{\theta, \phi}(s_t, a_t), \mathcal{B}_r^\pi Z_{\theta, \phi}(s_t, a_\pi))$$

implies maximising r_ϕ for the expert state-action pairs and minimising it for the model's state-action pairs.

In the mean time for a fixed r_ϕ function, the learner is trained to minimise

$$\mathbb{E}_{s_t, a_\pi \sim D_\pi} d(Z_{\theta, \phi}(s_t, a_\pi), \mathcal{B}_r^\pi Z_{\theta, \phi}(s_t, a_\pi))$$

We notice here an interesting property of our model. While GAIL or GCL leverage their adversarial architecture, our approach also employs both a discriminator/critic for the reward and a generator for the agent but they both share the same objective, succinctly expressed as:

$$\min_{\theta, \phi} \mathbb{E}_{s_t, a_\pi \sim D_\pi} d(Z_{\theta, \phi}(s_t, a_\pi), \mathcal{B}_r^\pi Z_{\theta, \phi}(s_t, a_\pi))$$

Hence, our model operates in a collaborative rather than adversarial manner, a characteristic we contend enhances its stability compared to adversarial counterparts.

To summarise, the reward model seeks to minimise the following objective:

$$\mathcal{L}(\phi) = \mathbb{E}_{s_t, a_t \sim D_\pi} d(Z_{\theta, \phi}(s_t, a_t), \mathcal{B}_r^\pi Z_{\theta, \phi}(s_t, a_\pi)) - \mathbb{E}_{s_t, a_t \sim D_E} d(Z_{\theta, \phi}(s_t, a_t), \mathcal{B}_r^\pi Z_{\theta, \phi}(s_t, a_t))$$

While the agent seeks to minimise this one:

$$\mathcal{L}(\theta) = \mathbb{E}_{s_t, a_\pi \sim D_\pi} d(Z_{\theta, \phi}(s_t, a_\pi), \mathcal{B}_r^\pi Z_{\theta, \phi}(s_t, a_\pi))$$

5.4.2 Handling Degenerate Reward Functions

For now we answered to the question 1 of section 5.3, but we still have to answer to the question 2. A corollary to question 2 is whether our approach confines the problem to a unique optimal reward function. Unfortunately, that is not the case. The cardinality of the set $H_{\bar{\pi}}$ is infinite, thus signifying the presence of an infinite number of optimal Q functions. Given the established bijection between H and C , we deduce that the collection of optimal reward functions is equally boundless. Nonetheless, the pertinent question remains: Does our model effectively avert degenerate solutions, wherein any constant function could rationalise the expert's behaviour?

The critic can be conceptualised as a separation mechanism that projects both D_E and D_π into a space that maximises the distance between them. In this configuration, the reward r signifies the distance between a specific point from either D_E or D_π and the hyperplane that segregates the two sets, analogous to a Maximum Margin approach as illustrated in figure 5.4.

With this interpretation, $C_{\bar{\pi}_E}$, the set of reward functions such that the optimal set-policy is the expert's, is also the set of reward functions that make D_π and D_E perfectly distinguishable and separated. Among this set of functions, there exists a reward function r_{min} such that the distance between D_E and D_π is minimal, yet the two sets remain perfectly separated. Consequently, any other reward function in $C_{\bar{\pi}_E}$ such that the distance between D_π and D_E is superior to the one output by r_{min} is deemed acceptable as D_π and D_E remain perfectly separated.

The immediate consequence is that constant reward functions for all states and actions are excluded, avoiding degenerate solutions. Indeed, in order to separate D_π and D_E , the reward associated to state-action pairs from D_E has to be different (in fact it is greater) from the one associated to state-action pairs from D_π . An extreme case would be a reward function that can only take

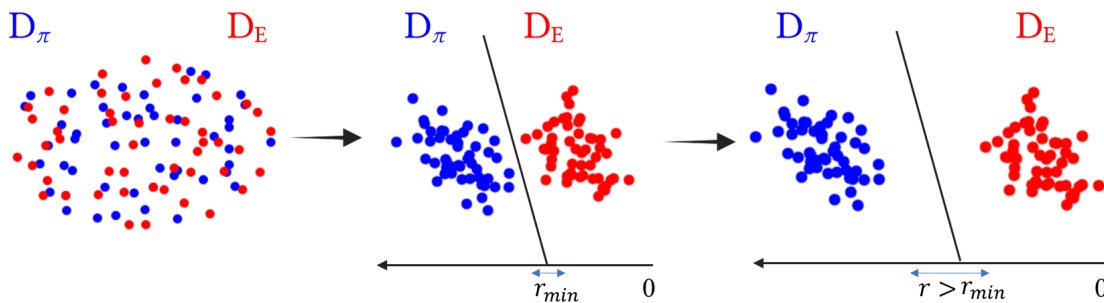


Figure 5.4: Given two sets of demonstrations D_E and D_π , the reward function determines the distance between their respective return distributions. There exist a reward threshold that guarantees complete separation of the two corresponding return distributions. Above this threshold there is no improvement in classification performance between D_E and D_π .

two values, a value r_E for expert state-action pairs and another value r_π for other state-action pairs. Even in this case, there is a guarantee that $r_E > r_\pi$.

Finally, we can note that this guarantee is also consolidated by the loss function $\mathcal{L}(\phi)$ which aims to maximise the difference between the reward given to the expert state-action pairs and the one given to the learning agent's; hence favouring the separation gap between D_E and D_π . In summary, our model and its associated loss function guarantee by design the absence of degenerate reward functions.

5.4.3 Model Overview

To summarise, there are 2 successive steps in this iterative process. In the first step, the model updates the reward function by minimising $\mathcal{L}(\phi)$ in order to maximise the distance between the expert's return distribution and the agent's. In the second step, the agent is updated using the new updated reward function. More formally, the process goes as follows:

1. For a fixed $Z_{\phi,\theta}$, find r_ϕ such that $\bar{\pi}_{r_\phi}^* = \bar{\pi}_E$. This is done by minimising $\mathcal{L}(\phi)$
2. For a fixed r_ϕ , find $Z_{\phi,\theta}$ such that $\arg \max_a Z_{\phi,\theta}(s, a) = \bar{\pi}_E(s), \forall s \in \mathcal{S}$. This is done by minimising $\mathcal{L}(\theta)$

We previously stated that our approach would provide (1) an incentive to imitate the demonstrated actions in demonstrated states, and (2) an incentive to take actions that lead it back to demonstrated states when it encounters new, out-of-distribution states. The first step above tackles (1) and the second step tackles (2) while avoiding compounding errors. Our method is detailed in Algorithm 5.

Algorithm 5 Our method: Collaborative IRL

Require: π_θ Learning agent, R_ϕ : reward model, γ : the discount factor, D_E : expert trajectories, epochs: number of epochs, iter: number of learner’s training iterations.

Initialise π_θ as in Algorithm 4 of chapter 4.

for epoch in epochs **do**:

 Sample trajectories D_π using π_θ

 Sample batch \bar{D}_π from D_π

 Sample batch \bar{D}_E from D_E

$r(s, a)_{\bar{D}_E} \leftarrow R_\phi(s, a)_{\bar{D}_E}$ ▷ evaluate the reward of each state-action pair

$r(s, a)_{\bar{D}_\pi} \leftarrow R_\phi(s, a)_{\bar{D}_\pi}$ ▷ using the current reward model

 Append $r(s, a)_{\bar{D}_E}$ and $r(s, a)_{\bar{D}_\pi}$ to \bar{D}_E and \bar{D}_π respectively

$Q \leftarrow \pi_\theta(s)_{\bar{D}_\pi}$ ▷ estimate the return distribution for each state-action pair

$Q_E, \log(P(Q_E)) \leftarrow \pi_\theta(s)_{\bar{D}_E}$

 Compute \hat{Q}_π and \hat{Q}_E the target Q distributions for the learner’s and expert’s trajectories using algorithm 4 of chapter 4.

 Update ϕ using $\mathcal{L}(\phi)$ and the Cramèr distance as IPM

for i in iter **do**

 Train π_θ using the same procedure as in chapter 4 and $\mathcal{L}(\theta)$

end for

end for

5.5 Experimental Results

As stated in the previous section, putting aside its simplicity and the stability allowed by collaborative models, our approach brings much of its benefits (i.e. learning the expert’s set-policy, aligning returns rather than occupancy measures, being more robust to out of distributions states, more stable than adversarial approaches) when confronted to highly stochastic environments as it does not rely on matching occupancy measures. Indeed aligning the expert and the agent’s return distributions as in Distributional RL ensures that the expert’s value distribution remains the optimal one under the learnt reward function. This has the benefit to avoid using occupancy measures (1) as it can present several flows as shown in figure 5.1 and guaranteeing that the learner and the expert share the same set-policy (2). In order to demonstrate the benefits (1) and (2) of our model, we will test again our model on the Frozen Lake environment while using a model that was trained using our distributional approach as an expert. We also compare the predicted reward values predicted by our model with the one predicted by the GCL algorithm that we use as baseline.

For the learning agent architecture, we used the same architecture as the expert, i.e. an architecture consisting of one hidden layer of 256 neurons outputting the flow parameters, one flow layer composed of 4 gaussians, which mixture’s CDF is used as a flow. We used $\mathcal{N}(0, 1)$ as a base distribution. The mean returns were computed over 100 draws from the base distribution. We used a learning rate of 0.001 and $\gamma = 0.9$. For the reward model, we used a feedforward neural network of one hidden layer of 64 neurons with a ReLU activation function. Empirically we noticed that our model achieves better results when the possible rewards are bounded, hence we used tanh



Figure 5.5: The 4×4 Frozen Lake environment.

scaled in $[-1, 1]$ as an activation function for the final layer with a unique unit. The model was trained for 500 epochs.

Figure 5.6 presents the learnt return distributions for each state-action pair on the standard stochastic Frozen Lake environment. It can be noticed that those distributions are very similar to those presented in figure 4.16 of chapter 4. The optimal action given the learnt reward function for each state is highlighted in blue. We observe that the learnt policy is indeed the optimal one.

Figure 5.7 illustrates the estimated unbounded rewards assigned to each state, depicting both the deterministic (upper left) and stochastic (upper right) scenarios. Notably, a discernible pattern emerges with rewards smoothly increasing along the optimal trajectory. Moreover, outside this trajectory, the rewards exhibit a gradual decline, approximately resembling the reciprocal of the distance between out-of-distribution states and the nearest state along the optimal path. This characteristic plays a pivotal role in our model’s capacity to navigate back from out-of-distribution states. The bottom left panel of figure 5.7 showcases the expert’s occupancy measure. A noteworthy negative correlation becomes apparent between the expert’s occupancy measure and the learned rewards. This correlation is particularly striking in a highly stochastic environment, where even an expert can deviate from the optimal trajectory. This observation further highlights instances where the expert intermittently revisits certain states before ultimately reaching more stable states, reflecting the challenges posed by the environment’s stochastic nature. Consequently, the initial states tend to exhibit elevated occupancy measures due to these transitional struggles. If the occupancy measure were to be fitted, it would confer significant importance to these transitional states. This shows up in the bottom right panel showcasing the attributed reward attributed to each state by our baseline, namely the GCL algorithm. Conversely, our model attributes comparatively lower rewards to such states, while assigning greater rewards to states aligning with the optimal path and situated closer to the ultimate goal.

Finally, the results showcased in figure 5.7 also hint at our model’s greater ability to tackle outer-alignment issues. Indeed, while GCL ultimately finds the optimal policy, the reward structure (decreasing towards the goal) shows that the intrinsic human objective is not “understood” by the model. On the other hand, the fact that our model outputs an increasing reward function as we get closer to the goal shows that the model accurately learnt the human objective, which is

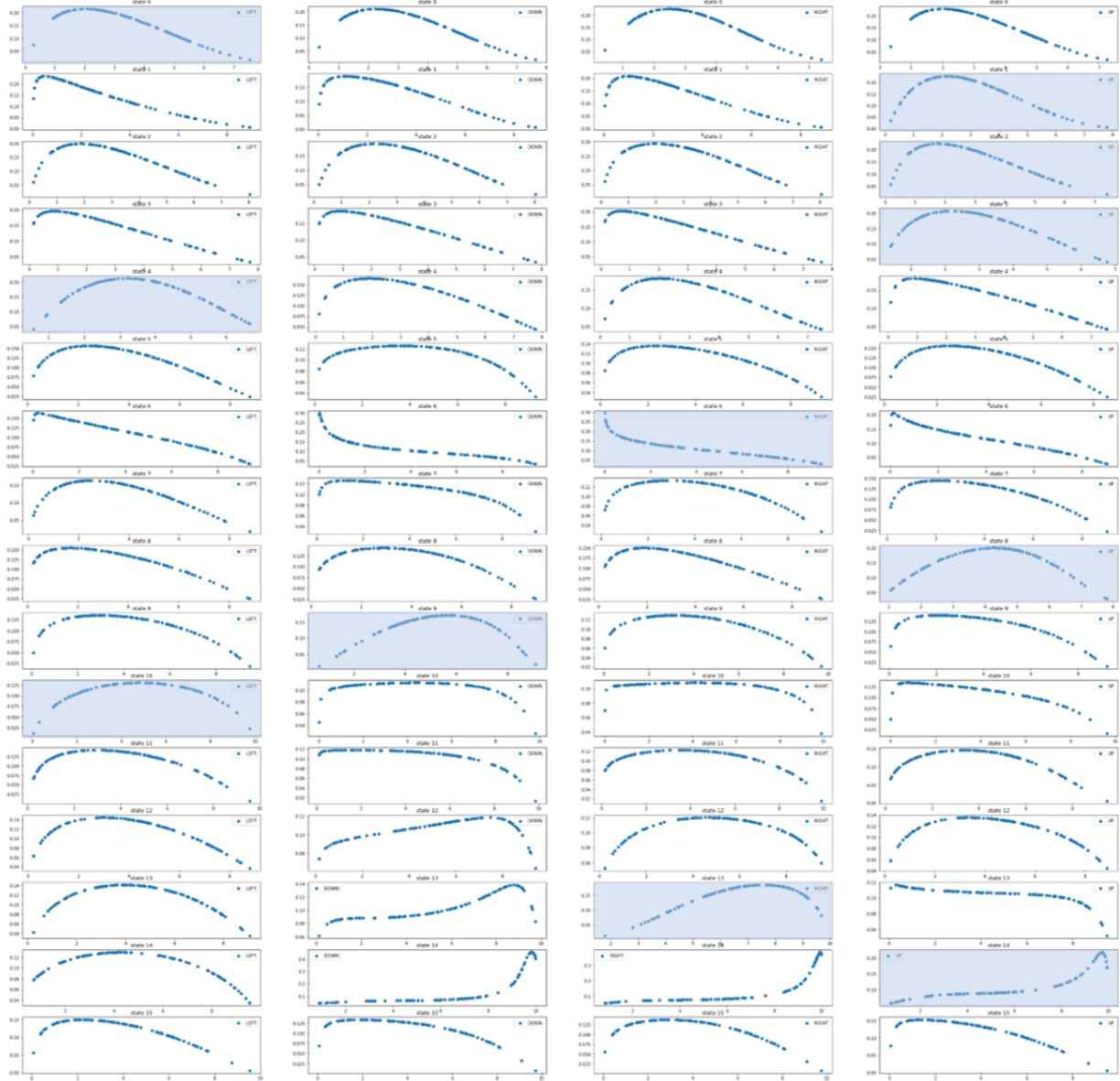


Figure 5.6: Our model predictions for all state-action pairs in the 4×4 Frozen Lake environment. Each line corresponds to a states and each column to an action. States are numbered from 0 to 8 starting from the upper left state and going to the right. The plots display the returns in the range $[0, 10]$ on the x axis and their corresponding predicted density. The optimal action given the learnt reward function for each state is highlighted in blue.

to get closer and eventually reach the goal state.

5.6 Conclusion

We addressed IRL using the Distributional RL framework. By harnessing the advantages offered by Normalizing flows and the Cramèr distance (as detailed in chapter 4), we have successfully

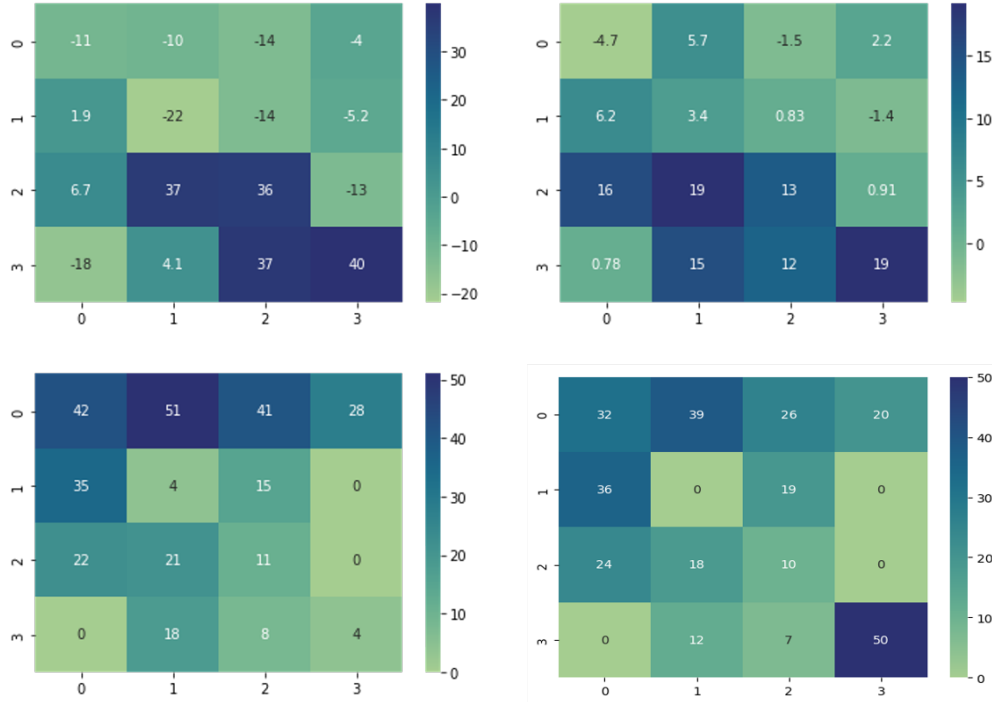


Figure 5.7: Predicted reward for each state in the deterministic setting (upper left) and stochastic setting (upper right). Bottom right: Expert’s occupancy measure. Bottom left: predicted reward for each state in the stochastic setting by the GCL algorithm.

devised a method for learning a reward function that accurately captures both the expert’s objectives and behaviours, while bypassing the necessity of matching occupancy measures as a primary objective. Building upon the insights presented in [PGP17] and the inherent connection between IL and IRL, we have demonstrated the feasibility of learning the expert’s set policy rather than relying on the conventional occupancy measure, thereby showcasing its efficacy, particularly in highly stochastic environments. In contrast to existing approaches centered around occupancy measures and adversarial techniques, our proposed framework adopts a collaborative model, wherein the shared objective between the agent and the critic promotes stability and robustness.

Our approach addresses the inherent challenges associated with the ill-posed nature of the IRL problem. Our model not only effectively incentivises the imitation of demonstrated actions within demonstrated states, but also facilitates the crucial ability to guide the agent towards returning to familiar states when navigating through novel, out-of-distribution scenarios.

Regarding the outer/inner alignment problems, our approach brings several benefits:

- While any IRL based method is deemed more relevant than a hard coded objective to ensure alignment, our approach captures the expert’s behaviour more accurately than existing approaches by learning its set-policy rather than matching occupancy measures. We believe this feature is a significant progress toward better outer-alignment.
- Using the link between IL and IRL, we devised a method that imitates the expert’s ac-

tions while ensuring that the model returns to demonstrated states if it encounters out-of-distribution situations. This property enables safer agents that can recover better in stochastic environments while staying close to the demonstrated path, hence limiting any drift risks. Safer agents are more robust to out-of-distribution scenarios and distributional shift, hence limiting the risk of inner-misalignment.

Our experimental results provide compelling evidence that our model successfully learns non-degenerate reward functions, effectively capturing the notion of distance between the current state and the nearest state within the demonstrated optimal trajectory. While our approach introduces several advantageous features compared to existing methods, there exists potential for further enhancement. To solidify the robustness and applicability of our approach, our next phase of experimentation will involve rigorous testing in more intricate and challenging environments. Specifically, we plan to evaluate our model’s performance on complex domains like the Atari 2600 games from the Arcade Learning Environment [Bel+13], as well as within the Mujoco robotic simulation framework [TET12]. This comprehensive evaluation aims to establish the effectiveness of our approach, showcasing its capabilities through comparisons with established techniques in the field.

In future work we will investigate how our model can be adapted to regularised MDPs as it has been proven that these do not contain degenerate solutions due to the uniqueness of the optimal policy for regularised MDPs [GSP19]. Moreover, [FLL18] introduced an adversarial framework to learn "disentangled rewards", which are reward functions that are decoupled from the environment dynamics and are specifically designed to be more robust when transferred to a different environment. Adapting this approach to our setup would be a nice addition to our model.

Chapter 6

Zero-Shot Clustering Through Metric Transfer Learning

Clustering in high dimension spaces is a difficult task; the usual distance metrics may no longer be appropriate under the curse of dimensionality. Indeed, the choice of the metric is crucial, and it is highly dependent on the dataset characteristics. However a single metric could be used to correctly perform clustering on multiple datasets of different domains. We propose to do so, providing a framework for learning a transferable metric. We show that we can learn a metric on a labelled dataset, then apply it to cluster a different dataset, using an embedding space that characterises a desired clustering in the generic sense. We learn and test such metrics on several datasets of variable complexity (synthetic, MNIST, SVHN, omniglot) and achieve results competitive with the state-of-the-art while using only a small number of labelled training datasets and shallow networks.

6.1 Introduction

In the previous chapter, we extended our Distributional RL framework from chapter 4 to the IRL context. We also referenced the work of [FLL18], which introduces an adversarial framework to obtain “disentangled rewards”. These rewards are intentionally designed to be independent of the complex environmental dynamics, enhancing their adaptability when applied to different environments or variations of the same environment with altered dynamics. For instance, this approach could enable training an IRL model to control a car based on expert demonstrations under sunny and dry conditions, while learning a reward function that remains applicable in rainy conditions with wet roads. This is a great way to handle distributional shift and inner alignment problem as defined in the introduction of this manuscript.

In a sense, the approach described in [FLL18] learns a global reward function that remains independent from the environment’s dynamics which can be described as a meta-reward function that is valid for a large range of dynamics variations for the same environment. One of the major stances of this thesis is that it should be possible to handle distributional shift by learning meta-objectives. In that, we agree with the authors of [FLL18]. However, the aim is to push the limit further by learning meta-rewards for different tasks (that remain similar) rather than keeping the

same task and only changing the dynamics.

In this section we will detail more exhaustively the aim of our work, i.e. zero-shot learning from an IRL perspective and why we restricted ourselves to the clustering problem. Finally we will present our view on metric learning for clustering.

6.1.1 From Meta-IRL to Zero-Shot Clustering

In the background section we presented Meta-learning for both Domain Adaptation and Multi-Task Learning. We highlighted the distinction between zero-shot and few-shot learning, with a specific focus on few-shot scenarios. Our exploration encompassed both metric-based and optimization-based approaches, with a comprehensive examination of Model Agnostic Meta Learning (MAML) [FAL17]. We also discussed the adaptation of MAML to RL [Rak+19] and Inverse IRL [Xu+19; Yu+19] contexts.

The underlying inspiration for these optimization-based methods stems from the concept of pre-training neural network weights for vision tasks. This established paradigm involves first pre-training a network on a large dataset such as ImageNet, then fine-tuning it on specific tasks using gradient descent. This pre-training enables networks to learn new tasks more effectively even with limited data. However, in few-shot scenarios, where data is extremely scarce, because the last layers of the network still need to be heavily adapted to the new task, the fine-tuning process can still lead to overfitting due to the need for substantial adaptation in the final layers of the network. MAML addresses this challenge by optimizing for an initial parameterization that can be rapidly adapted to new tasks with minimal examples and gradient steps as illustrated in figure 6.1. The meta-learner aims to discover an initialization that is not only versatile across a variety of tasks but also facilitates swift and efficient adaptation. In the context of MAML, this involves seeking parameters θ that, during the meta-learning process (indicated by the bold line), enable quick convergence to optimal parameters θ_i^* for specific tasks (illustrated by the grey lines) through a few gradient steps.

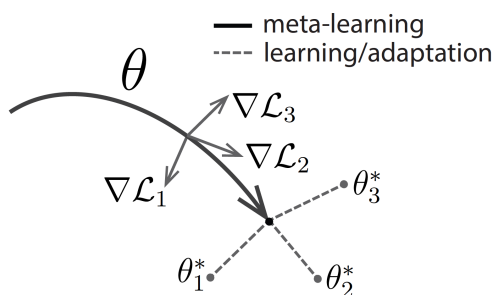


Figure 6.1: Optimisation based Meta-Learning. The model optimises for θ parameters that find a compromise between different tasks in order to optimise and adapt quickly to new tasks. Source: <https://bair.berkeley.edu/blog/2017/07/18/learning-to-learn/>

Despite their remarkable achievements, these approaches, characterized by their revolutionary simplicity and efficiency, encounter a significant limitation—they are unable to address the zero-shot learning scenario. In other words, they necessitate a collection of expert demonstrations

specific to the target task. Our overarching objective is to transcend this limitation and achieve success in the realm of zero-shot learning.

In zero-shot learning, the aspiration is to craft a reward function with the potential to guide an agent to success in a target domain or task, armed only with expert demonstrations from a different domain or task, and without any requirement for demonstrations within the target domain. This ambitious goal envisions a paradigm where a reward function derived from one context can be effectively transferred to another, enabling the agent to navigate uncharted territories and accomplish tasks beyond the confines of its training data.

Figure 6.2 provides a visual representation of this process. Consider the scenario of teaching a robot to ride a bike—an endeavor for which we lack a clear understanding of an appropriate reward function. One approach would involve employing IRL based on expert demonstrations. However, if we introduce the added constraint that no expert demonstrations are available for bike riding, the feasibility of IRL diminishes. A compelling alternative approach involves identifying an auxiliary task, such as car driving, which does possess available expert demonstrations. The proposition here is to formulate a reward function for the auxiliary task and subsequently endeavor to adapt it to the bicycle riding task. This knowledge transfer leverages the known attributes of one task to inform the learning process of another task, even if direct expert demonstrations for the latter task are unattainable.

This is clearly a hard problem for which a solution could be a revolution for all fields to which AI can be applied and even a gigantic step towards General AI. Therefore, while keeping this ultimate aim in mind, we humbly propose to simplify the problem at hand. For this matter, we draw the following constraints regarding the type of MDP considered:

- **One step problem:** Every state has to be reachable from any other state of the game
- **Unique optimum:** There should be only one optimal final state to find
- **Metrizability:** the reward function should behave like a metric. It should measure how close the current state is from the final desired state.

One can notice that the metrizability constraint illustrates the reward function learnt using our Distributional IRL approach, which can be interpreted as the inverse distance between the current state and the closest state located on the optimal path.

We also notice that clustering problems satisfy all three of these constraints:

- **One step problem:** A clustering can be seen as a one step partition problem
- **Unique optimum:** It has a unique perfect/desired partition (modulo the permutations between all partitions)
- **Metrizability:** A clustering depends on the metric used, it just finds the clustering that maximises/minimises the given metric. The metric can be interpreted as a distance between a proposed clustering and the perfect/desired solution

One direct consequence of metrizability is that optimisation based approaches like MAML are de facto ruled out. Only metric based approaches are possible in this setting.

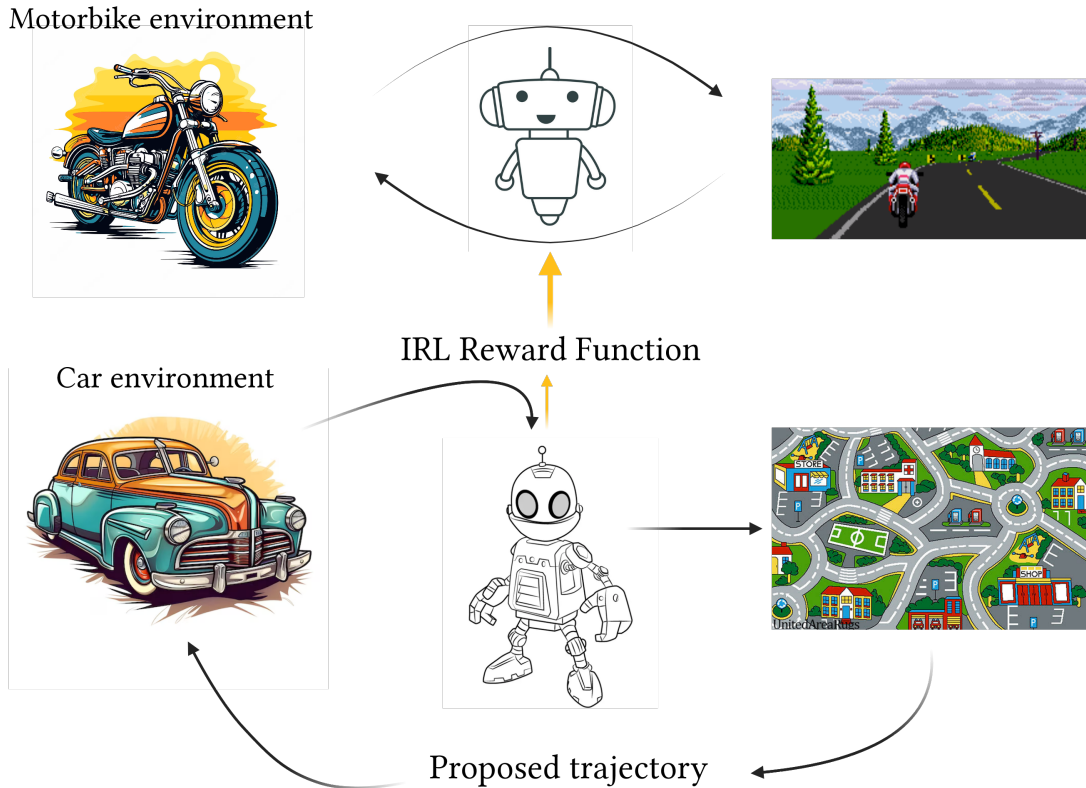


Figure 6.2: Given demonstrations of how to drive a car, a model can infer a reward function that can suit both the tasks of driving a car and riding a bike.

We also recall that there exist two types of transfers, namely domain and task transfer. Clustering can be adapted to handle those two types:

- **Domain transfer:** Given a dataset \mathcal{D} , it is possible to create instances \mathcal{D}_i of the dataset where the objects to cluster are not distributed the same way. The idea is then to learn a metric on a given instance \mathcal{D}_i and use the same metric to perform the clustering on \mathcal{D}_j .
- **Task transfer:** One can try to learn a clustering metric on a dataset \mathcal{D}_1 and then perform a clustering using the same metric on dataset \mathcal{D}_2 .

The clustering process is illustrated in figure 6.3

6.1.2 Transferable Metric Learning for Clustering

Given a dataset \mathbf{X} , clustering is the unsupervised task of assigning a categorical value $y_i \in \{1, \dots, k\}$ to each data point $x_i \in \mathbf{X}$, where no such example categories are given in the training data; i.e., we should map $\mathbf{X} = \{x_1, \dots, x_n\} \mapsto \mathbf{Y} = \{y_1, \dots, y_n\}$ with \mathbf{X} the input matrix of n data points, each of dimension d ; where $y_i = \kappa$ implies that data point x_i is assigned to the κ -th cluster.

Clustering methods complete this task by measuring similarity (the distance) between training

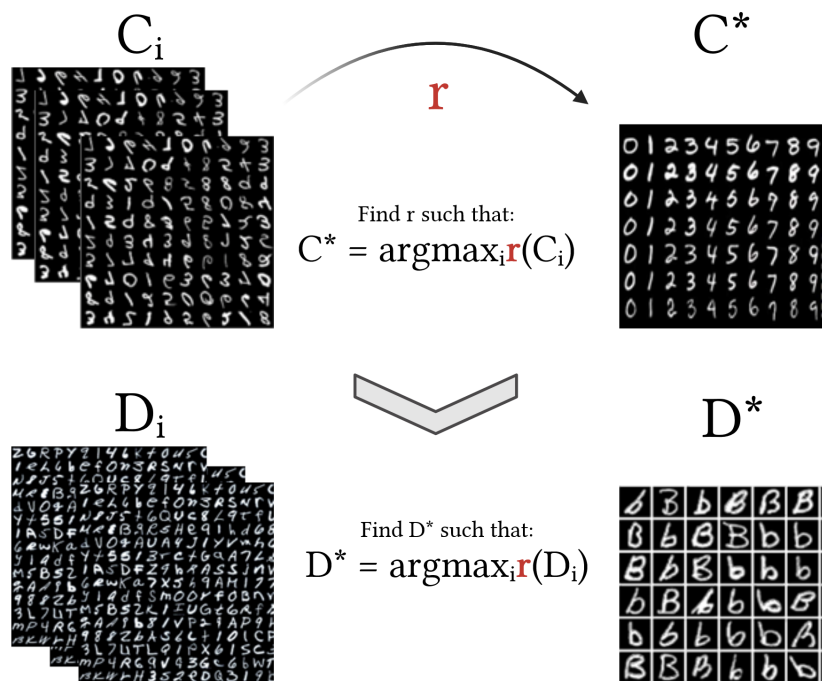


Figure 6.3: Given demonstrated clusterings C_i^* on instances of MNIST numbers dataset C_i , the aim is to find the metric r , such that $C_i^* = \operatorname{argmax}_r \hat{C}_i$. In a second step, we use r to find D_i^* such that $D_i^* = \operatorname{argmax}_r \hat{D}_i$. D_i are instances of the MNIST letter dataset.

pairs, using a similarity function

$$s(x_i, x_j) \in \mathbb{R}_+$$

Given a threshold $t \in \mathbb{R}_+$, a clustering algorithm typically allocates two points to the same cluster if $s(x_i, x_j) > t$. In order to form hard clusters, i.e. no point should belong to more than one unique cluster, it is necessary to find t given s to enforce that condition.

This similarity function should typically reflect subjective criteria fixed by the user. Basically, this means that the user decides what makes a good clustering. As mentioned in [HVZ19], “since classes are a high-level abstraction, discovering them automatically is challenging, and perhaps impossible since there are many criteria that could be used to cluster data (e.g., we may equally well cluster objects by colour, size, or shape). Knowledge about some classes is not only a realistic assumption, but also indispensable to narrow down the meaning of clustering”. Taking the example of MNIST [LCB10], one usually groups the same numbers together because these numbers share the highest amount of features (e.g., mutual information based models do that). However one may want to group numbers given their roundness. In this case, we may obtain two clusters, namely straight shaped numbers (i.e., 1, 4, 7) and round shaped numbers (i.e., all the others). Both clustering solutions are relevant, since each clustering addresses a different yet possible user subjective criteria (i.e., clustering semantics).

Table 6.1: Summary of notation.

\mathbf{X}	A dataset of n points, $x_i \in \mathbb{R}^d$
\mathbf{y}^*	True clustering (partition) of $\mathbf{X}, \mathbf{y}^* \in \mathbb{R}^n$, labels $y_i^* \in \{1, \dots, k\}$
\mathbf{y}	A possible clustering of $\mathbf{X}, \mathbf{y} \in \mathbb{R}^n$, labels $y_i \in \{1, \dots, k\}$
r	Metric $\mathbb{R}^{n \times d} \times \mathbb{N}^n \mapsto \mathbb{R}$, scoring of the clustering
$\hat{\mathbf{y}}$	Clustering retained after optimisation for a scoring function r , $\hat{\mathbf{y}} \in \mathbb{R}^n$, labels $\hat{y}_i \in \{1, \dots, k\}$
s	Similarity function $s : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}_+$
\mathbf{X}/\sim	\mathbf{X}/\sim the quotient space of \mathbf{X} generated by the equivalence relation \sim
\mathcal{D}	$\{\mathbf{X}_l, \mathbf{y}_l^*\}_{l=1}^\ell$ collection of labelled datasets
g	$\mathbb{R}^{n \times d} \times \mathbb{R}^n \rightarrow \mathbb{R}^e$ embedding function
$\mathbf{z}^*, \hat{\mathbf{z}}$	Embeddings, $\in \mathbb{R}^e$, of $(\mathbf{X}, \mathbf{y}^*)$, and (\mathbf{X}, \mathbf{y}) , respectively
$c_\theta(\mathbf{z})$	Output $\in \mathbb{R}^+$ from the WGAN critic taking the embedding \mathbf{z} as input
$\mathcal{G}(\mathbf{X}, \mathbf{y})$	Graph representing a clustered version of \mathbf{X}
\mathbf{A}	An adjacency matrix
CEM	Cross-entropy method
\mathcal{S}	Set of all intermediate clustering solutions found through CEM

Finding an automated way to derive and incorporate user criteria in a clustering task based on intended semantics can be very hard. Nowadays, the wide availability of shared annotated datasets is a valuable asset and provides examples of possible user criteria. Hence, we argue that, given “similar” annotated data, classification logic can be used to derive a user criteria that one can apply to clustering similar non-annotated data. For example, we consider the situation where a human is placed in front of two datasets, each one consisting of letters of a certain alphabet she does not understand. The first dataset is annotated, grouping the same letters together. Only by seeing the first dataset, the person can understand the grouping logic used (grouping same geometrical shapes together) and replicate that logic to the second non annotated dataset and cluster correctly its letters.

We are interested in tackling the problem of clustering data when the logic (i.e., user clustering criteria) is encoded into some available labelled datasets. This raises two main challenges, namely (1) find a solution that works well on the classification task but (2) ensure transferability in its decision mechanism so it is applicable to clustering data from a different domain.

We believe that addressing these challenges calls for the design of a scoring function that should be as general as possible to ensure transferability but is specific enough not to miss the user criteria. More specifically, the scoring function should be comparing the logic used to produce a certain clustering to the one used to produce clusterings of the already seen training datasets. Using the concept of logic is useful as a logic is general enough to be used on any dataset and specific enough as it is the main common property shared by all training dataset. Our goal is then

to find a suitable metric that retrieves and encapsulate the seen concept for scoring a clustering outcome.

Moreover, modern applications require solutions that are effective when data is of high dimension (i.e., large d). While distance-based approaches are broadly used for clustering (e.g., Euclidean distance), we argue that they are not suitable for our problem since they would yield in data specific models in addition to their poor performance in high dimensional spaces due to the curse of dimensionality. To lower dimensionality, a solution is to perform instance-wise embeddings $x_i \mapsto z_i \in \mathcal{Z}$, with $\dim(\mathcal{Z}) < \dim(\mathbf{X})$ e.g., with an autoencoder. However this mechanism is still domain specific.

To achieve training on more general patterns, we think it is necessary to take the dataset in its entirety. Therefore, instead of learning a metric that compares pairs of data points in a dataset instance (like a similarity measure), a learned metric is applied to sets of data points so comparison is done between sets. The metric can be intuitively understood as a distance between the logic underlying a given clustering and the general logic that was used to produce clusterings in training datasets.

For this, we propose a solution where we use a graph autoencoder [KW16] to embed a set of data points into a vector of chosen dimension. Then, we use the critic part of a Wasserstein GAN (WGAN) [ACB17] to produce a continuous score of the embedded clustering outcome. This critic represents the metric we seek. Thus, our main contributions are:

- We provide a framework for joint metric learning and clustering tasks.
- We show that our proposed solution yields a learned metric that is transferable to datasets of different sizes and dimensions, and across different domains (either vision or tabular) and tasks.
- We obtain results competitive to the state-of-the-art with only a small number of training datasets, relatively simple networks, and no prior knowledge (only an upper bound of the cluster number that can be set to a high value).
- Our method is scalable to large datasets both in terms of number of points or dimensions (e.g the SVHN dataset used in section 6.4) as it does not have to compute pairwise distances and therefore does not heavily suffer when the number of points or dimensions increase.
- We test the metric on datasets of varying complexity (synthetic [Ped+11], MNIST [LCB10; Coh+17; XRV17], SVHN [Net+11], omniglot [LST15]) and perform on par with the state-of-the-art while maintaining all the advantages cited above.

6.2 Related Work

Using auto-encoders before applying classic clustering algorithms resulted in a significant increase of clustering performance, while still being limited by these algorithms capacity. Deep Embedding Clustering (DEC) [XGF16] gets rid of this limitation at the cost of more complex objective functions. It uses an auto-encoder along with a cluster assignment loss as a regularisation. The obtained clusters are refined by minimising the KL-divergence between the distribution of

soft labels and an auxiliary target distribution. DEC became a baseline for deep clustering algorithms. Most deep clustering algorithms are based on classical center-based, divergence-based or hierarchical clustering formulations and hence bear limitations like the need for an *a priori* number of clusters.

MPCCKMeans [BBM04] is more related to metric learning as they use constraints for both metric learning and the clustering objective. However, their learned metrics remain dataset specific and are not transferable.

Constrained Clustering Network (CCN) [Ha18], learns a metric that is transferable across domains and tasks. Categorical information is reduced to pairwise constraints using a similarity network. Along with the learned similarity function, the authors designed a loss function to regularise the clustering classification. But, using similarity networks only captures local properties instance-wise rather than global geometric properties of dataset clustering. Hence, the learned metric remains non fully transferable, and requires to adapt the loss to the domain to which the metric is transferred to.

In Deep Transfer Clustering (DTC) [HVZ19] and Autonovel [HRa21], the authors tackle the problem of discovering novel classes in an image collection given labelled examples of other classes. They extended DEC to a transfer learning setting while estimating the number of classes in the unlabelled data. Autonovel uses self-supervised learning to train the representation from scratch on the union of labelled and unlabelled datasets then trains the data representation by optimising a joint objective function on the labelled and unlabelled subsets of data. We consider these two approaches as our state of the art baselines.

6.3 Our Framework

To restate our objective, we seek an evaluation metric

$$\begin{aligned} r : \mathbb{R}^{n \times d} \times \mathbb{N}^n &\rightarrow \mathbb{R} \\ (\mathbf{X}, \mathbf{y}) &\mapsto \mathbf{r}(\mathbf{X}, \mathbf{y}) \end{aligned} \tag{6.1}$$

where $\mathbf{X} \in \mathbb{R}^{n \times d}$ is a dataset of n points in d dimensions and $\mathbf{y} \in \mathbb{N}^n$ a partition of \mathbf{X} (i.e. a clustering of \mathbf{X}).

6.3.1 Formulating the Problem with Quotient Spaces

As we are looking for a metric comparing between different sets of points. The most natural formulation in this context is to use the concept of quotient spaces. Indeed, this formulation will allow us to formulate the problem as an optimisation one and to define the sought metric as a similarity measure between quotient spaces.

Quotient Maps Transform Similarity Between Clusters Into Distance Between Points.

Let \sim denote an equivalence relation where $x_i \sim x_j$ if and only if $s(x_i, x_j) > t \in \mathbb{R}_+$. When $x_i \sim x_j$ then x_i and x_j belong to the same cluster. We are interested in forming hard clusters, i.e. no point

should belong to more than one unique cluster, therefore it is necessary to find t given s such that $C_{\sim} = X/\sim$ is the quotient space of X generated from the considered equivalence relation.

As illustrated in figure 6.4, Given two sets of points D and A , the equivalence relation defines an application between D and A such that all points considered equivalent in D are mapped towards a single point in A . This mapping is called quotient map.

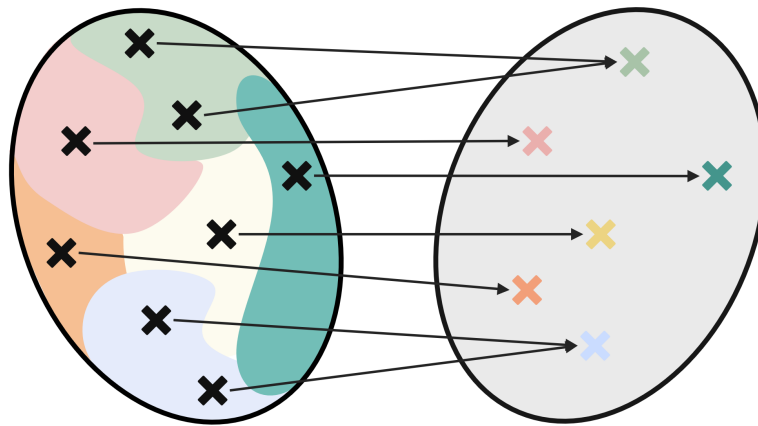


Figure 6.4: Illustration of quotient map. Given two sets of points D and A , the equivalence relation defines an application between D and A such that all points considered equivalent in D are mapped towards a single point in A .

Definition: quotient map: Let (X, τ) and (Y, τ') be topological spaces. Then a continuous surjective map $q : X \mapsto Y$ is a quotient map if $V \in \tau' \iff q^{-1}(V) \in \tau$. This means that for any open in Y its preimage for q is also open in X .

From Abstract Similarity to Optimising a Metric

An essential property of quotient maps is continuity. This means that neighbourhoods are preserved in both sets X and Y . Consequently, close clusters in the set X relative to some distance d are mapped to close points in Y . Therefore quotient maps not only describe an equivalence relation but they make it easier to define similar clusters. Indeed similar clusters are mapped to close points in Y .

The function r can then be designed to behave as a metric such that given two quotient spaces C_{\sim} and $C_{\sim'}$ respectively generated from equivalence relations \sim and \sim' , and C_{\sim^*} being the perfect clustering, if C_{\sim} is closer to be the correct clustering, then $r(C_{\sim}; C_{\sim^*}) < r(C_{\sim'}, C_{\sim^*})$. Therefore quotient maps are essential as they allow to reduce a similarity between clusters to a metric in a quotient space. For lighter notations, C_{\sim^*} will be considered implicit and we will denote $r(C_{\sim}) > r(C_{\sim'})$.

Obviously, such a function r implicitly encodes the choice of similarity measure s and its threshold t . Let $\{X_1, \dots, X_n\} \in \mathcal{D}$ be the collection of datasets used for training, and $\{X_1^*, \dots, X_n^*\}$ their

known clusterings. Such clustering problem could be approached as an optimisation problem:

$$\begin{aligned}
C_{\sim} &= \arg \max_{\mathbf{X}/\sim} r(\mathbf{X}/\sim) \\
\text{s.t } r(\mathbf{X}/\sim^*) &= 0 \\
\text{s.t } r(\mathbf{X}_i^*) &= 0 \quad \forall \mathbf{X}_i \in \mathcal{D} \\
\text{s.t } r(\mathbf{X}/\sim) &< r(\mathbf{X}/\sim^*) \quad \text{if } \sim \neq \sim^*
\end{aligned} \tag{6.2}$$

What we seek is a general function r that scores the general quality of a clustering. We seek a function that is effective for many (possibly high-dimensional) clustering problems i.e., a *transferable function*.

Metric r should provide a score for *any* labelled dataset of any dimensionality; and in particular this score should be such that $r(\mathbf{X}, \mathbf{y})$ is high when the hamming distance between the ground truth labels \mathbf{y}^* and \mathbf{y} is small (taking cluster label permutations into account). This would mean that we could perform clustering on any given dataset, simply by solving the optimisation problem 6.2 even if such a dataset had not been seen before. In section 6.3.5, this problem will be reformulated in terms of metric instead of equivalence relation.

6.3.2 Overview of our Model

Formally stated, our goal is:

1. To produce a metric r that grades the quality of a clustering such that $\mathbf{y}^* = \arg \max_{\mathbf{y}} r(\mathbf{X}, \mathbf{y})$;
2. Implement an optimisation algorithm that finds \mathbf{y}^* ;
3. Use (1) and (2) to perform a clustering on a new unrelated and unlabelled dataset.

We use a collection $\mathcal{D} = \{\mathbf{X}_l, \mathbf{y}_l^*\}_{l=1}^{\ell}$ of labelled datasets as examples of correctly ‘clustered’ datasets, and learn r such that $\mathbb{E}[r(\mathbf{X}, \mathbf{y})]$ is high. Indeed, as r can be considered either as a distance or a score, we will consider from now that r is a score to be maximised rather than a distance to minimise. In order to make r transferable between datasets, we embed each dataset with its corresponding clustering $(\mathbf{X}_l, \mathbf{y}_l)$ into a vector $\mathbf{z}_l \in \mathbb{R}^e$ using the embedding function g :

$$\begin{aligned}
g : \mathbb{R}^{n \times d} \times \mathbb{R}^n &\rightarrow \mathbb{R}^e \\
(\mathbf{X}, \mathbf{y}) &\mapsto \mathbf{z}
\end{aligned} \tag{6.3}$$

Therefore, the metric r is actually the composition of two functions: g and a subsequent scoring function from \mathbb{R}^e to \mathbb{R} . Our training procedure is structured around 3 blocs A, B and C detailed in next sections and depicted in figure 6.5 and is summarised in the following main steps:

Bloc A. step 1 Select a labelled dataset $(\mathbf{X}, \mathbf{y}^*) \sim \mathcal{D}$

Bloc A. step 2 Given a metric function r (output from bloc B step 2, or initialised randomly), we perform a clustering of dataset \mathbf{X} : $\hat{\mathbf{y}} = \arg \max_{\mathbf{y}} r(\mathbf{X}, \mathbf{y})$

Bloc B. step 1 \mathbf{y}^* and $\hat{\mathbf{y}}$ are represented as graphs where each clique represents a cluster.

- Bloc B. step 2 Graph convolutional autoencoders perform feature extraction from \hat{y} and y^* and output embeddings \hat{z} and z^*
- Bloc C. step 1 The metric r is modelled by a WGAN critic denoted c_θ that outputs evaluations of the clusterings: $r(\mathbf{X}, y^*) = c_\theta(z^*)$ and $r(\mathbf{X}, \hat{y}) = c_\theta(\hat{z})$
- Bloc C. step 2 Train the model using the error between $r(\mathbf{X}, y^*)$ and $r(\mathbf{X}, \hat{y})$.

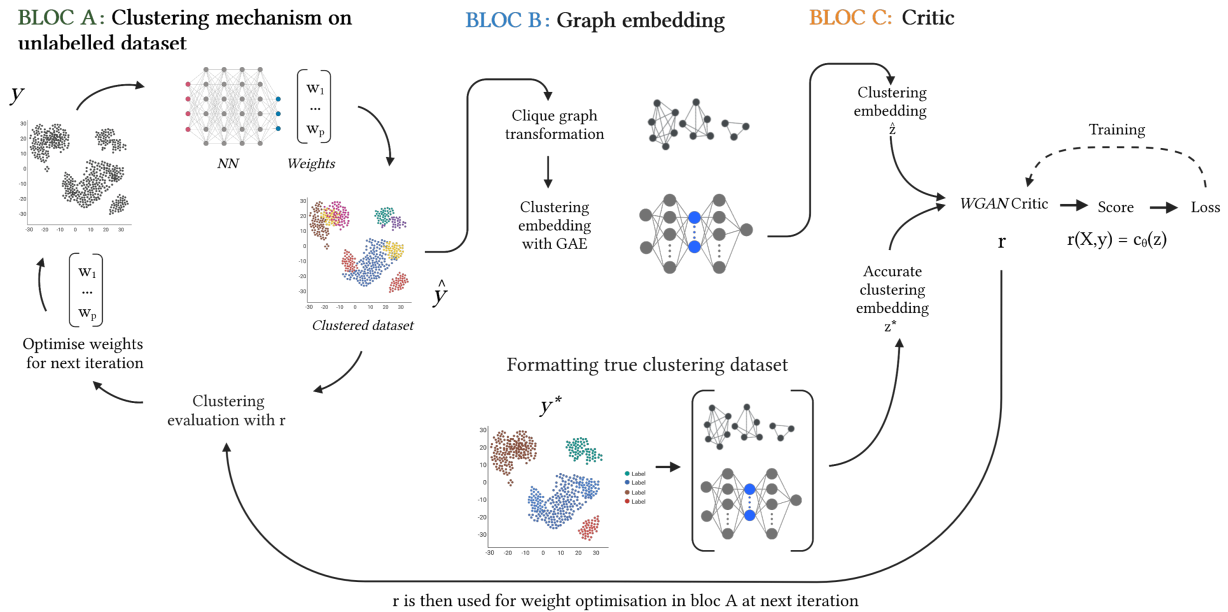


Figure 6.5: Our framework’s 3 components: the clustering mechanism (A), the GAE (B) and the WGAN (C). (A) takes an unlabelled dataset \mathbf{X} as input and outputs a clustering \hat{y} that maximises a metric r . \hat{y} is then turned into a graph $\mathcal{G}(\mathbf{X}, \hat{y})$ then into an embedding vector \hat{z} using (B). Same goes for the correctly labelled dataset, which is embedded as z^* . Then, (C), which is the metric itself, evaluates \hat{z} and z^* using c_θ and is trained to produce a new metric r which is then used for (A) in the next iteration.

6.3.3 Clustering Mechanism

We seek the most suitable optimisation algorithm for clustering given r . Considering a neural network parametrised by w that performs the clustering, we need to find its optimal weights w^* such that the metric r is maximised (see equation (6.4)). The type of algorithm to use depends on the nature of the metric r to optimise on.

$$\Phi_r(\mathbf{X}) \xrightarrow{\text{finds}} w^* = \arg \max_w r(\mathbf{X}, y^w) \quad (6.4)$$

Where Φ is an optimiser to determine and y^w is a clustering obtained with the weights w . The metric is assumed to hold certain properties, discussed in section 6.3.5:

- **Unique Maximum:** A unique optimal clustering. r has a unique maximum.

Algorithm 6 CEM Algorithm

Require: Dataset $\mathbf{X} \in \mathbb{R}^{n \times d}$; score function r ; $\mu \in \mathbb{R}^d$ and $\sigma \in \mathbb{R}^d$; elite percentage to retain p ; n number of samples w_i to produce; T number of iterations

for iteration = 1 to T **do**

 Produce n samples of neural network weights $w_i \sim \mathcal{N}(\mu, \text{diag}(\sigma))$

 Produce clusterings y_i of \mathbf{X} using each w_i

 Evaluate $r_i = r(\mathbf{X}, y_i)$

 Constitute the elite set of p best w_i

 Fit a Gaussian distribution with diagonal covariance to the elite set and get a new μ_t and σ_t

end for

return: μ, w^*

- **Continuity**¹: Any two clusterings \mathbf{y} and \mathbf{y}' should be similar if $r(\mathbf{y})$ and $r(\mathbf{y}')$ are close in \mathbb{R} space. Hence, r has to satisfy a continuity constraint.

There is no guarantee that the best metric for the clustering task is differentiable. Given the above assumptions, conditions are favourable for evolutionary strategies (ES) to iteratively converge towards the optimal solution. Indeed, if r is continuous and the series $\left((\mathbf{X}, \hat{y}_1), \dots, (\mathbf{X}, \hat{y}_p) \right)$ converges towards (\mathbf{X}, y^*) then $\left(r(\mathbf{X}, \hat{y}_1), \dots, r(\mathbf{X}, \hat{y}_p) \right)$ converges towards $r(\mathbf{X}, y^*)$.

We choose the Cross-Entropy Method (CEM) [BKa05], a popular ES algorithm for its simplicity, to optimise the clustering neural network weights by solving Eq.(6.4) (algorithm 6). Indeed, any other gradient free algorithm can be used, depending on the user preference.

6.3.4 Graph Based Dataset Embedding

To capture global properties and be transferable across different datasets, we argue that it is necessary to input all the points of a dataset at once. Hence, instead of pairwise similarities between random pairs of points, we propose to get a representation of the relation between a bunch of neighbouring points. Thus, we represent each dataset by a graph structure $\mathcal{G}(\mathbf{X}, \mathbf{y})$ where each node corresponds to a point in $x_i \in \mathbf{X}$ and where cliques represent clusters as shown in figure 6.5. This representation takes the form of a feature matrix X and an adjacency matrix A . Using X , and A , we embed the whole dataset into a vector $\mathbf{z} \in \mathbb{R}^e$. To do so, we use graph autoencoders (GAE). Our implementation is based on [KW16].

Specifically, we have $\{X, A\} \mapsto \mathbf{z}$, under the following mechanism:

$$GCN(X, A) = \text{ReLU}(\tilde{A}XW_0) = \bar{X} \tag{6.5}$$

With \tilde{A} the symmetrically normalized adjacency matrix and (W_0, W_1) the GCN weight matrices.

¹As a reminder, Let T and U be two topological spaces. A function $f : T \mapsto U$ is continuous in the open set definition if for every $t \in T$ and every open set u containing $f(t)$, there exists a neighbourhood v of t such that $f(v) \subset u$.

$$z = \tilde{A}\bar{X}W_1 \quad (6.6)$$

Finally, the decoder outputs a new adjacency matrix using the sigmoid function σ :

$$\hat{A} = \sigma(zz^T) \quad (6.7)$$

We obtain $z \in \mathcal{M}_{n,m}$ which is dependent of the shape of the dataset (where m is a user specified hyper-parameter). In order to make it independent from the number of points in \mathcal{X} , we turn the matrix z into a square symmetrical one $z \leftarrow z^T z \in \mathcal{M}_{m,m}$. The final embedding z corresponds to a flattened version of the principal triangular bloc of $z^T z$, which shape is $e = (\frac{m+1}{2}, 1)$. However, the scale of the output still depends on the number of points in the dataset. This could cause an issue when transferring to datasets with a vastly different number of data points. It should therefore require some regularisation; in order to simplify, we decided to use datasets with approximately the same number of points.

6.3.5 A Critic as a Metric

With embedded vectors of the same shape, we compare the clusterings proposed \hat{z} and the ground truth ones z using the metric r . Given optimisation problem stated in equation 6.2, we can enforce the constraints under the process highlighted in [NR00]. Learning viable metric is possible provided both the following constraints: (1) maximising the difference between the quality of the optimal decision and the quality of the second best; (2) minimising the amplitude of the metric function as using small values encourages the metric function to be simpler, similar to regularisation in supervised learning.

These two constraints enforce the exact same behaviour as for our sought reward function in our chapter 5 Distributional IRL model. Indeed, considering the perfect clustering as our expert dataset, denoted D_E in chapter 5 and D_π the proposals, (1) enforces the maximum margin analogy maximising the distance between D_E and D_π . On the other hand, (2) has the same effect as using tanh activation function for bounding the reward.

When maximising the metric difference between the two clusterings that have the highest scores, we get a similarity score as in traditional metric learning problems. Formalising the problem 6.2 in these terms, we obtain the equivalent problem (6.8) where \mathcal{S} is a set of solutions (i.e., clustering proposals) found using r_θ (the metric parameterised by the critic's weights θ) and \mathbf{y}^* is the true clustering, \mathbf{y}^{\max} is the best solution found in \mathcal{S} : $\mathbf{y}^{\max} = \arg \max_{\mathbf{y} \in \mathcal{S}} r_\theta(\mathbf{X}, \mathbf{y})$.

$$\begin{aligned} \min_{\theta} r_\theta(\mathbf{X}, \mathbf{y}^*) - \left[\max_{\theta} \min_{\mathbf{y}' \in \mathcal{S} \setminus \mathbf{y}^{\max}} r_\theta(\mathbf{X}, \mathbf{y}^{\max}) - r_\theta(\mathbf{X}, \mathbf{y}') \right] \\ \text{s.t. } \mathbf{y}^* = \arg \max_{\mathbf{y} \in \mathcal{Y}} r_\theta(\mathbf{y}) \end{aligned} \quad (6.8)$$

For more clarity, we will now break down this expression.

1. $\min_{y' \in \mathcal{S} \setminus y^{\max}} r_\theta(\mathbf{X}, y^{\max}) - r_\theta(\mathbf{X}, y')$ finds the second best clustering for r_θ . Indeed the aim is to find the closest clustering to y^{\max} under r_θ .
2. $\max_\theta \min_{y' \in \mathcal{S} \setminus y^{\max}} r_\theta(\mathbf{X}, y^{\max}) - r_\theta(\mathbf{X}, y')$: We find θ such that r_θ gives the highest possible value for the difference between the best and second best clusterings found under r_θ . This expression enforces the constraint (1).
3. Finally the whole expression finds θ such that the difference between the score obtained by the best found clustering is close (ideally equal) to the score obtained by the sought clustering. This has to be true under the constraint that the sought clustering obtains the maximum achievable score under r_θ .

The condition (2) is enforced using a tanh activation function as described in chapter 5.

Algorithm 7 Critic2Metric (C2M)

Require: b : batch size, n_epochs : number of epochs; p : percentage of elite weights to keep; $n_iterations$: number of CEM iterations; n_w : number of weights to generate; $\mu \in \mathbb{R}^d$: CEM mean; $\sigma \in \mathbb{R}^d$: CEM standard deviation, θ : critic's weights

for $n = 1$ to n_epochs **do**

for $k = 1$ to b **do**

Sample $(\mathbf{X}_k, \mathbf{y}_k^*) \sim \mathcal{D}$ a ground truth labelled dataset

Generate ground truth embeddings $\mathbf{z}_{(\mathbf{X}_k, \mathbf{y}_k^*)} = GAE(\mathcal{G}(\mathbf{X}_k, \mathbf{y}_k^*))$

Initialise clustering neural network weights $\{w_j\}_{j=1}^{n_w}$

for $i = 1$ to $n_iterations$ **do**

for $j = 1$ to n_w **do**

Generate clusterings $\hat{\mathbf{y}}_k^{w_j}$

Convert $\hat{\mathbf{y}}_k^{w_j}$ into a graph

$\mathbf{z}_{(\mathbf{X}_k, \hat{\mathbf{y}}_k^{w_j})} = GAE(\mathcal{G}(\mathbf{X}_k, \hat{\mathbf{y}}_k^{w_j}))$

Evaluate: $r(\mathbf{X}_k, \hat{\mathbf{y}}_k^{w_j}) = c_\theta(\mathbf{z}_{(\mathbf{X}_k, \hat{\mathbf{y}}_k^{w_j})})$

end for

Keep proportion p of best weights w_p

$w^* \leftarrow \text{CEM}(w_p, \mu, \sigma)$

end for

Generate clustering $\mathbf{y}_k^{w^*}$

$\mathbf{z}_{(\mathbf{X}_k, \hat{\mathbf{y}}_k^{w^*})} = GAE(\mathcal{G}(\mathbf{X}_k, \hat{\mathbf{y}}_k^{w^*}))$

Train critic as in [ACB17] using $\mathbf{z}_{(\mathbf{X}_k, \hat{\mathbf{y}}_k^{w^*})}$ and $\mathbf{z}_{(\mathbf{X}_k, \mathbf{y}_k^*)}$

end for

end for

To solve equation (6.8), we use a GAN approach where the clustering mechanism (i.e., CEM) plays the role of the generator while a critic (i.e., metric learning model) plays the role of the discriminator. In a classic GAN, the discriminator only has to discriminate between real and false samples, making it use a cross entropy loss. With this kind of loss, and in our case, the discriminator quickly becomes too strong. Indeed, the score output by the discriminator becomes quickly polarised around 0 and 1.

For this reason, we represent r as the critic of a WGAN [ACB17]. This critic scores the realness or fakeness of a given sample while respecting a smoothing constraint. The critic measures the Wasserstein distance between data distribution of the training dataset and the distribution observed in the generated samples. Since WGAN assumes that the optimal clustering provided is unique, the metric solution found by the critic satisfies equation (6.8) constraints. r reaching a unique maximum while being continuous, the assumptions made in section 6.3.3 are correctly addressed. To train the WGAN, we use the loss \mathcal{L} in equation (6.9) where $\hat{\mathbf{z}}$ is the embedding vector of a proposed clustering and \mathbf{z} is the embedding vector of the desired clustering. Our framework is detailed in algorithm 7.

$$\mathcal{L}(\mathbf{z}^*, \hat{\mathbf{z}}) = \max_{\theta} \mathbb{E}_{\mathbf{z}^* \sim p} [f_{\theta}(\mathbf{z}^*)] - \mathbb{E}_{\hat{\mathbf{z}} \sim p(\hat{\mathbf{z}})} [f_{\theta}(\hat{\mathbf{z}})] \quad (6.9)$$

6.4 Experiments

In this section, we carry out empirical evaluation of the proposed model using synthetic and real datasets. The purpose of this analysis is (1) to study the capacity of the model to provide a close to expected clustering under a learned metric, (2) to test its unsupervised clustering performance when transferring the metric to new different target datasets both in terms of domain and task.


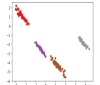

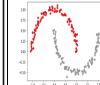





Dataset family	Synthetic data				MNIST			Street view house numbers	Omniglot
Dataset	Blob	Moon	Circles	Anisotropic	MNIST-digits [LCB10]	letters MNIST [Coh+17]	fashion MNIST [XRV17]	SVHN [Net+11]	Omniglot [LST15]
Snapshot									
Feature dimension	2	2	2	2	28 × 28	28 × 28	28 × 28	32 × 32	105 × 105
Maximum number of clusters	9 (custom)	9 (custom)	9 (custom)	9 (custom)	10	26	10	10	47
Size	200 (custom)	200 (custom)	200 (custom)	200 (custom)	60000	145600	60000	73257	32460

Table 6.2: Datasets description

For empirical evaluation, we parameterise our framework as follows: The critic (block C in Fig 6.5) is a 5 layer network of sizes 256, 256, 512, 512, and 1 (output) neurons. All activation functions are LeakyRelu ($\alpha = 0.2$) except last layer (no activation). RMSprop optimizer with 0.01 initial learning rate and a decay rate of 0.95. The CEM-trained neural network (bloc A in Fig 6.5) has 1 hidden layer of size 16 with Relu activation, and a final layer of size $k = 50$ (the maximum number of clusters). The GAE (bloc B in Fig 6.5) has 2 hidden layers; sized 32 and 16 for synthetic datasets, and 100 and 50 for real datasets.

We choose datasets based on 3 main criteria: having a similar compatible format; datasets should be large enough to allow diversity in subsampling configurations to guarantee against overfitting; datasets should be similar to the ones used in our identified baseline literature. All used datasets are found in table 6.2.

For training, we construct n sample datasets and their ground truth clustering, each containing 200 points drawn randomly from a set of 1500 points belonging to the training dataset. Each one

of these datasets, along with their clustering is an input to our model. To test the learned metric, we construct 50 new sample datasets from datasets that are different from the training one (e.g., if we train the model on MNIST numbers, we will use datasets from MNIST letters or fashion to test the metric). The test sample datasets contain 200 points each for synthetic datasets and 1000 points each otherwise. The accuracies are then averaged across the 50 test sample datasets. To test the ability of the model to learn using only a few samples, we train it using 5 (few shots) and 20 datasets (standard), each containing a random number of clusters. For few shots trainings, we train the critic for 1 epoch and 10 epochs for standard trainings.

To evaluate the clustering, we use Normalised-Mutual Information (NMI) [SG02] and clustering accuracy (ACC) [YXa10]. NMI provides a normalised measure that is invariant to label permutations while ACC measures the one-to-one matching of labels. For clustering, we only need that the samples belonging to the same cluster are attributed the same label, independently from the label itself. However, since we want to analyse the behaviour of the metric learned through our framework, we are interested in seeing whether it is permutation invariant or not. Hence, we need the two measures.

6.4.1 Results on 2D Synthetic Datasets

Analysis on synthetic datasets (see table 6.2) proves that our model behaves as expected. We do not compare our results to any baseline since existing unsupervised methods are well studied on them. We train our model using exclusively samples from blobs datasets. We then test the learned metric on the 4 different types of synthetic datasets (blobs, anisotropic, moons and circles). Results are displayed in table 6.3. We observe that the model obtains the best score on blobs since it is trained using this dataset. We can also notice that our model achieves high scores for the other types of datasets not included in training.

Types of datasets	Standard training		Few shots training	
	ACC	NMI	ACC	NMI
Blobs	98.4%	0.980	97.3%	0.965
Anisotropic	97.9%	0.967	97.2%	0.945
Circles	91.7%	0.902	92.7%	0.900
Moons	92.1%	0.929	92.8%	0.938

Table 6.3: Average ACC and NMI on synthetic test datasets.

Our model succeeds in clustering datasets presenting non linear boundaries like circles while blobs datasets used in training are all linearly separable. Hence, the model learns intrinsic properties of training dataset that are not portrayed in the initial dataset structure, and thus that the metric appears to be transferable.

Critic’s ablation study. To test if the critic behaves as expected, i.e., grades the clustering proposals proportionally to their quality, we test it on wrongly labelled datasets to see if the score decreases with the number of mislabelled points. We consider 50 datasets from each type of synthetic datasets, create 50 different copies and mislabel a random number of points in each copy.

A typical result is displayed in figure 6.6 and shows that the critic effectively outputs an ordering metric as the score increases when the number of mislabelled points decreases, reaching its maximum when there is no mislabelled point. This shows that the metric satisfies the constraints stated in equation 6.8.

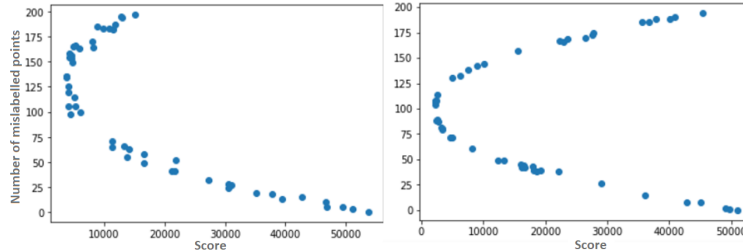


Figure 6.6: Metric values (i.e., scores given by the critic) for several clusterings of a dataset. Plots are from an anisotropic dataset (left) and a moons dataset (right). In a 2 cluster case (right), the formula used to compute mislabelled points has been made sensitive to label permutation to verify if permuted labels can fool the critic. The critic assigns a high score either when all the labels match the given ground truth or when all the labels are permuted (which again does not affect the correctness of the clustering)

An interesting behaviour is shown in figure 6.6. Recall that since we are in the context of a clustering problem, we only need for the samples belonging to the same cluster to get the same label, independently from the cluster label itself. Thus, the formula used to compute mislabelled points has been made sensitive to label permutation to verify if permuted labels can fool the critic. For instance, in a 2 clusters case, one can switch the labels of all points in each cluster and still get the maximum score. Switching all labels makes all the points wrongly labelled compared to the given ground truth but nonetheless the clustering itself remains true. This explains the rounded shape in figure 6.6 where the used datasets in the right panel only consisted of 2 clusters. The critic assigns a high score either when all the labels match the given ground truth or when all the labels are permuted (which does not affect the correctness of the clustering).

6.4.2 Results on MNIST Datasets

MNIST datasets give similar results both in terms of ACC and NMI on all test datasets regardless of the used training dataset (see table 6.4). Hence, the model effectively capture implicit features that are dataset independent. While standard training shows better results, the few shots training has close performance.

Table 6.5 shows the percentage of times the critic attributes the best score to the desired solution. It shows that ES algorithm choice has a significant impact on the overall performance. Even with a metric that attributes the best score to the desired clustering, the CEM may be stuck in a local optimum and fails to reconstruct back the desired clustering. Hence, a better optimisation can enhance the performance shown in table 6.4 closer to the one presented in table 6.5.

We compare our approach with baseline methods from the literature (tables 6.6 and 6.7). For some methods, we followed the procedure in [Ha18] and used their backbone neural network as a pairwise similarity metric. Table 6.6 reports results when training on SVHN and testing on MNIST

Training Dataset	Testing Dataset					
	Numbers		Letters		Fashion	
	ACC	NMI	ACC	NMI	ACC	NMI
Numbers (standard)	72.3%	0.733	81.3%	0.861	65.2%	0.792
Numbers (few shots)	68.5%	0.801	79.0%	0.821	61.8%	0.672
Letters (standard)	75.9%	0.772	83.7%	0.854	67.5%	0.800
Letters (few shots)	69.8%	0.812	78.7%	0.806	60.9%	0.641
Fashion (standard)	70.6%	0.706	83.4%	0.858	72.5%	0.762
Fashion (few shots)	70.1%	0.690	82.1%	0.834	70.7%	0.697

Table 6.4: Mean clustering performance on MNIST dataset.

Training Dataset	Testing Dataset					
	Numbers		Letters		Fashion	
	Best	Top 3	Best	Top 3	Best	Top 3
Numbers (standard)	78.3%	92.5%	86.0%	97.5%	69.2%	87.2%
Numbers (few shots)	75.8%	82.1%	83.3%	92.0%	65.1%	83.9%
Letters (standard)	77.4%	89.2%	88.8%	96.4%	70.2%	86.7%
Letters (few shots)	73.1%	80.6%	85.1%	91.5%	61.0%	76.3%
Fashion (standard)	70.1%	83.1%	85.0%	98.6%	76.9%	94.7%
Fashion (few shots)	67.9%	77.4%	83.5%	95.3%	70.2%	88.0%

Table 6.5: Critic based performance assessment: Best corresponds to the percentage of times the critic gives the best score to the desired solution. Top 3 is when this solution is among the 3 highest scores.

numbers. We obtain close ACC values to CCN and ATDA [SUH17]. These methods use Omniglot as an auxiliary dataset to learn a pairwise similarity function, which is not required for our model. Our model only uses a small fraction of SVHN, has shallow networks and does not require any adaptation to its loss function to achieve comparable results. Finally, other cited methods require the number of clusters as an a priori indication. We achieve comparable results without needing this information. When the loss adaptation through Omniglot is discarded (denoted source-only in table 6.6), or if the number of clusters is not given, their accuracy falls and our model surpasses them by a margin.

Table 6.7 reports results when training on Omniglot_{train} and testing on Omniglot_{test}. Values are averaged across 20 alphabets which have 20 to 47 letters. We set the maximum number of clusters $k = 100$. When the number of clusters is unknown, we get an ACC score relatively close to DTC and Autonovel. Compared to these two approaches, our method bears several significant advantages:

- **Deep Networks:** DTC and Autonovel used Resnets as a backbone which are very deep networks while we only used shallow networks (2 layers maximum)

Method	ACC	
	Loss Adaptation	Source Only
DANN [Ga16]	73.9%	54.9%
LTR [SSa16]	78.8%	54.9%
ATDA [SUH17]	86.2%	70.1%
CCN [Ha18]	89.1%	52%
Ours (standard)	–	84.3%
Ours (few shots)	–	81.4%

Table 6.6: Unsupervised cross-task transfer from SVHN to MNIST digits.

Method	ACC	NMI
k-means	18.9%	0.464
CSP [WQD14]	65.4%	0.812
MPCK-means [BBM04]	53.9%	0.816
CCN [Ha18]	78.18%	0.874
DTC [HVZ19]	87.0%	0.945
Autonovel [HRa21]	85.4%	–
Ours (standard)	83.4%	0.891

Table 6.7: Unsupervised cross-task transfer from Omniglot_{train} to Omniglot_{test} ($k = 100$ for all).

- **Pairwise similarity:** in Autonovel the authors used a pairwise similarity statistic between datasets instances which we aimed to avoid due to its significant computational bottleneck. Moreover, this metric is recalculated after each training epoch, which adds more complexity.
- **Vision tasks:** While DTC can only handle vision tasks, we present a more general framework which includes vision but also tabular datasets.
- **Number of classes:** DTC and Autonovel used the labelled dataset as a probe dataset, and estimates the number of classes iteratively, and when the labelled clusters are correctly recovered, they used the ACC metric to keep the best clustering. This approach is effective, but requires access to the labelled dataset at inference time to estimate the number of classes. This is a shortcoming (memory or privacy limitations). Our approach does not require the labelled dataset once the metric is learned. Our metric automatically estimates the number of clusters required to any new unlabelled dataset.

6.5 Conclusion and Discussion

We presented a framework for cross domain/task clustering by learning a transferable metric. This framework consisted of ES methods, and GAE alongside a critic. Our model extracts dataset-independent features from labelled datasets that characterise a given clustering, performs the

clustering and grades its quality. We showed successful results using only small datasets and relatively shallow architectures. Moreover, there is more room for improvement. Indeed, since our framework is composed of 3 different blocs (CEM, GAE, critic), overall efficiency can be enhanced by independently improving each bloc like replacing the CEM by a better performing algorithm for instance.

In future work, we will study the criteria that determine why some auxiliary datasets are more resourceful than others given a target dataset. In our case, this means to study for instance why using the MNIST letters dataset as training allowed a better performance on Fashion MNIST than when using MNIST numbers. This would allow to deliver a minimum performance guarantee at inference time by creating a transferability measure between datasets.

Although our obtained results are satisfying, this work remains a preliminary approach to the goal stated in the introduction section, which is transferable reward functions learnt through IRL. Indeed as we only tackled one step MDP problems with deterministic dynamics, there is no randomness induced by the environments. Therefore a distributional approach on our reward function (the metric) was not possible. This also explains why we used the Wasserstein distance, implementing a WGAN critic instead of using the Cramèr distance dual formulation as per [Bel+17]. Moreover, [Bel+17] is controversial as in dimensions greater than 1, the paper doesn't use the Cramer distance but the energy distance. It is also important to notice that our presented framework remains adversarial instead of collaborative contrary to the approaches presented in previous chapters. We believe a distributional approach applied to multi-step stochastic MDPs might allow to use collaborative models, avoiding the pitfalls of adversarial models.

Finally, while there is a huge amount of work towards learning transferable reward functions, we believe that the quotient space approach and the graph embedding process used in the presented models hold significant promise and will be under investigation in subsequent work tackling multi-step stochastic MDPs. Indeed as quotient maps capture the equivalence relation used in demonstrated clusterings they also capture this logic; this means that two clusters that are similar under the presented logic are mapped to close points in terms of computable distance. We believe the quotient space framework holds great potential as it allows to compare logics that induced two different quotient spaces.

Capturing and comparing logics is of great interest for the AI alignment field. Indeed, we pointed out in previous chapters that IRL approaches can better handle the outer alignment problem; the reward function describes the task by capturing the logic demonstrated by the expert. However, in practice demonstrations are often provided by different experts that do not follow the exact same logic or utility function. While our method could help capture better demonstrated logics, it may also be useful to compare them to each other.

Chapter 7

Conclusion

In this thesis we hypothesised that learning meta-rewards would help solving the mis-alignment issues in AI highlighted in the Introduction chapter 1.

We first gave in chapter 3 a vivid illustration of the outer alignment issue, showing that the same reward function can lead agents to converge towards different succeeding policies that differ greatly in their behaviour. We proposed an algorithm called CAMEO that took advantage of this situation allowing to sample succeeding policies with different behaviours on the fly from the distribution of succeeding policies. Such algorithm could be used for risk aware reinforcement learning, sampling policies that correspond to different risk profiles. Moreover, studying the resulting behaviours can help to craft a better reward function that matches better the desired behaviour, in favour of a better alignment between human and base objectives.

In chapter 4, we focused on the inner alignment problem and explained that in order to better take risk into account, using expected return maximisation as a mesa objective is not sufficient. We therefore proposed to use distributional approaches to RL. However, existing approaches present several limitations such as the inability to use loss functions in practice that guarantee convergence in theory or the incapacity to learn explicit target distributions, forbidding the evaluation of a given return density under a specified policy. We presented a distributional RL approach based on a particular invertible generative model, namely Normalizing Flows, that offers proper convergence guaranties. Indeed our approach allowed to compute easily the Cramèr distance and use it as a loss function even when only sample transitions are available.

In the Introduction chapter, we advocated for the use of Inverse RL to tackle the outer alignment issue, where the human objective is mis-aligned with the implemented reward function. Throughout this manuscript we laid some of the building blocks supporting that claim. Building on the contribution mentioned above, we proposed in chapter 5 a distributional approach to Inverse RL. While existing models are based on fitting occupancy measures and adversarial networks, we showed that we could obtain similar results with collaborative networks and comparing directly the return distributions of expert policies and learning agents. We showed that our approach is able to match the set of actions effectively chosen by the expert contrary to other approaches. Finally, while our approach is more straightforward and brings more convergence stability, we showed that it brought the same benefits as state of the art models, i.e. incentivizing

imitation of demonstrated actions and (2) promoting return to demonstrated states when faced with new situations.

Finally, we tackled the Meta-learning problem in chapter 6. Our main hypothesis was that learning a meta-reward can help tackle the distributional shift (inner alignment) and outer alignment in the same time if the meta-reward is learnt using expert demonstrations i.e IRL. As a first approach, in order to simplify the problem, we first tackled the clustering problem, viewing it as a one step MDP. We proposed a Wasserstein and GNN based approach that succeeded to learn metrics that would capture the logic used by the expert to solve some task on some dataset and apply it to rightfully cluster another dataset under the same logic.

The results showcased in this last chapter deserve some more in depth discussion. We argued in the introduction section that the main hurdle in the road of AI alignment is the incapacity for current models to build on previous knowledge and build abstract concepts. Humans and many other animals possess the remarkable ability of extrapolation, enabling them to apply their learned knowledge to novel situations. This capacity relies on two fundamental cognitive processes: the construction of abstract representations and the ability to create analogies that link these representations to new contexts. Furthermore, this cognitive framework is enhanced by core knowledge—a set of fundamental concepts shared by all humans. This core knowledge encompasses essential understandings of objects, space, and time. Together, these cognitive mechanisms empower organisms to accomplish various tasks, including:

- **Concept Learning:** Individuals can grasp new concepts based on a limited number of examples. By identifying common patterns and abstracting key features, they can generalise their understanding to classify and recognise novel instances.
- **Imitation and Behavioural Generation:** The ability to learn and generate behaviours extends beyond mere replication. Instead, organisms can comprehend the underlying concepts and principles behind behaviours, allowing them to adapt and apply these principles in different contexts.
- **Knowledge Transfer:** With a foundation of abstract representations and analogical reasoning, individuals can transfer knowledge from one domain to another. This capacity enables them to leverage what they have learned in one context to solve problems or navigate situations in unrelated domains.

How to know if a machine is capable of understanding? If the machine can perform an action (jump), recognise the concept when it is illustrated by others (show people jumping), be able to do the same thing with a different system (another robot jumping). In the work we presented in chapter 6, we showed that a metric based model should be able to understand a concept encompassed in the demonstrations and apply it to new situations. We therefore believe that our approach is a first step towards models that learn analogies. Indeed, Douglas Hofstadter defines a concept using the following property: “a concept is a package of analogies” in [HG95]¹. The ability of the human mind to make analogies is at the source of the most important cognitive abilities. The absence of these abilities is at least partly responsible for the fragility of current AI

¹As a fun fact, the book containing this statement, Fluid concepts and creative analogies: Computer models of the fundamental mechanisms of thought, was the first ever book sold on Amazon.com

and its difficulty in adapting knowledge and representations to new situations. As highlighted by Melanie Mitchell in [Mit21], several scientific revolutions are based on analogies (Darwin realizing that biological competition is analogous to economic competition, or Von Neumann making analogies between the brain and a computer). These examples illustrate two important facts:

- Analogy is not a rare occurrence, but rather a constant and ubiquitous mode of thought.
- Analogies are a key aspect not only of reasoning, but also of concept formation and abstraction.

We believe that learning core concepts and the ability to perform analogies represent the primary components for mitigating alignment challenges in AI. In this work, our results showed promise for the adoption of meta-reward learning through IRL, as exemplified by the promising outcomes observed in addressing the one-step MDP problem. Undoubtedly, the clustering framework served as an initial step and does not mark the culmination of the research pursued in this thesis.

Naturally, the subsequent phase involves the application of the approach delineated in chapter 6 to a genuine RL setup, specifically a multi-step MDP, building upon the methodologies introduced in chapters 4 and 5. Above all, we made a modest yet significant contribution through this work towards bringing us closer to the development of agents capable of demonstrating the generality and adaptability akin to human intelligence, and, perhaps, inching closer to unraveling the essence of general intelligence itself.

References

- [Mar52] Harry Markowitz. “Portfolio Selection”. In: *The Journal of Finance* 7.1 (1952), pp. 77–91. ISSN: 00221082, 15406261. URL: <http://www.jstor.org/stable/2975974> (visited on 12/14/2023).
- [BD62] Richard E. Bellman and Stuart E Dreyfus. *Applied Dynamic Programming*. Princeton: Princeton University Press, 1962. ISBN: 9781400874651. DOI: [doi:10.1515/9781400874651](https://doi.org/10.1515/9781400874651). URL: <https://doi.org/10.1515/9781400874651>.
- [Hub64] Peter J. Huber. “Robust Estimation of a Location Parameter”. In: *The Annals of Mathematical Statistics* 35.1 (1964), pp. 73–101. DOI: [10.1214/aoms/1177703732](https://doi.org/10.1214/aoms/1177703732). URL: <https://doi.org/10.1214/aoms/1177703732>.
- [BSA83] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. “Neuronlike adaptive elements that can solve difficult learning control problems”. In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13.5 (1983), pp. 834–846. DOI: [10.1109/TSMC.1983.6313077](https://doi.org/10.1109/TSMC.1983.6313077).
- [WD92] Christopher J. C. H. Watkins and Peter Dayan. “Q-learning”. In: *Machine Learning* 8.3 (May 1992), pp. 279–292. ISSN: 1573-0565. DOI: [10.1007/BF00992698](https://doi.org/10.1007/BF00992698). URL: <https://doi.org/10.1007/BF00992698>.
- [Put94] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. 1st. USA: John Wiley & Sons, Inc., 1994. ISBN: 0471619779.
- [HG95] D. R. Hofstadter and Fluid Analogies Research Group. *Fluid concepts and creative analogies: Computer models of the fundamental mechanisms of thought*. Basic Books, 1995.
- [Mül97] Alfred Müller. “Integral Probability Metrics and Their Generating Classes of Functions”. In: *Advances in Applied Probability* 29.2 (1997), pp. 429–443. ISSN: 00018678. URL: <http://www.jstor.org/stable/1428011> (visited on 07/16/2023).
- [Rus98] Stuart Russell. “Learning Agents for Uncertain Environments (Extended Abstract)”. In: *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*. COLT’ 98. Madison, Wisconsin, USA: Association for Computing Machinery, 1998, pp. 101–103. ISBN: 1581130570. DOI: [10.1145/279943.279964](https://doi.org/10.1145/279943.279964). URL: <https://doi.org/10.1145/279943.279964>.
- [Sut+99] Richard S Sutton et al. “Policy Gradient Methods for Reinforcement Learning with Function Approximation”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Solla, T. Leen, and K. Müller. Vol. 12. MIT Press, 1999. URL: https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

- [NR00] Andrew Y. Ng and Stuart J. Russell. “Algorithms for Inverse Reinforcement Learning”. In: *Proceedings of the Seventeenth International Conference on Machine Learning*. ICML ’00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 663–670. ISBN: 1558607072.
- [SG02] Alexander Strehl and Joydeep Ghosh. “Cluster ensembles—a knowledge reuse framework for combining multiple partitions”. In: *JMLR* 3(Dec) (2002), pp. 583–617.
- [And+03] Christophe Andrieu et al. “An Introduction to MCMC for Machine Learning”. In: *Machine Learning* 50 (2003), pp. 5–43. DOI: [10.1023/A:1020281327116](https://doi.org/10.1023/A:1020281327116). URL: <https://doi.org/10.1023/A:1020281327116>.
- [AN04] Pieter Abbeel and Andrew Y. Ng. “Apprenticeship Learning via Inverse Reinforcement Learning”. In: *Proceedings of the Twenty-First International Conference on Machine Learning*. ICML ’04. Banff, Alberta, Canada: Association for Computing Machinery, 2004, p. 1. ISBN: 1581138385. DOI: [10.1145/1015330.1015430](https://doi.org/10.1145/1015330.1015430). URL: <https://doi.org/10.1145/1015330.1015430>.
- [BBM04] Mikhail Bilenko, Sugato Basu, and Raymond J Mooney. “Integrating constraints and metric learning in semi-supervised clustering”. In: *ICML* (2004), p. 11.
- [BKa05] Pieter-Tjerk de Boer, Dirk P Kroese, and et. al. “A tutorial on the cross-entropy method”. en. In: *Annals of Operations Research* 134.1 (Feb. 2005), pp. 19–67.
- [Koe05] Roger Koenker. *Quantile Regression*. Econometric Society Monographs. Cambridge University Press, 2005. DOI: [10.1017/CBO9780511754098](https://doi.org/10.1017/CBO9780511754098).
- [Hof+07] Matthew Hoffman et al. “Bayesian Policy Learning with Trans-Dimensional MCMC”. In: *Advances in Neural Information Processing Systems*. Ed. by J. Platt et al. Vol. 20. Curran Associates, Inc., 2007. URL: https://proceedings.neurips.cc/paper_files/paper/2007/file/3a15c7d0bbe60300a39f76f8a5ba6896-Paper.pdf.
- [PDM08] PDMIA. *Processus Décisionnels de Markov en Intelligence Artificielle*. 2008. URL: <http://researchers.lille.inria.fr/~munos/papers/files/bouquinPDMIA.pdf>.
- [SBS08] Umar Syed, Michael Bowling, and Robert E. Schapire. “Apprenticeship Learning Using Linear Programming”. In: *Proceedings of the 25th International Conference on Machine Learning*. ICML ’08. Helsinki, Finland: Association for Computing Machinery, 2008, pp. 1032–1039. ISBN: 9781605582054. DOI: [10.1145/1390156.1390286](https://doi.org/10.1145/1390156.1390286). URL: <https://doi.org/10.1145/1390156.1390286>.
- [Zie+08] Brian D. Ziebart et al. “Maximum Entropy Inverse Reinforcement Learning”. In: *Proc. AAAI*. 2008, pp. 1433–1438.
- [Has10] Hado Hasselt. “Double Q-learning”. In: *Advances in Neural Information Processing Systems*. Ed. by J. Lafferty et al. Vol. 23. Curran Associates, Inc., 2010. URL: https://proceedings.neurips.cc/paper_files/paper/2010/file/091d584fced301b442654dd8c23b3fc9-Paper.pdf.
- [LCB10] Yann LeCun, Corinna Cortes, and CJ Burges. “MNIST handwritten digit database”. In: *ATT Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist> 2 (2010).
- [YXa10] Yi Yang, Dong Xu, and et. al. “Image clustering using local discriminant models and global integration”. In: *IEEE Transactions on Image Processing* 19(10) (2010), pp. 2761–2773.
- [Zie10] Brian D. Ziebart. “Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy”. AAI3438449. PhD thesis. USA, 2010. ISBN: 9781124414218.

- [Net+11] Yuval Netzer et al. “Reading Digits in Natural Images with Unsupervised Feature Learning”. In: *NIPS Workshop on Deep Learning and Unsupervised Feature Learning* (2011). URL: <http://ufldl.stanford.edu/housenumbers>.
- [Ped+11] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
- [RGB11] Stephane Ross, Geoffrey Gordon, and Drew Bagnell. “A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning”. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, Apr. 2011, pp. 627–635. URL: <https://proceedings.mlr.press/v15/ross11a.html>.
- [Bos12] Nick Bostrom. “The superintelligent Will: Motivation and Instrumental Rationality in Advanced Artificial Agents”. In: *Minds and Machines*. 2012.
- [Gro+12] Ivo Grondman et al. “A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients”. In: *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics* 42 (Nov. 2012), pp. 1291–1307. DOI: [10.1109/TSMCC.2012.2218595](https://doi.org/10.1109/TSMCC.2012.2218595).
- [TET12] Emanuel Todorov, Tom Erez, and Yuval Tassa. “MuJoCo: A physics engine for model-based control”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 5026–5033. DOI: [10.1109/IROS.2012.6386109](https://doi.org/10.1109/IROS.2012.6386109).
- [Bel+13] M. G. Bellemare et al. “The Arcade Learning Environment: An Evaluation Platform for General Agents”. In: *Journal of Artificial Intelligence Research* 47 (June 2013), pp. 253–279. DOI: [10.1613/jair.3912](https://doi.org/10.1613/jair.3912). URL: <https://doi.org/10.1613%2Fjair.3912>.
- [Mni+13] Volodymyr Mnih et al. “Playing Atari with Deep Reinforcement Learning”. In: *CoRR abs/1312.5602* (2013). arXiv: [1312.5602](https://arxiv.org/abs/1312.5602). URL: <http://arxiv.org/abs/1312.5602>.
- [Sej+13] Dino Sejdinovic et al. “Equivalence of distance-based and RKHS-based statistics in hypothesis testing”. In: *The Annals of Statistics* 41.5 (Oct. 2013). DOI: [10.1214/13-aos1140](https://doi.org/10.1214/13-aos1140). URL: <https://doi.org/10.1214%2F13-aos1140>.
- [Goo+14] Ian Goodfellow et al. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.
- [KW14] Diederik P Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: *International Conference on Learning Representations (ICLR)*. 2014. arXiv: [1312.6114](https://arxiv.org/abs/1312.6114) [stat.ML].
- [Sil+14] David Silver et al. “Deterministic Policy Gradient Algorithms”. In: *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32. ICML’14*. Beijing, China: JMLR.org, 2014, I–387–I–395.
- [WQD14] Xiang Wang, Buyue Qian, and Ian Davidson. “On constrained spectral clustering and its applications”. In: *Data Mining and Knowledge Discovery* (2014), pp. 1–30.
- [DKB15] Laurent Dinh, David Krueger, and Yoshua Bengio. “NICE: Non-linear Independent Components Estimation”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: <http://arxiv.org/abs/1410.8516>.

- [HGS15] Hado van Hasselt, Arthur Guez, and David Silver. “Deep Reinforcement Learning with Double Q-learning”. In: *Thirtieth AAAI Conference on Artificial Intelligence* abs/1509.06461 (2015). arXiv: [1509.06461](https://arxiv.org/abs/1509.06461). URL: <http://arxiv.org/abs/1509.06461>.
- [LST15] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. “Human-level concept learning through probabilistic program induction”. In: *Science* 350(6266) (2015), pp. 1332–1338.
- [Sch+15] John Schulman et al. “Trust Region Policy Optimization”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 1889–1897. URL: <https://proceedings.mlr.press/v37/schulman15.html>.
- [Bro+16] Greg Brockman et al. *OpenAI Gym*. 2016. arXiv: [1606.01540](https://arxiv.org/abs/1606.01540) [cs.LG].
- [FLA16] Chelsea Finn, Sergey Levine, and Pieter Abbeel. “Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization”. In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 2016, pp. 49–58. URL: <https://proceedings.mlr.press/v48/finn16.html>.
- [Fin+16] Chelsea Finn et al. *A Connection between Generative Adversarial Networks, Inverse Reinforcement Learning, and Energy-Based Models*. 2016. arXiv: [1611.03852](https://arxiv.org/abs/1611.03852) [cs.LG].
- [Ga16] Ganin and et al. “Domain-adversarial training of neural networks”. In: *JMLR* (2016).
- [HE16] Jonathan Ho and Stefano Ermon. “Generative Adversarial Imitation Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/cc7e2b878868cbae992d1fb743995d8f-Paper.pdf.
- [KW16] Thomas N. Kipf and Max Welling. “Variational Graph Auto-Encoders”. In: *Bayesian Deep Learning Workshop (NIPS 2016)*. arXiv, 2016.
- [Mni+16] Volodymyr Mnih et al. “Asynchronous Methods for Deep Reinforcement Learning”. In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, June 2016, pp. 1928–1937. URL: <https://proceedings.mlr.press/v48/mniha16.html>.
- [Sch+16] Tom Schaul et al. “Prioritized Experience Replay”. In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2016. URL: <http://arxiv.org/abs/1511.05952>.
- [SSa16] Ozan Sener, Hyun Oh Song, and et al. “Learning transfer able representations for unsupervised domain adaptation.” In: *NIPS* (2016), pp. 2110–2118.
- [WFL16] Ziyu Wang, Nando de Freitas, and Marc Lanctot. “Dueling Network Architectures for Deep Reinforcement Learning”. In: *Proceedings of the 33rd International Conference on Machine Learning* abs/1511.06581 (2016). arXiv: [1511.06581](https://arxiv.org/abs/1511.06581). URL: <http://arxiv.org/abs/1511.06581>.
- [XGF16] Junyuan Xie, Ross Girshick, and Ali Farhadi. “Unsupervised Deep Embedding for Clustering Analysis”. In: *ICML*. June 2016, pp. 478–487.
- [ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. “Wasserstein Generative Adversarial Networks”. In: *Proceedings of the 34th International Conference on*

- Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, Aug. 2017, pp. 214–223. URL: <https://proceedings.mlr.press/v70/arjovsky17a.html>.
- [BDM17] Marc G. Bellemare, Will Dabney, and Rémi Munos. “A Distributional Perspective on Reinforcement Learning”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, Aug. 2017, pp. 449–458. URL: <https://proceedings.mlr.press/v70/bellemare17a.html>.
- [Bel+17] Marc G. Bellemare et al. *The Cramer Distance as a Solution to Biased Wasserstein Gradients*. 2017. arXiv: [1705.10743](https://arxiv.org/abs/1705.10743) [cs.LG].
- [Coh+17] Gregory Cohen et al. “EMNIST: Extending MNIST to handwritten letters”. In: *2017 International Joint Conference on Neural Networks (IJCNN)* (2017). DOI: [10.1109/ijcnn.2017.7966217](https://doi.org/10.1109/ijcnn.2017.7966217).
- [Dab+17] Will Dabney et al. “Distributional Reinforcement Learning with Quantile Regression”. In: *AAAI*. 2017. arXiv: [1710.10044](https://arxiv.org/abs/1710.10044) [cs.AI].
- [FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, Aug. 2017, pp. 1126–1135. URL: <https://proceedings.mlr.press/v70/finn17a.html>.
- [Had+17] Dylan Hadfield-Menell et al. “Inverse Reward Design”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/32fdab6559cdfa4f167f8c31b9199643-Paper.pdf.
- [Pat+17] Deepak Pathak et al. “Curiosity-driven Exploration by Self-supervised Prediction”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, Aug. 2017, pp. 2778–2787. URL: <https://proceedings.mlr.press/v70/pathak17a.html>.
- [PGP17] Bilal Piot, Matthieu Geist, and Olivier Pietquin. “Bridging the Gap Between Imitation Learning and Inverse Reinforcement Learning”. In: *IEEE Transactions on Neural Networks and Learning Systems* 28.8 (2017), pp. 1814–1826. DOI: [10.1109/TNNLS.2016.2543000](https://doi.org/10.1109/TNNLS.2016.2543000).
- [SUH17] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. “Asymmetric tri-training for unsupervised domain adaptation”. In: *ICML* (2017), pp. 2988–2997.
- [Sch+17] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: [1707.06347](https://arxiv.org/abs/1707.06347) [cs.LG].
- [XRV17] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. Aug. 28, 2017. arXiv: [cs.LG/1708.07747](https://arxiv.org/abs/cs.LG/1708.07747) [cs.LG].
- [Dab+18] Will Dabney et al. “Implicit Quantile Networks for Distributional Reinforcement Learning”. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, July 2018, pp. 1096–1105. URL: <https://proceedings.mlr.press/v80/dabney18a.html>.

- [For+18] Meire Fortunato et al. “Noisy Networks For Exploration”. In: *International Conference on Learning Representations*. 2018. URL: <https://openreview.net/forum?id=rywHCPkAW>.
- [FLL18] Justin Fu, Katie Luo, and Sergey Levine. “Learning Robust Rewards with Adversarial Inverse Reinforcement Learning”. In: *International Conference on Learning Representations*. 2018. URL: <https://openreview.net/forum?id=rkHywl-A->.
- [Haa+18] Tuomas Haarnoja et al. “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor”. In: *International Conference on Machine Learning (ICML)* (2018).
- [Hes+18] Matteo Hessel et al. “Rainbow: Combining Improvements in Deep Reinforcement Learning”. In: *AAAI*. 2018. arXiv: [1710.02298](https://arxiv.org/abs/1710.02298) [cs.AI].
- [Ha18] Hsu and et al. “Learning to cluster in order to transfer across domains and tasks”. In: *ICLR*. arXiv, 2018.
- [KD18] Durk P Kingma and Prafulla Dhariwal. “Glow: Generative Flow with Invertible 1x1 Convolutions”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/d139db6a236200b21cc7f752979132d0-Paper.pdf.
- [Lev18] Sergey Levine. *Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review*. 2018. arXiv: [1805.00909](https://arxiv.org/abs/1805.00909) [cs.LG].
- [Row+18] Mark Rowland et al. “An Analysis of Categorical Distributional Reinforcement Learning”. In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. Ed. by Amos Storkey and Fernando Perez-Cruz. Vol. 84. Proceedings of Machine Learning Research. PMLR, Apr. 2018, pp. 29–37. URL: <https://proceedings.mlr.press/v84/rowland18a.html>.
- [SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [TOY18] Vahid Tavakol Aghaei, Ahmet Onat, and Sinan Yildirim. “A Markov chain Monte Carlo algorithm for Bayesian policy search”. In: *Systems Science and Control Engineering An Open Access Journal* Volume 6 (Sept. 2018), pp. 438–455. DOI: [10.1080/21642583.2018.1528483](https://doi.org/10.1080/21642583.2018.1528483).
- [Bur+19] Yuri Burda et al. “Large-Scale Study of Curiosity-Driven Learning”. In: *International Conference on Learning Representations*. 2019. URL: <https://openreview.net/forum?id=rJNwDjAqYX>.
- [Eys+19] Benjamin Eysenbach et al. “Diversity is All You Need: Learning Skills without a Reward Function”. In: *International Conference on Learning Representations*. 2019. URL: <https://openreview.net/forum?id=SJx63jRqFm>.
- [GSP19] Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. “A Theory of Regularized Markov Decision Processes”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 2160–2169. URL: <https://proceedings.mlr.press/v97/geist19a.html>.
- [HVZ19] Kai Han, Andrea Vedaldi, and Andrew Zisserman. “Learning to Discover Novel Visual Categories via Deep Transfer Clustering”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019.

- [Ho+19] Jonathan Ho et al. “Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 2722–2730. URL: <https://proceedings.mlr.press/v97/ho19a.html>.
- [Mül+19] Thomas Müller et al. “Neural Importance Sampling”. In: *ACM Trans. Graph.* 38.5 (Oct. 2019). ISSN: 0730-0301. DOI: [10.1145/3341156](https://doi.org/10.1145/3341156). URL: <https://doi.org/10.1145/3341156>.
- [Rak+19] Kate Rakelly et al. “Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 5331–5340. URL: <https://proceedings.mlr.press/v97/rakelly19a.html>.
- [TA19] Yunhao Tang and Shipra Agrawal. *Boosting Trust Region Policy Optimization by Normalizing Flows Policy*. 2019. arXiv: [1809.10326](https://arxiv.org/abs/1809.10326) [cs.AI].
- [WSB19] Patrick Nadeem Ward, Ariella Smofsky, and Avishek Joey Bose. “Improving Exploration in Soft-Actor-Critic with Normalizing Flows Policies”. In: *INNF workshop, International Conference on Machine Learning*. 2019. arXiv: [1906.02771](https://arxiv.org/abs/1906.02771) [cs.LG].
- [Xia+19] Huang Xiao et al. *Wasserstein Adversarial Imitation Learning*. 2019. arXiv: [1906.08113](https://arxiv.org/abs/1906.08113) [cs.LG].
- [Xu+19] Kelvin Xu et al. “Learning a Prior over Intent via Meta-Inverse Reinforcement Learning”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 6952–6962. URL: <https://proceedings.mlr.press/v97/xu19d.html>.
- [Yu+19] Lantao Yu et al. “Meta-Inverse Reinforcement Learning with Probabilistic Context Variables”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/30de24287a6d8f07b37c716ad51623a7-Paper.pdf.
- [ZR19] Zachary Ziegler and Alexander Rush. “Latent Normalizing Flows for Discrete Sequences”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 7673–7682. URL: <https://proceedings.mlr.press/v97/ziegler19a.html>.
- [Cam+20] Victor Campos et al. “Explore, Discover and Learn: Unsupervised Discovery of State-Covering Skills”. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, July 2020, pp. 1317–1327. URL: <https://proceedings.mlr.press/v119/campos20a.html>.
- [Fer+20] Antonio J. Fernández et al. “Some non-homogeneous Gagliardo-Nirenberg inequalities and application to a biharmonic non-linear Schrödinger equation”. In: *Journal of Differential Equations* 330 (2022), 1-65 (2020). DOI: [10.1016/j.jde.2022.04.037](https://doi.org/10.1016/j.jde.2022.04.037). eprint: [arXiv:2010.01448](https://arxiv.org/abs/2010.01448).

- [Kra+20] Victoria Krakovna et al. *Specification gaming: the flip side of AI ingenuity*. 2020. URL: <https://www.deepmind.com/blog/specification-gaming-the-flip-side-of-ai-ingenuity>.
- [Maz+20] Bogdan Mazouze et al. “Leveraging exploration in off-policy algorithms via normalizing flows”. In: *Proceedings of the Conference on Robot Learning*. Ed. by Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura. Vol. 100. Proceedings of Machine Learning Research. PMLR, Oct. 2020, pp. 430–444. URL: <https://proceedings.mlr.press/v100/mazouze20a.html>.
- [Sha+20] Archit Sharma et al. “Dynamics-Aware Unsupervised Discovery of Skills”. In: *International Conference on Learning Representations*. 2020. URL: <https://openreview.net/forum?id=HJgLZR4KvH>.
- [Cra21] Maria Cramer. “A.I. Drone May Have Acted on Its Own in Attacking Fighters, U.N. Says”. In: *New York Times* (June 4, 2021). URL: <https://www.nytimes.com/2021/06/03/world/africa/libya-drone.html>.
- [GRV21] Marylou Gabrié, Grant M. Rotskoff, and Eric Vanden-Eijnden. “Efficient Bayesian Sampling Using Normalizing Flows to Assist Markov Chain Monte Carlo Methods”. In: *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*. 2021. URL: <https://openreview.net/forum?id=mvtooHbjOwx>.
- [Gro+21] Oliver Groth et al. *Is Curiosity All You Need? On the Utility of Emergent Behaviours from Curious Exploration*. 2021. arXiv: [2109.08603](https://arxiv.org/abs/2109.08603) [cs.LG].
- [HRa21] Kai Han, Sylvestre-Alvise Rebuffi, and et. al. “AutoNovel: Automatically Discovering and Learning Novel Visual Categories”. In: *PAMI* (2021), pp. 1–1.
- [Hub+21] Evan Hubinger et al. *Risks from Learned Optimization in Advanced Machine Learning Systems*. 2021. arXiv: [1906.01820](https://arxiv.org/abs/1906.01820) [cs.AI].
- [KPB21] Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker. “Normalizing Flows: An Introduction and Review of Current Methods”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.11 (Nov. 2021), pp. 3964–3979. DOI: [10.1109/tpami.2020.2992934](https://doi.org/10.1109/tpami.2020.2992934). URL: <https://doi.org/10.1109/tpami.2020.2992934>.
- [Mit21] Melanie Mitchell. “Why AI is harder than we think”. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO ’21. Lille, France: Association for Computing Machinery, 2021, p. 3. ISBN: 9781450383509. DOI: [10.1145/3449639.3465421](https://doi.org/10.1145/3449639.3465421). URL: <https://doi.org/10.1145/3449639.3465421>.
- [Pap+21] George Papamakarios et al. “Normalizing Flows for Probabilistic Modeling and Inference”. In: *Journal of Machine Learning Research* 22.57 (2021), pp. 1–64. URL: <http://jmlr.org/papers/v22/19-1028.html>.
- [AF22] Prashanth L. A. and Michael Fu. *Risk-Sensitive Reinforcement Learning via Policy Gradient Search*. 2022. arXiv: [1810.09126](https://arxiv.org/abs/1810.09126) [cs.LG].
- [Bro+22] James Brofos et al. “Adaptation of the Independent Metropolis-Hastings Sampler with Normalizing Flow Proposals”. In: *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*. Ed. by Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera. Vol. 151. Proceedings of Machine Learning Research. PMLR, Mar. 2022, pp. 5949–5986. URL: <https://proceedings.mlr.press/v151/brofos22a.html>.

- [Cha+22] Alan Chan et al. “Greedification Operators for Policy Optimization: Investigating Forward and Reverse KL Divergences”. In: *Journal of Machine Learning Research* 23.253 (2022), pp. 1–79. URL: <http://jmlr.org/papers/v23/21-054.html>.
- [Li+22] Zewen Li et al. “A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects”. In: *IEEE Transactions on Neural Networks and Learning Systems* 33.12 (2022), pp. 6999–7019. DOI: [10.1109/TNNLS.2021.3084827](https://doi.org/10.1109/TNNLS.2021.3084827).
- [Ung22] Rebecca Ungarino. “Here are 9 fascinating facts to know about BlackRock, the world’s largest asset manager popping up in the Biden administration”. In: *Business Insider* (Mar. 10, 2022). URL: <https://www.businessinsider.com/what-to-know-about-blackrock-larry-fink-biden-cabinet-facts-2020-12?r=US&IR=T>.
- [Bla23] Inc. BlackRock. “2022 Form 10-K Annual Report”. In: *U.S. Securities and Exchange Commission* (Feb. 24, 2023).

Titre : Modèles Génératifs pour l'Apprentissage par Renforcement Inverse Distributionnel: Vers des Fonctions de Récompense Transférables

Mots clés : Apprentissage par renforcement, Apprentissage Profond, Modèle Génératifs, Apprentissage de Métriques, Transfert d'apprentissage, Meta-apprentissage

Résumé : Les humains possèdent une remarquable capacité à apprendre rapidement de nouveaux concepts et à s'adapter à des situations imprévues en s'appuyant sur des expériences antérieures. L'apprentissage par renforcement (RL) s'est révélé efficace pour résoudre des problèmes de prise de décision séquentielle dans des environnements dynamiques. Cependant, contrairement aux humains, les politiques apprises ne sont pas facilement transférables à différents environnements. En revanche, la fonction de récompense, représentant l'essence de la tâche à accomplir, présente un potentiel comme représentation transférable. Malheureusement, traduire l'intention humaine en fonctions mathématiques à optimiser n'est pas simple et la moindre erreur d'implémentation peut entraîner des comportements inattendus dramatiques. C'est ce qu'on appelle le problème d'alignement de l'IA.

L'apprentissage par renforcement inverse (IRL) tente d'apprendre une fonction de récompense à partir de démonstrations, mais sans garantie de transférabilité. L'hypothèse principale de cette thèse est que l'apprentissage de fonctions de récompense

transférables à plusieurs tâches similaires pourrait aider à atténuer le problème d'alignement de l'IA à travers l'apprentissage de concepts fondamentaux réutilisables, à la manière du raisonnement humain.

Dans cette thèse, nous proposons de coupler des modèles génératifs inversibles à une perspective distributionnelle en RL comme étape vers la résolution de ces défis. Cette approche se révèle avantageuse pour le problème IRL, car il devient possible d'apprendre la distribution des récompenses pour chaque état tout en l'interprétant comme une distance vis-à-vis l'état final voulu. Cela nous permet de montrer la possibilité de transférer les fonctions de récompense apprises pour les processus de décision de Markov à étapes uniques.

Cette thèse offre des perspectives pour la compréhension de l'adaptabilité en RL. Cette adaptabilité passe par l'apprentissage de concepts fondamentaux transférables sous la forme de fonctions de récompense. Nous espérons que les travaux exposés dans cette thèse permettent une meilleure atténuation des problèmes d'alignement en IA.

Title : Distributional Inverse Reinforcement Learning with Invertible Generative Models: Towards Transferable Reward Functions

Keywords : Reinforcement Learning, Deep Learning, Generative Models, Metric Learning, Transfer-Learning, Meta-Learning

Abstract : Humans possess a remarkable ability to quickly learn new concepts and adapt to unforeseen situations by drawing upon prior experiences and combining them with limited new evidence. Reinforcement Learning (RL) has proven effective in solving sequential decision-making problems in dynamic environments. However, unlike humans, learned policies are not efficiently transferable to different environments. Conversely, the reward function, representing the task's essence, holds promise as a transferable representation. Unfortunately, translating human intent into mathematical functions to optimise is not straightforward and the slightest implementation error can lead to dramatic unexpected behaviours. This is called the AI alignment issue.

Inverse Reinforcement Learning (IRL) attempts to learn a reward function from demonstrations, but there is no guarantee of transferability. The main hypothesis

of this thesis is that learning reward functions that are transferable to multiple similar tasks could help mitigate the AI alignment issue getting us closer to learning core concepts, akin to human reasoning.

In this thesis, we propose to explore the potential of invertible generative models along with a distributional perspective in RL as a step towards addressing these challenges. This approach proves advantageous for tackling IRL tasks, as we can learn the distribution of rewards for each state while interpreting the reward as a distance from the final state. Using this interpretation, we demonstrate transferability of learnt reward functions in single-step Markov Decision Processes.

This thesis offers insights into the understanding of adaptability in RL. This is done through the learning of transferable fundamental concepts in the form of reward functions. We hope that this work will allow to mitigate AI alignment issues.