HAL open science

On the parameter learning for Perturb-and-MAP models

Tatiana Shpakova

To cite this version:

Tatiana Shpakova. On the parameter learning for Perturb-and-MAP models. Machine Learning [cs.LG]. Université Paris sciences et lettres, 2019. English. NNT : 2019PSLEE077 . tel-03413229v2

HAL Id: tel-03413229

https://theses.hal.science/tel-03413229v2

Submitted on 3 Nov 2021

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.

THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences Lettres
PSL Research University

Préparée à l'École normale supérieure

On Parameter Learning for Perturb-and-MAP Models

Sur l'apprentissage des paramètres pour les modèles Perturb-and-MAP

École doctorale n°386

ÉCOLE DOCTORALE DE SCIENCES MATHÉMATIQUES DE PARIS CENTRE

Spécialité INFORMATIQUE

COMPOSITION DU JURY :

Mme. Florence d'Alché-Buc
TELECOM ParisTech, Rapporteur

M. Matthew Blaschko
KU Leuven, Rapporteur

M. Francis Bach
INRIA, Directeur de thèse

M. Ivan Laptev
INRIA, President du jury

M. Umut Şimşekli
TELECOM ParisTech, Membre du jury

Soutenue par Tatiana Shpakova
le 21.02.2019

Dirigée par **Francis BACH**

ENS ÉCOLE NORMALE SUPÉRIEURE | PSL RESEARCH UNIVERSITY PARIS

A mathematician is a device for turning coffee into theorems.

Alfréd Rényi

If I feel unhappy, I do mathematics to become happy. If I am happy, I do mathematics to keep happy.

Alfréd Rényi

Abstract

Probabilistic graphical models encode hidden dependencies between random variables for data modelling. Parameter estimation is a crucial and necessary part of handling such probabilistic models. These very general models have been used in plenty of fields such as computer vision, signal processing, natural language processing and many more. We mostly focused on log-supermodular models, which is a specific part of exponential family distributions, where the potential function is assumed to be the negative of a submodular function. This property will be very handy for maximum a posteriori and parameter learning estimations. Despite the apparent restriction of the models of interest, they cover a broad part of exponential families, since there are plenty of functions that are submodular, e.g., graph cuts, entropy and others. It is well known that a probabilistic treatment is challenging for most models, however we were able to tackle some of the challenges at least approximately.

In this manuscript, we exploit perturb-and-MAP ideas for partition function approximation and thus efficient parameter learning. Moreover, the problem can be also interpreted as a structure learning task, where each estimated parameter or weight represents the importance of the corresponding term. We propose a way of approximating parameter estimation and inference for models where exact learning and inference is intractable in general case due to the partition function calculation complexity.

The first part of the thesis is dedicated to theoretical guarantees. Given the log-supermodular models, we take advantage of the efficient minimization property related to submodularity. Introducing and comparing two existing upper bounds of the partition function, we are able to demonstrate their relation by proving a theoretical result. We introduce an approach for missing data as a natural subroutine of probabilistic modelling. It appears that we can apply a stochastic technique over the proposed perturb-and-map approximation approach and still maintain convergence while make it faster in practice.

The second main contribution of this thesis is an efficient and scalable generalization of the parameter learning approach. In this section we develop new algorithms to perform parameter estimation for various loss functions, different levels of supervision and we also work on the scalability. In particular, working with mostly graph cuts, we were able to incorporate various acceleration techniques.

As a third contribution we deal with the general problem of learning continuous signals. In this part, we focus on the sparse graphical models representations. We consider common sparsity-inducing regularizers as negative log-densities for the prior distribution. The proposed denoising techniques do not require choosing any precise regularizer in advance. To perform sparse representation learning, the signal processing

community often uses symmetric penalties such as ℓ_1, but we propose to parameterize the penalty and learn the weight of each loss component from the data. This is feasible via an approach which is similar to what we proposed in the previous sections.

For all aspects of the parameter estimation mentioned above we performed computational experiments to illustrate the ideas or compare with existing benchmarks, and demonstrate its performance in practice.

Résumé

Les modèles graphiques probabilistes codent des dépendances cachées entre des variables aléatoires pour la modélisation des données. L'estimation des paramètres est une partie cruciale et nécessaire du traitement de ces modèles probabilistes. Ces modèles très généraux ont été utilisés dans de nombreux domaines tels que la vision par ordinateur, le traitement du signal, le traitement du langage naturel et bien d'autres. Nous nous sommes surtout concentrés sur les modèles log-supermodulaires, qui constituent une partie spécifique des distributions de la famille exponentielle, où la fonction potentielle est supposée être l'opposée d'une fonction sous-modulaire. Cette propriété sera très pratique pour l'estimation par maximum a posteriori et l'apprentissage des paramètres. Malgré la restriction apparente des modèles d'intérêt, ils couvrent une grande partie des familles exponentielles, puisqu'il y a beaucoup de fonctions qui sont sous-modulaires, par exemple, les coupes dans de graphes, l'entropie et d'autres. Il est bien connu que le traitement probabiliste est un défi pour la plupart des modèles, mais nous avons été en mesure de relever certains de ces défis au moins approximativement.

Dans ce manuscrit, nous exploitons les idées "perturb-and-MAP" pour l'approximation de la fonction de partition et donc un apprentissage efficace des paramètres. De plus, le problème peut être considéré comme une tâche d'apprentissage structuré, où chaque paramètre ou poids estimé représente l'importance du terme correspondant. Nous proposons une méthode d'estimation et d'inférence approchée des paramètres pour les modèles où l'apprentissage et l'inférence exacts sont insolubles dans le cas général en raison de la complexité du calcul des fonctions de partition.

La première partie de la thèse est consacrée aux garanties théoriques. Étant donnés les modèles log-supermodulaires, nous tirons parti de la propriété de minimisation efficace liée à la sous-modularité. En introduisant et en comparant deux bornes supérieures existantes de la fonction de partition, nous sommes en mesure de démontrer leur relation en prouvant un résultat théorique. Nous introduisons une approche pour les données manquantes comme sous-routine naturelle de la modélisation probabiliste. Il semble que nous puissions appliquer une technique stochastique à l'approche d'approximation par perturbation tout en maintenant la convergence et la rendant plus rapide en pratique.

La deuxième contribution principale de cette thèse est une généralisation efficace de l'approche de l'apprentissage paramétrique. Dans cette section, nous développons de nouveaux algorithmes pour effectuer l'estimation des paramètres pour diverses fonctions de perte, différents niveaux de supervision et travaillons sur l'efficacité

à grande échelle. En particulier, en travaillant principalement avec des coupes de graphes, nous avons pu intégrer différentes techniques d'accélération.

Comme troisième contribution, nous traitons d'un problème général d'apprentissage des signaux continus. Dans cette partie, nous nous concentrons sur les représentations de modèles graphiques parcimonieux. Nous traitons des penalités classiques en estimation parcimonieuse en les considérant comme l'opposée de la log-vraiseullante de la distribution a priori. Les techniques de débruitage proposées ne nécessitent pas de paramêtre de régularisation précis à l'avance. Pour effectuer l'apprentissage de la représentation parcimonieuse la communauté utilise souvent des pénalités symétriques comme la meme ℓ_1, mais nous proposons de paramétrer la perte et d'apprendre le poids de chaque composante de la pénalité à partir des données. C'est faisable avec l'approche sur laquelle nous avons travaillé auparavant.

Pour tous les aspects de l'estimation des paramètres mentionnés ci-dessus, nous avons effectué des expériences pour illustrer l'idée ou la comparer aux algorithmes existants, et démontrer sa performance en pratique.

Acknowledgements

I am grateful that I had a chance to work on my PhD thesis under Francis Bach supervision during these three and a half years. Francis gave me the best thing he could, the enthusiasm and belief in academical research. I am thankful that you introduce the exciting top-tier scientific world to me and I hope I was a good student to accept and succeed. Thank you for making my dream to become a doctor come true and encouraging me to continue my academic career. I will always be grateful for that.

I also acknowledge support the European Union's H2020 Framework Programme (H2020-MSCA-ITN-2014) under grant agreement n°642685 MacSeNet.

I would like to acknowledge Florence d'Alché-Buc and Matthew Blaschko for reviewing this thesis. I am also thankful to the jury members Ivan Laptev and Umut Şimşekli who agreeing to participate. I deeply thank you all for the correction and remarks that I have got after your careful reading.

I would like also to mention Mike Davies here, who was not my co-advisor, but I often felt that he was. He teached me a different side of view and showed himself in the best qualities during our collaborative time. From Mike I have also learned to be sensitive to the details which is always important.

I am pleased that Anton Osokin and Dmitry Ostrovsky were always there and thank them for collaborations and interesting discussions related to this thesis. I would like to express my deepest gratitude for their continuous help as well.

I would like to highlight that SIERRA team is flourishing space for personal grow and idea sharing. The time I spent in SIERRA/WILLOW environment was the most productive in my life. The concentration of motivation and smartness was the highest ever and I am proud that I was a part of it. I am thankful for everyone who accompanied me on my path to the graduation during all these years.

I would especially like to thank my partner and college from SIERRA Dmitry Babichev for all his unlimited love and support, and thanks for pushing me forward sometimes. This PhD journey would be half less joyful and half harder without you by my side. I am thankful for my family as well, who always supports me, though probably do not have an idea what is this manuscript about.

There are many more people who had influenced me to become a researcher and conduct a PhD. I hope all of you feel my gratitude.

Contents

1 Introduction **3**
1.1 Probabilistic Graphical Models . . . 3
1.1.1 Markov Random Fields . . . 3
1.1.2 Conditional Random Fields . . . 4
1.1.3 Conditional Independence . . . 4
1.1.4 Model Parameterization . . . 5
1.1.5 Examples . . . 7
1.1.6 Probabilistic Inference . . . 8
1.2 Submodular Functions and Log-supermodular Models . . . 10
1.2.1 Submodularity . . . 10
1.2.2 Submodular Function Minimization . . . 11
1.2.3 Examples of Submodular Functions . . . 12
1.2.4 Log-supermodular Distributions . . . 13
1.2.5 Examples of Log-supermodular Distributions . . . 13
1.3 Parameter Learning and Inference . . . 13
1.3.1 Maximum Likelihood Estimation . . . 14
1.3.2 Optimization . . . 14
1.3.3 Missing Data Treatment . . . 14
1.3.4 Conditional Maximum Likelihood . . . 15
1.4 Partition Function Approximation . . . 15
1.4.1 L-Field bound . . . 15
1.4.2 Gumbel bound . . . 17

2 Parameter Learning for Log-supermodular Distributions **21**
2.1 Introduction . . . 21
2.2 Contributions . . . 22
2.3 Submodular functions and log-supermodular models . . . 22
2.3.1 Submodular functions . . . 23
2.3.2 Log-supermodular distributions . . . 23
2.3.3 Examples . . . 24
2.4 Upper-bounds on the log-partition function . . . 24
2.4.1 Base polytope relaxation with L-Field (Djolonga and Krause [2014]) . . . 24
2.4.2 “Pertub-and-MAP” with logistic distributions . . . 25
2.4.3 Comparison of bounds . . . 26

2.4.4 From bounds to approximate inference 27
2.5 Parameter learning through maximum likelihood 28
2.5.1 Learning with the L-field approximation 29
2.5.2 Learning with the logistic approximation with stochastic gradients 30
2.5.3 Extension to conditional maximum likelihood 31
2.5.4 Missing data through maximum likelihood 31
2.6 Experiments 32
2.7 Conclusion 34

3 Marginal Weighted Maximum Log-likelihood for Efficient Learning of Perturb-and-Map Models 35
3.1 Introduction 35
3.2 Contributions 37
3.3 Perturb-and-MAP 37
3.3.1 Gumbel perturbations 38
3.3.2 Parameter learning and Inference 39
3.3.3 Marginal probability estimation 41
3.4 Marginal Likelihood 41
3.4.1 Hamming loss 42
3.4.2 Weighted Hamming loss 44
3.4.3 Scalable algorithms for graph cuts 44
3.5 Parameter Learning in the Semisupervised Setup 45
3.6 Experiments 47
3.6.1 OCR dataset 47
3.6.2 HorseSeg dataset 48
3.6.3 Experiments analysis 51
3.7 Conclusion 52

4 Hyper-parameter Learning for Sparse Structured Probabilistic Models 53
4.1 Introduction 53
4.2 Contributions 54
4.3 Log-supermodular Distributions 54
4.3.1 Supermodular and Submodular Functions 55
4.3.2 Discrete Log-supermodular Distributions 56
4.3.3 Continuous Log-supermodular Distributions 56
4.3.4 Log-partition Function for Bayesian Learning 56
4.4 Perturb-and-MAP 57
4.4.1 Extension to the Continuous Case 58
4.4.2 Decoding with MMSE 59
4.5 Experiments 59
4.5.1 Synthetic Data. Experiments on decoding 59
4.5.2 Flow-based priors 60
4.5.3 Real Data. Experiments on the parameter learning and decoding. 60

4.6 Conclusion 62

5 Conclusion and Future Work **63**
5.1 Summary of the thesis 63
5.2 Perspectives 64

Contributions and thesis outline

Parameter estimation is a big part of probabilistic graphical model handling. Given the optimal parameter one can perform inference and evaluate marginal and conditional probabilities. Optimal parameters are often considered as a solution of the maximum likelihood problem, which is log-concave for exponential family distributions. However, the partition function term is hard to compute, and thus causes an unsolvable problem for the maximum likelihood learning process in the general case, and thereby we propose to use an effective and tight upper bound of the partition function based on the Gumbel perturbations and apply it for the broad but specific log-supermodular family of distributions.

Thus, this thesis brings new light on parameter learning for challenging probabilistic graphical models. We mostly provide some guarantees on the approach and develop new directions and possible applications in the rest of the thesis. Below we summarize each of the chapter contribution.

Chapter 1: We introduce some basic mathematical concepts that will be essential for this work. We provide an overview of probabilistic graphical models and its properties, submodularity and based on it log-supermodular distributions, and we also cover challenges and suitability of parameter learning for these models of interest.

Chapter 2: As the first contribution of this thesis, we present a new automatic parameter estimation procedure. Based on the idea of efficient maximum likelihood optimization, we propose to incorporate a suitable upper bound on the partition function, and thus make the parameter estimation problem feasible. The only assumption required is the existence of a maximum a posteriori solver. The sufficient condition for this is to consider the log-supermodular models, for which we can show the superiority over existing approaches.

Chapter 3: The next contribution we present is a generalization in terms of scalability, levels of supervision and various performance evaluation functions. To be specific, our intention was to make our approach more practical: we worked on the real losses like Hamming and weighted Hamming losses, we worked on a large-scale image segmentation problem with unlabeled data or only bounding-box given data, as these both cases are more common than the fully-labeled ones.

Chapter 4: This chapter considers the problem of a sparsity-inducing density learning supported on the continuous domain. Basically, we propose to consider the learned density function as a prior that encourages signal sparsity and thereby incorporate the derived prior into decoding, e.g., denoising problem.

Chapter 5: In this chapter we conclude the thesis, list the main contribution

and sum up the work that was done. We also discuss the possible future directions.

We note that the content of this thesis is based on the publications/submissions we list below:

(a) Chapter 2 is based on the work "Parameter Learning for Log-supermodular Distributions", T. Shpakova, F. Bach, published in Advances of the Conference on Neural Information Processing Systems (NIPS), 2016.

(b) Chapter 3 is based on the work "Marginal Weighted Maximum Log-likelihood for Efficient Learning of Perturb-and-Map Models", T. Shpakova, F. Bach, A. Osokin, published in Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), 2018.

(c) Chapter 4 is based on the work "Hyper-parameter Learning for Sparse Structured Probabilistic Models", T. Shpakova, F. Bach, M. Davies, which is under review for the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2019.

Chapter 1

Introduction

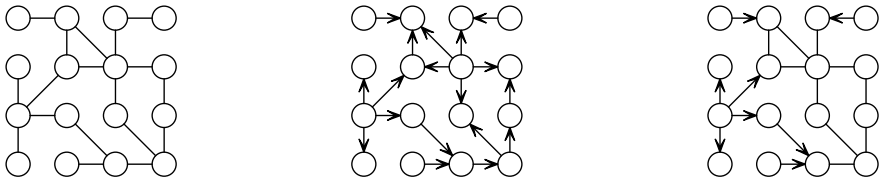
In this chapter we introduce in details the topic of probabilistic graphical models and the associated subroutines of the parameter learning problem. We will outline log-supermodular models which are crucial for this thesis, and discuss submodularity along with its properties.

1.1 Probabilistic Graphical Models

Probabilistic graphical models are a versatile tool proposed for modeling structured data and capturing the dependencies between variables via concepts from graph theory. Random variables of interest $(X_1, \ldots, X_n) = X$, where X is distributed according to a distribution $p(x)$, are dependent and connected in a predefined way. The corresponding graphical model can be described by a notation of a graph $G = (V, E)$ and encode the random variables via the node set V and the pairwise dependencies between them via the edge set E. A probabilistic graphical model takes advantage of probability theory to make a consistent probabilistic interpretation given the values of random variables (Wainwright and Jordan [2008b]). The goal would be to manipulate the model and evaluate various marginal and conditional probabilities. There are mainly two broad families: undirected and directed graphical models. We will briefly cover both, however undirected graphical models (they are also called Markov Random Fields) are more in demand for our needs in this work. In general there are all four types of distributions: that can be described by both UGM and DGM, or by only one of them, or by neither of them. Thus, none of two GM is more powerful or general than the other (see Murphy [2013] for more details). There are also models based on the chain graphs, graphs that include both ideas of undirected and directed graphical model. We present several examples in Figure 1-1.

1.1.1 Markov Random Fields

Undirected graphical models assume a representation where the local dependencies between the nodes are symmetric. This assumption makes sense for some domains, for example for images or other symmetric structured data as relational or spatial



(a) Example of UGM. (b) Example of DGM. (c) Example of chain model.

Figure 1-1: Undirected and directed graphical models on the same set of nodes V.

data. Another common way to model data is to use directed graphical models which are considered conditional dependencies and based on directed graphs. It is worth noting, that for DGMs is natural to decompose the density function using the chain rule, where each factor is a conditional probability, however is not natural for UGMs anymore. In this case we have to additionally normalize the model via the crafted normalizer term which is called the partition function. Thus, it becomes much harder to manipulate the model.

1.1.2 Conditional Random Fields

The general concept of Conditional Random Fields (CRFs, see Murphy [2013]) is that instead of working with joint probability distribution, we model the a-posteriori distribution, conditioned on the feature data. To make it more concrete, let us introduce the observed variables $X = (X_1, \ldots, X_n)$ which is called the evidence and the corresponding unobserved variables $Y = (Y_1, \ldots, Y_n)$ which are labels. Thus, using a discriminative model $p(y|x)$ instead of a generative model $p(x, y)$, we can consider the labels-to-predict behaviour only. This problem statement gives us more flexibility and allows us to focus on the target variables only. See Figure 1-2 for the example of a CRF, where the shaded nodes are meant to be the observed variables.

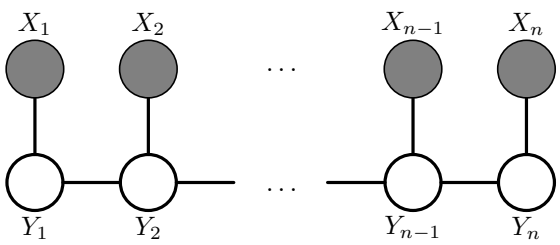


Figure 1-2: Example of a Linear-chain CRF.

1.1.3 Conditional Independence

By now the general idea was defined intuitively. Here we are going to discuss in more details the UGM's parameterization. As the chain rule can not be applied in

the undirected case, there is a way to formulate parametrization formally with the help of the Hammersley-Clifford theorem (Lauritzen [1996]).

Before formulating this, it is necessary to introduce the concept of conditional independence. Conditional independence's goal is to state whether two sets of variables are independent or not conditioned on a third set of variables. This can be decided by means of factoring the distribution into pieces or *factors*. We have already said that we assume some local dependencies between variables, as, e.g., in Figure 1-1, but now we would like to define it clearly.

Definition 1 (Markov property)**.** *Two subsets of variables X_A and X_B are* ***conditionally independent*** *given the third subset of variables X_C or equivalently $X_A \perp\!\!\!\perp X_B | X_C$, if every path from a node in X_A to a node in X_B in the graph G intersects the subset X_C at least in one node.*

We illustrate this definition with the following example in Figure 1-3. It is worth noting, that two nodes which are not connected with an edge, are conditionally independent given the rest of the graph.

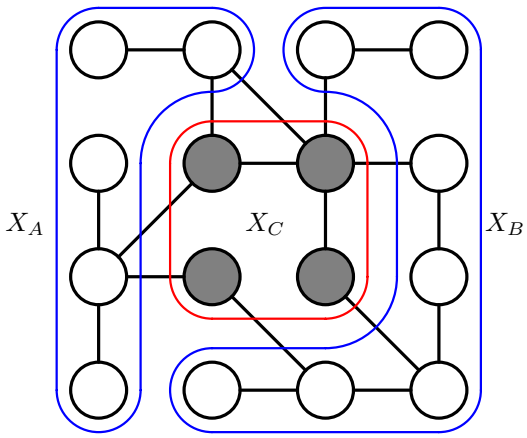


Figure 1-3: Conditionally independent subsets X_A and X_B given the subset X_C.

The Markov property makes concrete the intuitive statement that variables interact locally on the graph.

1.1.4 Model Parameterization

There is a common way for the undirected graphical model parametrization that we present next. We associate a potential function to each region or maximal *clique* in the graph and can formalize the probability mass function $p(x_1, \ldots, x_n) = p(x)$, where x_i is a realization of a discrete variable X_i, by means of the Hammersley-Clifford theorem (Hammersley and Clifford [1971]).

Theorem 1.1 (**Hammersley-Clifford**)**.** *Any positive distribution $p(x) > 0$ that satisfies the Markov property of an undirected graph $G = (V, E)$ can be expressed via*

the product of factors, where one factor depends on one maximal clique of the graph:

$$p(x) = \frac{1}{Z} \prod_{c \in C} \phi_c(x_c),$$

where C is a set of all maximal cliques of the graph G and $Z = \sum_{x \in \mathcal{X}} \prod_{c \in C} \phi_c(x_c)$ is the normalization constant.

Any non-negative function can serve as a potential function ϕ_c. We are going to represent the probability density function via the Gibbs distribution using log-potentials $f(x) = \sum_{c \in C} f_c(x_c) = \sum_{c \in C} \log \phi_c(x_c)$. Thus, for a MRF it has the following form:

$$p(x) = \frac{e^{f(x)}}{Z} = \frac{e^{f(x)}}{\sum_{x \in \mathcal{X}} e^{f(x)}}.$$

And a CRF model can be written as:

$$p(y|x) = \frac{e^{f(y|x)}}{Z(x)} = \frac{e^{f(y|x)}}{\sum_{y \in \mathcal{Y}} e^{f(y|x)}}.$$

While the interpretation with maximal cliques is classic but rather heavy, in practice pairwise models are more popular, where all cliques are of size one or two nodes, and thus the probability density function is written as:

$$p(x) \propto \prod_{i,j \in E} \phi_{i,j}(x_i, x_j) \prod_{i \in V} \phi_i(x_i).$$

This form is used due to its simplicity and strong coverage of various models. Another common treatment is to represent the log potentials as a linear function of some parameters θ: $f(x|\theta) = \psi(x)^T \theta$, where $\psi(x)$ is a feature vector of the data vector x. In many applications these features are created by hand, depending on the domain knowledge and user's needs. We will cover this part in the experimental settings. Thus, our final model has the form:

$$\log p(x|\theta) = \psi(x)^T \theta - \log Z(\theta),$$

which is called the log-linear model and the potential vector $\psi(x)$ contains only pairwise and unary terms.

The distribution of interest $p(x|\theta)$ clearly belongs to the exponential family as it can be rewritten as $p(x|\theta) = \exp[\psi(x)^T \theta - A(\theta)]$, where $\psi(x)$ is called a vector of sufficient statistics and $A(\theta) = \log Z(\theta)$ is a log-partition function. The distribution enjoys the exponential family's properties, most importantly it attains maximum entropy, and thus makes the weakest assumptions about the data (Murphy [2013]). Besides that, using the probability density function allows us to make use of the log-likelihood function concavity. For more details see Wainwright and Jordan [2008b], Murphy [2013], Bishop [2006].

1.1.5 Examples

In this part we will present some prominent examples of Markov Random Fields like Ising and Potts Models, and Gaussian MRFs.

Ising Model. This one is a pioneer and one of the classical graphical models created for the statistical physics needs (Ising [1925]). The idea is to model the system of directed magnetic spins, which are directed in one of the two possible directions: $x_i \in \{+1, -1\}$. This can be written down in the following way:

$$\log p(x|W, b) = \sum_{i \sim j} w_{ij} x_i x_j + \sum_i b_i x_i - \log Z(W, b) = \frac{1}{2} x^T W x + b^T x - \log Z(W, b).$$

A sign of the weight w_{ij} will influence the tendency of the two connected nodes to have the same or the opposite sign. It is worth noting, that the normalization constant $Z(W, b)$ is a sum of potential functions of all possible values of vector x, i.e., 2^D terms, and its calculation is $\#P$ hard in general case (Jerrum and Sinclair [1993]). A variation of this model will be covered in our experiments.

Potts Model. It is a straightforward generalization of the Ising Model for multiple number of discrete states: $x_i \in \{0, 1, \ldots, K\}$ (Potts and Domb [1952]). This model will also be covered in the experimental section of Chapter 4. Both Potts and Ising models can be applied for an image segmentation problem, since it is feasible to encourage an images smoothness using these types of models (Boykov and Kolmogorov [2004]). This usually is done via Conditional Random Fields (CRFs discussed before), where the features variables x_i are observed and the labels y_i are not. We present a grid model common for image segmentation problems in the Figure 1-4.

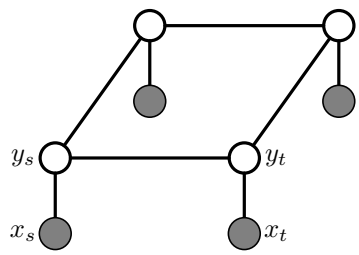


Figure 1-4: Grid-structured CRF for image segmentation task.

Gaussian MRFs. In the case of Gaussian MRFs, the joint probability density function is assumed to be Gaussian and has the following form:

$$p(x|\mu, \Sigma) \sim \mathcal{N}(\mu, \Sigma).$$

If $\Sigma^{-1}_{i,j}$ is zero, then there is no edge between nodes i and j and as follows no pairwise term of i and j (this follows from the conditional independence properties).

1.1.6 Probabilistic Inference

Given the probabilistic model we can perform inference with the tools provided by probabilistic theory. Using the axioms of probability, any marginal or conditional probability of interest can be defined, but not all calculations can be done in polynomial time. Let us discuss this problem of computing in more details as this knowledge will be crucial further in this thesis.

Let us consider the conditional probability of a subset X_A given the values for a subset X_B, while the rest random variables of the graphical model we denote as X_C. Our goal would be to evaluate the conditional probability $p(X_A|X_B)$, which is the classical problem of probabilistic inference.

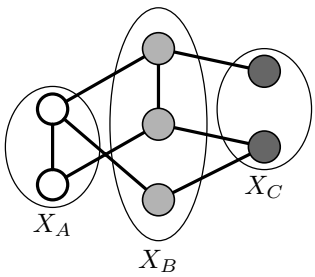


Figure 1-5: An example of the conditional probability calculation. The dark nodes correspond to the nodes on which we condition, light-shaded nodes are the nodes we marginalize over, and the white nodes are the nodes we want to calculate conditional probability for.

To make this easy to present, we will use Figure 1-5 with 3 shades of grey for the nodes coloring. The target variables X_A are unshaded, the observed variables X_B, which are also called the evidence nodes, are dark shaded, while light shaded are the rest nodes values X_C and they must be marginalized out to calculate $p(X_A|X_B)$. We derive the conditional probability in the following way:

$$p(X_A|X_B) = \frac{p(X_A, X_B)}{p(X_B)} = \frac{\sum\limits_{X_C} P(X_A, X_B, X_C)}{\sum\limits_{X_A, X_C} P(X_A, X_B, X_C)}.$$

Thus, the point of discussion in this section is an efficient calculation of this expression, which contains two expensive summations and can be of exponential complexity.

Exact Inference. There are plenty of algorithms for exact inference for the discrete graphical models. The most naive approach would be an elimination algorithm, that attains at a conditional probability of a single node (Wainwright and Jordan [2008b]), which is a quite specific problem to solve. However, a good point is that this approach is applicable for any graphical model. The most well-spread set of techniques are called message passing approaches. For example, the follow-up of the elimination algorithm is the belief propagation or sum-product message passing algorithm that can be applied for any acyclic graphical model, i.e., tree-based ones (Wainwright and

Jordan [2008b]), and it can compute all single-node marginals altogether. The next step of generalization would be the junction tree algorithm (Wainwright and Jordan [2008b]), which is mostly the combination of the previous two algorithms.

All of these algorithms are restricted either to a special graph structure, mostly to acyclic graphs or to the low-treewidth cases (the computation time is exponential in the treewidth of the graph, see more details in Wainwright and Jordan [2008b]). Moreover, these approaches do not work for continuous variables, except for jointly Gaussian distributions. It is explainable by the fact that exact inference is NP-hard in general (Wainwright and Jordan [2008b]). To perform general inference, one needs to call for approximated techniques covered in the next paragraph.

Approximate Inference. When exact inference is often intractable or too inefficient, an alternative would be approximate inference. It is interesting to note, that the sum-product message-passing algorithm from the previous paragraph can also be seen as an approximate inference technique applied to a cycled factor graph (Szeliski et al. [2008]). Here we aim to cover the most prominent of concepts like variational and Monte Carlo inference.

Variational Inference. The idea of variational inference is to approximate the intractable distribution of interest $p(x)$ with another close, but tractable distribution $q(x)$. Some examples of $q(x)$ could be from factored or Gaussian distributions. We usually evaluate the "closeness" via the KL divergence $\mathbb{KL}(q||p) = \sum_x q(x) \log \frac{q(x)}{p(x)}$ and, thus, can fit free parameters of $q(x)$ by minimizing the KL objective.

One of the most classic variational inference approaches is called "mean field" approximation, where the idea is to use a fully factorized posterior as $q(x) = \prod_{i=1}^{n} q(x_i)$ approximation, which is a product of marginals (Wainwright and Jordan [2008b]). Thus, the KL optimization problem is solvable by the coordinate descent method. The mean field approach belongs to the variational Bayes family of approaches, which aim to find the optimal parameters of the distribution $q(x)$ (Murphy [2013]). Moreover, there is also an extension: variational Bayes EM, where we aim to find both the parameters and the latent variables simultaneously (Murphy [2013]).

Another milestone is the loopy belief propagation technique, which is a generalization of the message passing (Murphy and Weiss [1999]). There are also more techniques that variational inference paradigm contains, however we will not cover it here (for more details see Wainwright and Jordan [2008b]). It is worth to mention, that variational inference is always approximate even if run to convergence.

Monte Carlo Inference. To break the bounds caused by variational inference, such as the special form of approximation or necessary properties as conjugate priors, we are going to introduce an alternative class of algorithms based on Monte Carlo approximation. The idea is to approximate any target value with averaged value obtained from sampling $\mathbb{E}[f|\mathcal{D}] \approx \frac{1}{N} \sum_{n=1}^{N} f(x^n)$.

The most prominent example for sampling in high dimensions is Markov Chain Monte Carlo (MCMC). This approach constructs a Markov chain on the state space $\mathcal{X}$, where the stationary distribution matches the target distribution $p(x)$. Thus, while performing a random walk on the states and converging to the stationary distribution, we are sampling from $p(x)$ and then integrate with respect to $p(x)$ (Robert and Casella [2013]).

A special case of MCMC is Gibbs sampling, which is very similar to mean field in the sense that we should compute each node's conditional and average out the neighbors. Random walk Monte Carlo methods include such approaches as Metropolis Hastings algorithm, slice sampling and others (for more details see Wainwright and Jordan [2008b], Doucet et al. [2001]).

Variational inference advantages would be its deterministic solution and a bound on loglikelihood it gives, but MCMC can be applied for broader types of distributions and usually is easier to implement, and, given enough time will converge to exact inference.

1.2 Submodular Functions and Log-supermodular Models

Moving away from the general theory of graphical models, in this section we would like to present a brand-new family of distributions, which is called family of log-supermodular distributions. They are practically based on the crucial notion of submodularity, and here we present its definition and properties in details.

1.2.1 Submodularity

We first start to consider a set function defined on the ground set V in the form

$$F(S) : 2^V \to \mathbb{R},$$

where $S \subseteq V$ and 2^V is a set of all possible subsets of the ground set V. Optimization of this function is a crucial problem of discrete optimization. The solution could represent the optimal set of pixels for the foreground/background image segmentation problem, or optimal locations for the fire alarms. Without loss of generality we always assume that $F(\emptyset) = 0$.

To make it looks more like a discrete optimization problem, we can biject each subset $S \subseteq V$ into a binary vector $x \in \{0,1\}^{|V|}$, where $x_i = 1$ means that the i-th element belongs to the subset S and $|V|$ is the ground set cardinality. In this interpretation the target function $f(x)$ has its values on the nodes of the hypercube $\{0,1\}^{|V|}$ (see example in Figure 1-6).

We first introduce submodular functions, which are set functions with strong properties, kind of analogous to convex functions in the continuous space. They are both very common in practice and both are feasible to be minimized, as we can minimize submodular function in polynomial time (Fujishige [2005], Bach [2016]).

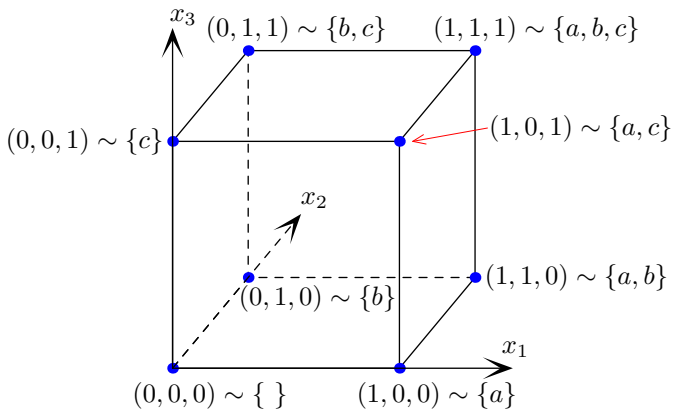


Figure 1-6: Hypercube representation for the ground set $V = \{a, b, c\}$.

Definition 2. *The set function $F(S) : 2^V \to \mathbb{R}$ is called **submodular**, if for any two subset $A, B \subseteq V$ and an element $s \in V$, such that $A \subseteq B, s \notin B$, the diminishing marginal returns property holds:*

$$F(A \cup \{s\}) - F(A) \geq F(B \cup \{s\}) - F(B),$$

*or equivalently $F(x) : \{0,1\}^{|V|} \to \mathbb{R}$ is **submodular**, if for any two vertices of hypercube $x, y \in \{0,1\}^{|V|}$ and a canonical basis vector $e_i \in \{0,1\}^{|V|}$, such that $x \leq y$ component-wise and $y_i = 0$, the following is true:*

$$F(x + e_i) - F(x) \geq F(y + e_i) - F(y).$$

We should also introduce a concept of a **supermodular** function, which is simply the negative of a submodular one. Another concept convenient for our purposes is the base polyhedron, which is a polyhedron that is associated to a submodular function.

Definition 3. *A base polyhedron $B(F)$ associated with a submodular function $F(S)$ is defined in the following form:*

$$B(F) = \{s \in \mathbb{R}^{|V|}, s(V) = F(V), \forall A \subseteq V, s(A) \leq F(A)\},$$

where $s(A) = \sum_{i \in A} s_i$. Thus this polyhedron is defined as the intersection of hyperplanes $\{s \in \mathbb{R}^{|V|}, s^T x_A = s(A) \leq F(A)\}$, where x_A are normals of the corresponding hyperplanes.

This polyhedron will play a crucial role for analysis and optimization, such as probabilistic inference in the log-supermodular models.

1.2.2 Submodular Function Minimization

As already been mentioned, there is a polynomial time algorithm for submodular function minimization, while maximization can be tackled only approximately. To

substantiate the statement, we introduce the **Lovász extension**, which is a convex extension that can be defined for any submodular function (Lovász [1983]).

Definition 4. *For any submodular function $F(x) : \{0,1\}^{|V|} \to \mathbb{R}$ we can define a* ***Lovász extension*** *$f(w) : \mathbb{R}^{|V|} \to \mathbb{R}$, which is a convex function and matches the initial function $F(x)$ at all vertices of the hyper-cube. It has the following form:*

$$f(w) = \sum_{v=1}^{|V|} w_{i_v}[F(\{i_1, \dots, i_v\}) - F(\{i_1, \dots, i_{v-1}\})],$$

where $(i_1, \dots, i_{|V|})$ is the following permutation of components of w : $w_{i_1} \geq w_{i_1} \geq \dots \geq w_{i_{|V|}}$.

It is worth noting, that the Lovász extension $f(w)$ is convex if and only if the initial function $F(x)$ is submodular (Lovász [1983]). Moreover, the solution of the Lovász extension minimization is binary (i.e., can be restricted to belong to $\{0,1\}^{|V|}$) and thus appears to be the minimizer for the submodular function $F(x)$ as well. This statement gives us a polynomial time guarantee. To sum up, we can link the submodular and convex analysis and profit from some of the convexity guarantees. This is used in machine learning both for regularizers (Bach [2010]) or losses (Yu and Blaschko [2015]).

Another interesting connection to mention is that the Lovász function is a support function of the base polyhedron $B(F)$ itself, and, thus, may be computed in the closed form $f(w) = \min_{s \in B(F)} w^T s$.

1.2.3 Examples of Submodular Functions

The simplest example of a submodular function would be the cardinality function. Having a look at Definition 2, we notice the inequality always happens to be equality. It is interesting to notice, that this function is supermodular as well. The functions which are both submodular and supermodular we will call **modular function** and it is known that they can be respesented as $F(x) = w^T x$, where $w \in \mathbb{R}^{|V|}$ is a vector of weights and x is a binary representation of a set.

Another crucial example is a cut function, which is defined for the undirected graph $G = (V, E)$ that builds on the ground set of nodes V. Basically the cut function $F(S)$ counts the number of edges connecting a node from S and a node from $V \backslash S$, i.e.

$$F(S) = F(x^S) = \sum_{i \sim j} |x_i^S - x_j^S|,$$

where x^S is a binary vector representation of the subset S, and x_i^S and x_j^S are representations of the nodes i and j correspondingly. This function is easily generalized for the weighted case.

Other prominent examples of the submodular functions are network flows, concave functions of a set cardinality, mutual entropy and matroids (see more details in Fujishige [2005], Bach [2016]).

1.2.4 Log-supermodular Distributions

Now we marry the two big parts introduced before: probabilistic models and submodular functions. Let us consider a probabilistic model defined on the binary space $\mathcal{X} = \{0, 1\}^K$ in the form of a Gibbs distribution:

$$p(x) = \frac{\exp(-f(x))}{Z(f)},$$

where now we assume that the potential $f(x)$ is a submodular function. Hence its negative $-f(x)$ is a supermodular one and, thus, the model is called log-supermodular probabilistic model or **log-supermodular distribution**. It was recently introduced by Djolonga and Krause [2014] and will be investigated throughout this thesis.

This common form defines a broad family of distributions on the one side, and allow us to perform efficient parameter learning of the other side. Maximum a posteriori estimation (MAP) for these models is known to be feasible (reduced to the submodular minimization problem), however the partition function calculation is not (Kolmogorov [2006]).

1.2.5 Examples of Log-supermodular Distributions

One of the examples to discuss is the binary pairwise MRF, which is also known as the Ising model and can be defined via the log-supermodular distribution, if the potentials satisfy the property of submodularity (so called attractive potentials that encourage the same signal sign among pairwise neighbors). Binary models can be naturally applied for computer vision purposes, e.g., for an image segmentation problem.

It is also possible to extend these models in two ways: to consider not only binary values, but discrete and even continuous (part of these work is focused on this), and also consider higher-order potentials instead of pairwise ones. The submodular machinery allow us to perform these extensions naturally. Thus, such models can be incorporated for example as sparsity-inducing priors or applied for semantic segmentation (see Chapter 4).

1.3 Parameter Learning and Inference

In this section we discuss a learning procedure for probabilistic graphical models and for the log-supermodular models in particular. There are plenty of approaches for parameter learning. We will cover the most prominent ones and will see that the learning procedure is usually computationally expensive.

1.3.1 Maximum Likelihood Estimation

We again consider a probabilistic model written as

$$p(x|\theta) = \frac{1}{Z(\theta)} \exp(\theta^T \psi(x)).$$

Given the datapoints $x_1, \dots x_N$, the averaged log-likelihood function $\ell(\theta)$ has the form:

$$\ell(\theta) = \frac{1}{N} \sum_{n=1}^{N} \log p(x_n|\theta) = \theta^T \Big(\frac{1}{N} \sum_{n=1}^{N} \psi(x_n) \Big) - \log Z(\theta) = \theta^T \hat{\psi}(x) - \log Z(\theta),$$

where $\hat{\psi}(x)$ is an empirical value of the expectation $\mathbb{E}\psi(x)$.

1.3.2 Optimization

Since we work with exponential families only, the given log-likelihood is concave and thereby has one global optimum that can be achieved by an approach from gradient ascent family. A gradient step would be possible since we evaluate the log-likelihood derivative:

$$\frac{\partial \ell(\theta)}{\partial \theta} = \hat{\psi}(x) - \frac{\partial \log Z(\theta)}{\partial \theta} = \hat{\psi}(x) - \mathbb{E}\psi(x|\theta),$$

where we use the exponential family property of the log partition function derivative.

We can notice that to make a gradient step, we have to perform an inference and evaluate the expectation $\mathbb{E}\psi(x|\theta)$, which is usually computational expensive and can be done in several ways we discussed in the probabilistic inference section. There is also a moment matching effect, that we can observe while converging to the optimum. For the optimal parameter θ the expectation $\mathbb{E}\psi(x|\theta)$ is going to match the empirical expectation $\hat{\psi}(x)$.

1.3.3 Missing Data Treatment

With the tools given by the probabilistic approach we can also tackle the missing data case:

$$p(x, z|\theta) = \frac{1}{Z(\theta)} \exp(\theta^T \psi(x, z)),$$

where we denote x as observed variables and z as unobserved or hidden. In the given circumstances, we will fit the optimal parameter θ to the observed data $x_1, \dots, x_N$ by solving an optimization problem in the form:

$$\max_{\theta \in \Theta} \ell(\theta) = \max_{\theta \in \Theta} \frac{1}{N} \sum_{n=1}^{N} \log \sum_{z \in \mathcal{Z}} p(x_i, z|\theta) = \max_{\theta \in \Theta} \frac{1}{N} \sum_{n=1}^{N} \log \sum_{z \in \mathcal{Z}} \theta^T \psi(x_i, z) - \log Z(\theta).$$

We should notice, that the summation term (where we marginalize out the hidden

values z) is of similar complexity as the partition function $Z(\theta)$ and thus provides another challenge to this optimization problem.

1.3.4 Conditional Maximum Likelihood

In this section we discuss the maximum likelihood formulation for the conditional random field model:

$$p(y|x,\theta) = \frac{1}{Z(x,\theta)} \exp(\theta^T \psi(x,y)),$$

where $\psi(x,y)$ is a potential vector of feature vector x and the label vector is y. This is the model of our particular interest in this thesis and we are going to stick with it in most of the times.

The new log-likelihood optimization problem can be written in the following form:

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \log p(y_n|x_n,\theta) - \lambda ||\theta||_2^2,$$

where the last term is a regularization term and can be seen as a log prior $p(\theta)$ on the parameters θ. In this common formulation the Gaussian prior was used, however there are plenty of priors that can be incorporated, such as Laplacian, mixed priors or even parameterized customized prior (see Chapter 4).

1.4 Partition Function Approximation

As we seen in the previous section, the partition function term restricts the tractability of the parameter estimation via maximum likelihood. In this section we review various partition function calculation techniques. When the challenge of exact partition function computation becomes impossible, we resort to approximation techniques. Some of the approximations are upper bounds.

1.4.1 L-Field bound

This approximation was designed by Djolonga and Krause [2014] directly for the log-supermodular distributions:

$$p(A) = \frac{\exp(-F(A))}{Z(-F)},$$

where $F(A)$ is a submodular function. The approach is named L-field as there is a natural relation to mean field method discussed earlier and it is based on the Lovász extension (explains the L in the approach's name).

The main idea is to benefit from the submodular function property, that it can be bounded from the both sides by modular functions $l, u : 2^V \to \mathbb{R}$.

Theorem 1.2. *If we consider bounds $l(A)$ and $u(A)$ of the submodular function $F(A)$: $l(A) \leq F(A) \leq u(A)$ holds for any $A \subseteq V$, then the following is correct:*

$$\log Z(-u) \leq \log Z(-F) \leq \log Z(-l), \tag{1.4.1}$$

where $\log Z(-F) = \log \sum_{A \in V} \exp(-F(A))$ is the log-partition function of the target distribution, and a log-partition function of any modular function $s(A) = s(x) = s^T x$ has the form :

$$\log Z(s) = \log \sum_{x \in \{0,1\}^{|V|}} \exp(-s^T x) = \log \sum_{x \in \{0,1\}^{|V|}} \prod_{i=1}^{|V|} \exp(-s_i x_i) =$$

$$= \log \prod_{i=1}^{|V|} \sum_{x_i \in \{0,1\}} \exp(-s_i x_i) = \sum_{i=1}^{|V|} \log(1 + e^{s_i}).$$

Equation (1.4.1) becomes obvious if we imagine the summation and log term before the bound inequalities $l(A) \leq F(A) \leq u(A)$. Via Theorem 1.2 we can bound the partition function from the both sides using some decomposable distributions based on modular functions. But there is no guarantee on the bounds tightness. Our motivation here would be to find the tightest possible modular function, to make our approximation the most accurate. Thus, we got a variational approach, that can be used to approximate the partition function from above:

$$\log Z(-F) \leq \min_l \log Z(-l) = \min_l \sum_{i=1}^{|V|} \log(1 + e^{-l_i}) \text{ subject to } l(A) \leq F(A).$$

Theorem 1.3. *The optimization problem $\min_l \sum_{i=1}^{|V|} \log(1 + e^{-l_i})$ subject to $l(A) \leq F(A)$ is equivalent to the following one:*

$$\min_{l \in B(F)} \sum_{i=1}^{|V|} \log(1 + e^{-l_i}),$$

where $B(F)$ is a base polytope of the submodular function $F(A)$.

This optimization problem is an existed problem of convex function optimization over the polytope, and can be solved via Frank-Wolfe algorithm (Lacoste-Julien et al. [2013]). Finally, we can write down the L-Field upper bound of the partition function in the following way

$$A_{\text{L-field}} = \min_{l \in B(F)} \sum_{i=1}^{|V|} \log(1 + e^{-l_i}) \geq \log Z(-F).$$

1.4.2 Gumbel bound

In this section we work with a general **Gibbs distribution** supported on discrete (not binary) set $\mathcal{X} = X_1 \times \cdots \times X_n$ in the form:

$$p(x) = \frac{\exp(f(x))}{Z(f)}.$$

The Gumbel approximation assumes that MAP (maximum a posteriori) inference problem is feasible, i.e., there is a solver for the following problem:

$$\arg\max_{x \in \mathcal{X}} p(x) = \arg\max_{x \in \mathcal{X}} f(x),$$

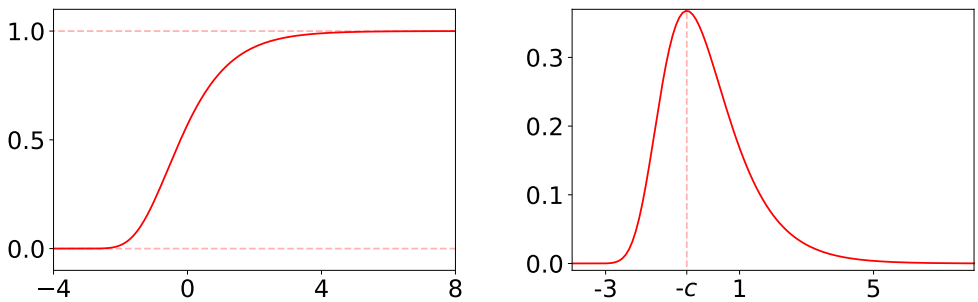
as the partition function is a constant and does not influence the solution. Thus, this makes the MAP inference problem much easier than the partition function calculation. While the partition function evaluation is $\#P$-hard problem (Jerrum and Sinclair [1993]), MAP is NP-hard in general, however it can be solved efficiently for many practical cases, e.g., for log-supermodular models.

Let us introduce the notation of *Gumbel distribution* as the approach we are going to present is based on it.

Definition 5. *We will call a distribution a* ***Gumbel distribution****, if it has the following cumulative function*

$$F(t) = e^{-e^{-(t+c)}},$$

where $c \approx 0.577$ is the Euler constant. The Gumbel distribution has a zero mean and a mode at the negative Euler constant. Its cumulative distribution and probability density functions are presented in Figure 1-7.



(a) Gumbel distribution cdf. (b) Gumbel distribution pdf.

Figure 1-7: Gumbel distribution cdf and pdf.

The idea of the approximation is to use Gumbel-distributed perturbations to substitute the partition function calculation with a bunch of related MAP inference problems (Hazan and Jaakkola [2012]).

Theorem 1.4. *Let $\{\gamma(x)\}_{x\in\mathcal{X}}$ be a collection of i.i.d. random variables from the Gumbel distribution. Then the random variable $\max_{x\in\mathcal{X}}\{f(x)+\gamma(x)\}$ belongs to the shifted Gumbel distribution, i.e., represents as Gumbel variable plus a constant, and the partition function can be represented as:*

$$\log Z = \mathbb{E}_\gamma[\max_{x\in\mathcal{X}}\{f(x)+\gamma(x)\}].$$

Proof. By definition we know that

$$F(t) = P(\gamma(y) \leq t).$$

Thus, we have, by independence,

$$\prod_{x\in\mathcal{X}} F(t-f(x)) = P(\max_{x\in\mathcal{X}}\{f(x)+\gamma(x)\} \leq t).$$

Since the shifted Gumbel cumulative distribution function is closed under multiplication, the following term is a cumulative distribution function as well:

$$\prod_{x\in\mathcal{X}} F(t-f(x)) = \exp\left(-\sum_{x\in\mathcal{X}}\exp(-(t-f(x)+c))\right) = \exp(-\exp(t+c)Z) = F(t-\log Z).$$

Expected value of the random variable $\max_{x\in\mathcal{X}}\{f(x)+\gamma(x)\}$ consequently equals to $\log Z$. In this way

$$\log Z = \mathbb{E}_\gamma[\max_{x\in\mathcal{X}}\{f(x)+\gamma(x)\}].$$

□

However, this theorem just proposes another appearance of the partition function term. This is still a $\#P$ hard problem, since we have exponential many Gumbel perturbations $\{\gamma(x)\}_{x\in\mathcal{X}}$. The next step would be to propose an upper bound with lower-dimensional perturbations.

Theorem 1.5. *Using the setup from the Theorem 1.4, we can rewrite and bound the partition function in the following way:*

$$\log Z = \mathbb{E}_{\gamma_1}\max_{x_1}\ldots\mathbb{E}_{\gamma_n}\max_{x_n}\{f(x)+\sum_{i=1}^n\gamma_i(x_i)\} \leq \mathbb{E}_\gamma\max_{x\in\mathcal{X}}\{f(x)+\sum_{i=1}^n\gamma_i(x_i)\}$$

See Hazan and Jaakkola [2012] for the detailed proof. The final upper bound and approximation has the following form:

$$A_{\text{Gumbel}} = \mathbb{E}_\gamma\max_{x\in\mathcal{X}}\{f(x)+\sum_{i=1}^n\gamma_i(x_i)\}.$$

Thus, if we can perform MAP inference $\max_{x\in\mathcal{X}}\{f(x)\}$, then it is typically feasible to calculate MAP for single node perturbations $\max_{x\in\mathcal{X}}\{f(x)+\sum_{i=1}^n\gamma_i(x_i)\}$ as well,

and also we can evaluate its expected value with Monte Carlo techniques.

Other partition function approximations can be constructed as well. The Mean Field approach gives a bound on the partition function and this bound can be used inside a learning algorithm. "Loopy" belief propagation algorithm also proposes an approximate treatment for the problem of inference and thus, provides a partition function approximation as a subroutine via message-passing technique. Other variations of this idea are Fractional Belief Propagation (Wiegerinck and Heskes [2003]), Tree-Reweighted Belief Propagation (Wainwright and Jordan [2008a]) and Generalized Belief Propagation (Yedidia et al. [2005]).

Chapter 2

Parameter Learning for Log-supermodular Distributions

Abstract

We consider log-supermodular models on binary variables, which are probabilistic models with negative log-densities which are submodular. These models provide probabilistic interpretations of common combinatorial optimization tasks such as image segmentation. In this thesis, we focus primarily on parameter estimation in the models from known upper-bounds on the intractable log-partition function. We show that the bound based on separable optimization on the base polytope of the submodular function is always inferior to a bound based on "perturb-and-MAP" ideas. Then, to learn parameters, given that our approximation of the log-partition function is an expectation (over our own randomization), we use a stochastic subgradient technique to maximize a lower-bound on the log-likelihood. This can also be extended to conditional maximum likelihood. We illustrate our new results in a set of experiments in binary image denoising, where we highlight the flexibility of a probabilistic model to learn with missing data.

This chapter is based on the work "Parameter Learning for Log-supermodular Distributions", T. Shpakova, F. Bach, published in Advances in Neural Information Processing Systems (NIPS), 2016.

2.1 Introduction

Submodular functions provide efficient and flexible tools for learning on discrete data. Several common combinatorial optimization tasks, such as clustering, image segmentation, or document summarization, can be achieved by the minimization or the maximization of a submodular function (Bach [2013], Golovin and Krause [2011], Lin and Bilmes [2011]). The key benefit of submodularity is the ability to model notions of diminishing returns, and the availability of exact minimization algorithms and approximate maximization algorithms with precise approximation guarantees (Krause

and Golovin [2014]).

In practice, it is not always straightforward to define an appropriate submodular function for a problem at hand. Given fully-labeled data, e.g., images and their foreground/background segmentations in image segmentation, structured-output prediction methods such as the structured-SVM may be used (Szummer et al. [2008]). However, it is common (a) to have missing data, and (b) to embed submodular function minimization within a larger model. These are two situations well tackled by *probabilistic modelling*.

Log-supermodular models, with negative log-densities equal to a submodular function, are a first important step toward probabilistic modelling on discrete data with submodular functions (Djolonga and Krause [2014]). However, it is well known that the log-partition function is intractable in such models. Several bounds have been proposed, that are accompanied with variational approximate inference (Djolonga and Krause [2015]). These bounds are based on the submodularity of the negative log-densities. However, parameter learning (typically by maximum likelihood), which is a key feature of probabilistic modeling, has not been tackled yet.

2.2 Contributions

In this chapter we highlight the following contributions:

- In Section 2.4, we review existing variational bounds for the log-partition function and show that the bound of Hazan and Jaakkola [2012], based on "perturb-and-MAP" ideas, formally dominates the bounds proposed by Djolonga and Krause [2014, 2015].
- In Section 2.5.1, we show that for parameter learning via maximum likelihood the existing bound of Djolonga and Krause [2014, 2015] typically leads to a degenerate solution while the one based on "perturb-and-MAP" ideas and logistic samples of Hazan and Jaakkola [2012] does not.
- In Section 2.5.2, given that the bound based on "perturb-and-MAP" ideas is an expectation (over our own randomization), we propose to use a stochastic subgradient technique to maximize the lower-bound on the log-likelihood, which can also be extended to conditional maximum likelihood.
- In Section 2.6, we illustrate our new results on a set of experiments in binary image denoising, where we highlight the flexibility of a probabilistic model for learning with missing data.

2.3 Submodular functions and log-supermodular models

In this section, we review the relevant theory of submodular functions and recall typical examples of log-supermodular distributions.

2.3.1 Submodular functions

We consider submodular functions on the vertices of the hypercube $\{0,1\}^D$. This hypercube representation is equivalent to the power set of $\{1,\dots,D\}$. Indeed, we can go from a vertex of the hypercube to a set by looking at the indices of the components equal to one and from set to vertex by taking the indicator vector of the set.

For any two vertices of the hypercube, $x, y \in \{0,1\}^D$, a function $f : \{0,1\}^D \to \mathbb{R}$ is submodular if

$$f(x) + f(y) \geqslant f(\min\{x,y\}) + f(\max\{x,y\}),$$

where the min and max operations are taken component-wise and correspond to the intersection and union of the associated sets. Equivalently, the function $x \mapsto f(x+e_i) - f(x)$, where $e_i \in \mathbb{R}^D$ is the i-th canonical basis vector, is non-increasing. Hence, the notion of diminishing returns is often associated with submodular functions. Most widely used submodular functions are cuts, concave functions of subset cardinality, mutual information, set covers, and certain functions of eigenvalues of submatrices (Bach [2013], Fujishige [2005]). Supermodular functions are simply negatives of submodular functions.

In this chapter, we are going to use a few properties of such submodular functions (see Bach [2013], Fujishige [2005] and references therein). Any submodular function f can be extended from $\{0,1\}^D$ to a convex function on $\mathbb{R}^D$, which is called the Lovász extension. This extension has the same value on $\{0,1\}^D$, hence we use the same notation f. Moreover, this function is convex and piecewise linear, which implies the existence of a polytope $B(f) \subset \mathbb{R}^D$, called the base polytope, such that for all $x \in \mathbb{R}^D$, $f(x) = \max_{s \in B(f)} x^\top s$, that is, f is the support function of $B(f)$.

The Lovász extension f and the base polytope $B(f)$ have explicit expressions that are presented in the Chapter 1. We will only use the fact that f can be efficiently minimized on $\{0,1\}^D$, by a variety of generic algorithms, or by more efficient dedicated ones for subclasses such as graph-cuts.

2.3.2 Log-supermodular distributions

Log-supermodular models are introduced in Djolonga and Krause [2014] to model probability distributions on a hypercube, $x \in \{0,1\}^D$, and are defined as

$$p(x) = \frac{1}{Z(f)} \exp(-f(x)),$$

where $f : \{0,1\}^D \to \mathbb{R}$ is a submodular function such that $f(0) = 0$ and the partition function is $Z(f) = \sum_{x \in \{0,1\}^D} \exp(-f(x))$. It is more convenient to deal with the convex log-partition function

$$A(f) = \log Z(f) = \log \sum_{x \in \{0,1\}^D} \exp(-f(x)).$$

In general, the calculation of the partition function $Z(f)$ or the log-partition function $A(f)$ is intractable, as it includes simple binary Markov random fields—the exact calculation is known to be $\#P$-hard (Jerrum and Sinclair [1993]). In Section 2.4, we review upper-bounds for the log-partition function.

2.3.3 Examples

Essentially, all submodular functions used in the minimization context can be used as negative log-densities (Djolonga and Krause [2014, 2015]). In computer vision, the most common examples are graph-cuts, which are essentially binary Markov random fields with attractive potentials, but higher-order potentials have been considered as well (Kohli et al. [2009]). In our experiments, we use graph-cuts, where submodular function minimization may be performed with max-flow techniques and is thus efficient (Boykov et al. [2001]). Note that there are extensions of submodular functions to continuous domains that could be considered as well (Bach [2016]).

2.4 Upper-bounds on the log-partition function

In this section, we review the main existing upper-bounds on the log-partition function for log-supermodular densities. These upper-bounds use several properties of submodular functions, in particular, the Lovász extension and the base polytope. Note that lower bounds based on submodular maximization aspects and superdifferentials (Djolonga and Krause [2014]) can be used to highlight the tightness of various bounds, which we present in Figure 2-1.

2.4.1 Base polytope relaxation with L-Field (Djolonga and Krause [2014])

This method exploits the fact that any submodular function $f(x)$ can be lower bounded by a modular function $s(x)$, i.e., a linear function of $x \in \{0,1\}^D$ in the hypercube representation. The submodular function and its lower bound are related by $f(x) = \max_{s \in B(f)} s^\top x$, leading to:

$$A(f) = \log \sum_{x \in \{0,1\}^D} \exp\left(-f(x)\right) = \log \sum_{x \in \{0,1\}^D} \min_{s \in B(f)} \exp\left(-s^\top x\right),$$

which, by swapping the sum and min, is less than

$$\min_{s \in B(f)} \log \sum_{x \in \{0,1\}^D} \exp\left(-s^\top x\right) = \min_{s \in B(f)} \sum_{d=1}^{D} \log\left(1 + e^{-s_d}\right) \stackrel{\text{def}}{=} A_{\text{L-field}}(f). \tag{2.4.1}$$

Since the polytope $B(f)$ is tractable (through its membership oracle or by maximizing linear functions efficiently), the bound $A_{\text{L-field}}(f)$ is tractable, i.e., computable in polynomial time. Moreover, it has a nice interpretation through convex duality as the

logistic function $\log(1+e^{-s_d})$ may be represented as $\max_{\mu_d \in [0,1]} -\mu_d s_d - \mu_d \log \mu_d - (1-\mu_d)\log(1-\mu_d)$, leading to:

$$A_{\text{L-field}}(f) = \min_{s \in B(f)} \max_{\mu \in [0,1]^D} -\mu^\top s + H(\mu) = \max_{\mu \in [0,1]^D} H(\mu) - f(\mu),$$

where $H(\mu) = -\sum_{d=1}^D \Big\{\mu_d \log \mu_d + (1-\mu_d)\log(1-\mu_d)\Big\}$. This shows in particular the convexity of $f \mapsto A_{\text{L-field}}(f)$. Finally, Djolonga and Krause [2015] shows the remarkable result that the minimizer $s \in B(f)$ may be obtained by minimizing a simpler function on $B(f)$, namely the squared Euclidean norm, thus leading to algorithms such as the minimum-norm-point algorithm (Fujishige [2005]).

2.4.2 "Pertub-and-MAP" with logistic distributions

Estimating the log-partition function can be done through optimization using "pertub-and-MAP" ideas. The main idea is to perturb the log-density, find the maximum a-posteriori configuration (i.e., perform optimization), and then average over several random perturbations (Hazan and Jaakkola [2012], Papandreou and Yuille [2011], Tarlow et al. [2012]).

The Gumbel distribution on $\mathbb{R}$, whose cumulative distribution function is $F(z) = \exp(-\exp(-(z+c)))$, where c is the Euler constant, is particularly useful. Indeed, if $\{g(y)\}_{y \in \{0,1\}^D}$ is a collection of independent random variables $g(y)$ indexed by $y \in \{0,1\}^D$, each following the Gumbel distribution, then the random variable $\max_{y \in \{0,1\}^D} g(y) - f(y)$ is such that we have

$$\log Z(f) = \mathbb{E}_g\Big[\max_{y \in \{0,1\}^D} \{g(y) - f(y)\}\Big],$$

from the [Hazan and Jaakkola, 2012, Lemma 1]. The main problem is that we need 2^D such variables, and a key contribution of Hazan and Jaakkola [2012] is to show that if we consider a factored collection $\{g_d(y_d)\}_{y_d \in \{0,1\}, d=1,\dots,D}$ of i.i.d. Gumbel variables, then we get an upper-bound on the log partition-function, that is,

$$\log Z(f) \leq \mathbb{E}_g \max_{y \in \{0,1\}^D} \{\sum_{d=1}^D g_d(y_d) - f(y)\}.$$

Writing $g_d(y_d) = [g_d(1) - g_d(0)]y_d + g_d(0)$ and using the fact that (a) $g_d(0)$ has zero expectation and (b) the difference between two independent Gumbel distributions has a logistic distribution (with cumulative distribution function $z \mapsto (1+e^{-z})^{-1}$) (Nadarajah and Kotz [2005]), we get the following upper-bound:

$$A_{\text{Logistic}}(f) = \mathbb{E}_{z_1,\dots,z_D \sim \text{logistic}}\Big[\max_{y \in \{0,1\}^D} \{z^\top y - f(y)\}\Big], \tag{2.4.2}$$

where the random vector $z \in \mathbb{R}^D$ consists of independent elements taken from the logistic distribution. This is always an upper-bound on $A(f)$ and it uses only the fact

that submodular functions are efficient to optimize. It is convex in f as an expectation of a maximum of affine functions of f.

2.4.3 Comparison of bounds

In this section, we show that $A_{\text{L-field}}(f)$ is always dominated by $A_{\text{Logistic}}(f)$. This is complemented by another result within the maximum likelihood framework in Section 2.5.

Proposition 2.1. *For any submodular function $f : \{0,1\}^D \to \mathbb{R}$, we have:*

$$A(f) \leqslant A_{\text{Logistic}}(f) \leqslant A_{\text{L-field}}(f). \tag{2.4.3}$$

Proof. The first inequality was shown by Hazan and Jaakkola [2012]. For the second inequality, we have:

$$\begin{aligned}
A_{\text{Logistic}}(f) = \;& \mathbb{E}_z\Big[\max_{y\in\{0,1\}^D} z^\top y - f(y)\Big] \\
= \;& \mathbb{E}_z\Big[\max_{y\in\{0,1\}^D} z^\top y - \max_{s\in B(f)} s^\top y\Big] \text{ from properties of the base polytope } B(f), \\
= \;& \mathbb{E}_z\Big[\max_{y\in\{0,1\}^D} \min_{s\in B(f)} z^\top y - s^\top y\Big], \\
= \;& \mathbb{E}_z\Big[\min_{s\in B(f)} \max_{y\in\{0,1\}^D} z^\top y - s^\top y\Big] \text{ by convex duality,} \\
\leqslant \;& \min_{s\in B(f)} \mathbb{E}_z\Big[\max_{y\in\{0,1\}^D} (z-s)^\top y\Big] \text{ by swapping expectation and minimization,} \\
= \;& \min_{s\in B(f)} \mathbb{E}_z\Big[\sum_{d=1}^D (z_d - s_d)_+\Big] \text{ by explicit maximization,} \\
= \;& \min_{s\in B(f)} \Big[\sum_{d=1}^D \mathbb{E}_{z_d}(z_d - s_d)_+\Big] \text{ by using linearity of expectation,} \\
= \;& \min_{s\in B(f)} \Big[\sum_{d=1}^D \int_{-\infty}^{+\infty} (z_d - s_d)_+ P(z_d) dz_d\Big] \text{ by definition of expectation,} \\
= \;& \min_{s\in B(f)} \Big[\sum_{d=1}^D \int_{s_d}^{+\infty} (z_d - s_d) \frac{e^{-z_d}}{(1+e^{-z_d})^2} dz_d\Big] \text{ by substituting the density function,} \\
= \;& \min_{s\in B(f)} \sum_{d=1}^D \log(1+e^{-s_d}), \text{ which leads to the desired result.}
\end{aligned}$$

□

In the inequality above, since the logistic distribution has full support, there cannot be equality. However, if the base polytope is such that, with high probability $\forall d, |s_d| \geq |z_d|$, then the two bounds are close. Since the logistic distribution is concentrated around zero, we have equality when $|s_d|$ is large for all d and $s \in B(f)$.

Running-time complexity of $A_{\text{L-field}}$ and A_{logistic}. The logistic bound A_{logistic} can be computed if there is efficient MAP-solver for submodular functions (plus

a modular term). In this case, the divide-and-conquer algorithm can be applied for L-Field (Djolonga and Krause [2014]). Thus, the complexity is dedicated to the minimization of $O(|V|)$ problems. Meanwhile, for the method based on logistic samples, it is necessary to solve M optimization problems. In our empirical bound comparison (next paragraph), the running time was the same for both methods. Note however that for parameter learning, we need a *single* SFM problem per gradient iteration (and not M).

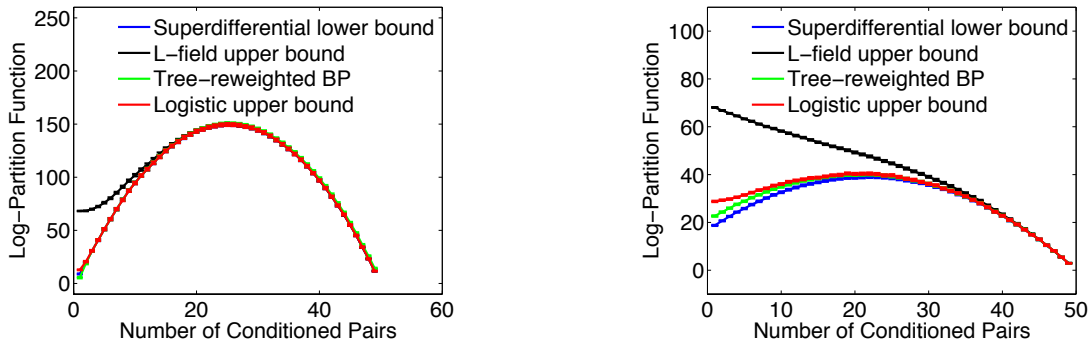
Empirical comparison of $A_{\text{L-field}}$ and A_{logistic}. We compare the upper-bounds on the log-partition function $A_{\text{L-field}}$ and A_{logistic}, with the setup used by Djolonga and Krause [2014]. We thus consider data from a Gaussian mixture model with 2 clusters in $\mathbb{R}^2$. The centers are sampled from $\mathcal{N}([3,3], I)$ and $\mathcal{N}([-3,-3], I)$, respectively. Then we sampled $n = 50$ points for each cluster. Further, these $2n$ points are used as nodes in a complete weighted graph, where the weight between points x and y is equal to $e^{-c||x-y||}$.

We consider the graph cut function associated to this weighted graph, which defines a log-supermodular distribution. We then consider conditional distributions, one for each $k = 1, \ldots, n$, on the events that at least k points from the first cluster lie on the one side of the cut and at least k points from the second cluster lie on the other side of the cut. For each conditional distribution, we evaluate and compare the two upper bounds. We also add the tree-reweighted belief propagation upper bound (Wainwright and Jordan [2008b]) and the superdifferential-based lower bound (Djolonga and Krause [2014]).

In Figure 2-1, we show various bounds on $A(f)$ as functions of the number on conditioned pairs. The logistic upper bound is obtained using 100 logistic samples: the logistic upper-bound A_{logistic} is close to the superdifferential lower bound from Djolonga and Krause [2014] and is indeed significantly lower than the bound $A_{\text{L-field}}$. However, the tree-reweighted belief propagation bound behaves a bit better in the second case, but its calculation takes more time, and it cannot be applied for general submodular functions.

2.4.4 From bounds to approximate inference

Since linear functions are submodular functions, given any convex upper-bound on the log-partition function, we may derive an approximate marginal probability for each $x_d \in \{0, 1\}$. Indeed, following Hazan and Jaakkola [2012], we consider an exponential family model $p(x|t) = \exp(-f(x) + t^\top x - A(f - t))$, where $f - t$ is the function $x \mapsto f(x) - t^\top x$. When f is assumed to be fixed, this can be seen as an exponential family with the base measure $\exp(-f(x))$, sufficient statistics x, and $A(f - t)$ is the log-partition function. It is known that the expectation of the sufficient statistics under the exponential family model $\mathbb{E}_{p(x|t)}x$ is the gradient of the log-partition function (Wainwright and Jordan [2008b]). Hence, any approximation of this log-partition gives an approximation of this expectation, which in our situation is the vector of marginal probabilities that an element is equal to 1.



(a) Mean bounds with confidence intervals, $c = 1$.

(b) Mean bounds with confidence intervals, $c = 3$.

Figure 2-1: Comparison of log-partition function bounds for different values of c. See text for details.

For the L-field bound, at $t = 0$, we have $\partial_{t_d} A_{\text{L-field}}(f - t) = \sigma(s_d^*)$, where s^* is the minimizer of $\sum_{d=1}^{D} \log(1 + e^{-s_d})$, thus recovering the interpretation of Djolonga and Krause [2015] from another point of view.

For the logistic bound, this is the inference mechanism from Hazan and Jaakkola [2012], with $\partial_{t_d} A_{\text{logistic}}(f - t) = \mathbb{E}_z y^*(z)$, where $y^*(z)$ is the maximizer of $\max_{y \in \{0,1\}^D} z^\top y - f(y)$. In practice, in order to perform approximate inference, we only sample M logistic variables. We could do the same for parameter learning, but a much more efficient alternative, based on mixing sampling and convex optimization, is presented in the next section.

2.5 Parameter learning through maximum likelihood

An advantage of log-supermodular probabilistic models is the opportunity to learn the model parameters from data using the maximum-likelihood principle. In this section, we consider that we are given N observations $x_1, \ldots, x_N \in \{0,1\}^D$, e.g., binary images such as shown in Figure 2-2.

We consider a submodular function $f(x)$ represented as

$$f(x) = \sum_{k=1}^{K} \alpha_k f_k(x) - t^\top x.$$

The modular term $t^\top x$ is explicitly taken into account with $t \in \mathbb{R}^D$, and K base submodular functions are assumed to be given with $\alpha \in \mathbb{R}_+^K$ so that the function f

remains submodular. Assuming the data $x_1, \dots, x_N$ are independent and identically (i.i.d.) distributed, then maximum likelihood is equivalent to minimizing:

$$\min_{\alpha \in \mathbb{R}_+^K,\ t \in \mathbb{R}^D} -\frac{1}{N} \sum_{n=1}^N \log p(x_n | \alpha, t) = \min_{\alpha \in \mathbb{R}_+^K,\ t \in \mathbb{R}^D} \frac{1}{N} \sum_{n=1}^N \Big\{ \sum_{k=1}^K \alpha_k f_k(x_n) - t^\top x_n + A(f) \Big\},$$

which takes the particularly simple form

$$\min_{\alpha \in \mathbb{R}_+^K,\ t \in \mathbb{R}^D} \sum_{k=1}^K \alpha_k \Big(\frac{1}{N} \sum_{n=1}^N f_k(x_n) \Big) - t^\top \Big(\frac{1}{N} \sum_{n=1}^N x_n \Big) + A(\alpha, t), \tag{2.5.1}$$

where we use the notation $A(\alpha, t) = A(f)$. We now consider replacing the intractable log-partition function by its approximations defined in Section 2.4.

2.5.1 Learning with the L-field approximation

In this section, we show that if we replace $A(f)$ by $A_{\text{L-field}}(f)$, we obtain a degenerate solution. Indeed, we have

$$A_{\text{L-field}}(\alpha, t) \quad = \quad \min_{s \in B(f)} \sum_{d=1}^D \log\big(1 + e^{-s_d}\big) = \min_{s \in B(\sum_{k=1}^K \alpha_K f_K)} \sum_{d=1}^D \log\big(1 + e^{-s_d + t_d}\big).$$

This implies that Eq. (2.5.1) becomes

$$\min_{\alpha \in \mathbb{R}_+^K,\ t \in \mathbb{R}^D} \min_{s \in B(\sum_{k=1}^K \alpha_K f_K)} \sum_{k=1}^K \alpha_k \Big(\frac{1}{N} \sum_{n=1}^N f_k(x_n) \Big) - t^\top \Big(\frac{1}{N} \sum_{n=1}^N x_n \Big) + \sum_{d=1}^D \log\big(1 + e^{-s_d + t_d}\big).$$

The minimum with respect to t_d may be performed in closed form with $t_d - s_d = \log \frac{\langle x \rangle_d}{1 - \langle x \rangle_d}$, where $\langle x \rangle = \frac{1}{N} \sum_{n=1}^N x_n$. Putting this back into the equation above, we get the equivalent problem:

$$\min_{\alpha \in \mathbb{R}_+^K} \min_{s \in B(\sum_{k=1}^K \alpha_K f_K)} \sum_{k=1}^K \alpha_k \Big(\frac{1}{N} \sum_{n=1}^N f_k(x_n) \Big) - s^\top \Big(\frac{1}{N} \sum_{n=1}^N x_n \Big) + \text{ const },$$

which is equivalent to, using the representation of f as the support function of $B(f)$:

$$\min_{\alpha \in \mathbb{R}_+^K} \sum_{k=1}^K \alpha_k \Big[\frac{1}{N} \sum_{n=1}^N f_k(x_n) - f_k\Big(\frac{1}{N} \sum_{n=1}^N x_n \Big) \Big].$$

Since f_k is convex, by Jensen's inequality, the linear term in α_k is non-negative; thus maximum likelihood through L-field will lead to a degenerate solution where all α's are equal to zero.

2.5.2 Learning with the logistic approximation with stochastic gradients

In this section we consider the problem (2.5.1) and replace $A(f)$ by $A_{\text{Logistic}}(f)$:

$$\min_{\alpha\in\mathbb{R}_+^K,\ t\in\mathbb{R}^D} \sum_{k=1}^K \alpha_k \langle f_k(x)\rangle_{\text{emp.}} - t^\top \langle x\rangle_{\text{emp.}} + \\ + \mathbb{E}_{z\sim\text{logistic}}\Big[\max_{y\in\{0,1\}^D} z^\top y + t^\top y - \sum_{k=1}^K \alpha_k f(y)\Big], \tag{2.5.2}$$

where $\langle M(x)\rangle_{\text{emp.}}$ denotes the empirical average of $M(x)$ (over the data). Denoting by $y^*(z,t,\alpha)\in\{0,1\}^D$ the maximizers of $z^\top y + t^\top y - \sum_{k=1}^K \alpha_k f(y)$, the objective function may be written:

$$\sum_{k=1}^K \alpha_k \Big[\langle f_k(x)\rangle_{\text{emp.}} - \langle f_k(y^*(z,t,\alpha))\rangle_{\text{logistic}}\Big] - \\ -t^\top\Big[\langle x\rangle_{\text{emp.}} - \langle y^*(z,t,\alpha)\rangle_{\text{logistic}}\Big] + \langle z^\top y^*(z,t,\alpha)\rangle_{\text{logistic}}.$$

This implies that at optimum, for $\alpha_k > 0$, then $\langle f_k(x)\rangle_{\text{emp.}} = \langle f_k(y^*(z,t,\alpha))\rangle_{\text{logistic}}$, while, $\langle x\rangle_{\text{emp.}} = \langle y^*(z,t,\alpha)\rangle_{\text{logistic}}$, the expected values of the sufficient statistics match between the data and the optimizers used for the logistic approximation (Hazan and Jaakkola [2012]).

In order to minimize the expectation in Eq. (2.5.2), we propose to use the projected stochastic gradient method, not on the data as usually done, but on our own internal randomization. The algorithm then becomes, once we add a weighted ℓ_2-regularization $\Omega(t,\alpha)$:

- **Input**: functions f_k, $k=1,\dots,K$, and expected sufficient statistics $\langle f_k(x)\rangle_{\text{emp.}}\in\mathbb{R}$ and $\langle x\rangle_{\text{emp.}}\in[0,1]^D$, regularizer $\Omega(t,\alpha)$.
- **Initialization**: $\alpha = 0, t = 0$
- **Iterations**: for h from 1 to H
  - Sample $z\in\mathbb{R}^D$ as independent logistics
  - Compute $y^* = y^*(z,t,\alpha)\in\arg\max_{y\in\{0,1\}^D} z^\top y + t^\top y - \sum_{k=1}^K \alpha_k f(y)$
  - Replace t by $t - \frac{C}{\sqrt{h}}\Big[y^* - \langle x\rangle_{\text{emp.}} + \partial_t\Omega(t,\alpha)\Big]$
  - Replace α_k by $\Big(\alpha_k - \frac{C}{\sqrt{h}}\Big[\langle f_k(x)\rangle_{\text{emp.}} - f_k(y^*) + \partial_{\alpha_k}\Omega(t,\alpha)\Big]\Big)_+$.
- **Output**: (α,t).

Since our cost function is convex and Lipschitz-continuous, the averaged iterates are converging to the global optimum (Nemirovski et al. [2009]) at rate $1/\sqrt{H}$ (for function values).

2.5.3 Extension to conditional maximum likelihood

In experiments in Section 2.6, we consider a joint model over two binary vectors $x, z \in \mathbb{R}^D$, as follows

$$p(x, z|\alpha, t, \pi) = p(x|\alpha, t)p(z|x, \pi) = \exp(-f(x) - A(f)) \prod_{d=1}^{D} \pi_d^{\delta(z_d \neq x_d)}(1 - \pi_d)^{\delta(z_d = x_d)}, \tag{2.5.3}$$

which corresponds to sampling x from a log-supermodular model and considering z that switches the values of x with probability π_d for each d, that is, a noisy observation of x. We have:

$$\log p(x, z|\alpha, t, \pi) \quad = \quad -f(x) - A(f) + \sum_{d=1}^{D} \Big\{ -\log(1 + e^{u_d}) + x_d u_d + z_d u_d - 2x_d z_d u_d \Big\},$$

with $u_d = \log \frac{\pi_d}{1-\pi_d}$ which is equivalent to $\pi_d = (1 + e^{-u_d})^{-1}$.

Using Bayes rule, we have

$$p(x|z, \alpha, t, \pi) \propto \exp(-f(x) - A(f) + x^\top u - 2x^\top (u \circ z)),$$

which leads to the log-supermodular model

$$p(x|z, \alpha, t, \pi) = \exp(-f(x) + x^\top (u - 2u \circ z) - A(f - u + 2u \circ z)).$$

Thus, if we observe both z and x, we can consider a conditional maximization of the log-likelihood (still a convex optimization problem), which we do in our experiments for supervised image denoising, where we assume we know both noisy and original images at training time. Stochastic gradient on the logistic samples can then be used. Note that our conditional ML estimation can be seen as a form of approximate conditional random fields (Lafferty et al. [2001]).

While supervised learning can be achieved by other techniques such as structured-output-SVMs (Szummer et al. [2008], Taskar et al. [2003], Tsochantaridis et al. [2005]), our approach also applies when we do not observe the original image, which we now consider.

2.5.4 Missing data through maximum likelihood

In the model in Eq. (2.5.3), we now assume we only observed the noisy output z, and we perform parameter learning for α, t, π. This is a latent variable model for which maximum likelihood can be readily applied. We have:

$$\begin{aligned} \log p(z|\alpha, t, \pi) \quad &= \quad \log \sum_{x \in \{0,1\}} p(z, x|\alpha, t, \pi) \\ &= \quad \log \sum_{x \in \{0,1\}^D} \exp(-f(x) - A(f)) \prod_{d=1}^{D} \pi_d^{\delta(z_d \neq x_d)}(1 - \pi_d)^{\delta(z_d = x_d)} \end{aligned}$$

(a) original image (b) noisy image (c) denoised image

Figure 2-2: Denoising of a horse image from the Weizmann horse database (Borenstein et al. [2004]).

$$= \quad A(f - u + 2u \circ z) - A(f) + z^\top u - \sum_{d=1}^{D} \log(1 + e^{u_d}).$$

In practice, we will assume that the noise probability π (and hence u) is uniform across all elements. While we could use majorization-minization approaches such as the expectation-minimization algorithm (EM), we consider instead stochastic subgradient descent to learn the model parameters α, t and u (now a non-convex optimization problem, for which we still observed good convergence).

2.6 Experiments

The aim of our experiments is to demonstrate the ability of our approach to remove noise in binary images, following the experimental set-up of Hazan and Jaakkola [2012]. We consider the training sample of $N_{train} = 100$ images of size $D = 50 \times 50$, and the test sample of $N_{test} = 100$ binary images, containing a horse silhouette from the Weizmann horse database (Borenstein et al. [2004]). At first we add some noise by flipping pixels values independently with probability π. In Figure 2-2, we provide an example from the test sample: the original, the noisy and the denoised image (by our algorithm).

We consider the model from Section 2.5.3, with the two functions $f_1(x)$, $f_2(x)$ which are horizontal and vertical cut functions with binary weights respectively, together with a modular term of dimension D. To perform minimization we use graph-cuts (Boykov et al. [2001]) as we deal with positive or attractive potentials.

Supervised image denoising. We assume that we observe $N = 100$ pairs (x_i, z_i) of original-noisy images, $i = 1, \ldots, N$. We perform parameter inference by maximum likelihood using stochastic subgradient descent (over the logistic samples), with regularization by the squared ℓ_2-norm, one parameter for t, one for α, both learned by cross-validation. Given our estimates, we may denoise a new image by computing the "max-marginal", e.g., the maximum a posteriori $\max_x p(x|z, \alpha, t)$ through a single graph-cut, or computing "mean-marginals" with 100 logistic samples. To calculate the error we use the normalized Hamming distance and 100 test images.

noise π	max-marg.	std	mean-marginals	std	SVM-Struct	std
1%	0.4%	<0.1%	0.4%	<0.1%	0.6%	<0.1%
5%	1.1%	<0.1%	1.1%	<0.1%	1.5%	<0.1%
10%	2.1%	<0.1%	2.0%	<0.1%	2.8%	0.3%
20%	4.2%	<0.1%	4.1%	<0.1%	6.0%	0.6%

Table 2.1: Supervised denoising results.

Results are presented in Table 2.1, where we compare the two types of decoding, as well as a structured output SVM (SVM-Struct Tsochantaridis et al. [2005]) applied to the same problem. Results are reported in proportion of correct pixels. We see that the probabilistic models here slightly outperform the max-margin formulation [1] and that using mean-marginals (which is optimal given our loss measure) lead to slightly better performance.

π	max-marg.	std	mean-marg.	std
1%	0.5%	<0.1%	0.5%	<0.1%
5%	0.9%	0.1%	1.0%	0.1%
10%	1.9%	0.4%	2.1%	0.4%
20%	5.3%	2.0%	6.0%	2.0%

Table 2.2: Unsupervised denoising results. Level of noise π is fixed.

π	max-marg.	std	mean-marg.	std
1%	1.0%	-	1.0%	-
5%	3.5%	0.9%	3.6%	0.8%
10%	6.8%	2.2%	7.0%	2.0%
20%	20.0%	-	20.0%	-

Table 2.3: Unsupervised denoising results. Level of noise π is not fixed.

Unsupervised image denoising. We now only consider $N = 100$ noisy images $z_1, \ldots, z_N$ to learn the model, without the original images, and we use the latent model from Section 2.5.4. We apply stochastic subgradient descent for the difference of the two convex functions A_{logistic} to learn the model parameters and use fixed regularization parameters equal to 10^{-2}.

We consider two situations, with a known noise-level π or with learning it together with α and t. The error was calculated using either max-marginals and mean-marginals.

1. Hazan and Jaakkola [2012] shows a stronger difference, which we believe (after consulting with authors) is due to lack of convergence for the iterative algorithm solving the max-margin formulation.

Note that here, structured-output SVMs cannot be used because there is no supervision. Results are reported in Tables 2.2 and 2.3. One explanation for a better performance for max-marginals in this case is that the unsupervised approach tends to oversmooth the outcome and max-marginals correct this a bit.

When the noise level is known, the performance compared to supervised learning is not degraded much, showing the ability of the probabilistic models to perform parameter estimation with missing data. When the noise level is unknown and learned as well, results are worse, still better than a trivial answer for moderate levels of noise (5% and 10%) but not better than outputting the noisy image for extreme levels (1% and 20%). In challenging fully unsupervised case the standard deviation is up to 2.2% (which shows that our results are statistically significant).

2.7 Conclusion

In this paper, we have presented how approximate inference based on stochastic gradient and "perturb-and-MAP" ideas could be used to learn parameters of log-supermodular models, allowing to benefit from the versatility of probabilistic modelling, in particular in terms of parameter estimation with missing data. While our experiments have focused on simple binary image denoising, exploring larger-scale applications in computer vision (such as done by Zhang et al. [2015], Tschiatschek et al. [2016]) should also show the benefits of mixing probabilistic modelling and submodular functions.

Chapter 3

Marginal Weighted Maximum Log-likelihood for Efficient Learning of Perturb-and-Map Models

Abstract

In this part we consider the structured-output prediction problem through probabilistic approaches and generalize the "perturb-and-MAP" framework to more challenging weighted Hamming losses, which are crucial in applications. While in principle our approach is a straightforward marginalization, it requires solving many related MAP inference problems. We show that for log-supermodular pairwise models these operations can be performed efficiently using the machinery of dynamic graph cuts. We also propose to use *double* stochastic gradient descent, both on the data and on the perturbations, for efficient learning. Our framework can naturally take weak supervision (e.g., partial labels) into account. We conduct a set of experiments on medium-scale character recognition and image segmentation, showing the benefits of our algorithms.

This chapter is based on the work "Marginal Weighted Maximum Log-likelihood for Efficient Learning of Perturb-and-Map Models", T. Shpakova, F. Bach, A. Osokin, published in Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), 2018.

3.1 Introduction

Structured-output prediction is an important and challenging problem in the field of machine learning. When outputs have a structure, often in terms of parts or elements (e.g., pixels, sentences or characters), methods that do take it into account typically outperform more naive methods that consider outputs as a set of independent

elements. Structured-output methods based on optimization can be broadly separated in two main families: *max-margin methods*, such as structured support vector machines (SSVM from Taskar et al. [2003], Tsochantaridis et al. [2005]) and *probabilistic methods based on maximum likelihoods* such as conditional random fields (CRF) (Lafferty et al. [2001]).

Structured-output prediction faces many challenges: (1) on top of large input dimensions, problems also have large outputs, leading to scalability issues, in particular when prediction or learning depends on combinatorial optimization problems (which are often polynomial-time, but still slow given they are run many times); (2) it is often necessary to use losses which go beyond the traditional 0-1loss to shape the behavior of the learned models towards the final evaluation metric; (3) having fully labelled data is either rare or expensive and thus, methods should be able to deal with weak supervision.

Max-margin methods can be used with predefined losses, and have been made scalable by several recent contributions (see, e.g., Lacoste-Julien et al. [2013] and references therein), but do not deal naturally with weak supervision. However, a few works (Yu and Joachims [2009], Kumar et al. [2010], Girshick et al. [2011]) incorporate weak supervision into the max-margin approach via the CCCP (Yuille and Rangarajan [2003]) algorithm.

The flexibility of probabilistic modeling naturally allows (a) taking into consideration weak supervision and (b) characterizing the uncertainty of predictions, but it comes with strong computational challenges as well as a non-natural way of dealing with predefined losses beyond the 0-1 loss. The main goal of this chapter is to provide new tools for structured-output inference with probabilistic models, thus making them more widely applicable, while still being efficient. There are two main techniques to allow for scalable learning in CRFs: stochastic optimization (Vishwanathan et al. [2006]) and piecewise training (Sutton and McCallum [2005, 2007], Kolesnikov et al. [2014]); note that the techniques above can also be used for weak supervision (and we reuse some of them in this work).

Learning and inference in probabilistic structured-output models recently received a lot of attention from the research community (e.g., Bakir et al. [2007], Nowozin and Lampert [2011], Smith [2011]). In this work we consider only models for which maximum-a posteriori (MAP) inference is feasible (a step often referred to as decoding in max-margin formulations, and which typically makes them tractable). A lot of efforts were spent to explore MAP-solvers algorithms for various problems, leveraging various structures, e.g., graphs of low tree-width (Bishop [2006], Wainwright and Jordan [2008b], Sontag et al. [2008], Komodakis et al. [2011]) and function submodularity (Boros and Hammer [2002], Kolmogorov and Zabih [2004], Bach [2013], Osokin and Vetrov [2015]).

Naturally, the existence of even an exact and efficient MAP-solver does not mean that the partition function (a key tool for probabilistic inference as shown below) is tractable to compute. Indeed, the partition function computation is known to be $\#P$-hard (Jerrum and Sinclair [1993]) in general. For example, MAP-inference is efficient for log-supermodular probabilistic models, while computation of their partition function is not (Djolonga and Krause [2014]).

For such problems where MAP-inference is efficient, but partition function computation is not, "perturb-and-MAP" ideas such as proposed by Papandreou and Yuille [2011], Hazan and Jaakkola [2012] are a very suitable treatment. By adding random perturbations, and then performing MAP-inference, they can lead to estimates of the partition function. In Section 3.3, we review the existing approaches to the partition function approximation, parameter learning and inference.

An attempt to learn parameters via "perturb-and-MAP" ideas was made by Hazan et al. [2013], where the authors have developed a PAC-Bayesian-flavoured approach for the non-decomposable loss functions. While the presented algorithm has something in common with ours (gradient descent optimization of the objective upper bound), it differs in the sense of the objective function and the problem setup, which is more general but that requires a different (potentially with higher variance) estimates of the gradients. Such estimates are usual in reinforcement learning, e.g., the log-derivative trick from the REINFORCE algorithm (Williams [1992]).

The goal of this chapter is to make the "perturb-and-MAP" technique applicable to practical problems, in terms of (a) scale, by increasing the problem size significantly, and (b) losses, by treating structured losses such as the Hamming loss or its weighted version, which are crucial to obtaining good performance in practice.

3.2 Contributions

Overall, in this chapter we make the following contributions:

- In Section 3.4, we generalize the "perturb-and-MAP" framework to more challenging weighted Hamming losses which are commonly used in applications. In principle, this is a straightforward marginalization but this requires solving many related MAP inference problems. We show that for graph cuts (our main inference algorithm for image segmentation), this can be done particularly efficiently. Besides that, we propose to use a *double* stochastic gradient descent, both on the data and on the perturbations.
- In Section 3.5, we show how weak supervision (e.g., partial labels) can be naturally dealt with. Our method in this case relies on approximating marginal probabilities that can be done almost at the cost of the partition function approximation.
- In Section 3.6, we conduct a set of experiments on medium-scale character recognition and image segmentation, showing the benefits of our new algorithms.

3.3 Perturb-and-MAP

In this section, we introduce the notation and review the relevant background. We study the following probabilistic model (a.k.a. a Gibbs distribution) over a discrete product space $Y = Y_1 \times \cdots \times Y_D$,

$$P(y) = \frac{1}{Z(f)} e^{f(y)}, \tag{3.3.1}$$

which is defined by a potential function $f : Y \to \mathbb{R}$. The constant $Z(f) = \sum_{y \in Y} e^{f(y)}$ is called the partition function and normalizes $P(y)$ to be a valid probability function, i.e., to sum to one. $Z(f)$ is in general intractable to compute as the direct computation requires summing over exponentially (in D) many elements.

Various partition function approximations methods have been used in parameter learning algorithms (Parise and Welling [2005]), e.g., mean-field (Jordan et al. [1999]), tree-reweighted belief propagation (Wainwright and Jordan [2008b]) or loopy belief propagation (Weiss [2001]). We will work with the upper bound on the partition function proposed by Hazan and Jaakkola [2012] as it allows us to approximate the partition function via MAP-inference, calculate gradients efficiently, approximate marginal probabilities and guarantee tightness for some probabilistic models. We introduce this class of techniques below.

3.3.1 Gumbel perturbations

Recently, Hazan and Jaakkola [2012] provided a general-purpose upper bound on the *log-partition function* $A(f) = \log Z(f)$, based on the "perturb-and-MAP" idea (Papandreou and Yuille [2011]): maximize the potential function perturbed by Gumbel-distributed noise. [1]

Proposition 3.1 (Hazan and Jaakkola [2012], Corollary 1)**.** *For any function $f : Y \to \mathbb{R}$, we have $A(f) \leqslant A_{\mathrm{G}}(f)$, where*

$$A_{\mathrm{G}}(f) = \mathbb{E}_{z_1,\ldots,z_D \sim \mathrm{Gumbel}}\Big[\max_{y \in Y}\Big(f(y) + \sum_{d=1}^{D} z_d(y_d)\Big)\Big]. \tag{3.3.2}$$

Gumbel *denotes the Gumbel distribution and $\{z_d(y_d)\}_{y_d \in Y_d}^{d=1,\ldots,D}$ is a collection of independent Gumbel samples.*

The bound is tight when $f(y)$ is a separable function (i.e., a sum of functions of single variables), and the tightness of this bound was further studied by Shpakova and Bach [2016] for log-supermodular models (where f is supermodular). They have shown that the bound A_G is always lower (and thus provide a better bound) than the "L-field" bound proposed by Djolonga and Krause [2014, 2015], which is itself based on separable optimization on the base polytope of the associated supermodular function.

The partition function bound A_G can be approximated by replacing the expectation by an empirical average. That is, to approximate it we need to solve a large number (as many as the number of Gumbel samples used to approximate the expectation) of MAP-like problems (i.e., maximizing f plus a separable function) which are feasible by our assumption. Strictly speaking, the MAP-inference is NP-hard in general, but firstly, it is much easier than the partition function calculation, secondly, there are solvers for special cases, e.g., for log-supermodular models (which include functions f which are negatives of cuts (Kolmogorov and Zabih [2004], Boykov and Kolmogorov

1. The Gumbel distribution on the real line has cumulative distribution function $F(z) = \exp(-\exp(-(z+c)))$, where c is the Euler constant.

[2004]) and those solvers are often efficient enough in practice. In this chapter, we will focus primarily on a subcase of supermodular potentials, namely negatives of graph cuts.

3.3.2 Parameter learning and Inference

In the standard supervised setting of structured prediction, we are given N pairs of observations $\mathcal{D} = \{(x^n, y^n)\}_{n=1}^N$, where x^n is a feature representation of the n-th object and $y^n \in Y = Y_1 \times \cdots \times Y_{D^n}$ is a structured vector of interest (e.g., a sequence of tags, a segmentation MAP or a document summarization representation). In the standard linear model, the potential function $f(y|x)$ is represented as a linear combination: $f(y|x) = w^T \Psi(x, y)$, where w is a vector of weights and the structured feature map $\Psi(x, y)$ contains the relevant information for the feature-label pair (x, y). To learn the parameters using the predefined probabilistic model, one can use the (regularized) maximum likelihood approach:

$$\max_w \frac{1}{N} \sum_{n=1}^N \log P(y^n|x^n, w) - \frac{\lambda}{2}\|w\|^2, \tag{3.3.3}$$

where $\lambda > 0$ is a regularization parameter and the likelihood $P(y|x, w)$ is defined as

$$P(y|x, w) = \frac{\exp(f(y|x))}{Z(f, x)} = \exp(f(y|x) - A(f, x)),$$

where $A(f, x)$ is the log-partition function (that now depends on x, since we consider conditional models).

Hazan and Jaakkola [2012] proposed to learn parameters based on the Gumbel bound $A_G(f, x)$ instead of the intractable log-partition function:

$$\begin{aligned}
&\log P(y|x) = f(y|x) - A(f, x) \leq f(y|x) - A_G(f, x) \\
&= f(y|x) - \mathbb{E}_z\Big[\max_{y \in Y}\Big\{\sum_{d=1}^D z_d(y_d) + f(y)\Big\}\Big] \\
&\approx f(y|x) - \frac{1}{M}\sum_{m=1}^M \max_{y^{(m)} \in Y}\Big\{\sum_{d=1}^D z_d^{(m)}(y_d^{(m)}) + f(y^{(m)})\Big\}.
\end{aligned}$$

Hazan and Jaakkola [2012] considered the fully-supervised setup where labels y^n were given for all data points x^n. Shpakova and Bach [2016] developed the approach, but also considered a setup with missing data (part of the labels y^n are unknown) for the small Weizmann Horse dataset from Borenstein et al. [2004]. Leveraging the additional stochasticity present in the Gumbel samples, Shpakova and Bach [2016] extend the use of stochastic gradient descent, not on the data as usually done, but on the Gumbel randomization. It is equivalent to the choice of parameter $M = 1$ for every gradient computation (but with a new Gumbel sample at every iteration). In our work, we use the stochastic gradient descent in a regime stochastic w.r.t. *both* the data and the Gumbel perturbations. This allows us to apply the method to large-scale datasets.

For linear models, we have $f(y|x) = w^T\Psi(x, y)$ and $\Psi(x, y)$ is usually given or takes zero effort to compute. We assume that the gradient calculation $\nabla_w f(y|x) = \Psi(x, y)$ does not add complexity to the optimization algorithm. The gradient of $\log P(y|x)$ is equal to

$$\nabla_w f(y|x) - \nabla_w \max_{y \in Y} \left\{ \sum_{d=1}^{D} z_d(y_d) + f(y|x) \right\} = \nabla_w f(y|x) - \nabla_w f(y^*|x),$$

where y^* lies in $\arg\max$ of the perturbed optimization problem. The gradient of $\langle \log P(y|x) \rangle$ (the average over a subsample of data, typically a mini-batch) has the form $\langle \nabla_w f(y|x) \rangle - \langle \nabla_w f(y^*|x) \rangle = \langle \Psi(x, y) \rangle_{emp.} - \langle \Psi(x, y^*) \rangle$, where $\langle \Psi(x, y) \rangle_{emp.}$ denotes the empirical average over the data. Algorithm 1 contains this double stochastic gradient descent (SGD) with stochasticity w.r.t. both sampled data and Gumbel samples. The choice of the stepsize $\gamma_h = \frac{1}{\lambda h}$ is standard for strongly-convex problems (Shalev-Shwartz et al. [2011]).

Algorithm 1 Double SGD: stochasticity w.r.t. data and Gumbel samples

Input: dataset $\mathcal{D} = \{(x^n, y^n)\}_{n=1}^N$, number of iterations H, size of the mini-batch T, stepsize sequence $\{\gamma_h\}_{h=1}^H$, regularization parameter λ

Output: model parameters w

1: **Initialization**: $w = 0$
2: **for** $h = 1$ **to** H **do**
3: Sample data mini-batch of small size T (that is, T pairs of observations)
4: Calculate sufficient statistics $\langle \Psi(x, y) \rangle_{emp.}$ from the mini-batch
5: **for** t=1 **to** T **do**
6: Sample $z_d(y_d)$ as independent Gumbels for all $y_d \in Y_d$ and for all d
7: Find $y^* \in \arg\max_{y \in Y} \left\{ \sum_{d=1}^{D} z_d(y_d) + f(y) \right\}$
8: Make a gradient step:

$$w_{h+1} \to w_h + \gamma_h \Big(\langle \Psi(x, y) \rangle_{emp.} - \langle \Psi(x, y^*) \rangle - \lambda w_h \Big)$$

Note, that the classic log-likelihood formulation (3.3.3) is implicitly considering a "0-1 loss" $l_{0\text{-}1}(y, \hat{y}) = [y \neq \hat{y}]$ as it takes probability of the entire output object y^n conditioned on the observed feature representation x^n.

However, in many structured-output problems 0-1 loss evaluation is not an adequate performance measure. The Hamming or weighted Hamming losses that sum mistakes across the D elements of the outputs, are more in demand as they count misclassification per element.

Table 3.1: Variants of the Objective Loss $\ell(w, x, y)$ Function. $\{\theta_d(y_d)\}_{d=1}^D$ are the weights of the weighted Hamming loss, $\{q_d(y_d)\}_{d=1}^D$ are the marginal probabilities $P(y_d|x)$.

Loss	Labelled Data	Unlabelled Data
0-1	$\log P(w, y\|x)$	$\log \sum_{y \in Y} P(w, y, x)$
Hamming	$\sum_{d=1}^{D} \log P(w, y_d\|x_d)$	$\sum_{d=1}^{D} \sum_{y_d \in Y_d} q_d(y_d) \log P(w, y_d\|x_d)$
Weighted Hamming	$\sum_{d=1}^{D} \theta_d(y_d) \log P(w, y_d\|x_d)$	$\sum_{d=1}^{D} \sum_{y_d \in Y_d} q_d(y_d)\theta_d(y_d) \log P(w, y_d\|x_d)$

3.3.3 Marginal probability estimation

Either at testing time (to provide an estimate of the uncertainty of the model) or at training time (in the case of weak supervision, see Section 3.5), we need to compute marginal probabilities for a single variable y_d out of the d ones, that is,

$$P(y_d|x) = \sum_{y_{-d}} P(y_{-d}, y_d|x),$$

where y_{-d} is a sub-vector of y obtained by elimination of the variable y_d. Following Hazan and Jaakkola [2012] and Shpakova and Bach [2016], this can be obtained by taking m Gumbel samples and the associated maximizers $y^m \in Y = Y_1 \times \cdots \times Y_D$, and, for any particular d, counting the number of occurrences in each possible value in all the d-th components y_d^m of the maximizers y^m.

While this provides an estimate of the marginal probability, this is not an easy expression to optimize at it depends on several maximizers of potentially complex optimization problems. In the next section, we show how we can compute a different (and new) approximation which is easily differentiable and on which we can apply stochastic gradient descent.

3.4 Marginal Likelihood

In this section, we demonstrate the learning procedure for the element-decoupled losses. We consider the regularized empirical risk minimization problem in a general form:

$$\max_w \frac{1}{N} \sum_{n=1}^{N} \ell(w, x^n, y^n) - \frac{\lambda}{2}\|w\|^2, \tag{3.4.1}$$

where $\ell(w, x, y)$ can take various forms from Table 3.1 and λ is the regularization parameter. The choice of the likelihood form is based on the problem setting such as presence of missing data and the considered test-time evaluation function.

3.4.1 Hamming loss

The Hamming loss is a loss function that counts misclassification per dimension:

$$l_h(y, \hat{y}) = \frac{1}{D} \sum_{d=1}^{D} [y_d \neq \hat{y}_d].$$

For this type of loss instead of the classic log-likelihood objective it is more reasonable to consider the following decoupling representation from Table 3.1:

$$\ell(w, x, y) = \sum_{d=1}^{D} \log P(y_d | x, w), \tag{3.4.2}$$

where

$$\begin{aligned} P(y_d|x, w) &= \sum_{y_{-d}} \exp(f(w, y_{-d}|y_d, x) \; - \; A(f, x)) \\ &= \exp(B(f, y_d, x) - A(f, x)) \end{aligned}$$

is the marginal probability of the single element y_d given the entire input x, and

$$B(f, y_d) = \log \sum_{y_{-d}} \exp(f(w, y_{-d}|y_d)),$$

where $y_{-d} \in Y_1 \times \ldots Y_{d-1} \times Y_{d+1} \times \cdots \times Y_D$. Thus, the log-marginal probability may be obtained from the difference of two log-partition functions (which we will approximate below with Gumbel samples).

This idea of considering the marginal likelihood was proposed by Kakade et al. [2002]. Our contribution is to consider the approximation by "perturb-and-MAP" techniques. We thus have a new objective function $\ell(w, x, y)$:

$$\ell(w, x, y) = \sum_{d=1}^{D} \left[(B(f, x, y_d) - A(f, x)) \right],$$

and now the following approximation could be applied:

$$\begin{aligned} A(f) &\approx A_G(f) = \mathbb{E}_z \Big\{ \max_{y \in Y} \sum_{d=1}^{D} z_d(y_d) + f(y) \Big\}, \\ B(f|y_d) &\approx B_G(f|y_d) \\ &= \mathbb{E}_z \Big\{ \max_{y_{-d} \in Y_{-d}} \sum_{s=1:s \neq d}^{D} z_s(y_s) + f(y_{-d}|y_d) \Big\}. \end{aligned}$$

It is worth noting, that the approximation is not anymore an upper bound of the marginal likelihood; moreover it is a difference of convex functions. Remarkably, the objective function exactly matches the log-likelihood in the case of unary potentials (separable potential function) as the log-likelihood function becomes the sum of marginal likelihoods.

As noted, the objective $\ell(w, x, y)$ is not convex anymore, but it is presented as the difference of two convex functions. We can still try to approximate with stochastic gradient descent (which then only converges to a stationary point, typically a local minimum). Algorithm 2 describes the implementation details.

Algorithm 2 Double SGD for Marginal Likelihood

Input: dataset $\mathcal{D} = \{(x^n, y^n)\}_{n=1}^N$, number of iterations H, size of the mini-batch T, stepsize sequence $\{\gamma_h\}_{h=1}^H$, regularization parameter λ

Output: model parameters w

1: **Initialization**: $w = 0$
2: **for** $h = 1$ **to** H **do**
3: Sample data mini-batch of small size T (that is, T pairs of observations)
4: **for** t=1 **to** T **do**
5: Sample $z_d(y_d)$ as independent Gumbels for all $y_d \in Y_d$ and for all d
6: Find $y_A^* \in \arg\max_{y \in Y} \Big\{ \sum_{d=1}^{D} z_d(y_d) + f(y) \Big\}$
7: **for** d=1 **to** D **do**
8: Find $y_B^* \in \arg \max_{y_{-d} \in Y_{-d}} \{ \sum_{s=1:s \neq d}^{D} z_s(y_s) + f(y_{-d}|y_d) \}$
9: Make a gradient step:

$$w_{h+1} \to w_h + \gamma_h \Big(\Big\langle \langle \Psi(x, y_B^*) \rangle - \Psi(x, y_A^*) \Big\rangle - \lambda w_h \Big)$$

Acceleration trick. Interestingly we can use the same Gumbel perturbation realizations for approximating $A_G(f)$ and $B_G(f|y_d)$ through an empirical average. On the one hand, this restriction should not influence on the result as with a sufficient large averaging number M, $A_G(f)$ and $B_G(f|y_d)$ converges to their expectations. This is the same for stochastic gradients: on every iteration, we use a different Gumbel perturbation, but we share this one for the estimation of the gradients of $A_G(f)$ and $B_G(f|y_d)$. This allows us to save some computations as shown below, while preserving convergence (the extra correlation added by using the same samples for the two gradients does not change the unbiasedness of our estimates).

Moreover, if y_A^* has the same label value y_d as the ground truth, then the MAP inference problem for y_A^* exactly matches the one for y_B^* (with the element y_d fixed from the ground truth). Then $y_A^* = y_B^*$ and the corresponding difference of gradients gives zero impact into the gradient step. This fact allows us to reduce the number of MAP-inference problems. We should thus calculate y_B^* only for those indices d that leads to a mismatch between d-th label of y_A^* and the ground truth one. Remarkably, during the convergence to the optimal value, the reduction will occur more often and decrease the execution time with the number of iteration increase. Besides that, in the experiments with graph cuts in Section 3.6 we use dynamic graph cut algorithm for solving several optimization problems of similar structure (here D marginal probabilities calculation).

We describe it in more details in Section 3.4.3.

3.4.2 Weighted Hamming loss

The weighted Hamming loss is used for performance evaluation in the models, where each dimension has its own level of importance, e.g., in an image segmentation with superpixels, proportional to the size of superpixels. It differs from the usual Hamming loss in this way:

$$l_h(y, \hat{y}) = \frac{1}{D} \sum_{d=1}^{D} \theta_d(y_d)[y_d \neq \hat{y}_d].$$

Thus we consider a dimension-weighted model as it can be adjusted for the problem of interest that gives the model more flexibility. The optimization problem of interest is transformed from the previous case by weighted multiplication:

$$\ell(w, x, y) = \sum_{d=1}^{D} \theta_d(y_d) \left[(B(f, x, y_d) - A(f, x)) \right]. \tag{3.4.3}$$

To justify this objective function, we notice that in the case of unit weights, the weighted loss and objective function (3.4.3) match the loss and the objective from the previous section. Furthermore, y_d with a large weight $\theta_d(y_d)$ puts more importance towards making the right prediction for this y_d, and that is why we put more weight on the d-th marginal likelihood. This corresponds to the usual rebalancing used in binary classification (see, e.g., Bach et al. [2006] and references therein). Then, the algorithm for this case duplicated the one for the usual Hamming loss and the acceleration trick can be used as well.

3.4.3 Scalable algorithms for graph cuts

As a classical efficient MAP-solver for pairwise potentials problem we will use graph cut algorithms from Boykov and Kolmogorov [2004]. The function $f(y|x)$ should then be supermodular, i.e., with pairwise potentials, all pairwise weights of w should remain negative.

In both Sections 3.4.1 and 3.4.2 we can apply the dynamic graphcut algorithm proposed by Kohli and Torr [2007], which is a modification of the Boykov-Kolmogorov graphcut algorithms. It is dedicated to situations when a sequence of graphcut problems with slightly different unary potentials need to be solved. Then, instead of solving them separately, we can use the dynamic procedure and find the solutions for slightly different problems with less costs. This makes graphcut scalable for a special class of problems.

It can easily be seen that our sequence of problems

$$y_B^* \in \arg \max_{y_{-d} \in Y_{-d}} \{ \sum_{s=1:s \neq d} z_s(y_s) + f(y_{-d}|y_d) \}$$

for $d = 1, \ldots, D$ is a perfect application for the dynamic graph cut algorithm. At each iteration we solve T sets of graph cut problems, each of set contains $1 + a_t$ problems solvable by the same dynamic cut, where a_t is the number of not matched pixels between y_A^* and ground truth y^n. Finally, using acceleration trick and dynamic cuts we reduce the gradient descent iteration complexity from $\sum_{t=1}^{T}(1 + D_t)$ graphcut problems to T dynamic graph cut problems. We make the approach scalable and can apply it for large datasets.

3.5 Parameter Learning in the Semisupervised Setup

In this section we assume the presence of objects with unknown labels in the train dataset. We can separate the given data in two parts: fully annotated data $\mathcal{D}_1 = \{(x^n, y^n)\}_{n=1}^N$ as in the supervised case and unlabeled data $\mathcal{D}_2 = \{x^l\}_{l=1}^L$. Then, the optimal model parameter w is a solution of the following optimization problem:

$$\max_w L_1(w) + \kappa L_2(w) - \frac{\lambda}{2}\|w\|^2, \tag{3.5.1}$$

where $L_1(w) = \sum_{n=1}^N \ell_1(w, x^n, y^n)$, $L_2(w) = \sum_{l=1}^L \ell_2(w, x^l)$ and the parameter κ governs the importance of the unlabeled data. $\ell_1(w, x^n, y^n)$ can have a form from the left column of the Table 3.1, and $\ell_2(w, x^l)$ from the right one.

Marginal calculations. It is worth reminding from Section 3.3.3, that we can approximate marginal probabilities $q(y)$ of holding $y_d = k$ along with the partition function approximation almost for free. This can be obtained by taking m Gumbel samples and the associated maximizers $y^m \in Y = Y_1 \times \cdots \times Y_D$, and, for any particular d, counting the number of occurrences in each possible value in all the d-th components y_d^m of the maximizers y^m. The approximation accuracy depends on number of samples M. To calculate this we already need to have a trained weight vector w which we can obtained from the fully annotated dataset $\mathcal{D} = \{(x^n, y^n\}_{n=1}^N)$. We will calculate $q(y)$ for the unlabelled data $\mathcal{D}_2 = \{x^l\}_{l=1}^l$. Those marginal probabilities contain much more information than MAP inference for the new data as can be seen on the example in Figure 3-2. We believe that proper use of the marginal probabilities will help to gain better result than using labels from the MAP inference (which we observe in experiments).

It is worth noting that for the inference and learning phases we use a different number of Gumbel samples. During the learning phase, we incorporate the double stochastic procedure and use 1 sample per 1 iteration and 1 label. For the marginal calculation (inference) we should use large number of samples (e.g. 100 samples) to get accurate approximation.

We provide the sketch of the proposed optimization algorithm in Algorithm 3. The optimization of L_1 is fully supervised and this can be done with tools of the previous

Algorithm 3 Sketch for the semisupervised algorithm.

Input: fully annotated dataset $\mathcal{D}_1 = \{(x^n, y^n)\}_{n=1}^N$, number of iterations H, size of the mini-batch T, stepsize sequence $\{\gamma_h\}_{h=1}^H$, regularization param. λ

Output: model parameters w_1

Initialization: $w_1 = 0$

Find w_1 **via Algorithm 2**

Input: fully annotated dataset $\mathcal{D}_1 = \{(x^n, y^n)\}_{n=1}^N$, unlabeled dataset $\mathcal{D}_2 = \{x^l\}_{l=1}^L$, number of iterations H, size of the mini-batch T, stepsize sequence $\{\gamma_h\}_{h=1}^H$, regularization parameter λ

Output: model parameters $w_{1,2}$

Initialization: $w_{1,2} = w_1$

Calculate: $q(y)$ for unlabeled data via w_1

Find $w_{1,2}$ **via mixture of Algorithms 2 and 4**

section. The optimization of L_2 requires the specification of $\ell_2(w, x)$, which we take as

$$\ell_2(w, x) = \sum_{d=1}^{D} \sum_{y_d \in \{0,\dots,K\}} q_d(y_d) \log P(w, y_d | x_d) = \sum_{d=1}^{D} \sum_{y_d \in \{0,\dots,K\}} q_d(y_d) B(f | y_d) - DA(f),$$

that is, the average of the fully supervised cost function with labels generated from the model q. The term L_2 corresponds to the common way of treating unlabeled data through the marginal likelihood. The sub-algorithm for the optimization of ℓ_2 is presented as Algorithm 4.

Algorithm 4 Double SGD for Unsupervised Learning

Input: unlabeled dataset $\mathcal{D}_2 = \{x^l\}_{l=1}^L$, parameter estimate w_1, number of iterations H, size of the mini-batch T, stepsize sequence $\{\gamma_h\}_{h=1}^H$, regularization parameter λ

Output: model parameters $w_{1,2}$

Initialization: $w_{1,2} = w_1$

for $h = 1$ **to** H **do**

  Sample data mini-batch of small size T (that is, T pairs of observations)

  for t=1 **to** T **do**

    Sample $z_d(y_d)$ as independent Gumbels for all $y_d \in Y_d$ and for all d

    Find $y_A^* \in \arg\max_{y \in Y} \Big\{ \sum_{d=1}^{D} z_d(y_d) + f(y) \Big\}$

    for d=1 **to** D **and** k=0 **to** K **do**

      Find $y_{B,d,k}^* \in \arg \max_{y_{-d} \in Y_{-d}} \{ \sum_{s=1:s \neq d}^{D} z_s(y_s) + f(y_{-d} | y_d = k) \}$

  Make a gradient step:

$$w_{h+1} \to w_h + \gamma_h \Big(\Big\langle \langle \sum_{k=0}^{K} q_d(k) \Psi(x, y_{B,d,k}^*) \rangle - \Psi(x, y_A^*) \Big\rangle - \lambda w_h \Big)$$

Acceleration trick. Suppose, that y_d can take values in the range $\{0, \dots, K\}$. Again we use the same Gumbel perturbation for estimating $A_G(f)$ and $B_{dkG}(f|y_d = k)$ for all $k \in \{0, \dots, K\}$. The consequence of using the same perturbations is that if the d-th label y_d of y_A^* takes value k, than the corresponding d-th gradient will cancel out with one of the y_{Bk}^*. Thus, we will calculate only K (instead of $K+1$ labels) structured labels $y_{Bl}^*(l \neq k)$ and reduce the number of optimization problems to be solved. Dynamic graph cuts are applied here as well.

Finally in Table 3.1 we see the relationships between the proposed objective functions. Firstly, the known labels y^n in the supervised case are equivalent to the binary marginal probabilities $q(y^n) \in \{0,1\}^{D^n}$. Secondly, the unit weights $\theta_d(y_d) = 1$ in the weighted Hamming loss are equivalent to the basic Hamming loss.

Partial labels. Another case that we would like to mention is annotation with partial labels, e.g., in an image segmentation application, the bounding boxes of the images are given. Then denote y^{given} as the set of given labels. In this setup the marginal probabilities become conditional ones $q(y_d|y^{given})$ and to approximate this we need to solve several conditional MAP-inference problems. The objective function

$$\ell_2(w) = \sum_d \sum_{y_d \in \{0,\dots,K\}} q_d(y_d|y^{given}) \log P(w, y_d|x_d, y^{given})$$

remains feasible to optimize.

3.6 Experiments

The experimental evaluation consists of two parts: Section 3.6.1 is dedicated to the chain model problem, where we compare the different algorithms for supervised learning; Section 3.6.2 is focused on evaluating our approach for the pairwise model on a weakly-supervised problem.

3.6.1 OCR dataset

The given OCR dataset from Taskar et al. [2003] consists of handwritten words which are separated in letters in a chain manner, see examples in Figure 3-1. The OCR dataset contains 10 folds of $\sim$ 6000 words overall. The average length of the word is $\sim$ 9 characters. Two traditional setups of these datasets are considered: 1) "small" dataset when one fold is considered as a training data and the rest is for test, 2) "large" dataset when 9 folds of 10 compose the train data and the rest is the test data. We perform cross-validation over both setups and present results in Table 3.2.

As the MAP oracle we use the dynamic programming algorithm of Viterbi [1967]. The chain structure also allows us to calculate the partition function and marginal probabilities exactly. Thus, the CRF approach can be applied. We compare its performance with the structured SVM from Osokin et al. [2016], perturb-and-MAP (Hazan and Jaakkola [2012]) and the one we propose for marginal perturb-and-MAP (as Hamming loss is used for evaluation).

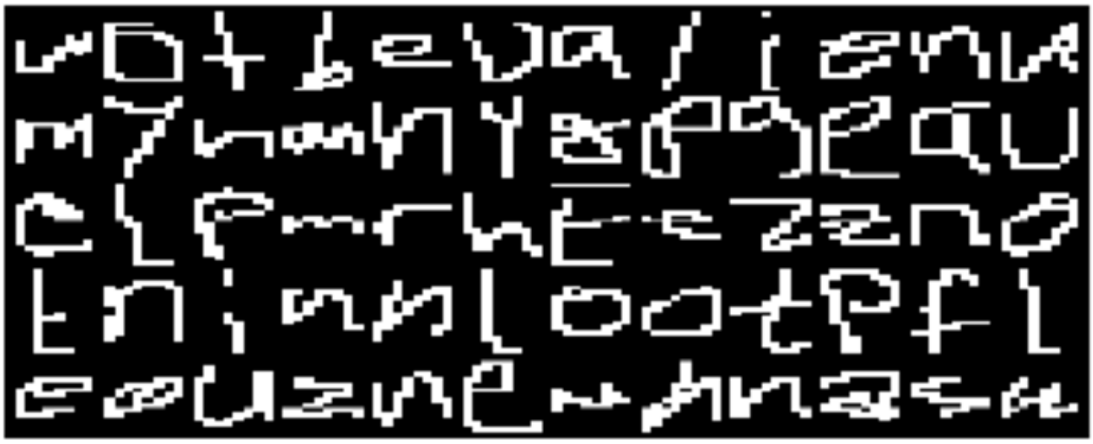


Figure 3-1: Samples from OCR dataset (Taskar et al. [2003]).

The goal of this experiment is to demonstrate that the CRF approach with exact marginals shows a slightly worse performance as the proposed one with approximated marginals but correct Hamming loss.

Table 3.2: OCR Dataset. Performance Comparison.

method	small dataset	large dataset
CRF	19.5 ± 0.4	13.1 ± 0.8
S-SVM+BCFW	19.5 ± 0.4	12.0 ± 1.1
perturb&MAP	19.1 ± 0.3	12.5 ± 1.1
marg. perturb&MAP	19.1 ± 0.3	12.8 ± 1.2

For the OCR dataset, we performed 10-fold cross-validation and the numbers of Table 3.2 correspond to the averaged loss function (Hamming loss) values over the 10 folds. As we can see from the result in Table 3.2, the approximate probabilistic approaches slightly outperforms the CRF on both datasets. The Gumbel approximation (with or without marginal likelihoods) does lead to a better estimation for the Hamming loss. Note that S-SVM performs better in the case of a larger dataset, which might be explained by stronger effects of model misspecification that hurts probabilistic models more than S-SVM (Pletscher et al. [2011]).

3.6.2 HorseSeg dataset

The problem of interest is foreground/background superpixel segmentation. We consider a training set of images $\{x^n\}_{n=1\ldots N}$ that contain different numbers of superpixels. A hard segmentation of the image is expressed by an array $y^n \in \{0,1\}^{D^n}$, where D^n is the number of superpixels for the n-th image.

The HorseSeg dataset was created by Kolesnikov et al. [2014] and contains horse images. The "small" dataset has images with manually annotated labels and contains 147 images. The second "medium" dataset is partially annotated (only bounding

Table 3.3: HorseSeg Dataset. Performance Comparison.

method	"small"	"medium"	"large"
S-SVM+BCFW	12.3	10.9	10.9
perturb&MAP	20.9	21.0	20.9
w.m. perturb&MAP	11.6	10.9	10.9

boxes are given) and contains 5974 images. The remaining "large" one has 19317 images with no annotations at all. A fully annotated hold out dataset was used for the test stage. It consists of 241 images.

The graphical model is a pairwise model with loops. We consider log-supermodular distribution and thus, the max oracle is available as the graph cut algorithm by Boykov and Kolmogorov [2004]. Note that CRFs with exact inference cannot be used here.

Following Kolesnikov et al. [2014], for the performance evaluation the weighted Hamming loss is used, where the weight is governed by the superpixel size and foreground/background ratio in the particular image.

That is,

$$l_h(y, \hat{y}) = \frac{1}{D} \sum_{d=1}^{D} \theta_d(y_d)[y_d \neq \hat{y}_d],$$

where

$$\theta_d(y_d) = \begin{cases} \dfrac{V_d}{2V_{foreground}}, & \text{if } \; y_d = 1. \\ \dfrac{V_d}{2V_{background}}, & \text{if } \; y_d = 0. \end{cases}$$

V_d is the size of superpixel d, $V_{background}$ and $V_{foreground}$ are the sizes of the background and the foreground respectively. In this way smaller object sizes have more penalized mistakes.

Since we incorporate $\theta(y)$ into the learning process and for its evaluation we need to know the background and foreground sizes of the image, this formulation is only applicable for the supervised case, where y_d is given for all superpixels. However, in this dataset we have plenty of images with partial or zero annotation. For these set of images $\mathcal{D}_2 = \{x^l\}_{l=1}^{L}$, we handle approximate marginal probabilities q_d^l associated to the unknown labels. Using them we can approximate the foreground and background volumes: $V_{foreground}^l \approx \sum_{d=1}^{D^l} q_d^l$ and $V_{background}^l \approx \sum_{d=1}^{D^l} (1 - q_d^l)$.

We provide an example of the marginal and MAP inference in Figure 3-2. The difference of the information compression between these two approaches is visually comparable. We believe that the smoother and accurate marginal approach should have a positive impact on the result, as the uncertainty about the prediction is well propagated.

As an example of the max-margin approaches we take S-SVM+BCFW from the paper of Osokin et al. [2016] which is well adapted to large-scale problems. For S-SVM+BCFW and perturb-and-MAP methods we use MAP-inference for labelling unlabelled data using w_1 (see Section 3.5).

(a) Original image (b) Marginal inference (c) MAP inference

Figure 3-2: Example of the marginal and MAP inference for an image from the HorseSeg database (Kolesnikov et al. [2014]).

For the HorseSeg dataset (Table 3.3), the numbers correspond to the averaged loss function (weighted Hamming loss) values over the hold out test dataset. The results of the experiment in Table 3.3 demonstrate that the approaches taking into account the weights of the loss $\theta_d(\cdot)$ (S-SVM+BCFW and w.m. perturb&MAP) give a much better accuracy than the regular perturb&MAP. S-SVM+BCFW uses loss-augmented inference and thereby augments the weighted loss structure into the learning phase. Weighted marginal perturb-and-MAP plugs the weights of the weighted Hamming loss inside the objective log-likelihood function. Basic perturb-and-MAP does not use the weights $\theta_d(\cdot)$ and loses a lot of accuracy. This shows us that the predefined loss for performance evaluation has a significant influence on the result and should be taken into account.

Small dataset size influence. We now investigate the effect of the reduced "small" dataset. We preserve the setup from the previous section and the only thing that we change is N, the size of the "small" fully-annotated dataset $\mathcal{D}_1 = \{(x^n, y^n)\}_{n=1}^N$. The new "small" dataset is 10% the size of the initial one, i.e., only 14 images. By taking a small labelled dataset, we test the limit of supervised learning when few labels are present.

For the HorseSeg dataset (Table 3.4), the numbers correspond to the averaged loss function (weighted Hamming loss) values over the hold out test dataset. The results of this experiments are presented in Table 3.4. In this setup, the probabilistic approach "weighted marginal perturb-and-MAP" gains more than max-margin "S-SVM+BCFW". This could happen because of very limited fully supervised data. The learned parameter w_1 gives a noisy model and this noisy model produces a lot of noisy labels for the unlabeled data, while weighted perturb-and-MAP is more cautious as it

Table 3.4: Reduced HorseSeg Dataset. Performance Comparison.

method	10% of "small"	"medium" with bbox	"medium" w/o bbox
S-SVM+BCFW	17.3	14.0	16.1
perturb&MAP	23.2	23.4	23.0
w.m. p.&MAP	18.1	13.7	14.4

uses probabilities that contain more information (see Figure 3-2).

Acceleration trick impact

We now compare the execution time of the algorithm with and without our acceleration techniques (namely Dynamic Cuts [DC] and Gumbel Reduction[GR]) to get an idea on how helpful they are. Table 3.5 shows the execution time for calculating all y_B^* (Algorithm 2) for different numbers of iterations on the HorseSeg small dataset. We conclude that the impact of DC does not depend on the total number of iterations always leading to acceleration around 1.3. For GR, acceleration goes from 3.5 for 100 iterations to 7.6 for one million iterations. Overall, we get acceleration of factor around 10 for one million iterations.

method \ it	100	10^3	10^4	10^5	10^6
basic	0.9	9.2	89.5	900	8993
DC	0.7	6.9	69.0	696	7171
GR	0.3	2.1	15.5	133.4	1186
DC+GR	0.2	1.5	10.9	83.5	727

Table 3.5: Execution time comparison in seconds. HorseSeg small dataset.

3.6.3 Experiments analysis

The experiments results mainly show that not taking into account the right loss in the learning procedure is detrimental to probabilistic technique such as CRFs, while taking it into account (our novelty) improves results. Also, Tables 3.2 and 3.3 show that the proposed methods achieves (and sometimes surpasses) the level performance of the max-margin approach (with loss-augmented inference).

Further, we observed that the size of the training set influences the SSVM and perturb-and-MAP approaches differently. For smaller datasets, the max-margin approaches tend to lose information due to usage of the hard estimates for the unlabelled data (e.g. in Table 3.4: 16.1 against 14.4 for "medium" dataset without bounding boxes labeling).

Table 3.4 reports an experiment about using weakly-labeled data at the training stage (the results on the partially annotated "medium" dataset). This experiment studied the impact on the final prediction quality of the training set of "medium" size

on top of the reduced "small" fully-labelled set. The results of Table 3.4 mean that the usage of our approach adopted to the correct test measure outperforms the default perturb-and-MAP by a large margin. Our approach also significantly outperformed the comparable baseline of SSVM due to reduced size of the "small" fully-labelled set.

3.7 Conclusion

In this chapter, we have proposed an approximate learning technique for problems with non-trivial losses. We were able to make marginal weighted log-likelihood for perturb-and-MAP tractable. Moreover, we used it for semi-supervised and weakly-supervised learning. Finally, we have successfully demonstrated good performance of the marginal-based and weighted-marginal-based approaches on the middle-scale experiments. As a future direction, we can go beyond the graph cuts and image segmentation application and consider other combinatorial problems with feasible MAP-inference, e.g., matching.

Chapter 4

Hyper-parameter Learning for Sparse Structured Probabilistic Models

Abstract

In this chapter we consider the estimation of hyperparameters for regularization terms commonly used for obtaining structured sparse parameters in signal estimation problems, such as signal denoising. By considering the convex regularization terms as negative log-densities, we propose approximate maximum likelihood estimation for estimating parameters for *continuous* log-supermodular distributions, which is a key property that many sparse priors have. We then show how "perturb-and-MAP" ideas based on the Gumbel distribution and efficient discretization can be used to approximate the log-partition function for these models, which is a crucial step for approximate maximum likelihood estimation. We illustrate our estimation procedure on a set of experiments with flow-based priors and signal denoising.

This chapter is based on the work "Hyper-parameter Learning for Sparse Structured Probabilistic Models", T. Shpakova, F. Bach, M. Davies, is under review for the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2019.

4.1 Introduction

Structured sparsity has emerged a versatile tool to go beyond plain parsimonious models. Indeed, taking into account the potential structure between the signal coefficients to be set to zero, both interpretability and predictive performance can be improved. Structured sparse priors can be handled in the various frameworks that have emerged for sparse methods, within convex optimization (Yuan and Lin [2006], Jacob et al. [2009]) or non-convex optimization (Baraniuk et al. [2010], He and Carin [2009], Huang et al. [2011]).

While encoding structure has several benefits, it comes with the extra task of

specifying a certain number of hyper-parameters, for example, for tree-structured sparsity, the weights to be given to each depth of the tree. The goal of this chapter is to propose data-driven estimation procedures for all of these hyper-parameters for a class of priors based on submodular functions. These include tree-structured priors or group-based priors and are commonly used in signal estimation problems (Kim and Xing [2010], Zhao et al. [2006]).

4.2 Contributions

In this chapter, we make the following *contributions*:

- We propose in Section 4.3 approximate maximum likelihood estimation for estimating parameters in *continuous* log-supermodular distributions, whose negative log-densities are commonly used as structured sparse priors in signal processing applications.
- We incorporate sparsity-inducing and challenging l_p norms treatment.
- We show in Section 4.4 how "perturb-and-MAP" ideas based on the Gumbel distribution and efficient discretization can be used to approximate the log-partition function for these models, which is the key step for approximate maximum likelihood estimation. Here, the fact that submodular functions can be efficiently minimized is crucial.
- We illustrate our estimation procedure in Section 4.5 on a set of experiments with flow-based priors and signal denoising.

4.3 Log-supermodular Distributions

As a probabilistic model, we are going to work with the family of log-supermodular distributions discussed by Djolonga and Krause [2014]. These distributions are a special case of a Gibbs distribution over some variable $x \in \mathcal{X}$, which could at this point be discrete or continuously valued:

$$dp(x) = \frac{e^{-f(x)}}{Z(f)} d\mu(x),$$

where $d\mu(x)$ is a base measure, and $f(x)$ is a potential function and the normalizer

$$Z(f) = \int_{\mathcal{X}} e^{-f(x)} d\mu(x).$$

It is worth noting that the partition function $Z(f)$ is intractable in the general case, continuous or discrete, and its handling constitutes the core computational difficulty of probabilistic inference (Wainwright and Jordan [2008b]). If the potential function $f(x)$ is submodular (see definition in the next section), then the distribution above is called log-supermodular (because $-f$ is supermodular).

We use these models as they cover a broad family of distributions and allow us to perform an efficient gradient descent optimization of the likelihood objective due to submodularity (see below).

4.3.1 Supermodular and Submodular Functions

Submodular functions can be defined on sets $\mathcal{X}$ which are products of intervals. These functions have the particular property to have polynomial-time minimization algorithms (Krause and Golovin [2014]). In this work, we consider the special cases $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{X} = \{0,1\}^n$, which will lead respectively to continuous and discrete submodular functions.

Discrete functions. A function $f : \{0,1\}^n \to \mathbb{R}$ can be uniquely identified to a *set-function* defined over subsets of $\{1,\dots,n\}$, by defining $F(A) = f(1_A)$ where $1_A \in \{0,1\}^n$ is the indicator vector of the set A. It is then said submodular if it satisfies the following diminishing return property $\forall A, B \subseteq \{1,\dots,n\}$ such that $A \subseteq B$ and for all i, then $F(A \cup i) - F(A) \geq F(B \cup i) - F(B)$. Classical submodular functions include network flows, graph cuts, as well as concave functions of the cardinality and appear in many areas of signal processing and machine learning (Fujishige [2005], Bach [2013]). In particular, non-decreasing submodular functions are commonly used to penalize the support of signal in compressed sensing (Obozinski and Bach [2012]). Classical examples include group-based priors (counting the number of active groups) or tree-based priors (cardinality of the smallest rooted tree containing a given set), which are commonly used in signal processing (Baraniuk et al. [2010]).

Continuous functions. In the continuous setting, a twice differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is submodular if for all $x \in \mathbb{R}^n$, the cross second-order derivatives are non-positive, that is, for all $i \neq j$, then $\frac{\partial^2 f}{\partial x_i \partial x_j}(x) \leqslant 0$. For non-differentiable functions, see more details in Bach [2016]. Important examples include the Lovász extensions of discrete submodular functions defined on $\{0,1\}^n$ (Bach [2013], Fujishige [2005]). These extended functions are typically used for regularization as follows: when a vector w whose support $A = \{i, w_i \neq 0\}$ should have a small value $F(A)$ where F is the corresponding set-function, then the penalty $f(|w|)$, where f is the Lovász extension, and the absolute values are taken component-wise, is a natural convex relaxation (Bach [2013], Obozinski and Bach [2012]). Classical example is the Lovász extension of a set cover function and it has the following form:

$$f(|w|) = \sum_{A \in \mathcal{A}} d_A \max_{a \in A} |w_a|,$$

where $\mathcal{A}$ is a set of subsets of $\{1,\dots,n\}$ (Bach et al. [2012]); see more details in Section 4.5.2. Other non-convex but tighter relaxations are of the form $f(|w|^p)$ where the power is taken component-wise (El Halabi et al. [2018]):

$$f(|w|^p) = \sum_{A \in \mathcal{A}} d_A \max_{a \in A} |w_a|^p.$$

By changing the free parameter p we can manipulate the desired level of sparsity

in some sense (as we know that the Laplacian prior does not really provide sparse samples (Davies and Gribonval [2009])).

4.3.2 Discrete Log-supermodular Distributions

An important example of a submodular minimization problem is a graphcut problem, which is very applicable in a variety of fields. For example it is popular to incorporate graphcut functions as potential functions in Markov random fields (MRFs) and perform an image segmentation as a side effect (Kolmogorov and Zabih [2004], Boykov and Kolmogorov [2004]). Other examples can be mutual entropy, certain functions of eigenvalues of submatrices and many others (Fujishige [2005]). Seeing them as log-densities thus leads to a probabilistic treatment (see, e.g., Djolonga and Krause [2014]), which similar to what we try to achieve in this paper for continuous distributions.

4.3.3 Continuous Log-supermodular Distributions

Several examples have been considered in other settings and then called "multivariate-totally positive of order 2", and include the multivariate logistic, Gamma distributions, as well as characteristic roots of random Wishart matrices (Karlin and Rinott [1980]). We propose to extend their use in signal processing, in order to learn parameters of the associated negative log-densities (which are submodular functions), in structured sparsity problems.

4.3.4 Log-partition Function for Bayesian Learning

Let us consider the following standard denoising problem, with $x = D\alpha + \varepsilon$, where ε is a Gaussian noise $\sim \mathcal{N}(0, \sigma^2 I)$. Given the noisy signal $x \in \mathbb{R}^n$ and a dictionary $D \in \mathbb{R}^{n\times k}$, we try to recover the initial representation $\alpha \in \mathbb{R}^k$. The decoding problem often has the following form:

$$\min_{\alpha} \frac{1}{2}\|x - D\alpha\|^2 + \Lambda(\alpha),$$

where $\Lambda(\alpha)$ serves as a regularizer. This formulation can be considered from **three** points of view: 1) as penalized least squares regression without any probabilistic meaning, as 2) *maximum a posteriori* (MAP) or 3) *minimum-mean-square-error* (MMSE) estimation within some probabilistic model (Gribonval [2011]). The probabilistic interpretation allows to learn parameters of $\Lambda(\alpha)$, which we aim to do here. The joint probability has the form:

$$P(x, \alpha) = P(\alpha)P(x|\alpha) = P(\alpha)P(\varepsilon = x - D\alpha),$$

where $P(\alpha) = \frac{\exp(-\Lambda(\alpha))}{Z}$ is a log-supermodular prior on α. Thus, we would like to parameterize and learn the prior $P(\alpha)$ from the training dataset before performing the denoising on the test dataset.

In a sparse set-up we would like to encourage the argument α to contain plenty of zero elements and thus we can use the usual ℓ_1-formulation (Chen et al. [2001]):

$$\min_{\alpha} \frac{1}{2}\|x - D\alpha\|^2 + \Lambda(\alpha),$$

$$s.t. \;\; \Lambda(\alpha) = f(|\alpha|),$$

where $|\alpha|$ is meant component-wise. Here $f(|\alpha|)$ is encoding the structured sparsity, see an example in the Section 4.5.2. In this paper, for simplicity, we consider only orthonormal dictionaries.

Maximum likelihood for parameter learning. The main goal here is to consider $\Lambda(\alpha)$ as a negative log-density with the functional parameter $f(\cdot)$ and to find its optimal data-dependent parameters by maximum likelihood. To do so, we need to solve an optimization problem that involves the log-partition function in the way described below. We use the notation, for a submodular function f defined on $\mathbb{R}^k_+$,

$$\log Z(f) = A(f) = \log \int_{\mathbb{R}^k_+} \exp(-f(\alpha))d\alpha.$$

Note that, by a simple change of variable, we have:

$$\log \int_{\mathbb{R}^k} \exp(-f(|\alpha|))d\alpha = k \log 2 + \log \int_{\mathbb{R}^k_+} \exp(-f(\alpha))d\alpha.$$

We thus have $-\log P(\alpha) = f(\alpha) + A(f) + k \log 2$. The next step is to go from $p = 1$ to $p < 1$. We can done the transformation by changing of variable $\beta = |\alpha|^p$ (component-wise):

$$\log Z(f) = \log \int_{\mathbb{R}^k} \exp(-g(|\alpha|^p))d\alpha = k \log 2 + \log \int_{\mathbb{R}_+} \exp(-g(\beta)) \prod_{i=1}^{k} \Big(\frac{1}{p}\beta_i^{1/p-1}\Big)d\beta.$$

which changes g to g plus some separable terms. As a result, g is submodular because it is obtained by a monotonic separable change of variable. Later in the discretization, this simply means adding a power p in the discretization.

In order to learn the distribution of α we need to perform maximum likelihood for the densities defined above. To perform this we need to approximate the log-partition function A, which is always convex, but usually hard to compute. Two questions arise: (1) approximation of $A(f)$, and (2) parameterization of f in a suitable form so that our optimization problems are easily solved.

4.4 Perturb-and-MAP

To perform an effective parameter learning, we would like to approximate the log-partition function $A(f)$ by making use of the "perturb-and-MAP" approach from Hazan and Jaakkola [2012]. One necessary point is to have an access to an efficient

MAP-oracle, e.g., a graphcut solver for our particular example of flow-based priors (see Section 4.5.2).

Our algorithm is based on an approximation result from Hazan and Jaakkola [2012], which states that for any real-valued function g defined on a discrete set $\mathcal{T} = \prod_{i=1}^{n} \mathcal{T}_i$, then

$$\log \sum_{t \in \mathcal{T}} e^{g(t)} \leqslant \mathbb{E}_{h_1,\ldots,h_n \sim \text{Gumbel}} \Big[\max_{t \in \mathcal{T}} \Big(g(t) + \sum_{i=1}^{n} h_i(t_i) \Big) \Big],$$

where Gumbel denotes the Gumbel distribution [1] and with a collection $\{h_i(t_i)\}_{t_i \in \mathcal{T}_i}^{i=1,\ldots,n}$ of independent Gumbel samples. This allows to define an upper-bound on the log-partition function, which is based on perturbing g and performing maximization; it is thus efficient only for functions g for which adding a separable terms leads to efficient optimization. This is exactly the case for negatives of submodular functions.

Via the proposed approximation we achieve a direct way of parameter learning in the discrete case. We can approximate the expectation over Gumbels using Monte-Carlo ideas by replacing the expectation by sampling. An efficient number of Gumbel samples depends on application, however in our experiments, $M = 100$ seems to be enough for most of the setups. Following Shpakova et al. [2018], in practice, when embedding the approximation result above in an optimization problem (for maximum likelihood) we can use stochastic gradient descent as a subroutine (Shpakova et al. [2018]), rather than using a fixed set of Gumbel samples.

4.4.1 Extension to the Continuous Case

To work with continuous data, we need to perform a discretization of the partition function to cast its calculation as a discrete optimization problem.

Approximation of $A(f) = \log \int_{\mathbb{R}_+^k} \exp(-f(\alpha)) d\alpha$. In order to approximate $A(f)$, we are going to discretize each α_i into r values $0 = u_0 < u_1 < \cdots < u_{r-1}$, and consider the measure on $\mathbb{R}_+$ define as the weighted sum of Diracs $\sum_{j=0}^{r-1} \pi_j \delta(\beta = u_j)$. A simple example (based on the trapezoidal rule) for the weights (π_j) is $\pi_0 = \frac{u_1 - u_0}{2}$, $\pi_{r-1} = \frac{u_{r-1} - u_{r-2}}{2}$, with the rest as $\pi_j = \frac{u_{j+1} - u_{j-1}}{2}$. We then discretize the integral in the following way

$$\hat{A}(f) = \log \sum_{z \in \{0,\ldots,r-1\}^k} \Big(\prod_{i=1}^{k} \pi_{z_i} \Big) \exp(-f(u_{z_i}, \ldots, u_{z_k})).$$

This is done by considering kr Gumbel variables $h_{i,j}$, $i \in \{1, \ldots, k\}$ and $j \in \{0, \ldots, r-1\}$, and the perturb-and-MAP approximation

$$\hat{A}_G(f) = \mathbb{E}_h \Bigg(\max_z \Big\{ -f(\cdot) + \sum_{i=1}^{k} h_{i,z_i} + \sum_{i=1}^{k} \log \pi_{z_i} \Big\} \Bigg).$$

1. The Gumbel distribution on the real line has the cumulative distribution function $F(z) = \exp(-\exp(-(z + c)))$, where c is the Euler constant.

The separable term $h_{i,z_i} + \log \pi_{z_i}$ does not change the nature of the problem and the function remains submodular (but now defined on a finite number of values).

4.4.2 Decoding with MMSE

In this section we discuss several ways of performing inference. Decoding with MAP is straightforward, as we can get the approximation with a discrete solver or an exact solution via divide-and-conquer algorithms (Groenevelt [1991]), if the objective is convex. Instead of using MAP decoder we can use MMSE, which is known to be the Bayesian optimal classifier for the ℓ_2-loss function (see, e.g., Gribonval [2011]). However, it is a challenge to calculate, and as one of our contributions in this work we propose a way of its approximation via perturb-and-MAP ideas. Indeed, we have the estimator

$$\psi_{MMSE}(x) = \mathbb{E}(\alpha|x)$$

$$P(\alpha|x) = \frac{P(x,\alpha)}{\sum_\alpha P(x,\alpha)} = \frac{\exp(-s(\alpha,x))}{Z(s)},$$

where $s(\alpha, x) = \frac{1}{2\sigma^2}||x - D\alpha||^2 + \Lambda(\alpha) = \frac{1}{2\sigma^2}||D^T x - \alpha||^2 + \Lambda(\alpha)$ for an orthonormal dictionary D. Then, the optimal estimator can be computed as:

$$\psi_{MMSE}(x) = \mathbb{E}(\alpha|x) \approx$$

$$\frac{1}{M}\sum_{m=1}^{M} \arg\min_{\alpha_m} \left\{ s(\alpha_m) - \sum_k h_k(\alpha_{km}) \right\},$$

as the expectation of the sufficient statistics is known to be a gradient of the log-partition function (Wainwright and Jordan [2008b]). Thus, we can get an approximate solution via approximate mean marginals and discretization.

4.5 Experiments

In the experiment section we illustrate the proposed parameter learning technique and compare the performance of two decoding approaches.

4.5.1 Synthetic Data. Experiments on decoding

We consider a one dimensional distribution $P(\alpha) = \frac{\exp^{-|\alpha|}}{Z}$. Closed-form solution for MMSE in discrete setup can then be calculated exactly for this one dimensional setup. We present denoising results for 10,000 randomly sampled data points in Table 4.1. We can clearly see that MMSE outperforms MAP in this synthetic setup, in a set-up where the Gumbel approximation of the log-partition function is exact.

Approach	ℓ_2 loss value
continuous MAP solution	0.693
MAP-oracle	0.744
MMSE exact	0.626
MMSE approx $[M = 100]$	0.632
MMSE approx $[M = 1000]$	0.628

Table 4.1: Denoising results. Synthetic data.

4.5.2 Flow-based priors

We now show examples where our maximization problems can be cast a a max-flow / min-cut. Following Sections 6.3 and 6.4 of Bach [2013], we consider prior $\Lambda(\alpha)$ as functions f of the form

$$f(\alpha) = \sum_{A \in \mathcal{A}} d_A \max_{a \in A} \alpha_a,$$

where d_A are parameters that we are interested of. In order to minimize the function $f(u_{z_1}, \ldots, u_{z_k}) + \sum_{i=1}^{k} v_{i,z_i}$, which is required in Section 4.4.1, following Ishikawa [2003], we can create a weighted directed graph where the (st)-minimum cut gives the optimal solution, thus making the optimization efficient.

4.5.3 Real Data. Experiments on the parameter learning and decoding.

Figure 4-1: Classical boat image for image denoising.

We work with patches 8×8 of the 'boat' image(see Figure 4-1). Half of them are considered as a train dataset to learn hyperparameteters of our priors, and another half as a test dataset. The original image is 512×512. After Gaussian noise was added to the test dataset, we try to denoise it with the proposed decoding techniques. We compare two ways of denoising: 1) model the prior directly for the pixels values and 2) model the prior for some wavelet coefficients.

We work under the following assumption

$$P(\alpha) = \frac{e^{-f(\alpha)}}{Z(f)},$$

where $f(\alpha) = \sum_{A \in \mathcal{A}} d_A \max_{a \in A} |\alpha_a|$, where $\mathcal{A}$ is a set of predefined groups of variables and α could be either the pixels values, either the wavelet coefficients. These groups could be based on the image grid or on the wavelet tree structure. Moreover, D is a orthogonal wavelet transform (we use Haar wavelets).

Before we report results in Tables 4.2 and 4.3, we first comment on the discretization grid which influences a lot the performance. We use a uniform grid with a step Δ (either 1 or 10). The quality and the running time consumption of the discrete MAP and MMSE depends on the discretization grid and on Δ correspondingly. For a fair comparison we spend the same amount of time for both: MAP and $\Delta = 1$ consume approximately the same amount of time as MMSE and $\Delta = 10$. For MMSE we use $M = 100$ Gumbel perturbations. The method "c. MAP" refers to the exact MAP solution achieved through direct continuous optimization via divide-and-conquer algorithm (Groenevelt [1991]). In Tables 4.2 and 4.3 we compute signal-to-noise ratios (SNR), where $\text{SNR} = 20 \log_{10} \frac{||x||_2}{||\hat{x}-x||_2}$ and x is a test image and $\hat{x}$ its prediction. We accompany the results with their standard deviation, across 10 different noise samples.

First set of experiments. We compare several models: "baseline", "unary" and "grid" models. "baseline" is a model with no learning and we set parameters d_A equal to zeros, "unary" corresponds to groups of size one (independent but not identically distributed variables), and "grid" corresponds to groups of size one and two in a grid manner.

method	baseline	unary	grid
c. MAP	22.33 ± 0.02	22.32 ± 0.02	26.16 ± 0.02
$\text{MAP}_{\Delta 10}$	21.95 ± 0.01	21.95 ± 0.01	25.36 ± 0.02
$\text{MAP}_{\Delta 1}$	22.32 ± 0.02	22.31 ± 0.02	26.16 ± 0.02
MMSE	22.31 ± 0.02	22.30 ± 0.01	25.99 ± 0.01
Lhood	356.4	355.7	253.5

Table 4.2: Primal approach.

Second set of experiments. The model "tree" corresponds to groups of size one and two that represent the wavelet quad-tree dependencies of two-dimensional Haar

wavelets.

method	baseline	unary	grid
c. MAP	22.33 ± 0.02	24.96 ± 0.01	24.66 ± 0.01
$\text{MAP}_{\Delta 10}$	21.98 ± 0.01	24.63 ± 0.08	24.44 ± 0.05
$\text{MAP}_{\Delta 1}$	22.33 ± 0.02	25.02 ± 0.03	24.72 ± 0.03
MMSE	22.29 ± 0.03	25.17 ± 0.02	25.15 ± 0.02
Lhood	480	185.5	183.0

Table 4.3: Wavelets approach.

Summary: From Tables 4.2 and 4.3, we can see that we are able to learn the structure. In the primal approach (directly on pixels), the Markov random field ("grid") performs best, while for the wavelet approach, unary potentials already work best (with little gains in likelihood and denoising performance for the "tree" approach). Note that MMSE sometimes outperform MAP, however it depends on the discretization grid.

4.6 Conclusion

We proposed a new parameter learning approach in the non-trivial structured and continuous setup. We demonstrated its performance for a denoising problem. We also propose a way to perform approximate MMSE estimation which is the optimal decoder for ℓ_2-loss evaluation. There is still some space for further investigation: non-uniform discretization grids, ℓ_p-looking norms such as done by Shervashidze and Bach [2015], where more heavy-tailed priors were considered.

Chapter 5

Conclusion and Future Work

5.1 Summary of the thesis

In this thesis, we focused on efficient parameter learning methods specific to various submodularity-based probabilistic models. We start our analysis with basic binary models, and expand it to discrete and continuous cases by the end of the thesis.

As our first contribution, we consider the problem of the partition function approximation for the specific log-supermodular distributions and provide some analysis. We presented two existed bounds and have shown that the L-Field bound by Djolonga and Krause [2014, 2015] is always inferior than the Gumbel bound by Hazan and Jaakkola [2012]. Then, we investigate the ability of these two approaches to perform approximate parameter learning under the maximum likelihood framework, and demonstrated that L-Field bound leads to a degenerate solution under linear parametrization, while the Gumbel approach can be applied successfully.

Another part of the thesis covers the learning rate guarantees. We bring the problem of parameter learning down to the optimization problem of a convex function and tackle it by gradient descent techniques. Our novelty is to incorporate the stochastic techniques over the Gumbel perturbations to accelerate the approach, thus, make it significantly faster. We can do either stochastic over Gumbels either double-stochastic over both data and Gumbels, if necessary.

Taking advantage of the probabilistic modelling, we also tackle the problem of partially missing data and propose an approach for parameter learning treatment. This extension does not conserve the convexity of the objective, but we use a local optimum as a solution.

We highlight that we extended the base approach into such directions as high scale and low levels of supervision. Besides that, we also covered various popular structured losses and reformulated the MLE objective in accordance with the picked loss function.

And the last part of the thesis contributes to the possible structure learning problem and sparsity-inducing solutions of the signal decoding problems. In this part we also move away from standard MAP inference solution and propose to deal with appropriate for squared loss MMSE approximation.

5.2 Perspectives

There are several possible directions that can be a prolongation of this work.

— Recently log-supermodular models with higher order potentials are gaining attention (Zhang et al. [2015], Tschiatschek et al. [2016]). They possibly can be modeled and tackled by our approach.

— Another attractive field is to consider an application of the other submodular functions with feasible MAP solvers besides graphcuts, e.g., matching (Taskar et al. [2005]). And in this framework we also can consider models, where MAP inference can be solved approximately, e.g., solvers based on linear programming relaxations (Sontag et al. [2008]).

— We can also try to tackle a dictionary learning problem altogether with prior learning in a expectation-maximization way (Mairal et al. [2009]). This can give promising results.

— For the continuous distribution support our treatment is still limited and this can be extended for wiser non-uniform grid approximations, as long as Monte Carlo integration.

— Even larger-scale application can be considered if we see our approach as a last layer of a neural network (Goodfellow et al. [2016]). We can also possibly achieve better accuracy, if we incorporate neural networks during the feature engineering process.

— We could incorporate non-convex optimization approaches for parameter learning in the unsupervised case.

Bibliography

F. Bach. Structured sparsity-inducing norms through submodular functions. In *Advances in Neural Information Processing Systems*, pages 118–126, 2010.

F. Bach. Learning with submodular functions: a convex optimization perspective. *Foundations and Trends in Machine Learning*, 6(2-3):145 – 373, 2013.

F. Bach. Submodular functions: from discrete to continuous domains. *Mathematical Programming*, pages 1–41, 2016.

F. Bach, D. Heckerman, and E. Horvitz. Considering cost asymmetry in learning classifiers. *Journal of Machine Learning Research (JMLR)*, 7:1713–1741, 2006.

F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Structured sparsity through convex optimization. *Statistical Science*, 27(4):450–468, 2012.

G. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan. *Predicting Structured Data*. The MIT Press, 2007.

R. G. Baraniuk, V. Cevher, M. F. Duarte, and C. Hegde. Model-based compressive sensing. *IEEE Transactions on Information Theory*, 56(4):1982–2001, 2010.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

E. Borenstein, E. Sharon, and S. Ullman. Combining top-down and bottom-up segmentation. In *Proc. ECCV*, 2004.

E. Boros and P. L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics (DAM)*, 123(1):155–225, 2002.

Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(9):1124–1137, 2004.

Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23 (11):1222–1239, 2001.

S. S. Chen, D. L Donoho, and M. A Saunders. Atomic decomposition by basis pursuit. *SIAM Review*, 43(1):129–159, 2001.

M. Davies and R. Gribonval. Restricted isometry constants where ℓ^p sparse recovery can fail for $0 < p \leq 1$. *IEEE Transactions on Information Theory*, 55(5):2203–2214, 2009.

J. Djolonga and A. Krause. From MAP to Marginals: Variational Inference in Bayesian Submodular Models. In *Adv. NIPS*, 2014.

J. Djolonga and A. Krause. Scalable Variational Inference in Log-supermodular Models. In *Proc. ICML*, 2015.

A. Doucet, N. De Freitas, and N. Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001.

M. El Halabi, F. Bach, and V. Cevher. Combinatorial penalties: Which structures are preserved by convex relaxations? In *International Conference on Artificial Intelligence and Statistics*, pages 1551–1560, 2018.

S. Fujishige. *Submodular Functions and Optimization*. Annals of discrete mathematics. Elsevier, 2005.

R. B. Girshick, P. F. Felzenszwalb, and D. A. Mcallester. Object detection with grammar models. In *Adv. NIPS*, 2011.

D. Golovin and A. Krause. Adaptive Submodularity: Theory and Applications in Active Learning and Stochastic Optimization. *Journal of Artificial Intelligence Research*, 42:427–486, 2011.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

R. Gribonval. Should penalized least squares regression be interpreted as maximum a posteriori estimation? *IEEE Transactions on Signal Processing*, 59(5):2405–2410, 2011.

H. Groenevelt. Two algorithms for maximizing a separable concave function over a polymatroid feasible region. *European Journal of Operational Research*, 54(2): 227–236, 1991.

J. M Hammersley and P. Clifford. Markov fields on finite graphs and lattices. 1971.

T. Hazan and T. Jaakkola. On the partition function and random maximum a-posteriori perturbations. In *Proc. ICML*, 2012.

T. Hazan, S. Maji, J. Keshet, and T. Jaakkola. Learning efficient random maximum a-posteriori predictors with non-decomposable loss functions. In *Adv. NIPS*, 2013.

L. He and L. Carin. Exploiting structure in wavelet-based bayesian compressive sensing. *IEEE Transactions on Signal Processing*, 57(9):3488–3497, 2009.

J. Huang, T. Zhang, and D. Metaxas. Learning with structured sparsity. *Journal of Machine Learning Research*, 12(Nov):3371–3412, 2011.

H. Ishikawa. Exact optimization for markov random fields with convex priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1333–1336, 2003.

E. Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik*, 31(1): 253–258, 1925.

L. Jacob, G. Obozinski, and J. Vert. Group lasso with overlap and graph lasso. In *Proceedings of the 26th annual international conference on machine learning*, pages 433–440. ACM, 2009.

M. Jerrum and A. Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM Journal on Computing*, 22(5):1087–1116, 1993.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

S. Kakade, Y. W. Teh, and S. T. Roweis. An alternate objective function for markovian fields. In *Proc. ICML*, 2002.

S. Karlin and Y. Rinott. Classes of orderings of measures and related correlation inequalities. i. multivariate totally positive distributions. *Journal of Multivariate Analysis*, 10(4):467–498, 1980.

S. Kim and E. P. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *ICML*, pages 543–550, 2010.

P. Kohli and P. H. Torr. Dynamic graph cuts for efficient inference in markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29(12):2079–2088, 2007.

P. Kohli, L. Ladicky, and P. H. S. Torr. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009.

A. Kolesnikov, M. Guillaumin, V. Ferrari, and C. H. Lampert. Closed-form training of conditional random fields for large scale image segmentation. In *Proc. ECCV*, 2014.

V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE transactions on pattern analysis and machine intelligence*, 28(10):1568–1583, 2006.

V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(2):147–159, 2004.

N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(3):531–552, 2011.

A. Krause and D. Golovin. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press, February 2014.

M. P. Kumar, B. Packer, and D. Koller. Self-paced learning for latent variable models. In *Adv. NIPS*, 2010.

S. Lacoste-Julien, M. Jaggi, M. Schmidt, and P. Pletscher. Block-coordinate Frank-Wolfe optimization for structural SVMs. In *Proc. ICML*, 2013.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*, 2001.

S. L. Lauritzen. *Graphical models*, volume 17. Clarendon Press, 1996.

H. Lin and J. Bilmes. A class of submodular functions for document summarization. In *Proc. NAACL/HLT*, 2011.

L. Lovász. Submodular functions and convexity. In *Mathematical Programming The State of the Art*, pages 235–257. Springer, 1983.

J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, pages 689–696. ACM, 2009.

K. P. Murphy and M. I. Weiss, Y.and Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 467–475. Morgan Kaufmann Publishers Inc., 1999.

K.P. Murphy. *Machine learning : a probabilistic perspective*. MIT Press, 2013.

S. Nadarajah and S. Kotz. A generalized logistic distribution. *International Journal of Mathematics and Mathematical Sciences*, 19:3169 – 3174, 2005.

A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

S. Nowozin and C. H. Lampert. Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 6(3–4):185–365, 2011.

G. Obozinski and F. Bach. Convex relaxation for combinatorial penalties. *arXiv preprint arXiv:1205.1240*, 2012.

A. Osokin and D. P. Vetrov. Submodular relaxation for inference in Markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TRAMI)*, 37(7):1347–1359, 2015.

A. Osokin, J.-B. Alayrac, I. Lukasewitz, P. K. Dokania, and S. Lacoste-Julien. Minding the gaps for block Frank-Wolfe optimization of structured SVMs. In *Proc. ICML*, 2016.

G. Papandreou and A. Yuille. Perturb-and-MAP random fields: Using discrete optimization to learn and sample from energy models. In *Proc. ICCV*, 2011.

S. Parise and M. Welling. Learning in Markov random fields: an empirical study. In *Joint Statistical Meeting (JSM)*, 2005.

P. Pletscher, S. Nowozin, P. Kohli, and C. Rother. Putting MAP back on the map. In *33rd Annual Symposium of the German Association for Pattern Recognition*, 2011.

R. B. Potts and C. Domb. Some generalized order-disorder transformations. *Proceedings of the Cambridge Philosophical Society*, 48:106, 1952.

C. Robert and G. Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical Programming*, 127(1):3–30, 2011.

N. Shervashidze and F. Bach. Learning the structure for structured sparsity. *IEEE Transactions on Signal Processing*, 63(18):4894–4902, 2015.

T. Shpakova and F. Bach. Parameter learning for log-supermodular distributions. In *Adv. NIPS*, 2016.

T. Shpakova, F. Bach, and A. Osokin. Marginal weighted maximum log-likelihood for efficient learning of perturb-and-map models. In *Proc. Uncertainty in Artificial Intelligence (UAI)*, 2018.

N. A. Smith. Linguistic structure prediction. *Synthesis Lectures on Human Language Technologies*, 4(2):1–274, 2011.

D. Sontag, T. Meltzer, A. Globerson, Y. Weiss, and T. Jaakkola. Tightening LP relaxations for MAP using message-passing. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 503–510, 2008.

C. Sutton and A. McCallum. Piecewise training of undirected models. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 568–575, 2005.

C. Sutton and A. McCallum. Piecewise pseudolikelihood for efficient training of conditional random fields. In *Proc. ICML*, 2007.

R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE transactions on pattern analysis and machine intelligence*, 30(6):1068–1080, 2008.

M. Szummer, P. Kohli, and D. Hoiem. Learning CRFs using graph cuts. In *Proc. ECCV*, 2008.

D. Tarlow, R.P. Adams, and R.S. Zemel. Randomized optimum models for structured prediction. In *Proc. AISTATS*, 2012.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. 2003.

B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd international conference on Machine learning*, pages 896–903. ACM, 2005.

S. Tschiatschek, J. Djolonga, and A. Krause. Learning probabilistic submodular diversity models via noise contrastive estimation. In *Proc. AISTATS*, 2016.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR)*, 6:1453–1484, 2005.

S.V.N. Vishwanathan, N. N. Schraudolph, M. W. Schmidt, and K. P. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In *Proc. ICML*, 2006.

A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13(2):260–269, 1967.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008a.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008b.

Y. Weiss. Comparing the mean field method and belief propagation for approximate inference in MRFs. *Advanced Mean Field Methods: Theory and Practice*, pages 229–240, 2001.

W. Wiegerinck and T. Heskes. Fractional belief propagation. In *Advances in Neural Information Processing Systems*, pages 438–445, 2003.

R. J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on information theory*, 51(7):2282–2312, 2005.

C.-N. J. Yu and T. Joachims. Learning structural SVMs with latent variables. In *Proc. ICML*, 2009.

J. Yu and M. Blaschko. Learning submodular losses with the lovász hinge. In *International Conference on Machine Learning*, pages 1623–1631, 2015.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

A. L. Yuille and A. Rangarajan. The concave-convex procedure. In *Adv. NIPS*, 2003.

J. Zhang, J. Djolonga, and A. Krause. Higher-order inference for multi-class log-supermodular models. In *Proc. ICCV*, 2015.

P. Zhao, G. Rocha, and B. Yu. Grouped and hierarchical model selection through composite absolute penalties. *Department of Statistics, UC Berkeley, Tech. Rep*, 703, 2006.

List of Figures

1-1 Undirected and directed graphical models on the same set of nodes V. 4
1-2 Example of a Linear-chain CRF. 4
1-3 Conditionally independent subsets X_A and X_B given the subset X_C. . 5
1-4 Grid-structured CRF for image segmentation task. 7
1-5 An example of the conditional probability calculation. The dark nodes correspond to the nodes on which we condition, light-shaded nodes are the nodes we marginalize over, and the white nodes are the nodes we want to calculate conditional probability for. 8
1-6 Hypercube representation for the ground set $V = \{a, b, c\}$. 11
1-7 Gumbel distribution cdf and pdf. 17

2-1 Comparison of log-partition function bounds for different values of c. See text for details. 28
2-2 Denoising of a horse image from the Weizmann horse database (Borenstein et al. [2004]). 32

3-1 Samples from OCR dataset (Taskar et al. [2003]). 48
3-2 Example of the marginal and MAP inference for an image from the HorseSeg database (Kolesnikov et al. [2014]). 50

4-1 Classical boat image for image denoising. 60

List of Tables

2.1 Supervised denoising results. 33
2.2 Unsupervised denoising results. Level of noise π is fixed. 33
2.3 Unsupervised denoising results. Level of noise π is not fixed. 33

3.1 Variants of the Objective Loss $\ell(w, x, y)$ Function. $\{\theta_d(y_d)\}_{d=1}^{D}$ are the weights of the weighted Hamming loss, $\{q_d(y_d)\}_{d=1}^{D}$ are the marginal probabilities $P(y_d|x)$. 41
3.2 OCR Dataset. Performance Comparison. 48
3.3 HorseSeg Dataset. Performance Comparison. 49
3.4 Reduced HorseSeg Dataset. Performance Comparison. 51
3.5 Execution time comparison in seconds. HorseSeg small dataset. . . . 51

4.1 Denoising results. Synthetic data. 60
4.2 Primal approach. 61
4.3 Wavelets approach. 62

Résumé

Les modèles graphiques probabilistes codent les dépendances entre les variables aléatoires et l'estimation des paramètres fait partie du traitement des modèles probabilistes. Ces modèles ont été utilisés dans des domaines tels que la vision par ordinateur, le traitement du signal, le traitement du langage naturel. Nous nous sommes concentrés sur les modèles log-supermodulaires, qui font partie des distributions familiales exponentielles, où la fonction potentielle est la fonction négative d'une fonction sous-modulaire. Malgré la restriction du modèle, est couvre une grande partie des familles exponentielles, car il ya beaucoup de fonctions qui sont sous-modulaires, par exemple, les coupes graphiques, entropie et autres. Le traitement probabiliste est habituellement difficile, mais nous avons été en mesure de relever certains des défis au moins approximativement.

Nous exploitons les idées perturb-and-MAP pour l'approximation des fonctions de partition et l'apprentissage efficace des paramètres. Nous proposons une méthode d'estimation et d'inférence approximative des paramètres pour les modèles où l'apprentissage et l'inférence exacts sont difficiles à gérer dans le cas général.

La première partie de la thèse est consacrée aux garanties théoriques. Étant donné les modèles log-supermodulaires, nous tirons parti de la propriété de minimisation efficace liée à la sous-modularité. En introduisant et en comparant deux limites supérieures existantes de la fonction de partition, nous démontrons leur relation en prouvant un résultat théorique. Nous introduisons une approche pour les données manquantes comme sous-routine naturelle de la modélisation probabiliste. Il semble que nous puissions appliquer une technique stochastique à l'approche d'approximation par perturbation et carte proposée tout en maintenant la convergence tout en la rendant plus rapide dans la pratique.

Une autre contribution est une généralisation efficace et évolutive de l'approche d'apprentissage des paramètres. Nous développons des algorithmes pour effectuer l'estimation des paramètres pour diverses fonctions de perte, différents niveaux de supervision et nous travaillons sur l'évolutivité. Nous incorporons également certaines techniques d'accélération.

Comme troisième contribution, nous abordons le problème général de l'apprentissage des signaux continus. Nous nous concentrons sur les représentations de modèles graphiques clairsemés et nous considérons les régularisateurs à faible densité comme des densités logarithmiques négatives pour la distribution antérieure. Les techniques de débruitage proposées ne nécessitent pas le choix d'un redresseur précis à l'avance. Pour effectuer un apprentissage de représentation clairsemée, la communauté du traitement du signal utilise souvent des pertes symétriques telles que ℓ_1, mais nous proposons de paramétrer la perte et d'apprendre le poids de chaque composante de perte à partir des données.

Nous avons effectué des expériences informatiques pour illustrer l'idée générale ou la comparer à des repères existants, et démontrer sa performance dans la pratique.

Abstract

Probabilistic graphical models encode hidden dependencies between random variables for data modelling. Parameter estimation is a crucial part of handling such probabilistic models. These very general models have been used in plenty of fields such as computer vision, signal processing, natural language processing. We mostly focused on log-supermodular models, which is a specific part of exponential family distributions, where the potential function is assumed to be the negative of a submodular function. This property is handy for maximum a posteriori and parameter learning estimations. Despite the apparent restriction of the model, is covers a broad part of exponential families, since there are plenty of functions that are submodular, e.g., graph cuts, entropy and others. Probabilistic treatment is challenging for most models, however we were able to tackle some of the challenges at least approximately.

In this manuscript, we exploit perturb-and-MAP ideas for partition function approximation and efficient parameter learning. Moreover, the problem can be also interpreted as a structure learning task, where each estimated parameter or weight represents the importance of the corresponding term. We propose a way of approximate parameter estimation and inference for models where exact learning and inference is intractable in general case due to the partition function calculation complexity.

The first part of the thesis is dedicated to theoretical guarantees. Given the log-supermodular models, we take advantage of the efficient minimization property related to submodularity. Introducing and comparing two existing upper bounds of the partition function, we are able to demonstrate their relation by proving a theoretical result. We introduce an approach for missing data as a natural subroutine of probabilistic modelling. It appears that we can apply a stochastic technique over the proposed perturb-and-map approximation approach and still maintain convergence while make it faster in practice.

The second main contribution is an efficient and scalable generalization of the parameter learning approach. In this section we develop new algorithms to perform parameter estimation for various loss functions, different levels of supervision and we also work on the scalability. In particular, working with mostly graph cuts, we were able to incorporate various acceleration techniques.

As a third contribution we deal with the general problem of learning continuous signals. We focus on the sparse graphical models representations. We consider common sparsity-inducing regularizers as negative log-densities for the prior distribution. The proposed denoising techniques do not require choosing any precise regularizer in advance. To perform sparse representation learning, the signal processing community often uses symmetric losses such as ℓ_1, but we propose to parameterize the loss and learn the weight of each loss component from the data. This is feasible via an approach which is similar to what we proposed in the previous sections.

For all aspects of the parameter estimation mentioned above we performed computational experiments to illustrate the idea or compare with existing benchmarks, and demonstrate its performance in practice.