Méthodes de sélection de voisinage pour la prévision à court-terme du trafic urbain

Julien Salotti

N° d'ordre NNT : 2019LYSEI077

Thèse de doctorat de l'Université de Lyon
opérée au sein de
L'INSA de Lyon

Ecole Doctorale N° 512
Mathématiques et Informatique (InfoMaths)

Spécialité / discipline de doctorat : Informatique

À soutenir publiquement par
Julien Salotti

Méthodes de sélection de voisinage et de prévision à court-terme pour l'analyse du trafic urbain

Devant le jury composé de: 24/09/2019

Elisa Fromont	Professeure des Universités	Université de Rennes	Rapporteure
Vincent Aguiléra	Docteur HDR	Direction des Routes Île-de-France	Rapporteur
Azedine Boulmakoul	Professeur des Universités	Université Hassan II de Casablanca	Examinateur
Peter Sturm	Directeur de Recherches	INRIA Grenoble Rhône-Alpes	Examinateur
Serge Fenet	Maître de Conférences	Université Claude Bernard Lyon 1	Co-encadrant
Christine Solnon	Professeure des Universités	INSA Lyon	Directrice de thèse
Nour-Eddin El Faouzi	Directeur de Recherches	Ifsttar - ENTPE	Co-directeur de thèse

Département FEDORA – INSA Lyon - Ecoles Doctorales – Quinquennal 2016-2020

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **http://www.edchimie-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage **secretariat@edchimie-lyon.fr** INSA : R. GOURDON	M. Stéphane DANIELE Institut de recherches sur la catalyse et l'environnement de Lyon IRCELYON-UMR 5256 Équipe CDFA 2 Avenue Albert EINSTEIN 69 626 Villeurbanne CEDEX **directeur@edchimie-lyon.fr**
E.E.A.	**ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE** **http://edeea.ec-lyon.fr** Sec. : M.C. HAVGOUDOUKIAN **ecole-doctorale.eea@ec-lyon.fr**	M. Gérard SCORLETTI École Centrale de Lyon 36 Avenue Guy DE COLLONGUE 69 134 Écully Tél : 04.72.18.60.97 Fax 04.78.43.37.17 **gerard.scorletti@ec-lyon.fr**
E2M2	**ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODÉLISATION** **http://e2m2.universite-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : H. CHARLES **secretariat.e2m2@univ-lyon1.fr**	M. Philippe NORMAND UMR 5557 Lab. d'Ecologie Microbienne Université Claude Bernard Lyon 1 Bâtiment Mendel 43, boulevard du 11 Novembre 1918 69 622 Villeurbanne CEDEX **philippe.normand@univ-lyon1.fr**
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTÉ** **http://www.ediss-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : M. LAGARDE **secretariat.ediss@univ-lyon1.fr**	Mme Emmanuelle CANET-SOULAS INSERM U1060, CarMeN lab, Univ. Lyon 1 Bâtiment IMBL 11 Avenue Jean CAPELLE INSA de Lyon 69 621 Villeurbanne Tél : 04.72.68.49.09 Fax : 04.72.68.49.16 **emmanuelle.canet@univ-lyon1.fr**
INFOMATHS	**INFORMATIQUE ET MATHÉMATIQUES** **http://edinfomaths.universite-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage Tél : 04.72.43.80.46 **infomaths@univ-lyon1.fr**	M. Luca ZAMBONI Bât. Braconnier 43 Boulevard du 11 novembre 1918 69 622 Villeurbanne CEDEX Tél : 04.26.23.45.52 **zamboni@maths.univ-lyon1.fr**
Matériaux	**MATÉRIAUX DE LYON** **http://ed34.universite-lyon.fr** Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bât. Direction **ed.materiaux@insa-lyon.fr**	M. Jean-Yves BUFFIÈRE INSA de Lyon MATEIS - Bât. Saint-Exupéry 7 Avenue Jean CAPELLE 69 621 Villeurbanne CEDEX Tél : 04.72.43.71.70 Fax : 04.72.43.85.28 **jean-yves.buffiere@insa-lyon.fr**
MEGA	**MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE** **http://edmega.universite-lyon.fr** Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bât. Direction **mega@insa-lyon.fr**	M. Jocelyn BONJOUR INSA de Lyon Laboratoire CETHIL Bâtiment Sadi-Carnot 9, rue de la Physique 69 621 Villeurbanne CEDEX **jocelyn.bonjour@insa-lyon.fr**
ScSo	**ScSo*** **http://ed483.univ-lyon2.fr** Sec. : Véronique GUICHARD INSA : J.Y. TOUSSAINT Tél : 04.78.69.72.76 **veronique.cervantes@univ-lyon2.fr**	M. Christian MONTES Université Lyon 2 86 Rue Pasteur 69 365 Lyon CEDEX 07 **christian.montes@univ-lyon2.fr**

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Table des matières

Liste des figures **vii**

Liste des tables **ix**

1 Introduction Générale **1**

2 Cadre théorique de l'apprentissage supervisé **7**
2.1 Notions de probabilités et statistique 8
2.1.1 Probabilités 8
2.1.2 Des probabilités aux statistiques 11
2.1.3 Relation entre plusieurs variables 13
2.2 Apprentissage supervisé 15
2.3 Séries temporelles 28
2.4 Discussion 31

3 Introduction à la prévision de trafic **33**
3.1 Approches par modèles de trafic 34
3.2 Modèles statistiques 36
3.2.1 Famille des modèles ARIMA 36
3.2.2 Modèle espace-état 37
3.3 Méthodes d'apprentissage artificiel 38
3.3.1 Approches des k plus proches voisins 38
3.3.2 Méthodes à noyaux 39
3.3.3 Réseaux de neurones artificiels 40
3.4 Différentes grilles de lectures 42

4 Méthodes de prévision **45**
4.1 Régression linéaire 46
4.2 Régression Ridge 48
4.3 Régression lasso 49
4.4 Combinaisons polynomiales de variables 50

4.5 Régression à vecteurs de support 51
4.5.1 Cas linéaire 51
4.5.2 Cas non-linéaire – utilisation du noyau 55
4.6 Régression k-NN 58
4.7 ARIMA 60
4.7.1 Modèle autorégressif AR(p) 60
4.7.2 Modèle moyenne mobile MA(q) 60
4.7.3 ARMA(p,q) 61
4.7.4 ARIMA(p,q,d) 62
4.8 Modèle Vecteur Autorégressif (VAR(p)) 63

5 Sélection de variables 65
5.1 `TiGraMITe` 67
5.2 Apprendre un graphe avec `Lasso` 73
5.3 Sélection à partir du graphe de dépendance 74

6 Résultats expérimentaux 79
6.1 Données 80
6.1.1 Jeu de données de Lyon 80
6.1.2 Jeu de données de Marseille 83
6.1.3 Différences entre trafic urbain et autoroutier 84
6.2 Mesures de performance 85
6.3 Comparaison des approches de prévision 87
6.3.1 Comparaison des méthodes de régression linéaire . . . 88
6.3.2 Comparaison des variantes SVR 89
6.3.3 Comparaison des variantes MLP 90
6.3.4 Comparaison des variantes k-`NN` 90
6.3.5 Comparaison des meilleures méthodes 94
6.4 Étude de l'impact de la résolution temporelle 99

7 Discussion 103
7.1 Synthèse 103
7.2 Limites et perspectives 105

Table des figures

2.1 Les différentes configurations du dilemme biais-complexité . . 22

4.1 Illustration de l'utilisation de la projection dans un nouvel espace. Dans cet exemple, on a $\Phi : \mathbb{R}^2 \to \mathbb{R}^2$. 57

5.1 Sélection de variables avec le graphe de dépendance (cas n° 1) 75
5.2 Sélection de variables avec le graphe de dépendance (cas n° 2) 76

6.1 Lyon – Mesures de débit sur un capteur de trafic pour une semaine (du lundi au dimanche) 81
6.2 Disposition de tous les capteurs pour la ville de Lyon 82
6.3 Disposition des capteurs pour le sous-ensemble des données nettoyées pour la ville de Lyon 83
6.4 Disposition des capteurs pour la ville de Marseille 84
6.5 Comparaison méthodes linéaires 88
6.6 Comparaison SVR 89
6.7 Comparaison MLP 91
6.8 Comparaison k-`NN` 94
6.9 Comparaison des meilleures méthodes 95
6.10 Étude de l'impact du pas d'agrégation sur la prévision 101

Liste des tableaux

6.1 Marseille - MASE 92
6.2 Lyon - MASE 93

Chapitre 1

Introduction Générale

« Quoi de pire que d'être coincé dans les bouchons ? » C'est la question que se posent de nombreux automobilistes sur leur trajet quotidien vers leur travail. Avec l'augmentation du nombre d'automobilistes et la concentration des habitants dans des zones urbaines, la congestion des infrastructures de transport est devenue un problème majeur dans le monde. Au coût économique et de qualité de vie lié au temps perdu par les usagers sur la route s'ajoute le coût écologique lié à la pollution et à la consommation de carburant qui se traduit par des émissions de gaz à effet de serre.

Face à ce problème, la mission des gestionnaires d'infrastructures est multiple : informer les usagers des conditions de trafic, anticiper les perturbations et la congestion, et proposer des actions en conséquence pour mitiger voir même résoudre ces problèmes.

Dans les dernières décennies sont apparus de nouveaux outils pour aider les gestionnaires à atteindre ces objectifs : les systèmes de transport intelligents (abrégés ITS en anglais). Ces derniers ont pu voir le jour grâce à l'essor des nouvelles technologies de l'information. Ils intègrent différents composants permettant de surveiller le réseau grâce à des capteurs ou de la vidéo par exemple, de gérer automatiquement certains systèmes comme les feux des carrefours et de fournir de l'information aux usagers, comme les assistants de navigation GPS, en s'appuyant parfois sur des modules prédictifs, pour fournir par exemple des estimations de temps de trajet.

Les applications d'assistance à la navigation, comme Waze, permettent de collecter une quantité massive d'information sur les usagers du réseau et fournissent en temps-réel des informations sur l'état du trafic de l'ensemble du réseau. Les données, considérées comme le nouvel or noir, sont un enjeu majeur pour les entreprises, qui s'appuient dessus pour développer de nouveaux produits innovants. La création et la possession de ces données est souvent restreinte à quelques entreprises qui dominent le marché et les revendent aux

autres acteurs souhaitant proposer des services autour de la donnée. Avoir la main sur la création et la distribution de la donnée est un enjeu majeur pour les gestionnaires d'infrastructure et plus largement les métropoles qui s'interrogent sur la question de la gouvernance des données.

La Métropole de Lyon a notamment mis en place dans le cadre de sa démarche « Lyon Métropole intelligente » la plateforme *Data Grand Lyon* qui permet entre autre l'utilisation libre et gratuite (open data) de nombreuses données. Dans le cadre du trafic, la ville continue d'installer de nombreux capteurs sur son réseau de transport afin de pouvoir surveiller en temps-réel son activité et créer une base de données historique. Les données ainsi collectées servent d'entrée à de nombreuses applications, dont la prévision du trafic. On distingue la prévision à long terme (typiquement 24h à l'avance) et la prévision à court terme (typiquement à moins d'une heure).

En quelques décennies, cette évolution technologique a considérablement modifié les objectifs dans le domaine de la prévision à court-terme. On est passé de l'observation localisée de quelques tronçons d'autoroute à la nécessité de produire des prévisions à l'échelle d'un réseau urbain complet. La méthodologie et les algorithmes utilisés sont donc amenés à évoluer pour répondre à ces nouveaux enjeux.

Une grande diversité de travaux de recherche ont été menés dans ce domaine, s'appuyant sur différentes spécialités comme les approches par simulation, l'étude statistique des séries temporelles, et différents champs de l'apprentissage artificiel, comme l'étude des réseaux de neurones ou des machines à vecteurs support. Les jeux de données étudiés varient également par le système considéré : autoroute, périphériques, grandes villes avec un réseau très structuré (comme le quartier de Manhattan) ou villes avec un centre historique, comme de nombreuses métropoles européennes. Enfin les dimensions des données (nombre de capteurs) et les variables de trafic considérées (vitesse, débit, temps de parcours, trajectoires) s'ajoutent à cette diversité. Il devient ainsi important de pouvoir dégager de cette littérature une connaissance permettant de choisir la méthode la plus appropriée au système que l'on souhaite étudier, en tenant compte de ses spécificités.

Un point important pour améliorer la qualité des modèles de prévision réside dans la sélection des variables utilisées pour faire la prévision. Dans le contexte de la prévision de trafic, on dispose généralement de données provenant d'un grand nombre de capteurs et une question délicate est : quels capteurs considérer pour prévoir le futur d'un capteur donné, et à quels horizons temporels ? Pour répondre à cette question, il est possible d'exploiter des connaissances sur la topologie du réseau de transport ou sur le trafic urbain, ou également d'utiliser des méthodes d'apprentissage artificiel.

L'évolution technologique a également permis de collecter des données

avec une résolution temporelle de plus en plus fine. Cependant, ces données présentent d'importantes fluctuations à haute fréquence qui comportent de l'information sur le phénomène de trafic mais également une part de bruit qui peut gêner l'apprentissage des modèles. La problématique du choix de la meilleure résolution temporelle est une question ouverte pour laquelle il n'existe pas de méthode de référence.

Un des objectifs de cette thèse est de pouvoir comparer sur les mêmes jeux de données, de nombreuses méthodes de prévision à court-terme provenant des différents champs de recherche cités précédemment. Disposant de données du réseau urbain de la Métropole de Lyon, et de données d'autoroutes urbaines de Marseille, l'objectif est également de pouvoir mettre en lumière les différences de performances des méthodes dans ces deux contextes différents, et de dégager les caractéristiques à l'origine de ces différences de performances. Nous étudions également l'intérêt de différents procédés de sélection de variables, dont l'objectif est d'apprendre sur un sous-ensemble plus pertinent des données, afin de simplifier les modèles et donc de rendre leur apprentissage et leur utilisation moins coûteuses. Enfin, nous proposons une étude de l'impact de la résolution temporelle des données sur la qualité des prévisions.

Le **chapitre deux** de ces travaux est une introduction aux grands principes de l'apprentissage automatique, que l'on retrouve sous différentes formes dans chaque algorithme de prévision basé sur l'historique. Il débute par un rappel de concepts importants de probabilités et de statistiques, qui forment le langage dans lequel nous décrirons ensuite les différents modèles. Puis, nous présentons les principes fondamentaux de l'apprentissage. Enfin, nous étendons les concepts de probabilités présentés précédemment à la notion plus spécifique de série temporelle, qui est le formalisme employé lorsqu'on étudie des phénomènes de nature temporelle, comme c'est le cas pour des données issues de capteurs de trafic. Le **chapitre trois** présente l'état de l'art des travaux de prévision du trafic à court-terme. Nous y présentons les différentes familles d'approches qui ont été utilisées pour résoudre ce problème et identifions deux problématiques majeures : la prévision à l'échelle d'un réseau urbain et la sélection d'un voisinage pertinent. Dans le **chapitre quatre**, les différentes méthodes de prévision que nous avons étudiées et comparées sont présentées en détail : des modèles classiques de séries temporelles (ARIMA, VAR), la régression Lasso, la méthodes des k plus proches voisins et la régression à vecteurs de support (SVR). Le **chapitre cinq** décrit les approches de sélection de variables et nous y détaillons les deux méthodes évaluées dans

la thèse : la sélection Lasso et `TiGraMITe`, une méthode issue de la physique et appliquée avec succès pour l'étude de phénomènes climatiques complexes. Le **chapitre six** présente les expérimentations effectuées et la comparaison des différentes approches de prévision, ainsi qu'une étude de l'intérêt des méthodes de sélection de variables. Il comporte également une étude de l'impact de la résolution temporelle des données sur la performance des méthodes de prévision. Enfin, nous présentons dans le **chapitre sept** une discussion afin d'identifier les contributions de la thèse, ainsi que les limites de ces travaux et les perspectives de recherche qui en découlent.

Liste des travaux et communications

Conférences internationales avec comité de lecture

- Julien Salotti, Serge Fenet, Romain Billot, Nour-Eddin El Faouzi, Christine Solnon. Comparison of traffic forecasting methods in urban and suburban context. *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI), Vólos, Grèce, 2018.*

Conférences nationales avec comité de lecture

- Julien Salotti, Serge Fenet, Romain Billot, Nour-Eddin El Faouzi, Christine Solnon. Comparaison de méthodes de prévision à court terme du trafic en milieux urbain et périurbain. *CAp (Conférence sur l'Apprentissage Automatique), Rouen, 2018.*

Communications pour des journées thématiques

- Julien Salotti, Serge Fenet, Romain Billot, Nour-Eddin El Faouzi, Christine Solnon. Comparaison de méthodes pour la prévision à court-terme du trafic urbain. *RFTM2018, Rencontres Francophones Transport Mobilité, Lyon, 2018.*

- Julien Salotti, Serge Fenet, Romain Billot, Nour-Eddin El Faouzi, Christine Solnon. Vers l'utilisation de graphes de liens causaux pour l'amélioration de la prévision court-terme du trafic routier. *RFP-IA 2016, Journée Transports Intelligents, Clermont-Ferrand, 2016.*

- Julien Salotti, Serge Fenet, Romain Billot, Nour-Eddin El Faouzi, Christine Solnon. Extract space-time dynamics from sensor network to build urban traffic prediction model : a machine learning point of

view *Urban Modelling Symposium : Towards Integrated Modelling of Urban Systems, Lyon, 2014.*

Chapitre 2

Cadre théorique de l'apprentissage supervisé

Sommaire

2.1 Notions de probabilités et statistique 8
2.1.1 Probabilités 8
2.1.2 Des probabilités aux statistiques 11
2.1.3 Relation entre plusieurs variables 13
2.2 Apprentissage supervisé 15
2.3 Séries temporelles 28
2.4 Discussion 31

Dans ce chapitre sont présentées au lecteur les notions mathématiques et les intuitions nécessaires pour comprendre les approches de prévision de trafic présentées par la suite. La tâche de prévision de trafic consiste à apprendre à partir d'un historique de mesures des modèles ou des lois permettant de prédire le trafic futur à partir d'un ensemble d'observations.

La théorie des probabilités est l'étude des phénomènes caractérisés par le hasard et l'incertitude. La statistique est l'étude d'un phénomène par la collecte de données, leur traitement et leur analyse. Le phénomène de trafic sur un réseau de transport comporte une grande part de hasard et d'incertitude. On peut observer et mesurer ce phénomène grâce à différentes technologies de collecte de données, comme les capteurs installés sur les routes. Il est ainsi naturel que les champs des mathématiques que sont les probabilités et la statistique soient au cœur des approches de prévision de trafic.

Les approches appartenant au domaine de l'apprentissage supervisé consistent à apprendre une fonction de prédiction à partir d'un ensemble d'exemples annotés. L'objectif étant de pouvoir utiliser cette fonction pour effectuer des

prévisions à partir de nouveaux exemples non annotés. Il existe une très grande diversité de modèles et de techniques d'apprentissage supervisé. L'objectif de ce chapitre est de présenter les grands principes qui sous-tendent cet apprentissage et que l'on retrouve nécessairement, sous des formes variées, dans les méthodes d'apprentissage supervisé. Les notions de probabilités et de statistique présentées au préalable constitueront le langage dans lequel nous exprimerons ces grands principes de l'apprentissage.

Enfin, le phénomène de trafic que l'on souhaite étudier présente la particularité d'être un phénomène temporel dont on souhaite comprendre l'évolution dans le temps. Pour décrire de tels phénomènes, des cadres théoriques et représentations spécifiques ont été développées. On présente donc en fin de chapitre les notions de processus stochastiques et de séries temporelles, qui seront utiles pour comprendre les modèles de prévision de trafic.

2.1 Notions de probabilités et statistique

Dans cette partie, nous allons introduire les notions de bases en probabilités et statistiques permettant de comprendre les travaux de la thèse. Il est possible de décrire très formellement et dans un cadre très général les concepts appartenant à la théorie des probabilités, mais cela sort du cadre de la thèse. L'objectif est de donner au lecteur l'intuition derrière ces notions, et les outils nécessaires pour comprendre les calculs présentés par la suite.

2.1.1 Probabilités

Variable aléatoire réelle. Une variable aléatoire X est une fonction qui décrit le résultat d'une expérience aléatoire par un nombre réel. Si l'ensemble des valeurs possibles est fini ou infini dénombrable, on parle de variable *discrète*. Si l'ensemble des valeurs possibles est infini, comme l'ensemble des réels, ou un intervalle sur les réels, on parle de variable *continue*.

Distribution de probabilité continue. Une distribution (ou loi) de probabilité associe une probabilité aux valeurs que peut prendre une variable aléatoire. Dans le cas d'une variable discrète, on peut associer une probabilité à chacune des valeurs possibles. Dans le cas d'une variable continue, la probabilité que la variable prenne une valeur précise est toujours nulle (car il y en a une infinité). On s'intéresse donc à la probabilité que la valeur appartienne à un certain intervalle.

Fonction de répartition. On peut décrire la loi de probabilité d'une variable aléatoire X, discrète ou continue, par sa fonction de répartition :

$$\begin{array}{rccl} F & : & \mathbb{R} & \to & [0,1] \\ & & x & \mapsto & \mathbb{P}(X \leq x) \end{array}$$

On peut donc voir que la probabilité que X prenne une valeur dans l'intervalle $[a, b]$ s'exprime de la manière suivante :

$$\mathbb{P}(a \leq X \leq b) = F(b) - F(a)$$

Densité de probabilité. Dans le cas d'une variable continue (au sens des probabilités), la fonction de répartition F est souvent continue et dérivable par morceaux. On peut alors s'intéresser à sa dérivée f que l'on appelle *densité de probabilité*. On peut donc réécrire les égalités précédentes :

$$\mathbb{P}(X \leq x) = F(x) = \int_{-\infty}^{x} f(x)\, dx$$

et

$$\mathbb{P}(X \in [a, b]) = \int_{a}^{b} f(x)\, dx$$

Espérance. L'espérance $\mathbb{E}[X]$ d'une variable aléatoire X est la valeur moyenne des observations que l'on s'attend à observer si on répète l'expérience aléatoire un grand nombre de fois.

Espérance d'une variable discrète. L'espérance d'une variable discrète est la somme des valeurs, pondérées par leur probabilité. Notons Ω l'ensemble des valeurs que peut prendre la variable. On note alors l'espérance comme suit :

$$\mathbb{E}[X] = \sum_{x \in \Omega} x\, \mathbb{P}(X = x)$$

Par exemple, considérons X comme le gain en euros obtenu dans un jeu de dé, dont les règles sont les suivantes : si on tombe sur 1, on perd 7 euros, si on tombe sur 2, on gagne 10 euros, sinon le gain est nul. Notons D la valeur du dé. On peut donc calculer l'espérance de gain de ce jeu :

$$\begin{array}{rcl} \mathbb{E}[X] & = & -7 \times \mathbb{P}(X = -7) + 10 \times \mathbb{P}(X = 10) + 0 \times \mathbb{P}(X = 0) \\ & = & -7 \times \mathbb{P}(D = 1) + 10 \times \mathbb{P}(D = 2) + 0 \times \mathbb{P}(D \in \{3, 4, 5, 6\}) \\ & = & -7 \times \frac{1}{6} + 4 \times \frac{1}{6} + 0 \times \frac{4}{6} \\ & = & \frac{1}{2} \end{array}$$

On en déduit donc que si l'on joue de nombreuses fois à ce jeu, on peut s'attendre à gagner, en moyenne, cinquante centimes par partie. À titre d'exemple, les jeux proposés dans les casinos sont généralement conçus de façon à ce que l'espérance de gain des joueurs soit négative (et donc que celle du casino soit positive).

Espérance d'une variable continue. Pour une variable X continue possédant une densité, on peut également calculer l'espérance. L'idée est la même que dans le cas discret, on remplace simplement la somme sur les valeurs possibles par une intégrale, et la probabilité d'une valeur par la fonction de densité :

$$\mathbb{E}[X] = \int_{\mathbb{R}} x f(x)\, dx$$

Variance. La variance d'une variable aléatoire X, notée $\mathrm{Var}(X)$, est définie comme l'espérance de l'écart au carré entre la variable X et son espérance $E[X]$:

$$\mathrm{Var}(X) = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right]$$

Il existe une identité remarquable entre l'espérance et la variance : le théorème de König-Huygens. Ce théorème permet de ré-écrire la variance sous une forme pouvant simplifier certains calculs :

$$\mathrm{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$$

Écart-type. L'écart-type d'une variable aléatoire X, noté σ_X, est la racine carrée de sa variance :

$$\sigma_X = \sqrt{\mathrm{Var}(X)}$$

L'écart-type est donc positif ou nul. Malgré l'apparente redondance avec la notion de variance (l'un est le carré de l'autre), les deux concepts sont utiles. L'écart-type présente l'avantage d'être exprimé dans la même unité que la variable X. La variance a des fondements mathématiques plus profonds (facilité de calcul, possibilité de la traduire en terme d'énergie, distance dans un espace euclidien, généralisable par la notion de *moment d'ordre supérieur*, etc.).

Position et dispersion. On peut parler de l'espérance comme d'un *indicateur de position* et de la variance comme d'un *indicateur de dispersion* de la distribution de probabilité suivie par une variable aléatoire.

Pour illustrer, imaginons que la variable X représente la note à un examen d'un élève appartenant à une classe. L'espérance $\mathbb{E}[X]$ correspond à la note moyenne que l'on s'attend à observer si l'on considère un grand nombre d'élèves de cette classe. Cela *positionne* la classe sur l'échelle des notes allant de zéro à vingt. Une autre classe sera représentée par une autre distribution dont l'espérance et la variance seront peut-être différentes. La variance représente l'écart au carré moyen par rapport à la moyenne. C'est donc un indicateur de la dispersion des notes. Plus la variance est faible, plus les notes seront concentrées autour de la moyenne. Plus elle est élevée, plus les notes seront dispersées.

Pour résumer notre exemple, l'espérance nous informe sur le niveau moyen de la classe, et nous permet donc de *positionner* plusieurs classes sur l'axe des notes. La variance permet de caractériser la *dispersion* des notes autour de la moyenne. Elle nous renseigne donc sur l'homogénéité du niveau des élèves.

2.1.2 Des probabilités aux statistiques

Nous avons jusqu'à présent introduit des notions de probabilités qui permettent de décrire le résultat d'une expérience aléatoire qui se déroule suivant certaines règles fixées et connues.

Maintenant, nous allons porter notre attention sur des notions de statistiques. En partant d'un ensemble d'observations du résultat d'une expérience aléatoire, nous souhaitons retrouver les règles qui régissent cette expérience. En d'autres termes, nous souhaitons estimer les caractéristiques de la distribution de probabilités suivie par la variable aléatoire qu'on observe.

Échantillon. On appelle *échantillon* un ensemble de réalisations d'une variable aléatoire. C'est à partir de cet ensemble d'observations que l'on va tenter d'étudier la distribution de probabilité.

Statistique. Ce paragraphe a pour objectif de présenter une homonymie qui introduit souvent de la confusion chez le lecteur. Les expressions « les statistiques » et « la statistique » font souvent référence au domaine mathématique que l'on vient d'introduire. Mais il existe aussi un concept mathématique bien précis que l'on appelle « une statistique ».

Une statistique est le résultat d'un calcul portant sur les valeurs contenues dans un échantillon. Par exemple, si on a observé n fois la variable X, on dispose d'un échantillon de n valeurs $(x_1, \ldots, x_n)$. Le résultat d'un calcul impliquant ces n valeurs est *une statistique*.

Estimateur. Soit X une variable aléatoire, et θ un paramètre de sa distribution (par exemple, son espérance ou sa variance). Un *estimateur* $\hat{\theta}$ du paramètre θ est une statistique sur un échantillon, dont on espère que la valeur soit proche du vrai paramètre. On peut donc définir plusieurs estimateurs pour un même paramètre, et leur qualité va dépendre des quatre propriétés suivantes :

Biais Le biais mesure la tendance de l'estimateur à sous-estimer ou surestimer la valeur du paramètre estimé. Il est défini comme suit :

$$\text{Biais}(\hat{\theta}) \equiv \mathbb{E}[\hat{\theta}] - \theta$$

Si cette valeur est nulle pour toute les valeurs possibles de θ, l'estimateur est *sans biais*. La valeur de l'estimateur dépend de l'échantillon aléatoire à partir duquel il est calculé, et on ne peut donc pas garantir qu'il sera toujours égal à θ. Le fait que l'estimateur soit *sans biais* nous indique qu'il sera en moyenne égal à θ, si on le calcule sur plusieurs échantillons.

Convergence Un estimateur $\hat{\theta}$ est convergent s'il tend en probabilité vers θ lorsque n (la taille de l'échantillon) augmente. Autrement dit, c'est le cas si la probabilité que $\hat{\theta}$ s'éloigne de θ tend vers zéro quand $n \to \infty$.

Efficacité Un estimateur est calculé sur un échantillon tiré au hasard. On peut donc s'intéresser à sa variance $\text{Var}(\hat{\theta})$ sur l'ensemble des échantillons possibles. Plus cette valeur est faible, moins les fluctuations sont importantes. Un estimateur avec une faible variance – ce qui est souhaitable – est dit *efficace*.

Robustesse Un estimateur est *robuste* si sa valeur n'est pas trop impactée par la présence d'une valeur extrême et peu probable dans l'échantillon.

Deux estimateurs classiques sont la moyenne et la variance empiriques :

— La moyenne empirique est un estimateur sans biais convergent de l'espérance. Pour un échantillon de n observations de X, on a donc :

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

— La variance empirique est un estimateur de la variance, basé sur la formule classique de la variance :

$$s_n = \frac{1}{n} \sum_{i=1}^{n} (x_i^2 - \overline{x}^2)$$

Mais cet estimateur est en réalité biaisé. On lui préfère donc souvent en pratique sa version corrigée, qui est un estimateur sans biais convergent :

$$\hat{\sigma}^2 = \frac{n}{n-1} s_n = \frac{1}{n-1} \sum_{i=1}^{n} (x_i^2 - \overline{x}^2)$$

2.1.3 Relation entre plusieurs variables

Jusqu'à présent, nous avons parlé de l'étude d'une variable aléatoire, mais il est fréquent que l'on observe plusieurs phénomènes aléatoires à la fois, et que l'on souhaite savoir si les résultats de certains d'entre eux ont une influence sur les autres. Pour ne pas alourdir, on considère uniquement deux variables continues, mais ces notions peuvent s'étendre pour des variables discrètes et un nombre de variables plus grand que deux.

Généralement, on donne les formules avec deux variables X et Y. Mais par soucis de cohérence avec les notations des futurs chapitres, nous nommerons les variables X_1 et X_2.

Fonction de répartition à deux variables. On peut étendre la notion de fonction de répartition pour les distributions de probabilités à plusieurs variables assez naturellement :

$$F_{X_1X_2}(x_1, x_2) = \mathbb{P}(X_1 < x_1, X_2 < x_2)$$

Densité de probabilité jointe. Si la densité de probabilité jointe existe, on l'obtient en dérivant la fonction de répartition une fois par rapport à chaque variable.

$$f_{X_1X_2}(x_1, x_2) = \frac{\partial F_{X_1X_2}}{\partial x_1 \partial x_2}(x_1, x_2)$$

Densité de probabilité marginale. La densité de probabilité marginale d'une variable X_1 caractérise sa distribution de probabilité. Elle donne la probabilité de ses valeurs sans référence aux valeurs de X_2, contrairement à la densité de probabilité jointe qui caractérise comment les valeurs des deux variables dépendent l'une de l'autre.

$$f_{X_1}(x_1) = \int_{\mathbb{R}} f_{X_1X_2}(x_1, x_2)\, d_{x_2}$$

Intuitivement, cette intégrale calcule le total de probabilité de tous les couples (x_1, x_2) possibles avec x_1 fixée et pour toutes les valeurs x_2 possibles.

Densité de probabilité conditionnelle. La densité de probabilité conditionnelle de X_1 sachant que $X_2 = x_2$ s'écrit comme suit :

$$f_{X_1|X_2}(x_1, x_2) = \frac{f_{X_1X_2}(x_1, x_2)}{f_{X_2}(x_2)}$$

Ce concept nous permet de répondre à la question suivante : « Si j'ai observé que la variable X_2 vaut x_2, quelle est la probabilité que la variable X_1 prenne la valeur x_1 ? ». La notion de probabilité conditionnelle permet donc de décrire comment la connaissance de la variable X_2 modifie notre croyance sur les valeurs possibles pour X_1, ou du moins sur leur probabilité associée. Par exemple : si on sait que le sol est mouillé, notre croyance dans le fait qu'il pleut augmente.

Indépendance entre variables. Les variables X_1 et X_2 sont indépendantes si la probabilité conditionnelle est égale à la probabilité marginale :

$$\forall x_2, \quad f_{X_1}(x_1) = f_{X_1|X_2}(x_1, x_2)$$

Cela signifie donc que la connaissance de la valeur d'une des variables ne nous fournit pas d'information sur les probabilités des valeurs de l'autre variable. On peut réécrire cette égalité sous la forme suivante :

$$f_{X_1X_2}(x_1, x_2) = f_{X_1}(x_1) \times f_{X_2}(x_2)$$

Covariance. La covariance est une mesure de dépendance entre deux variables. Elle mesure si les deux variables varient proportionnellement par rapport à leurs espérances respectives :

$$\mathrm{Cov}(X_1, X_2) = \mathbb{E}\left[(X_1 - \mathbb{E}[X_1])(X_2 - \mathbb{E}[X_2])\right]$$

Si les variables X_1 et X_2 sont indépendantes, leur covariance $\mathrm{Cov}(X_1, X_2)$ est nulle. En effet, l'indépendance implique que leur variations respectives ne sont aucunement liées.

La réciproque n'est pas toujours vraie. Des variables peuvent avoir une covariance nulle sans être indépendantes. Cela provient du fait que la covariance ne détecte que le fait que deux variables s'éloignent de leur espérance de façon plus ou moins proportionnelle. Considérons par exemple une variable X_1 et son carré $X_2 = X_1^2$. Il existe une relation totalement déterministe entre les deux, elle sont donc dépendantes : connaître X_1 permet de connaître immédiatement X_2. Supposons maintenant les deux propriétés suivantes :

$\mathbb{E}[X_1] = 0$ et $\mathbb{E}[X_1^3] = 0$ et calculons la covariance de X_1 et X_2 :

$$\begin{aligned}\mathrm{Cov}(X_1, X_2) &= \mathrm{Cov}(X_1, X_1^2) \\ &= \mathbb{E}[X_1 X_1^2] - \mathbb{E}[X_1]\,\mathbb{E}[X_1^2] \\ &= \mathbb{E}[X_1^3] - \mathbb{E}[X_1]\,\mathbb{E}[X_1^2] \\ &= 0 - 0 \times \mathbb{E}[X_1^2] \\ &= 0\end{aligned}$$

La covariance est donc capable de capturer une relation *linéaire* entre deux variables. Mais si cette dépendance est plus complexe, il est possible qu'elle ne soit pas détectée.

Estimateur de covariance. La covariance entre deux variables peut être estimée à partir d'un échantillon, par l'estimateur suivant :

$$\widehat{\mathrm{Cov}}(X_1, X_2) = \frac{1}{n-1} \sum_{i=1}^{n} (x_{1,i} - \overline{x_1})(x_{2,i} - \overline{x_2})$$

Corrélation. La corrélation est une version normalisée de la covariance. En effet, cette dernière présente deux défauts : l'unité dans laquelle elle est exprimée est le produit des unités de X_1 et X_2, ce qui n'est pas toujours simple à interpréter, et surtout les valeurs de la covariance dépendent des valeurs prises par les variables X_1 et X_2.

Le coefficient de *corrélation*, au contraire, est sans unité et ses valeurs sont comprises entre -1 et 1. Il correspond à la covariance divisée par le produit des variances respectives de X_1 et X_2 :

$$\mathrm{Cor}(X_1, X_2) = \frac{\mathrm{Cov}(X_1, X_2)}{\sqrt{\mathrm{Var}(X_1)\,\mathrm{Var}(X_2)}}$$

Il est possible de l'estimer de la manière suivante (coefficient de corrélation de Pearson) :

$$r_{X_1 X_2} = \frac{1}{n-1} \sum_{i=1}^{n} \frac{(x_{1,i} - \overline{x_1})}{s_{X_1}} \frac{(x_{2,i} - \overline{x_2})}{s_{X_2}}$$

2.2 Apprentissage supervisé

L'apprentissage artificiel (ou apprentissage automatique) est un domaine de l'intelligence artificielle qui étudie les algorithmes ou modèles statistiques

qui permettent à un ordinateur de réaliser une tâche spécifique sans un ensemble d'instructions explicites, mais en s'appuyant sur la découverte de motifs et sur le principe d'inférence.

On présente à l'algorithme des données d'apprentissage, à partir desquelles il va apprendre un modèle mathématique permettant de prendre des décisions ou effectuer des prédictions sans qu'on l'ait programmé explicitement pour effectuer cette tâche. Suite à cette phase d'apprentissage, le modèle appris est utilisé dans une phase de production pour réaliser la tâche à partir de nouvelles données d'entrée.

On trouve de nombreuses applications d'apprentissage artificiel pour les tâches humaines que nous savons effectuer sans pour autant savoir décrire explicitement comment on les réalise. Les champs d'applications sont très diversifiés : on peut citer le domaine de la vision par ordinateur, avec notamment la reconnaissance de caractères, la classification d'images, la détection d'objets ou de personnes dans une vidéo, ou encore le traitement naturel du langage, avec par exemple la traduction de documents ou la reconnaissance vocale. Les algorithmes de recommandation utilisés dans les plate-formes d'hébergement de vidéos, de réservation de logements ou de billets d'avions appartiennent également à ce domaine. L'application de prévision de trafic appartient au domaine de la prévision de séries temporelles, pour lequel on retrouve des applications dans la finance ou l'étude des phénomènes climatiques par exemple.

On distingue différents types d'apprentissage artificiel, selon le type de données utilisées ou le type de tâche que l'on apprend. Les algorithmes d'apprentissage supervisé construisent un modèle mathématique à partir de données étiquetées, qui contiennent l'entrée et la sortie désirée du modèle. Le modèle peut donc être entraîné pour diminuer son erreur sur les exemples de sortie. Les algorithmes d'apprentissage non-supervisé apprennent à partir de données d'entrées, et découvrent de la structure dans ces données, par exemple des groupes d'entrées similaires (*clustering*). Ces algorithmes ne sont pas guidés par des données étiquetées décrivant la sortie attendue pour une entrée donnée. Les algorithmes d'apprentissage par renforcement s'intéressent au comportement d'agents logiciels, qui peuvent effectuer des actions dans un environnement. L'agent est récompensé (ou puni) pour le résultat de ses actions et l'objectif est d'apprendre à maximiser une notion de gain cumulé sur ses actions.

Les algorithmes de prévision de trafic présentés dans la thèse sont des algorithmes d'apprentissage supervisé : à partir de l'historique, on peut fournir en entrée des valeurs passées et en sortie les valeurs que le modèle doit pré-

voir. Nous présentons donc plus en en détail dans cette section les concepts de l'apprentissage supervisé.

Notations Les notations utilisées ici ont été largement reprises du livre *Element of Statistical Learning* [36] et des supports du cours du professeur Stéphane Mallat au Collège de France, intitulé « science des données » [70]. Ces deux sources ont également influencé les choix de présentation pour les différents concepts d'apprentissage statistique présentés dans cette section.

Notations génériques pour les variables :
— X : variable aléatoire explicative ;
— Y : variable aléatoire expliquée ;
— X_j : si X est un vecteur, X_j est son j^{e} élément.

Les observations des variables sont écrites en minuscules :
— x_i : la i^{e} observation de X. C'est un scalaire ou un vecteur.

Les matrices sont écrites en **gras** :
— $\mathbf{X}$: un ensemble de N vecteurs x_i (avec $i = 1, \ldots, N$), chacun de dimension d, est représenté par une matrice $\mathbf{X}$ de dimension $N \times d$.

Les vecteurs ne sont pas écrits en gras sauf pour lever une ambiguïté entre :
— x_i : le vecteur de taille d correspondant à la i^{e} observation de X ;
— $\mathbf{x}_j$: le vecteur contenant les N observations de la variable X_j.

Par convention, les vecteurs sont toujours des colonnes, donc la i^{e} ligne de $\mathbf{X}$ est notée x_i^T, le vecteur transposé de x_i.

Les notations sont coiffées d'un accent circonflexe (par ex : $\hat{y}$) lorsque l'on fait référence à une valeur qui a été estimée ou prédite, pour la différencier de la vraie valeur théorique ou observée.

Classification versus régression. L'objectif de l'apprentissage supervisé est d'apprendre un algorithme – ou un modèle – capable de donner la réponse Y à une question portant sur des données X. Les données X et Y sont toutes les deux des variables aléatoires. On peut nommer les variables comme suit : X est la variable *explicative* et Y la variable *expliquée*. On peut également dire que X est la *variable d'entrée* de l'algorithme et que Y en est la variable de *sortie*.

Une variable peut être soit *quantitative*, si elle représente une quantité, soit *catégorielle* ou *qualitative*, si ses valeurs sont des catégories qui ne représentent pas de quantité. Par exemple, le kilométrage d'une voiture est une variable quantitative, tandis que la marque d'une voiture est une variable catégorielle.

Lorsque la variable Y que l'on souhaite prédire est catégorielle, on parle de *classification*. C'est par exemple le cas dans la classification d'images, où X est la valeur de chaque pixel de l'image et Y est une catégorie, par exemple *chat*, *chien*, *voiture*, etc.

Dans un problème de *régression*, la variable que l'on souhaite prédire est quantitative. Par exemple, on peut souhaiter apprendre à prédire le prix de vente Y d'une voiture à partir de ses caractéristiques X : marque, modèle, année de production, kilomètres parcourus, etc.

Dans cette thèse, nous nous concentrerons sur des problèmes de régression.

Données d'apprentissage. Pour apprendre un algorithme qui répondra correctement aux questions qu'on lui pose, nous allons l'entraîner en lui présentant un ensemble de questions, mais également les réponses, qui sont connues.

Pour reprendre l'exemple précédent, on dispose d'un historique d'un certain nombre d'exemples de transactions déjà effectuées, pour lesquelles on connaît les caractéristiques de la voiture, ainsi que le prix auquel elle a été vendue.

Si on dispose de n exemples, les données d'apprentissage sont représentées par l'ensemble $\{(x_i, y_i)\}_{i \leq n}$. Dans notre exemple x_i correspond aux caractéristiques de la i^{e} voiture et y_i est le prix auquel elle a été vendue. Considérons la variable $X = (\dots, X_j, \dots)_{j \leq d}$, elle-même composée de d variables quantitatives. Alors $x_i = (x_{i1}, \dots, x_{id})$ est un vecteur de dimension d décrivant la i^{e} voiture et y_i est un scalaire représentant son prix.

Approximation de fonction. On peut voir le problème de régression comme un problème d'approximation de fonction. Tout d'abord, considérons $X \in \mathbb{R}^d$, un vecteur aléatoire d'entrée, et $Y \in \mathbb{R}$ la variable aléatoire de sortie, ainsi que leur distribution jointe $P(X, Y)$. Trouver un bon algorithme de prédiction revient à apprendre une fonction $\hat{f} : \mathbb{R}^d \to \mathbb{R}$ qui, à partir d'une entrée x fournit une bonne approximation $\hat{y} = \hat{f}(x)$ de la réponse y.

Risque. Pour savoir si $\hat{y}$ est une bonne approximation de la réponse y, il nous faut définir une mesure $r(y, \hat{y}) \in \mathbb{R}$ de l'erreur d'approximation. Cette mesure est appelée *risque*, ou parfois *perte*. Pour les problèmes de régression, la mesure classiquement utilisée est le *risque quadratique* :

$$r(y, \hat{y}) = (y - \hat{y})^2$$

Risque empirique. À partir des données d'entraînement $\{(x_i, y_i)\}_{i \leq n}$, l'algorithme d'apprentissage va sélectionner une fonction $\hat{f}$ parmi un ensemble de fonctions $\mathcal{H}$ qui aura été défini à l'avance. Le risque moyen de la fonction $\hat{f}$ sur les données d'entraînement s'appelle le *risque empirique* :

$$\hat{R}_e(\hat{f}) = \frac{1}{n} \sum_{i=1}^{n} r(y_i, \hat{f}(x_i))$$

Risque de généralisation. Le risque empirique permet de quantifier l'erreur de la fonction $\hat{f}$ sur les données d'entraînement. Mais pour s'assurer d'avoir appris une bonne fonction, nous aimerions quantifier l'erreur moyenne qu'elle produirait pour un nouveau couple (x, y) qui n'était pas présent dans la base d'apprentissage. Cette mesure s'appelle le *risque de généralisation* car il s'agit de savoir si ce qui a été appris sur certaines données observées se généralise bien à l'ensemble des données qu'il sera possible d'observer dans le futur.

Pour définir cette mesure, il faut considérer que les exemples $\{(x_i, y_i)\}_{i \leq n}$ sont les réalisations de n tirages aléatoires et indépendants suivant la même loi de probabilité $P(X, Y)$. On dit que les exemples sont i.i.d. (indépendants, identiquement distribués). On peut alors définir le risque de généralisation sur cette distribution de probabilité jointe :

$$R(\hat{f}) = \mathbb{E}[r(\hat{f}(X), y)]$$

Si on pouvait calculer directement le risque de généralisation, alors le meilleur choix de fonction serait de sélectionner $\hat{f} \in \mathcal{H}$ qui minimise ce risque. Mais le problème vient justement du fait qu'on ne connaît pas la distribution $P(X, Y)$ qui gouverne les données, ce qui rend le calcul impossible.

Minimisation du risque empirique. En pratique, on pourrait se dire que la meilleur solution est de choisir la fonction qui se comporte le mieux sur le jeu de données d'apprentissage, c'est à dire :

$$\hat{f} = \arg\min_{h \in \mathcal{H}} \hat{R}_e(h)$$

Idéalement, on aimerait que le risque empirique $\hat{R}_e(\hat{f})$ soit un bon estimateur du risque de généralisation $R(\hat{f})$, ce qui nous garantirait que le choix de $\hat{f}$ était judicieux. Malheureusement, le fait que $\hat{f}$ ait été choisie pour être performante spécifiquement sur le jeu d'apprentissage implique que le risque empirique est un estimateur potentiellement très biaisé du risque de généralisation, et probablement de façon trop optimiste.

Quelles garanties théoriques ? Pour commencer, considérons f_a, la meilleure fonction qu'on aurait pu choisir au sein de l'ensemble de fonctions $\mathcal{H}$, c.a.d. :

$$f_a = \arg\min_{h \in \mathcal{H}} R(h)$$

Il existe un résultat théorique qui permet de borner l'écart de risque de généralisation entre le minimisateur du risque empirique $\hat{f}$ et la fonction qui minimise le risque de généralisation f_a. Cela permet de savoir en théorie, de combien on se trompe au maximum en choisissant $\hat{f}$. On ne rentrera pas ici dans les détails du calcul permettant d'arriver à ce résultat, mais le voici :

$$R(f_a) \leq R(\hat{f}) \leq R(f_a) + 2 \max_{h \in \mathcal{H}} |R(h) - R_e(h)|$$

Cette inégalité est un résultat incontournable de la théorie de l'apprentissage et est connu sous le nom de *dilemme biais-complexité*. On rencontre souvent sous l'appellation *dilemme biais-variance* un résultat dans le même esprit, mais plus spécifiquement lié au *risque quadratique* que l'on a introduit précédemment. Cette inégalité contient de nombreux enseignements sur ce qu'est l'apprentissage supervisé.

Tout d'abord, intéressons nous à la partie droite de l'inégalité qui est une somme composée de deux termes. La première chose à remarquer est que les deux termes dépendent de la famille de fonctions $\mathcal{H}$ parmi laquelle on autorise l'algorithme à choisir une fonction. Le choix de $\mathcal{H}$ traduit les hypothèses que l'on fait *a priori* sur la nature de la fonction f qu'il va falloir approximer pour bien prédire Y à partir de X. Voyons maintenant le rôle que joue ce choix dans le processus d'apprentissage.

Le premier terme est $R(f_a)$. C'est le risque de généralisation de la meilleure fonction appartenant à $\mathcal{H}$. On voit donc que ce terme diminue lorsque l'on fait grandir $\mathcal{H}$. En effet, plus on cherche parmi une famille diverse de fonctions, plus on a de chance de trouver f_a qui soit une bonne approximation de la vraie fonction f. On appelle ce terme *biais* ou *erreur d'approximation*.

Le deuxième terme est : $2 \max_{h \in \mathcal{H}} |R(h) - R_e(h)|$. Pour toutes les fonctions $h \in \mathcal{H}$, on s'intéresse à la différence entre le risque empirique et le risque réel de généralisation. Autrement dit, pour une fonction h, on veut savoir à quel point le fait d'avoir observé spécifiquement ce jeu de données $(x_i, y_i)_{i \leq n}$ plutôt qu'un autre tiré de la même distribution $P(X, Y)$ a biaisé notre estimation du risque $R(h)$ par le risque empirique $R_e(h)$. On prend le maximum sur $\mathcal{H}$ car on veut connaître l'erreur maximum que l'on peut produire en tentant d'estimer le risque de généralisation par le risque empirique, quelle que soit la fonction h retenue. On voit que ce terme augmente lorsque $\mathcal{H}$ grandit, puisque c'est un maximum. En effet, plus on a un large choix de fonctions,

plus il sera simple d'en trouver une qui a de très bonnes performances sur le jeu de données d'apprentissage par rapport à ses performances moyennes sur l'ensemble des jeux de données possibles, c.-à-d. $Re(h)$ très inférieur à $R(h)$. Ce terme est donc appelé terme de *variance*, *fluctuation* ou encore *complexité* car il exprime le fait que la complexité de $\mathcal{H}$ va faire fluctuer ou varier l'écart entre le risque empirique et le risque de généralisation.

L'analyse de ces deux termes permet donc de comprendre pourquoi cette inégalité constitue un dilemme ou un compromis. En effet, pour avoir des garanties sur la qualité de la fonction $\hat{f}$ – qui minimise le risque empirique – il faut choisir une bonne famille de fonctions $\mathcal{H}$. Plus elle est grande (diverse, complexe), plus il y a de chances qu'elle contienne une bonne fonction f_a candidate, mais plus on a de chances que les performances de généralisation de la fonction choisie $\hat{f}$ soient éloignées des performances de f_a.

L'enjeu fondamental de l'apprentissage supervisé est de trouver une famille d'hypothèse $\mathcal{H}$ qui contient des fonctions adaptées au problème à résoudre, mais est suffisamment petite pour que cette fonction soit réellement choisie par l'algorithme. C'est ici qu'interviennent les connaissances *a priori* de l'expert sur le problème, sur la distribution qui génère les données, notamment sur les régularités de la fonction que l'on souhaite apprendre. Toutes ces connaissances lui permettront d'envisager quels types de fonctions doivent appartenir à $\mathcal{H}$.

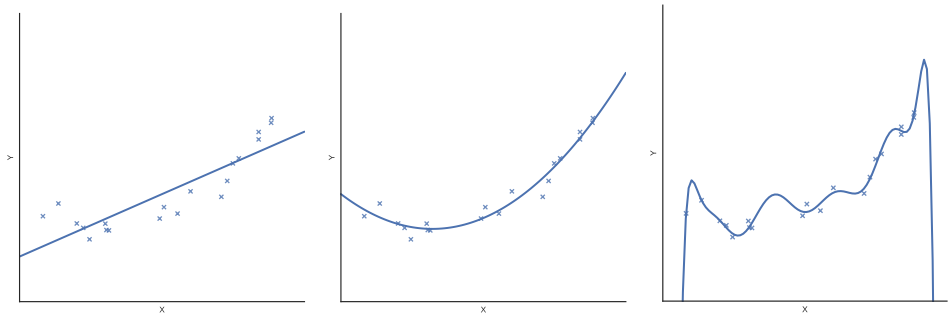
Sous-apprentissage et sur-apprentissage Le *sous-apprentissage* et le *sur-apprentissage* sont des situations que l'on rencontre lorsque l'on n'a pas réussi à trouver un bon équilibre pour la complexité de $\mathcal{H}$. Nous allons expliquer ces notions via l'exemple présenté dans la figure 2.1. La fonction utilisée pour générer les données, $f : X \mapsto Y$, est un polynôme du second degré auquel on a ajouté un peu de bruit.

Le sous-apprentissage (fig. 2.1a) se présente lorsque $\mathcal{H}$ est trop simple. Dans cet exemple, $\mathcal{H}$ est l'ensemble des polynômes de degré au plus 1. On voit que la droite apprise $\hat{f}$ est proche de la meilleure fonction $f_a \in \mathcal{H}$ qu'on aurait pu trouver. Cependant, cela reste très éloigné de la vraie fonction f car notre ensemble de fonctions est trop biaisé.

Dans la figure 2.1b, on présente un cas ou la famille de fonctions est adaptée au problème. On a choisi $\mathcal{H}$ comme étant l'ensemble des polynômes de degré au plus 2. Cette famille contient un bon polynôme candidat f_a et est assez peu complexe pour que le minimisateur du risque empirique $\hat{f}$ soit une bonne approximation de f_a

Le *sur-apprentissage* correspond au cas où $\mathcal{H}$ est trop complexe. Dans la figure 2.1c, on s'est autorisé à chercher parmi des polynômes de degrés

bien plus grands que 2. On voit qu'on a été capable de trouver un polynôme qui est proche de tous les points du jeu d'entraînement, sans pour autant ressembler à la fonction f qui a généré les données. On voit en effet qu'entre les observations, le polynôme fluctue de manière excessive. Le problème va donc se poser pour les observations futures, qui n'ont aucune raison d'être proches de ces fluctuations, et seront donc mal prédites. On est donc ici dans le cas où $\mathcal{H}$ contient une très bonne estimation f_a de la vraie fonction f (on aurait pu apprendre un polynôme de degré 2), mais où la fonction $\hat{f}$ apprise est très éloignée de f_a.



(a) sous-apprentissage (b) bon apprentissage (c) sur-apprentissage

FIGURE 2.1 – Les différentes configurations du dilemme biais-complexité

Influence du nombre d'observations. Si on considère un ensemble d'hypothèses $\mathcal{H}$ fini, on peut réécrire l'inégalité précédente avec une borne sur l'erreur de fluctuations qui dépend du nombre d'exemples. Pour n et δ fixés, le résultat suivant est vrai avec une probabilité $1 - \delta$:

$$R(h) \leq R(f_a) + 2\sqrt{\frac{1}{n}\log\frac{2\,|\mathcal{H}|}{\delta}}$$

On observe que plus la taille de l'échantillon n est élevée, meilleure sera la borne. Mais on voit aussi que la borne fait intervenir la complexité (ici la taille) de la famille d'hypothèses $\mathcal{H}$. On voit que si l'on souhaite conserver une borne constante, plus le cardinal de $\mathcal{H}$ est grand, plus le nombre d'exemples n nécessaires sera élevé. Ainsi, le choix de modèles complexes peut se révéler risqué si jamais on dispose d'une faible quantité de données.

Notons que, la plupart du temps, on travaille avec $\mathcal{H}$ de taille infinie, et ce résultat n'est donc plus valable. Il faut alors recourir à des outils théoriques plus poussés, qui sortent du cadre de cette thèse. Ce qu'il faut conserver, c'est l'intuition qu'un modèle complexe nécessite plus d'échantillons pour

converger vers une solution satisfaisante et qu'on court un plus grand risque de *sur-apprentissage*. Nous allons maintenant voir un autre phénomène qui peut rendre l'apprentissage difficile.

Fléau de la haute dimension. Le fléau de la dimension est un phénomène que l'on rencontre lorsque la variable d'entrée $X = (\dots, X_j, \dots)_{j \leq d}$ est composée d'un grand nombre d de variables. Chaque réalisation x_i de X est un vecteur de taille d, et peut donc être vue comme un point dans un espace $\mathbb{R}^d$ de dimension d.

Beaucoup d'algorithmes d'apprentissage utilisent une notion de similarité entre deux entrées x_i et $x_{i'}$ pour apprendre. On voit que l'intuition géométrique est de considérer que x_i et $x_{i'}$ sont similaires, si la distance entre les deux points dans $\mathbb{R}^d$ est faible. Dans les espaces de faible dimension comme $\mathbb{R}^2$ ou $\mathbb{R}^3$, nous sommes capables de raisonner intuitivement avec cette notion de distance. Malheureusement, en haute dimension, notre intuition est souvent mise à rude épreuve.

On dit parfois que les espaces de haute dimension sont très « vastes ». On court ainsi le risque que toutes les observations x_i du jeu d'apprentissage soient loin les unes des autres. Autrement dit, si l'on mesure la distance euclidienne dans un tel espace, la différence de distance entre les différentes paires d'observations (x_i, x'_i) sera probablement faible et la notion de proximité spatiale risquera donc de perdre son utilité pour l'apprentissage. Tous les algorithmes d'apprentissage ne sont pas affectés de la même manière par ce fléau, car certains d'entre eux intègrent des mécanismes permettant d'en limiter les effets. Dans le pire des cas théoriques, le nombre n d'exemples nécessaires pour apprendre augmente exponentiellement avec la dimension d, ce qui rend évidemment la tâche d'apprentissage quasi-impossible pour des valeurs de d élevées.

Espace de représentation. Nous avons vu précédemment qu'il est plus facile d'apprendre une bonne approximation d'une fonction $f : X \mapsto Y$ si cette fonction est peu complexe – autrement dit, régulière. Mais il arrive parfois que dans l'espace d'origine ($\mathbb{R}^d$ dans notre cas), la fonction soit très irrégulière et donc difficile à apprendre.

Une approche face à ce problème consiste à utiliser une fonction ϕ qui transforme les vecteurs d'entrée $x \in \mathbb{R}^d$ en une nouvelle représentation $\phi(x) \in \mathbb{R}^{d'}$ dans un nouvel espace dans lequel il sera plus facile d'apprendre. Plus précisément, on espère choisir ϕ telle qu'il existe une fonction $g : \mathbb{R}^{d'} \to \mathbb{R}$, régulière et donc facile à apprendre, avec $Y = f(X) = g(\phi(X))$.

L'idée est donc de passer d'un espace de représentation dans lequel la

fonction f est difficile à apprendre, à un nouvel espace dans lequel il existe une fonction g plus simple, qui fasse le lien entre la nouvelle représentation de la variable d'entrée $\phi(X)$ et la variable de sortie Y. Cela permet d'aborder le processus d'apprentissage d'une nouvelle manière. Plutôt que de devoir concevoir des algorithmes capables d'approximer des fonctions complexes comme f, on peut se concentrer sur des algorithmes qui seront performants pour apprendre des fonctions simples comme g, puis faire en sorte pour chaque problème de se ramener à un cas dans lequel on a le droit d'utiliser ces algorithmes. Dans cette approche, le plus important est donc de trouver une bonne représentation des données, c'est-à-dire choisir – ou apprendre – la fonction ϕ adaptée au problème. Nous reparlerons plus tard de cette approche lorsque nous introduirons les méthodes à noyaux et la sélection de variables.

Séparation entre jeu d'apprentissage et jeu de test. Nous avons vu précédemment que lors de l'apprentissage d'un modèle $\hat{f}$ sur un jeu de données $\mathcal{D} = \{x_i, y_i\}$, l'observation du risque empirique nous fait sous-estimer le risque de généralisation. Avant d'utiliser un algorithme, on souhaite avoir une bonne estimation de ce risque de généralisation, car c'est lui qu'on peut s'attendre à observer en pratique. Pour répondre à cette question, on peut partitionner le jeu de données $\mathcal{D}$ en deux parties : le jeu d'apprentissage $\mathcal{D}_a = \{x_i^a, y_i^a\}$ et le jeu de test $\mathcal{D}_t = \{x_i^t, y_i^t\}$.

Définissons pour un modèle h le risque empirique sur le jeu d'apprentissage (ou *erreur d'apprentissage*) $R_e^a(h)$ et le risque empirique sur le jeu de test (ou *erreur de test*) $R_e^t(h)$ comme suit :

$$R_e^a(h) = \frac{1}{|\mathcal{D}_a|} \sum_{i=1}^{n} r(y_i^a, h(x_i^a)) \quad \text{et} \quad R_e^t(h) = \frac{1}{|\mathcal{D}_t|} \sum_{i=1}^{n} r(y_i^t, h(x_i^t))$$

Dans un premier temps, le modèle $\hat{f}$ est appris en minimisant le risque empirique sur $\mathcal{D}_a$. Ainsi, seuls les exemples $(x_i^a, y_i^a) \in \mathcal{D}_a$ sont utilisés pour apprendre le modèle. On a donc :

$$\hat{f} = \arg\min_{h \in \mathcal{H}} \hat{R}_e^a(h)$$

Ensuite, pour obtenir une estimation du risque de généralisation $R(\hat{f})$, on calcule le risque empirique sur le jeu de test :

$$\hat{R}(\hat{f}) = \hat{R}_e^t(\hat{f})$$

Comme les exemples du jeu de test n'ont pas été utilisés pour choisir $\hat{f}$, cet estimateur ne sera pas biaisé par la procédure d'apprentissage.

Hyperparamètres et sélection de modèle. Lors de l'apprentissage d'un modèle, on fait la différence entre les paramètres internes du modèle dont la valeur est déterminée pendant l'apprentissage et les hyperparamètres, dont la valeur doit être fixée avant l'apprentissage. Si l'on reprend l'exemple de la figure 2.1, un hyperparamètre peut être le degré maximal du polynôme, et les paramètres sont les valeurs des coefficients du polynôme. Comme autres exemples d'hyperparamètres, on a par exemple le coefficient de régularisation dans une régression LASSO ou la bande passante dans une régression à noyau gaussien. Le choix des valeurs des hyperparamètres est déterminant pour la qualité du modèle que l'on va apprendre. Il faut donc disposer d'une méthode pour choisir une valeur pertinente.

Validation Croisée. Les approches de type validation croisée servent à évaluer (estimer) l'erreur de généralisation d'un modèle, à fixer la valeur des hyperparamètres d'un modèle et à sélectionner un modèle parmi un ensemble de modèles candidats. Un des objectifs est d'éviter le sur-apprentissage. L'idée fondamentale est de décomposer le jeu de données, afin qu'un exemple ayant servi à l'apprentissage ne soit pas utilisé pour évaluer la performance du modèle appris. On évite ainsi de biaiser l'évaluation. Dans la validation croisée, plusieurs partitionnements du jeu de données sont considérés : pour chacun d'entre eux, le modèle est appris et une estimation de l'erreur de généralisation est produite, puis on moyenne les estimations obtenues sur les différents partitionnements pour obtenir l'estimation finale de l'erreur.

Apprentissage, Validation, Test L'approche la plus élémentaire est de séparer le jeu de données en un jeu d'apprentissage, un jeu de validation et un jeu de test. Cette approche peut être vue comme la plus simple ou un cas dégénéré de validation croisée, car un seul partitionnement est considéré et l'erreur de généralisation obtenue n'est donc pas une moyenne.

Supposons que le modèle que l'on souhaite estimer possède un hyperparamètre λ. On souhaite choisir la meilleure valeur $\hat{\lambda}$ pour cet hyperparamètre parmi un ensemble de valeurs $\{\lambda_1, \ldots, \lambda_m\}$. Sur le jeu d'apprentissage, on va apprendre un modèle pour chacune des valeurs λ_i, que l'on notera $\hat{f}_{\lambda_i}$. L'objectif est de conserver le modèle avec la plus petite erreur de généralisation. Estimer cette erreur sur le jeu d'apprentissage serait problématique, car il contient des exemples ayant servi à l'apprentissage. On va donc calculer le risque empirique sur le jeu de validation, qui ne contient que des exemples

n'ayant pas été utilisés jusqu'à présent. On choisit donc la valeur $\hat{\lambda}$ comme suit :

$$\hat{\lambda} = \arg\min_{\lambda_i} \hat{R}_e^v(\hat{f}_{\lambda_i}) \tag{2.1}$$

avec $\hat{R}_e^v(\hat{f}_{\lambda_i})$ le risque empirique calculé sur le jeu de validation pour le modèle d'hyperparamètre λ_i. On pourrait alors penser que ce risque empirique est une bonne estimation du risque de généralisation du modèle choisi, mais notre procédure souffre en réalité de deux biais.

Le premier biais est pessimiste, dans le sens où il va nous conduire à surestimer l'erreur de généralisation. Il provient du fait qu'au lieu d'utiliser tous les exemples hors jeu de test pour apprendre le modèle, on les a séparés en un jeu d'apprentissage de taille n_a et un jeu de validation de taille n_v. Si l'on dispose de peu de données et/ou si la qualité du modèle est très sensible à la quantité de données d'apprentissage, il est possible que l'erreur du modèle appris avec n_a exemples soit plus importante que le modèle appris avec n_a+n_v exemples, causant donc la surestimation de l'erreur.

Le deuxième biais est optimiste : il nous pousse à sous-estimer l'erreur. En effet, précédemment, on a mentionné le fait que les exemples du jeu de validation n'étaient pas contaminés, car ils n'avaient pas été utilisés pour apprendre le modèle. Mais maintenant que l'on conserve le modèle appris avec la valeur $\hat{\lambda}$, on voit que les exemples du jeu de validation ont été utilisés pour discriminer parmi tous les candidats f_{λ_i}. On peut alors avoir choisi un modèle particulièrement bon sur le jeu de validation. Le risque empirique calculé sur le jeu de validation peut donc être optimiste par rapport au risque empirique calculé sur de nouvelles données. Plus le nombre de valeurs λ_i candidates est élevé, plus on court le risque qu'un modèle f_{λ_i} soit, par hasard, beaucoup plus performant sur le jeu de validation que sa performance moyenne sur des nouvelles données.

Une solution pour estimer alors l'erreur de généralisation en évitant ces deux biais, est de réapprendre un modèle $f_{\hat{\lambda}}$ d'hyperparamètre $\hat{\lambda}$ sur l'ensemble des données d'apprentissage et de validation. On évaluera alors sa performance en calculant le risque empirique sur le jeu de test, qui n'aura pas encore servi, ni lors de l'estimation des paramètres, ni lors du choix de l'hyperparamètre.

Validation *Leave-p-out* Dans la méthode *leave-p-out*, on sépare les données en un jeu d'apprentissage, de taille n_a et un jeu de test. On va ensuite explorer tous les partitionnements possibles du jeu d'apprentissage en deux sous-jeux de tailles respectives $n_a - p$ et p. Le sous-jeu de taille $n_a - p$ est utilisé pour apprendre les modèles f_{λ_i} et le sous-jeu de taille p est

utilisé en validation, pour calculer le risque empirique $\hat{R}_e^v(\hat{f}_{\lambda_i})$. En totalité, on considère donc $\binom{n_a}{p}$ partitionnements différents, et donc autant d'évaluations du risque empirique de validation pour chaque valeur de λ_i. On obtient une estimation finale du risque empirique en calculant la moyenne de toutes ces estimations, et on retient le $\hat{\lambda}$ avec le plus faible risque empirique moyen. Enfin, pour avoir une estimation finale non biaisée de sa performance, on calcule le risque empirique sur le jeu de test, $\hat{R}_e^t(\hat{f}_{\hat{\lambda}})$.

Un avantage de cette méthode est que chaque exemple a servi à l'apprentissage pour certains partitionnements, et à la validation pour d'autres. En outre, on a considéré tous les partitionnements possibles, ce qui évite d'introduire un biais lié à la sélection du partitionnement. Par contre, on voit que le nombre de modèles a apprendre croît rapidement avec p, ce qui rend cette approche peu applicable en pratique.

Validation *Leave-one-out*. La validation *leave-one-out* (LOO) correspond au cas particulier où l'on conserve uniquement 1 exemple pour la validation et $n_a - 1$ pour apprendre le modèle. On évite ainsi l'explosion du nombre de partitionnements, puisqu'on en considère n_a au total. La validation LOO est utilisée quand on dispose de peu de données, car elle permet d'utiliser $n_a - 1$ exemples d'entraînement. Elle possède aussi l'avantage d'être peu biaisée, car les modèles sont entraînés sur $n_a - 1$ exemples au lieu de n_a.

Validation k-Fold. La validation k-fold consiste à partitionner le jeu d'apprentissage en k partitions de même taille. Chaque partition est utilisée une fois comme jeu de validation, en entraînant le modèle sur l'union des $k-1$ autres partitions. On obtient donc k estimations de l'erreur de généralisation, qui sont moyennées pour obtenir l'estimation finale. A la différence de la validation LOO qui est exhaustive, ici on ne teste pas toutes les façons de diviser le jeu d'apprentissage en k partitions. La méthode la plus simple est de diviser les données de façon aléatoire, mais il existe d'autres approches comme la validation stratifiée, utile pour les problèmes de classification avec des classes déséquilibrées.

Dans la pratique, on utilise souvent de la validation 5-fold ou 10-fold. Un avantage de cette approche est son faible coût de calcul, par rapport à la *leave-one-out*.

Sélection de modèles. Dans les paragraphes précédents, on a traité du choix par validation croisée du meilleur hyperparamètre pour une méthode donnée. On a également évoqué l'estimation de l'erreur de généralisation

pour le modèle sélectionné. Il reste à aborder la question de la comparaison de différentes méthodes, chacune possédant (ou non) des hyperparamètres.

Validation $(k \times l)$-fold. Dans cette approche (également nommée *nested cross-validation*), il y a deux boucles imbriquées de validation croisée. La boucle intérieure est utilisée pour apprendre les meilleurs hyperparamètres pour chaque famille de modèles, et la boucle extérieure pour comparer les modèles ainsi optimisés. L'inconvénient de cette approche est le coût lié au nombre de fois où l'on apprend chaque modèle.

k-fold et jeu de validation. Une façon moins coûteuse, bien que moins robuste, est de garder un jeu de validation suffisamment grand. Les hyperparamètres des modèles sont appris par validation k-fold sur le jeu d'apprentissage puis comparés sur le jeu de validation.

2.3 Séries temporelles

Dans cette section nous présentons les notions de processus stochastiques et de séries temporelles qui permettent de décrire plus précisément le phénomène de trafic qui est naturellement temporel. Une des spécificités des données temporelles est que le temps fournit un ordre naturel pour les observations, ce qui les différencie des données classiques. Nous présentons également comment représenter le problème de prévision de séries temporelles de façon à pouvoir utiliser certains algorithmes d'apprentissage supervisé qui ne sont pas conçus spécifiquement pour les données temporelles.

Notations. Pour passer à la notion de variables temporelles, il faut introduire les notations suivantes :

- N : nombre de séries observées
- T : nombre de pas de temps observés
- Z : un processus stochastique multivarié
- $Z(t) = (Z_1(t), \ldots, Z_N(t))$ le vecteur aléatoire à l'instant t
- $z(t) = (z_1(t), \ldots, z_N(t)$ l'observation du vecteur aléatoire $Z(t)$,
- p : le lag pour l'embedding

Série temporelle. Une série temporelle est une séquence de valeurs qui ont été observées séquentiellement dans le temps. Très souvent, on suppose que l'intervalle de temps – ou *pas de temps* – séparant deux observations est constant.

Série temporelle univariée. On parle de série temporelle *univariée* quand une seule quantité est observée au cours du temps. Une série temporelle univariée comportant T observations successives sera représentée par un vecteur :

$$z = (z(1), \ldots, z(T))$$

avec $z(t)$ la valeur observée à l'instant t.

Série temporelle multivariée. On parle de série temporelle *multivariée* quand on observe plusieurs quantités simultanément, au cours du temps. Si on observe N quantités, alors une observation simultanée, à l'instant t sera représentée par un vecteur colonne :

$$z(t) = \begin{pmatrix} z_1(t) \\ \vdots \\ z_N(t) \end{pmatrix}$$

et donc l'ensemble des données par la matrice :

$$z = \begin{pmatrix} z_1(1), & \ldots & z_1(T) \\ \vdots & \ddots & \vdots \\ z_N(1), & \ldots & z_N(T) \end{pmatrix}$$

Processus stochastique. La notion de *processus stochastique* est une généralisation de la notion de variable aléatoire, permettant d'analyser des phénomènes temporels. Un processus stochastique *à temps discret*, $Z = (Z(1), \ldots, Z(T))$, est une famille de variables aléatoires $Z(t)$, chacune associée à un instant t. Par la suite, on parlera toujours de processus stochastiques à temps discret ; qu'on appellera simplement processus stochastiques.

Ainsi, on peut considérer qu'une série temporelle univariée (resp. multivariée) – notée z – est une réalisation d'un processus stochastique univarié (resp. multivarié) – noté Z. L'analyse de séries temporelles implique donc de s'intéresser aux propriétés du processus stochastique qui a généré les données.

Stationnarité. La *stationnarité* est une propriété de certains processus stochastiques, qui décrit une invariance du processus lorsqu'on le décale dans le temps. Beaucoup de modèles d'apprentissage utilisés pour l'analyse ou la prédiction de séries temporelles supposent que le processus stochastique sous-jacent respecte cette hypothèse de stationnarité. Il existe deux types de stationnarité décrits ci-dessous.

Stationnarité forte : Un processus Z est dit stationnaire si pour tout temps t_1 et t_2 et décalage k, on peut écrire l'égalité suivante :

$$P\big(Z(t_1),\dots,Z(t_2)\big) = P\big(Z(t_1+k),\dots,Z(t_2+k)\big)$$

On peut lire cette propriété comme suit : la distribution de probabilité jointe de toute famille de variables $(Z(t_1),\dots,Z(t_2))$ n'est pas affectée par un décalage temporel.

En pratique, cette propriété n'est pas très utile car il est difficile d'obtenir des informations précises sur la distribution jointe entière à partir d'une unique réalisation (la série temporelle). C'est pourquoi on introduit une variante moins générale, mais plus facile à tester en pratique.

Stationnarité faible : La stationnarité faible – ou encore *stationnarité d'ordre 2* – est une propriété moins stricte que la précédente. Comme son nom l'indique, elle ne porte que sur les moments d'ordre 1 (moyenne) et d'ordre 2 (covariance) des distributions. Un processus Z est donc *faiblement stationnaire* si :

1. la moyenne est finie et indépendante de l'instant t :

$$\forall t, \quad \mathbb{E}[Z(t)] = \mu$$

2. le moment d'ordre 2 existe pour tout t :

$$\forall t, \quad \mathbb{E}\left[Z(t)^2\right] < \infty$$

3. la covariance entre deux variables $Z(t)$ et $Z(t+k)$ dépend uniquement du décalage temporel k :

$$\forall t, \forall k, \quad \mathrm{Cov}(Z(t), Z(t+k)) = \mathbb{E}\big[(Z(t)-\mu)(Z(t+k)-\mu)\big] = \gamma(k)$$

Prévision de séries temporelles. L'objectif de la prévision de séries temporelles, est de produire à partir d'une série temporelle observée, une estimation de ses valeurs futures. Prenons par exemple une série temporelle univariée, $z = \big(z(1),\dots,z(T)\big)$. La prévision à un instant t, pour un horizon de prévision h, consiste à trouver une estimation $\hat{z}(t+h)$, la plus proche possible de la valeur qui sera réellement observée, $z(t+h)$, en utilisant notre connaissance des valeurs observées du passé jusqu'à l'instant t, c'est-à-dire $\big(z(1),\dots,z(t)\big)$.

Représentation classique Pour pouvoir utiliser les méthodes de régression présentées précédemment pour résoudre un problème de prévision de séries temporelle, il faut pouvoir se ramener à une représentation des données avec une matrice d'entrée $\mathbf{X}$ et un vecteur de sortie Y. Le jeu de données est composé d'exemples correspondants à un instant t, et est donc de la forme $\{(\mathbf{x}_t, y_t\})$. La composition de ces exemples dépend du cadre de série temporelle dans lequel on se trouve :

Dans le cas univarié :

$$\begin{cases} y_t = z(t+h) \\ \mathbf{x}_t = (z(t), \ldots, z(t-d)) \end{cases} \tag{2.2}$$

Dans le cas multivarié :

$$\begin{cases} y_t = (z_1(t+h), \ldots, z_N(t+h)) \\ \mathbf{x}_t = (z_1(t), \ldots, z_1(t-d), \ldots, z_N(t), \ldots, z_n(t-d)) \end{cases} \tag{2.3}$$

Dans le cas où on a sélectionné p variables spatio-temporelles pertinentes pour prédire le futur d'une série z_i :

$$\begin{cases} y_t = z_i(t+h) \\ \mathbf{x}_t = \big(z_{i_1}(t-\tau_1), \ldots, z_{i_p}(t-\tau_p)\big) \end{cases} \tag{2.4}$$

avec i_j et τ_j correspondant respectivement à l'identifiant de la série temporelle et au décalage temporel de la j^{e} variable spatio-temporelle sélectionnée.

2.4 Discussion

Dans ce chapitre, nous avons introduit la notion de variables aléatoires et de densité de probabilité à plusieurs variables, ainsi que les coefficients de covariance et corrélation qui permettent de décrire la dépendance linéaire entre deux variables.

Puis, nous avons présenté l'apprentissage supervisé comme un problème d'approximation d'une fonction qui relie les entrées X à la sortie Y. La notion de risque empirique permet d'évaluer l'erreur d'approximation, et il est possible d'approximer la fonction en minimisant le risque empirique sur le jeu d'entraînement. Mais, ce que l'on souhaite réellement minimiser est le risque de généralisation, c.-à-d. l'erreur moyenne d'approximation sur de nouvelles données. Or, il est possible que la fonction apprise soit très bonne

sur les données d'entraînement, et mauvaise sur de nouvelles données : c'est le sur-apprentissage.

Nous avons présenté le dilemme qui existe entre la complexité de la famille de fonctions que l'algorithme peut apprendre, et le biais systématique induit par cette famille de fonctions. Si la famille est peu complexe, il est possible qu'elle ne contienne pas de fonction très proche de la fonction que l'on souhaite approximer, elle est donc biaisée. Si la famille est très complexe, ce biais sera moins fort car elle contiendra une fonction plus proche de la fonction à approximer, mais il y a des chances de ne pas converger vers le choix de cette fonction parmi toutes les autres, la variance étant élevée.

Nous avons présenté la notion d'hyperparamètre, que l'on fixe avant l'apprentissage du modèle et qui peut déterminer la complexité de la famille d'hypothèse (par ex. le degré maximal du polynôme). Nous avons vu que les approches par validation croisée permettent de chercher un compromis entre biais et complexité, en découpant le jeu d'entraînement et en utilisant une partie pour l'apprentissage et l'autre pour la validation. Nous avons mentionné le fléau de la haute dimension qui pose des problèmes d'apprentissage du modèle lorsque le nombre de variables d'entrée est trop élevé. Ce problème provient du fait que la notion de distance dans un espace à haute dimension n'a pas le comportement auquel on est intuitivement habitué pour les espaces de faible dimension. Nous avons vu qu'il est possible de changer d'espace de représentation des données par une transformation, ce nouvel espace pouvant être de plus faible dimension ou simplement permettre d'obtenir une représentation plus simple des données pour pouvoir utiliser des algorithmes classiques d'apprentissage.

Enfin, nous avons rappelé que le phénomène de trafic est un phénomène temporel et qu'il existe des outils mathématiques pour décrire ce phénomène et ses observations : les processus stochastiques et les séries temporelles. Nous avons présenté la propriété de stationnarité qui est souvent une hypothèse pour les modèles de prévision de séries temporelles. Enfin, nous avons vu comment passer d'une représentation sous forme de séries temporelles, à une représentation classique qui permet d'utiliser des modèles d'apprentissage supervisé classiques, qui n'ont pas été conçus spécifiquement pour les séries temporelles.

Dans le chapitre suivant, nous présentons la littérature de la prévision de trafic, avec les différentes approches qui ont été proposées pour résoudre ce problème. Nous y identifions les problématiques qui ont ensuite guidé les travaux de la thèse.

Chapitre 3

Introduction à la prévision de trafic

Sommaire

3.1 Approches par modèles de trafic 34
3.2 Modèles statistiques 36
3.2.1 Famille des modèles ARIMA 36
3.2.2 Modèle espace-état 37
3.3 Méthodes d'apprentissage artificiel 38
3.3.1 Approches des k plus proches voisins 38
3.3.2 Méthodes à noyaux 39
3.3.3 Réseaux de neurones artificiels 40
3.4 Différentes grilles de lectures 42

Le chapitre précédent a permis d'introduire les concepts principaux de l'apprentissage supervisé, qui sont à la base des algorithmes de prévision. On a également présenté les notions de processus stochastique et de séries temporelles, qui permettent de décrire les phénomènes se déroulant dans le temps. Suite à cette introduction théorique, nous allons maintenant nous intéresser aux travaux existants sur la prévision de trafic.

L'objectif de chapitre est de présenter les différents enjeux de la prévision de trafic, ainsi que l'évolution des problématiques rencontrées dans ce domaine depuis sa création. Les différentes approches de la prévision de trafic y sont présentées succinctement, afin de donner une vue d'ensemble de la diversité des grandes familles de méthodes et des différentes façon de modéliser le problème de prévision. Le chapitre suivant permettra de rentrer plus en détail dans le fonctionnement des méthodes qui ont été retenues pour notre étude expérimentale.

La prévision de trafic est un enjeu central de la recherche sur les transports depuis plus de trois décennies. Cette problématique découle du besoin de développer des applications utiles aux usagers des réseaux de transports et aux gestionnaires des infrastructures. Il est devenu crucial de pouvoir informer les conducteurs sur les conditions de trafic, afin de les aider à préparer leurs trajets en amont et à les optimiser en temps-réel, pour faire face aux différents imprévus, dont l'apparition de la congestion. Pour les gestionnaires, il est nécessaire de pouvoir surveiller l'état du trafic sur tout le réseau, et de l'anticiper, afin de prendre les mesures permettant d'assurer la meilleure qualité de service possible. Alors que certaines de ces décisions sont ensuite directement liées à l'expertise du gestionnaire, les dernières décennies ont également vu apparaître de nouvelles applications permettant notamment d'optimiser en temps-réel la signalisation pour maintenir un trafic aussi fluide que possible, en atténuant le plus tôt les congestions naissantes [63, 32].

Cette évolution applicative a été rendue possible par les progrès technologiques, couplés à une progression dans les méthodes d'analyse des données. Les progrès matériels ont permis d'équiper de capteurs un nombre croissant de tronçons routiers, de collecter en temps-réel les mesures de trafic et de les centraliser dans des bases de données au volume de plus en plus conséquent. La résolution temporelle à laquelle on est capable de mesurer le trafic a également beaucoup gagné en finesse. Dans le même temps s'est déroulée une évolution des méthodes d'analyse. En parallèle des modèles mathématiques très détaillés fondés sur des théories microscopiques et macroscopiques du trafic, s'est développée la recherche sur les algorithmes empiriques guidés par les données. Cela a donné lieu à l'utilisation croissante de nombreux algorithmes d'apprentissage automatique et de fouille de données pour analyser le phénomène de trafic. On a alors observé un déplacement de l'intérêt pour les modèles statistiques classiquement utilisés (la famille ARIMA) vers des modèles d'intelligence artificielle comme les méthodes à noyaux, les réseaux de neurones ou les algorithmes évolutionnistes.

3.1 Approches par modèles de trafic

La théorie du trafic propose des modèles mathématiques décrivant le phénomène de trafic. L'utilisation de ces modèles permet de développer des simulations du trafic sur le réseau. Afin de pouvoir faire fonctionner ces simulations, il est nécessaire de disposer d'une description du réseau de transport et de connaître complètement les conditions initiales ainsi que les profils de demande. L'évolution du trafic sur le réseau est alors simulée, afin de produire une prévision. Ces modèles sont distingués en trois familles, selon le

niveau de granularité avec lequel on modélise le phénomène de trafic : les modèles macroscopiques, microscopiques et mésoscopiques.

Modèles macroscopiques Les modèles macroscopiques permettent de décrire le phénomène de trafic à un niveau agrégé. On ne se concentre donc pas sur le comportement individuel des véhicules, mais sur des mesures agrégées en un point du réseau, telles que le débit, la densité et la vitesse moyenne des véhicules. L'objectif de ces modèles est de décrire la dynamique du flux de trafic sur le réseau.

Cette famille d'approche est née de l'hypothèse que l'écoulement du trafic dans le réseau est similaire à l'écoulement d'un fluide. Lighthill and Whitham ont apporté en 1955 une première contribution majeure en comparant les véhicules à des particules dans un fluide [62]. Cette idée fût complétée en 1956 par Richards, en introduisant la notion d'onde de congestion, qui décrit le phénomène par lequel la congestion se propage en sens inverse du trafic. Ces travaux donnent naissance au modèle LWR [81] (du nom de ses inventeurs).

Plusieurs raffinements de ce modèle ont été proposés par la suite : le modèle LWR multi-classe [110] introduit des distributions de vitesse, le modèle de Payne [76] permet de mieux décrire les changements d'état de trafic, le modèle à transmission cellulaire de Daganzo [21] est une méthode de résolution numérique permettant de trouver des solutions aux équations cinétiques.

La difficulté d'utilisation de ces modèles pour la prévision à court-terme provient de la nécessité d'estimer très précisément les conditions initiales, et d'intégrer en continu les données récentes pour produire des prévisions en temps-réel. Si cela est possible pour des autoroutes très bien équipées en capteurs, la prévision à l'échelle d'un réseau urbain est compliquée par la mauvaise couverture du réseau, ainsi que des dynamiques plus complexes de trafic, notamment causées par les cycles de feux parfois imprévisibles.

Modèles microscopiques Les modèles microscopiques décrivent le phénomène de trafic à l'échelle du véhicule. Ainsi, on ne décrit pas le trafic par des mesures agrégées comme dans les modèles macroscopiques, mais par des propriétés propres à chaque véhicule, comme leur vitesse et leur position dans le réseau. L'objectif est donc de modéliser les interactions entre les véhicules. De ces dynamiques à l'échelle des véhicules émergent les mesures agrégées, comme le débit.

On peut distinguer trois types d'approches microscopiques : les modèles à loi de poursuite qui sont des modèles continus [6, 50, 80], les modèles de type automate cellulaire, qui s'appuient sur une discrétisation du temps et de l'espace [20, 73, 54], et les modèles à changement de voie, qui décrivent

la façon dont les conducteurs décident de changer de voie et les dynamiques que cela crée [7].

Le coût de calcul élevé de ces simulations les rend peu adaptées pour la prévision à court-terme en temps-réel sur un réseau urbain.

Modèles mésoscopiques Les modèles mésoscopiques sont à mi-chemin entre les modèles microscopiques et macroscopiques. Dans le modèle mésoscopique, on ne distingue pas les véhicules individuellement, mais on peut spécifier leur comportement, par exemple de façon probabiliste. Le comportement du trafic est spécifié pour des groupes de véhicules, mais les interactions entre ces véhicules sont décrites avec un faible niveau de détail [78, 47, 43].

Le problème principal des approches basées sur les modèles de la théorie du trafic est la nécessité de savoir décrire mathématiquement le comportement des véhicules ou du flux de trafic. Or, dans le cas du trafic urbain à l'échelle du réseau, il existe de nombreuses sources d'incertitude et de dynamiques complexes qui sont difficile à modéliser.

Dans le cadre de cette thèse, nous avons fait le choix de nous concentrer sur les approches guidées par les données historiques. Parfois au prix de la perte d'une compréhension humaine fine du phénomène étudié ou de la modélisation retenue, les méthodes d'apprentissage artificiel découvrent dans l'historique des dépendances statistiques qui permettent par extrapolation de produire des prévisions. Si la structure simple de certain de ces modèles permet encore une analyse humaine du modèle appris, d'autres modèles comme les réseaux de neurones possèdent un nombre très élevé de paramètres, dont il est difficile d'expliquer le rôle exact qu'ils jouent de la prévision. Cependant, cette concession sur l'explicabilité est compensée par de nouvelles possibilités en terme de précision de prévision.

3.2 Modèles statistiques

3.2.1 Famille des modèles ARIMA

Le modèle ARIMA a été introduit en 1970 par Box and Jenkins [8]. Il permet de prévoir les valeurs futures d'une série temporelle univariée. Ce modèle comporte : une partie auto-régressive (AR) qui décrit la dépendance entre un instant et les instants passés, et une partie à moyenne mobile (MA) qui capture l'erreur de prévision aux instants passés. La notion d'intégration (I) signifie qu'on modélise parfois la version différenciée de la série. Cette

méthodologie a très rapidement été utilisée pour prévoir le trafic à court-terme [1] et s'est vite imposée comme une base incontournable à laquelle se comparer [35].

De nombreuses extensions de cette méthode ont été proposées par la suite. La méthode ARIMAX permet d'intégrer des variables exogènes au modèle. Elle est notamment utilisée pour intégrer des données météorologiques dans la prévision [99], mais aussi des informations issues d'un algorithme de partitionnement de données (*clustering*) [112]. Le modèle *Seasonal* ARIMA (SARIMA) permet de modéliser la saisonnalité des données de trafic [118, 90].

Pour répondre à l'objectif de prévoir plusieurs sections du réseau avec le même modèle, les travaux se sont orientés vers des versions multivariées d'ARIMA. Le modèle *Space-Time* ARIMA (STARIMA) [77] permet de prédire plusieurs séries temporelles en intégrant de la connaissance a priori sur leur dépendance spatiale et temporelle. Dans le cas du trafic, ce modèle a été utilisé pour intégrer la connaissance du réseau physique dans la prévision [53, 72, 16, 24]. Les paramètres estimés dans ce modèle sont globaux : ils sont communs à tous les capteurs. Le modèle *Generalized* STARIMA (GSTARIMA) intègre l'a priori sur la dépendance de la même manière, mais permet d'apprendre des paramètres différents pour chaque capteur [71]. Enfin, les modèles Vecteur Autorégressif (VAR) [12] et Vecteur Autorégressif et Moyenne Mobile (VARMA) [69] permettent d'apprendre la dépendance spatio-temporelle plutôt que de l'intégrer a priori.

3.2.2 Modèle espace-état

Un modèle espace-état est une représentation mathématique d'un système physique qui est composée d'un ensemble de variables d'entrées, de variables de sorties et de variables d'état. Les variables d'état sont choisies pour représenter le comportement « interne » du système. La façon dont les variables d'état évoluent dans le temps dépendent de leurs valeurs passées et des variables d'entrées, qui sont exogènes. Les valeurs des variables de sorties dépendent uniquement des variables d'état.

Les variables d'entrées, de sorties, et d'état sont reliées par un ensemble d'équations différentielles du premier ordre (équations aux différences dans le cas discret). L'estimation de ces quantités passe par la résolution du système d'équations, le plus souvent par une approche de type filtre de Kalman.

Les modèles espace-états possèdent un certain nombre d'avantages : ils permettent d'analyser la structure de la série (tendance, saisonnalité), le passage d'une version univariée à multivariée est directe, contrairement aux modèles de la famille ARIMA qui nécessite de proposer de nouveaux outils théoriques [93, 33]. Ces modèles ont de bonnes performances de prédiction

dans les situations de trafic habituelles, supérieures notamment aux modèles ARIMA [93] et SARIMA [33]. Le paramétrage de ces modèles reste cependant chronophage, ce qui complique leur application à un système complexe et volumineux comme un réseau urbain.

3.3 Méthodes d'apprentissage artificiel

3.3.1 Approches des k plus proches voisins

La méthode des k plus proches voisins (k-NN) appartient à la famille de méthodes non-paramétriques. Ces dernières permettent d'estimer une fonction de régression sans formuler d'hypothèse forte sur sa forme [5]. L'algorithme est basé sur une description d'un état de trafic, ainsi qu'une mesure de distance entre deux états. A partir de l'état courant (à l'instant t), il est ainsi possible de chercher dans l'historique les états passés les plus proches. Le nombre d'états retenus (appelés voisins) est noté k. C'est un paramètre estimé lors de l'apprentissage. Les valeurs dans le proche futur de ces états (déjà observé, donc présent dans l'historique) sont ensuite combinées (par exemple en calculant la moyenne) pour produire une prédiction.

La méthode de régression k-`NN` a été appliquée à la prévision de trafic sur autoroute à partir des années 1990 [22, 91] dans une formulation univariée. Elle fournit alors des résultats comparables à ceux d'un modèle ARIMA et d'un réseau de neurones à une couche cachée, voire meilleurs dans certains cas. Smith et al. [90] proposent plusieurs modifications de l'algorithme : (i) une nouvelle définition de l'état incluant la valeur historique moyenne pour l'instant de la journée considéré, (ii) plusieurs manières de combiner les futurs des k états voisins, en pondérant l'impact des voisins par la distance à l'état courant. Le modèle retenu est alors légèrement moins performant que le modèle SARIMA, mais présente l'avantage de ne pas nécessiter les efforts conséquents de spécification de ce dernier. Gong et Wang ajoutent dans la description de l'état observé par un capteur l'information des deux capteurs en amont (la topologie du réseau étant connue) [34]. Chang et al. soulignent l'importance de bien choisir le nombre de pas de temps utilisés pour décrire l'état courant [14]. Zheng et Su proposent de contraindre la sélection d'états voisins pour ne conserver que des états observés à une heure de la journée proche de celle de l'état courant. Ils présentent également une méthode pour diminuer l'impact des valeurs aberrantes lors de l'étape de combinaison des k valeurs futures [121]. Wu et al. montrent que les capteurs situés en aval sont importants dans la description de l'état courant, au même titre que ceux situés en amont [113] et obtiennent des résultats supérieurs à un réseau de

neurones à une couche cachée. Dell'Acqua et al. considèrent un nombre plus important de capteurs en amont et en aval du capteur à prédire, mais toujours sur un segment linéaire d'autoroute. Les auteurs intègrent également dans la mesure de distance une pénalité pour les états observés à un moment de la journée différent de l'état courant [23]. Cai et al. utilisent la régression k-NN pour prédire le trafic d'un district de Pékin [9]. Pour sélectionner les capteurs voisins spatialement, la notion de distance équivalente est introduite. Pour déterminer si deux capteurs sont proches, des propriétés du réseau physique sont utilisées (distance physique entre les deux capteurs, nombre de capteurs entre eux), ainsi que des propriétés des données réelles (corrélation entre les séries temporelles observées par ces capteurs). Des poids sont intégrés dans la mesure de distance entre deux états, par l'intermédiaire de deux fonctions gaussiennes : une pour considérer la différence d'instant de la journée, l'autre pour tenir compte de la distance équivalente entre les capteurs. Xia et al. présentent une adaptation MapReduce (Hadoop) de l'algorithme afin de distribuer l'apprentissage et paralléliser la prévision [115, 116].

3.3.2 Méthodes à noyaux

Les méthodes à noyaux permettent de transformer un classifieur linéaire en un classifieur non-linéaire, par l'intermédiaire d'un changement d'espace de représentation des données. La fonction noyau permet de calculer un produit scalaire dans le nouvel espace à partir de la représentation des données dans l'espace d'origine. Cela permet donc d'apprendre un classifieur linéaire dans un espace de grande dimension (parfois même infinie), sans avoir à calculer explicitement la projection des données dans cet espace.

Régression par séparateur à vaste marge (SVR) La régression par séparateur à vaste marge appartient à la famille des méthodes utilisant l'astuce du noyau (*Kernel trick*). Les machines à vecteur de support (SVM) sont les méthodes à *Kernel trick* les plus utilisées, principalement pour des problèmes de classification. La régression SVR est l'adaptation de ces méthodes pour résoudre des problèmes de régression. Elle combine les notions de fonction noyau et de marge maximale. Seul un sous-ensemble des exemples du jeu de données aura une influence sur la fonction de prédiction apprise par le modèle. Ces exemples sont appelés vecteurs supports.

Dans les dernières décennies, cette méthode à rencontré un grand succès en apprentissage artificiel [19, 102], notamment grâce à ses résultats sur des données en haute dimension, ses garanties théoriques, son petit nombre d'hyperparamètres et ses résultats probants dans de nombreuses applications.

Les premières applications de cette méthode pour la prévision de trafic sont proposées au début des années 2000 [2, 111, 101]. La version univariée est alors montrée plus performante que des méthodes basiques, comme la moyenne historique, ou bien la répétition de la dernière valeur observée. Des performances supérieures à un réseau de neurone simple (MLP) ont été observés pour un petit jeu de données (une journée), alors que des résultats comparables ont été observés avec un jeu de données plus grand (4 jours) [101]. Ces résultats sont reproduits en utilisant un modèle SVR dont les hyperparamètres sont fixés par un algorithme génétique [119]. Castro-Neto et al. proposent une version incrémentale (*online*) du SVR pour limiter le coût d'apprentissage du modèle et montrent que ce modèle est plus performant en situation inhabituelle que des modèles paramétriques comme les réseaux de neurones [10]. De nombreuses variantes de méta-heuristiques ont été proposées pour améliorer l'apprentissage des hyperparamètres [42, 18, 61, 41, 117]. Une adaptation du SVR pour prédire à l'échelle du réseau (un seul modèle pour tous les capteurs) est proposé dans la thèse de P.-A. Laharotte [59].

Régression à noyaux La régression à noyaux est plus simple que la régression SVR, dans le sens où on n'utilise pas la notion de marge maximale. Seul le noyau est utilisé pour apprendre un modèle de régression classique dans l'espace de haute-dimension induit par le noyau. Sun et al. proposent une version basée sur la distance de Mahalanobis [95]. Haworth et al. proposent une version incrémentale (*online*) et locale pour la prévision de trafic urbain [38].

3.3.3 Réseaux de neurones artificiels

La famille des réseaux de neurones comprend une grande variété d'architectures différentes [39]. Un réseau de neurones est composé de plusieurs composants simples, les neurones artificiels. Un neurone artificiel est une vue très simplifiée du fonctionnement d'un neurone biologique. Un neurone reçoit de l'information de plusieurs entrées. Ces entrées sont combinées par une fonction de combinaison (très souvent une somme pondérée), puis le résultat de cette combinaison est fournie en entrée de la fonction d'activation (souvent non-linéaire) qui déterminera la sortie du neurone. Les réseaux de neurones sont souvent organisés en couches successives, la sortie des neurones d'une couche servant d'entrée aux neurones de la couche suivante. Les neurones de la première couche correspondent aux données d'entrées de la tâche de prévision et la dernière couche fournit la réponse au problème de prévision. Les autres couches sont appelées couches cachées. Les paramètres du modèle sont les poids utilisés par la fonction de combinaison. Il peut aussi

y avoir des hyperparamètres quand la structure du réseau n'est pas fixée, mais qu'on souhaite l'apprendre. L'entraînement du réseau se fait traditionnellement par l'algorithme de rétropropagation du gradient, qui permet de calculer le gradient de l'erreur pour chaque neurone, en remontant de la dernière couche vers la première, et de corriger la valeur des poids associés aux entrées de chaque neurones. Une partie du travail de modélisation est de choisir la bonne structure du réseau, i.e. le nombre de couches, le nombre de neurones par couche, le choix des liens entre les neurones d'une couche et de la suivante (est-ce que tous les neurones seront une entrée de tous ceux de la couche suivante ou non ?), le choix de la fonction d'activation, ainsi que le choix d'hyperparamètres (par exemple pour la régularisation lors de l'apprentissage).

Lorsque les connections entres les neurones ne forment pas de boucles, on parle de réseaux à propagation directe. C'est le cas du perceptron multicouche (MLP) [39] qui est constitué d'une couche d'entrée, d'une ou plusieurs couches cachées et d'une couche de sortie. Certaines structures peuvent également être plus complexes en incluant des boucles (dans les réseaux récurrents [17], dont la sortie d'un neurone peut aussi être une entrée du même neurone) ou des blocs spécifiques (mémoire dans les réseaux LSTM [40] par exemple). Ce choix de structure peut être guidé par l'expérience du modélisateur et/ou par l'utilisation de métaheuristiques comme les algorithmes génétiques.

Depuis les années 2010, les réseaux de neurones profonds connaissent un grand succès dans de nombreux domaines, notamment la vision par ordinateur, la reconnaissance de la parole, le traitement automatique des langues naturelles, etc. Leur particularité est de posséder un grand nombre de couches cachées, et donc de paramètres. Pour entraîner ces réseaux, il est nécessaire de disposer d'une grande quantité de données et d'une grande puissance de calcul, généralement des processeurs graphiques (GPU). Cela explique que ces algorithmes profonds, dont les plus anciens datent des années 1970 et d'autres des années 1990 aient été redécouverts récemment grâce aux progrès effectués dans la collecte et le stockage des données, ainsi que dans la puissance de calcul. Les principaux inconvénients de ces réseaux complexes sont le risque de surapprentissage, les ressources et le temps de calcul importants, le risque de ne pas converger si on dispose d'un trop petit jeu de données, et la difficulté d'interprétation (on parle encore de boîte noire, même si la recherche sur l'explicabilité de ces réseaux commence à se développer).

Les réseaux de neurones ont été adoptés avec succès pour étudier le trafic depuis le début des années 1990. Dougherty [25] propose un résumé des travaux de cette époque. Plus récemment, Vlahogianni et al. ont produit en 2004 [104] et en 2014 [108] deux synthèses sur les travaux en prévision de trafic à court-terme listant, entre autres, les approches de type réseau

de neurones. Un grand nombre d'études utilisent un perceptron multicouche (MLP) avec une couche cachée [100, 49, 57, 60, 122, 107, 96, 105]. Un réseau de neurones récurrent (RNN) est utilisé dans [74]. D'autres contributions proposent des modèles hybrides, mélangeant filtre de Kalman et MLP avec des règles floues [94] ou du partitionnement flou (*fuzzy clustering*) avec un réseau de neurone [97]. Récemment, des approches d'apprentissage profond ont été proposées : une pile d'auto-encodeurs (SAE) suivie d'une couche de régression logistique pour la prévision dans [65], une combinaison de machine de Boltzmann restreinte (RBM) et d'un réseau récurrent dans [68] ou encore un réseau LSTM [98, 67, 120]. Enfin, une combinaison de réseau de neurones convolutifs (CNN) et de réseau récurrent a également été testée [114].

3.4 Différentes grilles de lectures

Les sections précédentes ont permis de présenter différentes familles d'approches pour la prévision de trafic, ainsi que la grande variété de raffinement des méthodes au sein de chaque famille. Cependant, on observe que tout au long de l'évolution du domaine, des grandes problématiques ou défis se sont posés aux chercheurs, indépendamment de la famille de méthodes étudiée. Vlahogianni et al. présentent ces défis dans une revue de l'état de l'art [108].

Le choix du voisinage spatio-temporel Les premiers travaux du domaine se sont concentrés sur la prévision du trafic sur un segment de route localisé. Les données d'entrées était alors limitées à une série temporelle, et seule l'activité passée du segment en question était utilisée pour la prévision. L'installation d'une succession de capteurs sur des segments consécutifs d'autoroute a permis de soulever la question de l'utilisation du voisinage spatial pour la prévision. Certains modèles univariés comme ARIMA n'étant pas adapté, cela a également encouragé une évolution vers des modèles plus sophistiqués. Dans un premier temps, l'information d'un ou deux capteurs en amont dans le trafic a été incorporées aux modèles, en suivant l'analogie entre l'écoulement du trafic et celui d'un fluide. Puis l'information des capteurs en aval a également été incorporée, ce qui est cohérent avec la notion d'onde de congestion qui se déplace dans le sens inverse du trafic.

Lorsque l'objet d'étude a évolué d'une portion linéaire d'autoroute vers de petits réseaux, on observe une évolution dans la façon de décrire le voisinage. La méthode STARIMA propose par exemple de définir des matrices de voisinages, similaire à une matrice d'adjacence de graphe, mais pondérée [52, 53]. Ces matrices doivent être définies par le modélisateur. Cette définition peut être directement liée à la connaissance de la topologie du réseau quand le ré-

seau étudié est de petite taille. Une étude préalable de la corrélation croisée entre les capteurs peut guider le modélisateur dans cette tâche. Les différentes matrices de voisinage correspondent au voisinage pertinent pour une prévision à un certain horizon. En ce sens, le voisinage identifié est qualifié de spatio-temporel.

La prévision à l'échelle d'un réseau entier fait apparaître la nécessité d'automatiser l'identification du voisinage pertinent. Les méthodes de sélection de variables sont utilisées, dont la régression Lasso [51]. Cette identification du voisinage est d'autant plus importante que le nombre de capteurs est grand, car si l'on conserve l'information de tous les capteurs, la dimensionnalité des données en entrée du modèle augmente et on court le risque du fléau de la haute dimension.

Le type de réseau Le passage historique de l'étude de portions d'autoroute à l'étude de réseaux urbains a ouvert de nouvelles questions méthodologiques. En effet, les dynamiques de trafic dans un réseau avec une forte demande sont complexes. A cela s'ajoute la présence de signalisation, parfois imprévisible [75]. Ces facteurs créent la nécessité d'utiliser des algorithmes robustes face à la complexité des dynamiques observées [96]. Le réseau n'est couvert que partiellement, ce qui peut poser des problèmes à des modèles faisant certaines hypothèses de conservation du flux de trafic. Certains travaux ont étudiés la question du meilleur emplacement pour des nouveaux capteurs [31]. Comme mentionné dans le paragraphe précédent, l'étude des réseaux urbains nécessite de modéliser les dépendances spatio-temporelles entre les capteurs.

La résolution temporelle Les mesures de trafic telles que le débit ou la vitesse moyenne sont mesurées à intervalle constant par les capteurs. Plus la résolution temporelle est élevée (par ex. 30 secondes), plus les données comportent du bruit, qui rend difficile l'apprentissage des modèles [79, 64, 108]. Si l'on diminue la résolution temporelle en agrégeant les données, il est possible d'éliminer une partie de ce bruit. Certaines propriétés statistiques des données peuvent disparaître comme la non-linéarité et la non-stationnarité [103]. Il existe un compromis entre le bruit qu'on arrive à enlever des données et l'information que l'on perd pour la prévision à court-terme. Il n'existe pas de méthode de référence pour choisir le pas d'agrégation temporelle, même si cela reste un choix important qui s'impose au modélisateur.

Il ressort de ces travaux, la nécessité actuelle de prévoir le trafic à court-terme à l'échelle d'un réseau urbain. Comme mentionné précédemment, la

sélection d'un voisinage spatio-temporel pertinent est une problématique centrale, qui nécessite d'être automatisée pour pouvoir passer à l'échelle quand la taille du réseau augmente. Il n'existe pas encore de méthodologie de référence pour résoudre ce problème. Des travaux concluants ont été obtenu avec la régression Lasso sur un réseau d'autoroutes. Cependant, le modèle Lasso est basé sur une hypothèse de dépendance spatio-temporelle linéaire entre les capteurs. Or, il est peu probable que cette hypothèse soit vérifiée dans le cadre du trafic urbain. Dans cette thèse, nous comparons la sélection de voisinage par la méthode Lasso avec une méthode non-paramétrique (`TiGraMITe`) issue du domaine de la physique et de l'étude des système complexes. Ces mécanismes de sélection de variables sont comparés sur un jeu de données de trafic urbain (Lyon) et sur une portion d'autoroute (Marseille).

On observe aussi une grande diversité dans les familles d'approches utilisées pour prédire le trafic, ainsi que dans les type de réseau sur lesquelles ces méthodes sont évaluées. Disposant d'un jeu de données de trafic urbain collecté dans un centre-ville européen historique (qui se distingue par sa topologie irrégulière de réseaux organisés en grilles comme par exemple à Manhattan) et d'un jeu de données d'autoroute urbaine, l'objectif de ces travaux est également de pouvoir mettre en lumière les avantages et inconvénients de ces différentes approches selon le contexte dans lesquelles elles sont appliquées.

Enfin, les travaux de la thèse ont également pour objectif d'apporter une contribution à la problématique du choix de la résolution temporelle des données fournies en entrée aux modèles. Pour cela, nous proposons une étude empirique sur le jeu de données de Marseille, pour lequel nous avons fait varier le pas d'agrégation temporel.

Le prochain chapitre présente de façon plus formelle et détaillée les différentes méthodes de prévision et de sélection de voisinage qui ont été étudiées au cours de la thèse.

Chapitre 4

Méthodes de prévision

Sommaire

4.1 Régression linéaire 46
4.2 Régression Ridge 48
4.3 Régression lasso 49
4.4 Combinaisons polynomiales de variables 50
4.5 Régression à vecteurs de support 51
4.5.1 Cas linéaire 51
4.5.2 Cas non-linéaire – utilisation du noyau 55
4.6 Régression k-NN 58
4.7 ARIMA 60
4.7.1 Modèle autorégressif AR(p) 60
4.7.2 Modèle moyenne mobile MA(q) 60
4.7.3 ARMA(p,q) 61
4.7.4 ARIMA(p,q,d) 62
4.8 Modèle Vecteur Autorégressif (VAR(p)) 63

Dans ce chapitre, nous présentons en détail les méthodes de prévision que nous allons comparer par la suite. La section 4.1 introduit la régression linéaire classique, puis des variations de cette méthode utilisant un terme de pénalité sont présentées : la régression ridge en section 4.2 et la régression lasso en section 4.3. Une méthode permettant d'introduire de la non-linéarité dans le modèle de régression est présentée en section 4.4 et sert d'introduction à la section 4.5 sur la régression à vecteur support (SVR). La notion de non-linéarité et son intérêt pour la prévision de trafic est présentée dans cette section sur le SVR. La méthode de régression k-`NN` est présentée en section 4.6. Enfin, les modèles classiques de séries temporelles sont présentés : le modèle ARIMA en section 4.7 et le modèle VAR en section 4.8.

4.1 Régression linéaire

Notation Pour faciliter l'écriture des vecteurs colonnes dans un paragraphe de texte, ils seront écrits sous la forme suivante :

$$v \in \mathbb{R}^d = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_d \end{pmatrix} = (v_1, \ldots, v_d)^\intercal \tag{4.1}$$

Le jeu d'apprentissage est composé des exemples $\{(x_{i1}, \ldots, x_{id}, y_i)\}_{i=1}^T$. Pour simplifier la modélisation, on introduit également $x_{i0} = 1$. On a ainsi :

$$\mathbf{x}_i = (1, x_{i1}, \ldots, x_{id})^\intercal \in \mathbb{R}^{d+1}, \quad \text{et} \quad y_i \in \mathbb{R} \tag{4.2}$$

L'hypothèse est qu'il existe une relation linéaire entre les composantes du vecteur d'entrée $\mathbf{x}$ et de la variable de sortie y, que l'on souhaite modéliser. On suppose l'existence d'une variable aléatoire non observée ϵ, qui ajoute du bruit à cette relation linéaire. Le modèle de régression linéaire prend la forme suivante :

$$y_i = \beta_0 1 + \beta_1 x_{i1} + \cdots + \beta_d x_{id} + \epsilon_i \tag{4.3}$$

Les composantes du vecteur $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_d)^\intercal$ sont les paramètres du modèle, que l'on souhaite estimer. Le terme β_0 correspond à l'ordonnée à l'origine, c'est à dire la valeur de y_i quand les variables $x_1, \ldots, x_d$ sont toutes nulles. On peut réécrire le modèle sous une forme plus compacte :

$$y_i = \mathbf{x}_i^\intercal \boldsymbol{\beta} + \epsilon_i \tag{4.4}$$

où $\mathbf{x}_i^\intercal \boldsymbol{\beta}$ est le produit scalaire entre $\mathbf{x}_i$ et $\boldsymbol{\beta}$.

Enfin, il est courant de regrouper toutes ces équations en une expression matricielle :

$$Y = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{4.5}$$

avec :

$$\mathbf{X} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1d} \\ 1 & x_{21} & \cdots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{T1} & \cdots & x_{Td} \end{pmatrix}, \quad Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_T \end{pmatrix}, \quad \boldsymbol{\epsilon} = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_T \end{pmatrix} \tag{4.6}$$

Estimation des paramètres

Il existe plusieurs méthodes pour estimer le vecteur de paramètre $\boldsymbol{\beta}$. La plus courante est la méthode des moindres carrés ordinaires. L'objectif est alors de minimiser la somme du carré de l'erreur de prédiction du modèle sur les exemples du jeu d'entraînement :

$$\sum_{i=1}^{T} \epsilon_i^2 = \sum_{i=1}^{T} (y_i - \hat{f}(\mathbf{x}_i))^2 = \sum_{i=1}^{T} (y_i - \mathbf{x}_i^\intercal \boldsymbol{\beta})^2 \tag{4.7}$$

On cherche donc :

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^{d+1}}{\arg\min} \sum_{i=1}^{T} (y_i - \mathbf{x}_i^\intercal \boldsymbol{\beta})^2 = \underset{\boldsymbol{\beta} \in \mathbb{R}^{d+1}}{\arg\min} \|Y - \mathbf{X}\boldsymbol{\beta}\|^2 \tag{4.8}$$

Ce problème de minimisation admet la solution de forme close suivante :

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\intercal \mathbf{X})^{-1} \mathbf{X} Y \tag{4.9}$$

sous réserve de certaines conditions :

• **Linéarité** Il faut que le modèle soit correctement spécifié, c.à.d. que la variable prédite soit réellement une combinaison linéaire de la variable d'entrée. Nous verrons plus tard qu'il est possible d'introduire de la non-linéarité par un prétraitement, en calculant de nouvelles variables par des combinaisons ou transformations des variables d'entrée.

• **Exogénéité stricte** $\mathbb{E}[\epsilon_i | \mathbf{X}] = 0$ pour $(i = 1, \ldots, T)$. Cette condition a plusieurs implications :
— la moyenne du terme d'erreur est nulle : $\mathbb{E}[\epsilon_i] = 0$
— les variables d'entrées ne sont pas corrélées avec le terme d'erreur : $\mathbb{E}[\mathbf{X}^\intercal \boldsymbol{\beta}] = 0$
Dans la pratique, cette hypothèse est surtout importante si l'on souhaite tirer des interprétations de type « cause à effet » entre les variables d'entrées et la variable de sortie. Si l'on souhaite simplement faire des prévisions, sans les interpréter, il est possible d'ignorer cette hypothèse.

• **Variance sphérique de l'erreur** Cette hypothèse se décompose en deux partie :
— homoscédasticité : $\mathrm{Var}[\epsilon_i | X] = \sigma^2$ pour $(i = 1, \ldots, T)$
— absence d'autocorrélation : $\mathrm{Cov}(\epsilon_i, \epsilon_j | X) = 0$.

• **Absence de multicolinéarité parfaite** Il y a multicolinéarité parfaite lorsque l'une des variables d'entrée peut s'exprimer exactement comme une combinaison linéaire des autres variables d'entrées. Dans ce cas, la méthode des moindres carrés ordinaires ne permet pas d'estimer le vecteur $\boldsymbol{\beta}$. L'absence de multicolinéarité parfaite peut s'exprimer par la propriété suivante : $\mathrm{rg}(\mathbf{X}) = d + 1$. En pratique, la multicolinéarité parfaite peut arriver lorsque des variables qualitatives ont été codées de façon redondante, mais la plupart des logiciels de régression repèrent ces redondances et appliquent certaines stratégies pour les corriger [29].

4.2 Régression Ridge

La régression ridge (également appelée régularisation de Tikhonov) permet de réduire la taille des coefficients en ajoutant une pénalité dans la fonction de coût que l'on minimise. Dans un modèle de régression linéaire avec beaucoup de variable corrélées, il est possible que l'on obtienne de mauvaises estimations de certains coefficients, par exemple avec un très grand coefficient pour une variable compensé par un très grand coefficient négatif pour une autre variable corrélée. Cela introduit une variance élevée des estimateurs $\{\beta_i\}_{i=1}^d$, ce qui n'est pas souhaitable. Ce problème est corrigé grâce à la contrainte de taille imposée par la régression ridge. Le problème à résoudre se formule comme suit :

$$\hat{\boldsymbol{\beta}}^{\text{ridge}} = \arg\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{T} (y_i - \beta_0 - \sum_{j=1}^{d} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{d} \beta_j^2 \right\}, \tag{4.10}$$

avec $\lambda \geq 0$ un paramètre de régularisation qui contrôle à quel point on contraint les coefficients. On peut également le réécrire sous la forme équivalente :

$$\begin{aligned} \hat{\boldsymbol{\beta}}^{\text{ridge}} = \arg\min_{\boldsymbol{\beta}} & \left\{ \sum_{i=1}^{T} (y_i - \beta_0 - \sum_{j=1}^{d} x_{ij}\beta_j)^2 \right\} \\ & \text{avec} \sum_{j=1}^{d} \beta_j^2 \leq t, \end{aligned} \tag{4.11}$$

dans laquelle t joue un rôle équivalent à λ dans la formulation précédente. On voit ici apparaître plus explicitement la contrainte de taille imposée sur la somme des coefficients au carré. Notons que le coefficient β_0, qui représente l'ordonnée à l'origine n'est pas inclut dans la pénalité.

Estimation des paramètres En pratique, l'estimation est effectuée en deux temps. Dans un premier temps, on calcule une estimation de l'ordonnée à l'origine par la formule $\hat{\beta}_0 = \frac{1}{T}\sum_{i=1}^{T} y_i$, puis on apprend une régression ridge sans ordonnée à l'origine, sur la version centrée $(\mathbf{X}_c)$ des variables d'entrée (i.e. dont on a soustrait la moyenne empirique), notée comme suit :

$$\mathbf{X}_c = \begin{pmatrix} x_{11} - \bar{x}_1 & \cdots & x_{1d} - \bar{x}_d \\ x_{21} - \bar{x}_1 & \cdots & x_{2d} - \bar{x}_d \\ \vdots & \ddots & \vdots \\ x_{T1} - \bar{x}_1 & \cdots & x_{Td} - \bar{x}_d \end{pmatrix}, \quad \text{avec } \bar{x}_j = \frac{1}{T}\sum_{i=1}^{T} x_{ij} \tag{4.12}$$

On peut alors exprimer la solution de ce problème par la formule :

$$\hat{\boldsymbol{\beta}}^{\text{ridge}} = (\mathbf{X}_c^\intercal \mathbf{X}_c + \lambda \mathbf{I})^{-1} \mathbf{X}_c Y,$$

avec $\hat{\boldsymbol{\beta}}^{\text{ridge}} = (\hat{\beta}_1, \ldots, \hat{\beta}_d)^\intercal$, car $\hat{\beta}_0$ est appris séparément, et $\mathbf{I}$ est la matrice identité $d \times d$. L'ajout du coefficient positif λ aux élements diagonaux de la matrice $\mathbf{X}_c^\intercal \mathbf{X}_c$, permet de s'assurer que la matrice sera inversible, même si $rg(\mathbf{X}_c^\intercal \mathbf{X}_c) < d$. C'est cette motivation qui était à l'origine des premiers travaux sur la régression ridge.

La régression ridge a donc la capacité d'améliorer l'estimation des paramètres, en diminuant la variance de l'estimateur (au prix d'un léger biais) par la pénalisation des trop gros coefficients. Mais il lui reste cependant l'inconvénient de ne pas sélectionner un sous-ensemble des variables. Les coefficients, bien que plus faibles, ne tendent pas nécessairement vers zéro. Or pour des raisons d'interprétation et pour améliorer la précision de la prédiction, on souhaite parfois ne conserver qu'un sous-ensemble des variables d'entrées.

4.3 Régression lasso

La régression lasso est une méthode de régression linéaire avec une pénalité, comme la régression ridge. Dans le cas du lasso, la pénalité porte sur la valeur absolue des coefficients, contrairement au carré dans le cas ridge. On obtient donc l'estimateur :

$$\hat{\boldsymbol{\beta}}^{\text{lasso}} = \arg\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{T} (y_i - \beta_0 - \sum_{j=1}^{d} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{d} |\beta_j| \right\} \tag{4.13}$$

Comme pour la régression ridge, on peut réécrire ce problème sous la forme suivante :

$$\hat{\boldsymbol{\beta}}^{\text{lasso}} = \arg\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{T} (y_i - \beta_0 - \sum_{j=1}^{d} x_{ij}\beta_j)^2 \right\} \quad \text{avec} \sum_{j=1}^{d} |\beta_j| \leq t, \tag{4.14}$$

Ce nouveau problème ne possède pas de solution en forme close, mais il existe des algorithmes permettant de calculer les solutions pour un chemin de valeurs du paramètre λ avec le même coût de calcul que la régression ridge.

La contrainte $\sum_{j=1}^{d} |\beta_j| \leq t$, appelée pénalité L_1, assure que pour des valeurs de t suffisamment petites, certains coefficients seront nuls.

4.4 Combinaisons polynomiales de variables

Cette section a pour objectif de présenter une approche permettant d'introduire de la non-linéarité dans un modèle. Elle sert ainsi d'introduction à la section suivante sur la régression à vecteurs de support, dans laquelle on présentera plus en profondeur cette notion de non-linéarité et son intérêt pour la prévision de trafic.

On a mentionné précédemment que le modèle de régression linéaire est en réalité linéaire par rapport à ses paramètres et non par rapport aux variables d'entrées. On signifie par cette affirmation que la méthodologie reste valide si on applique des transformations aux variables d'entrées. Prenons par exemple le modèle simple :

$$y = \beta_0 + \beta_1 x + \epsilon \tag{4.15}$$

Ce modèle est linéaire par rapport à la variable d'entrée x. Mais on pourrait lui préférer le modèle suivant :

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \epsilon \tag{4.16}$$

On a rendu le modèle non-linéaire par rapport à la variable d'entrée x, mais il est toujours linéaire par rapport aux coefficients $\{\beta_i\}$, et on peut donc appliquer la même méthodologie que précédemment pour estimer les paramètres du modèle.

La manière la plus classique d'introduire de la non-linéarité est de créer de nouvelles variables qui sont des combinaisons polynomiales de degré au

plus k des variables d'entrées. Notons cette transformation Φ^k. Pour deux variables d'entrées X_1 et $X2$, on aura alors :

$$\begin{aligned}\Phi^1(X_1, X_2) =&(1, X_1, X_2)\\ \Phi^2(X_1, X_2) =&(1, X_1, X_2, X_1^2, X_1X_2, X_2^2)\\ \Phi^3(X_1, X_2) =&(1, X_1, X_2, X_1^2, X_1X_2, X_2^2, X_1^3, X_1^2X_2, X_1X_2^2, X_2^3)\end{aligned} \tag{4.17}$$

Il est important de noter que le nombre de variables ainsi créées croit de façon polynomiale par rapport au nombre de variables d'entrées et de façon exponentielle par rapport au degré max k. Il faut donc faire attention à ne pas augmenter trop le degré si l'on veut éviter le risque de sur-apprentissage.

Le modèle appris suite à la transformation Φ^2 serait donc le suivant :

$$Y = \beta_0 + \beta_1X_1 + \beta_2X_2 + \beta_3X_1^2 + \beta_4X_1X_2 + \beta_5X_2^2 + \epsilon \tag{4.18}$$

On peut choisir d'estimer ce modèle par une régression linéaire classique ou par une régression ridge ou lasso.

4.5 Régression à vecteurs de support

Dans cette section, nous présentons la régression à vecteurs de support (SVR), qui permet d'introduire de la non-linéarité dans le modèle de régression par l'utilisation d'une fonction noyau.

4.5.1 Cas linéaire

Dans cette section, on a $\mathbf{x}_i = (x_{i1}, \ldots, x_{id}) \in \mathbb{R}^d$. On cherche à apprendre une fonction de la forme :

$$\begin{aligned}\hat{y}_i &= \hat{f}(\mathbf{x}_i)\\ &= \sum_{j=1}^{d} w_j x_{ij} + b\\ &= \langle \mathbf{w}, \mathbf{x}_i \rangle + b\end{aligned} \tag{4.19}$$

où $\langle \cdot, \cdot \rangle$ représente le produit scalaire dans $\mathbb{R}^d$ et $b \in \mathbb{R}$ est une constante.

L'objectif est d'apprendre une fonction $\hat{f}$ telle que l'erreur de prédiction est inférieure à une valeur ϵ fixée, pour tous les y_i du jeu d'apprentissage. Ceci est équivalent à dire que les exemples $(\mathbf{x}_i, y_i)$ sont tous compris dans un ϵ-tube autour de la fonction de régression $\hat{f}$. Comme dans le cas de la régression ridge, on souhaite garder le vecteur de paramètres $\mathbf{w}$ petit en minisant sa norme $\|\mathbf{w}\|$, ce qui nous amène au problème d'optimisation convexe suivant :

$$\text{minimiser} \quad \frac{1}{2}\|\mathbf{w}\|^2 \tag{4.20}$$

$$\text{sous les contraintes} \quad |y_i - \hat{f}(\mathbf{x}_i)| \leq \epsilon \tag{4.21}$$

Ces dernières contraintes se réécrivent sous la forme suivante :

$$\begin{cases} y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b \leq \epsilon \\ \langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i \leq \epsilon \end{cases} \tag{4.22}$$

Cette formulation du modèle suppose qu'il existe réellement une fonction $\hat{f}$ qui approxime avec une précision ϵ tous les exemples $(\mathbf{x}_i, y_i)$ du jeu d'entraînement, c.à.d. que le problème admet une solution. Il peut arriver que ce ne soit pas le cas, et on peut aussi souhaiter tolérer certaines violations de cette contrainte, par exemple en présence d'observations aberrantes. A cet effet, il faut reformuler le modèle de manière à faire apparaître un compromis entre l'objectif initial, qui est de minimiser $\|\mathbf{w}\|$, et la quantité de violation de la contrainte que l'on s'autorise. Cela se fait par l'introduction de variables ressorts ξ_i et ξ_i^* et d'un coût C. Le problème se formule comme suit :

$$\begin{aligned} &\text{minimiser} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{T}(\xi_i + \xi_i^*) \\ &\text{sous les contraintes} \quad \begin{cases} y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b \leq \epsilon + \xi_i \\ \langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i* \geq 0 \end{cases} \end{aligned} \tag{4.23}$$

Cela correspond à l'utilisation d'une ϵ-fonction de perte insensible :

$$|\xi|_\epsilon = \begin{cases} 0, & \text{si } |\xi| \leq \epsilon \\ |\xi| - \epsilon, & \text{sinon.} \end{cases} \tag{4.24}$$

Formulation duale du problème Le problème défini précédemment est souvent plus facile à résoudre dans sa formulation duale, obtenue par la méthode des multiplicateurs de Lagrange. Cette méthode permet de trouver le minimum (ou maximum) d'une fonction dérivable, sous contrainte.

L'idée est d'intégrer les contraintes dans la fonction objectif en construisant une nouvelle fonction, appelée lagrangien. Pour construire cette fonction, on introduit de nouvelles variables $\alpha_i, \alpha_i^*, \eta_i, \eta_i^* \geq 0$, nommées multiplicateurs de Lagrange, ou encore variables duales. Par opposition, les variables du problème d'origine $(\mathbf{w}, b, \xi_i, \xi_i*)$ sont appelées variables primales. Le lagrangien

est donné par la formule suivante :

$$\begin{aligned} L := \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{T}(\xi_i + \xi_i^*) - \sum_{i=1}^{T}(\eta_i\xi_i + \eta_i^*\xi_i^*) \\ - \sum_{i=1}^{T}\alpha_i(\epsilon + \xi_i - y_i + \langle \mathbf{w}, \mathbf{x}_i\rangle + b) \\ - \sum_{i=1}^{T}\alpha_i^*(\epsilon + \xi_i^* + y_i - \langle \mathbf{w}, \mathbf{x}_i\rangle - b) \end{aligned} \tag{4.25}$$

Dans cette formule, les deux termes de gauche correspondent à la fonction objectif de l'équation (4.23), et les termes de droites correspondent aux contraintes.

On peut montrer que cette fonction admet un point-selle au niveau de la solution. Il en découle que ses dérivées partielles par rapport aux variables primales ($\frac{\partial L}{\partial \mathbf{w}}, \frac{\partial L}{\partial b}, \frac{\partial L}{\partial \xi_i}, \frac{\partial L}{\partial \xi_i^*}$) doivent être annulées. Ces conditions vont permettre de simplifier la formule du lagrangien. Dans un premier temps, écrivons-les :

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{T}(\alpha_i^* - \alpha_i)\mathbf{x}_i = 0 \tag{4.26}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{T}(\alpha_i^* - \alpha_i) = 0 \tag{4.27}$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \eta_i = 0 \tag{4.28}$$

$$\frac{\partial L}{\partial \xi_i^*} = C - \alpha_i^* - \eta_i^* = 0 \tag{4.29}$$

Tout d'abord, observons que les équations (4.28) et (4.29) permettent de réécrire les variables η_i et η_i^* en fonction des variables C, α_i, α_i^*. Ainsi, il sera possible de les faire disparaître de la formule du lagrangien, par une substitution. En outre, ces deux équations et le fait que les variables duales sont positives permettent de déduire que $\alpha_i, \alpha_i^* \in [0, C]$. En substituant ces quatre équations dans l'expression du lagrangien (4.25), on obtient le problème d'optimisation suivant :

$$\begin{aligned} \underset{\alpha_i,\alpha_i^*}{\text{maximiser}} \quad & -\frac{1}{2}\sum_{i,j=1}^{T}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)\langle \mathbf{x}_i,\mathbf{x}_j\rangle \quad -\epsilon\sum_{i=1}^{T}(\alpha_i+\alpha_i^*) \\ & \qquad\qquad +\sum_{i=1}^{T} y_i(\alpha_i-\alpha_i^*) \qquad (4.30) \\ \text{sous les contraintes} \quad & \begin{cases} \sum_{i=1}^{T}(\alpha_i^*-\alpha_i)=0 \\ \alpha_i,\alpha_i^*\in[0,C] \end{cases} \end{aligned}$$

L'équation (4.26) permet de remplacer $\mathbf{w}$ dans l'écriture du modèle 4.31, qui s'écrit alors sous la forme suivante :

$$\begin{aligned} \hat{y} &= \hat{f}(\mathbf{x}) \\ &= \sum_{i=1}^{T}(\alpha_i-\alpha_i^*)\langle \mathbf{x}_i,\mathbf{x}\rangle + b \end{aligned} \qquad (4.31)$$

On observe que lors de l'optimisation, ou pour effectuer une prédiction, on n'a pas directement besoin des valeurs des $\mathbf{x}_i$, mais simplement du résultat d'un produit scalaire entre eux. Cette observation est à l'origine de la technique nommée astuce du noyau (*Kernel Trick*), qui sera utile pour rendre la fonction de prédiction non-linéaire.

L'estimation des paramètres α_i, α_i^* a permis d'estimer $\mathbf{w}$. Cependant, il reste encore à estimer le paramètres b. Les conditions de Karush-Kuhn-Tucker généralisent la méthode des multiplicateurs de Lagrange, au cas où l'on a des contraintes d'inégalité. Ces conditions sont les suivantes :

$$\begin{aligned} \alpha_i(\epsilon+\xi_i-y_i+\langle \mathbf{w},\mathbf{x}_i\rangle+b)=0 \\ \alpha_i^*(\epsilon+\xi_i^*+y_i-\langle \mathbf{w},\mathbf{x}_i\rangle-b)=0 \end{aligned} \qquad (4.32)$$

et

$$\begin{aligned} (C-\alpha_i)\xi_i=0 \\ (C-\alpha_i^*)\xi_i^*=0 \end{aligned} \qquad (4.33)$$

Rappelons-nous que l'un des coefficients ξ_i et ξ_i^* est non-nul si l'exemple $(\mathbf{x}_i, y_i)$ est en dehors de l'ϵ-tube. Comme chacun de ces coefficients correspond à un côté de l'ϵ-tube, ils ne peuvent pas être tous les deux non-nuls, i.e. $\xi_i\xi_i^*=0$.

Pour tout exemple en dehors de l'ϵ-tube, la condition (4.33) permet alors de déduire que le coefficient α_i (ou α_i*) associé à la variable non-nulle ξ_i (ou ξ_i^*) est nécessairement égal à C.

Pour tout exemple à l'intérieur de l'ϵ-tube, les termes de droites de la condition (4.32) sont non-nuls. Il en découle que les coefficients α_i et α_i^* sont nuls.

À la lumière de ces nouvelles conclusions, en reprenant la formule (4.31), on constate que seuls les exemples en dehors de l'ϵ-tube participent à la construction de la fonction de régression $\hat{f}$. On appelle ces exemples les vecteurs supports. La quantité de vecteurs supports détermine en quelque sorte la complexité de la fonction de régression.

Pour déterminer b, on s'intéresse aux différentes valeurs possibles pour α_i et α_i^* et ce que cela signifie en terme de position de l'exemple $(\mathbf{x}_i, y_i)$ par rapport à l'ϵ-tube.

Chaque point n'appartenant pas exactement à la frontière de l'ϵ-tube, vérifie deux de ces conditions :

$$\alpha_i < C \implies y_i - \hat{f}(\mathbf{x}_i) \leq \epsilon \qquad \implies -\epsilon + y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle \leq b \tag{4.34}$$

$$\alpha_i^* > 0 \implies y_i - \hat{f}(\mathbf{x}_i) \geq -\epsilon \qquad \implies -\epsilon + y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle \leq b \tag{4.35}$$

$$\alpha_i > 0 \implies y_i - \hat{f}(\mathbf{x}_i) \geq \epsilon \qquad \implies -\epsilon + y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle \geq b \tag{4.36}$$

$$\alpha_i^* < C \implies y_i - \hat{f}(\mathbf{x}_i) \leq -\epsilon \qquad \implies -\epsilon + y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle \geq b \tag{4.37}$$

Cela permet de donner un encadrement de b :

$$\begin{aligned} \max_i \{-\epsilon + y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle |\, \alpha_i < C \text{ ou } \alpha_i* > 0\} \leq b \leq \\ \min_i \{-\epsilon + y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle |\, \alpha_i > 0 \text{ ou } \alpha_i* < C\} \end{aligned} \tag{4.38}$$

Si certains point appartiennent exactement à la frontière de l'ϵ-tube, les inégalités deviennent des égalités.

4.5.2 Cas non-linéaire – utilisation du noyau

Non-linéarité De nombreux travaux ont mis en évidence la présence de non-linéarité dans les données de trafic. Le terme non-linéarité est un peu ambigu, car il se définit d'abord par une opposition : une série temporelle est non-linéaire si on ne peut pas la modéliser en utilisant des approches linéaires [37]. Dans les données de trafic, on peut identifier différentes formes de non-linéarité comme les transitions abruptes entre les états de trafic (fluide, congestionné) et les fluctuations intenses autour d'une tendance plus linéaire [89]. La variance non constante de ces fluctuations est une forme de non-linéarité nommée hétéroscédasticité. Ce comportement est présent dans les données de trafic, avec une variance plus importante lors des pics d'utilisation du réseau, le matin et en fin d'après-midi [106, 15]. Une possibilité serait de lisser les données pour éliminer les fluctuations à trop haute

fréquence, mais cette approche est contraire à la nature du phénomène de trafic. On court alors le risque de perdre de l'information utile à l'approche de la congestion, ce qui impacte peu les prévisions à long terme mais est crucial pour la prévision à court-terme où l'on veut prédire précisément ces changements d'état [106].

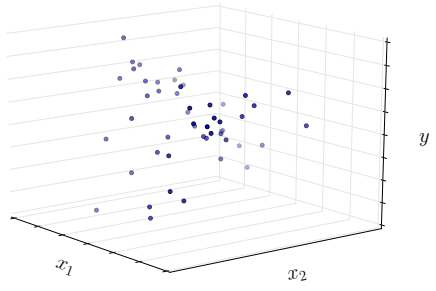
Quand la structure des données dans l'espace d'entrée est trop compliquée pour apprendre un modèle linéaire, il est possible de projeter les données dans un nouvel espace dans lequel on espère obtenir une structure plus simple. Un exemple a été fourni dans la partie 4.4 sur les combinaisons polynomiales de variables. On notera cette projection $\phi : \mathbb{R}^d \to \mathcal{F}$, où $\mathcal{F}$ est un espace vectoriel muni d'un produit scalaire. Si $\mathcal{F}$ est de dimension finie d', l'image d'un exemple $\mathbf{x}_i \in \mathbb{R}^d$ sera notée $\Phi(\mathbf{x}_i) = (\Phi_1(\mathbf{x}_i), \Phi_2(\mathbf{x}_i), \ldots, \Phi_{d'}(\mathbf{x}_i))$. En réalité, Φ n'est pas vraiment une projection au sens mathématique, mais plutôt une application de $\mathbb{R}^d$ dans $\mathcal{F}$. Malgré cela, le terme « projection » est souvent employé pour décrire un tel changement de représentation. On peut maintenant s'intéresser à ce que devient le problème d'optimisation du SVR quand on ajoute ce changement de représentation :

$$\begin{aligned} \underset{\alpha_i,\alpha_i^*}{\text{maximiser}} \quad & -\frac{1}{2}\sum_{i,j=1}^{T}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)\langle\Phi(\mathbf{x}_i),\Phi(\mathbf{x}_j)\rangle \quad -\epsilon\sum_{i=1}^{T}(\alpha_i+\alpha_i^*) \\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad +\sum_{i=1}^{T} y_i(\alpha_i-\alpha_i^*) \\ \text{sous les contraintes} \quad & \begin{cases} \sum_{i=1}^{T}(\alpha_i^*-\alpha_i)=0 \\ \alpha_i,\alpha_i^* \in [0,C] \end{cases} \end{aligned} \tag{4.39}$$

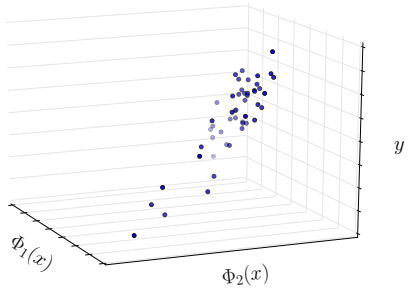
On note que la seule différence est l'apparition de Φ au sein du produit scalaire. Qu'en est-il du vecteur $\mathbf{w}$ et de la formule de $\hat{f}$? On a :

$$\begin{aligned} \mathbf{w} &= \sum_{i=1}^{T}(\alpha_i-\alpha_i^*)\Phi(\mathbf{x}_i) \\ \hat{f}(\mathbf{x}) &= \sum_{i=1}^{T}(\alpha_i-\alpha_i^*)\langle\Phi(\mathbf{x}_i),\Phi(\mathbf{x})\rangle + b \end{aligned} \tag{4.40}$$

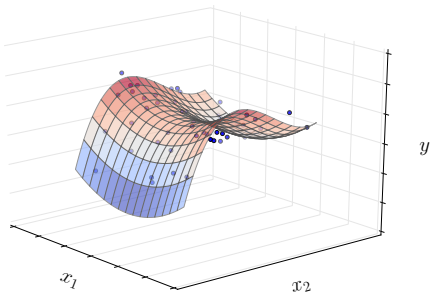
On remarque que seule l'expression de $\mathbf{w}$ nécessite de connaître explicitement la valeur de $\Phi(\mathbf{x}_i)$. La résolution du problème d'optimisation, ainsi que le calcul de la prédiction $\hat{f}(\mathbf{x})$ font simplement intervenir le résultat du produit scalaire $\langle\Phi(\mathbf{x}_i),\Phi(\mathbf{x})\rangle$.



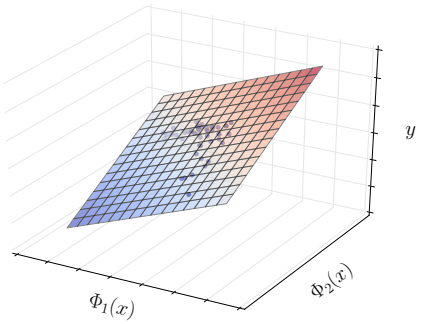
(a) Espace de départ



(b) Espace image



(c) Représentation de la fonction $\hat{f}$ dans l'espace de départ



(d) Représentation de la fonction $\hat{f}$ dans l'espace image

FIGURE 4.1 – Illustration de l'utilisation de la projection dans un nouvel espace. Dans cet exemple, on a $\Phi : \mathbb{R}^2 \to \mathbb{R}^2$.

Noyau Admettons que l'on connaisse une expression analytique permettant de calculer ce produit scalaire à partir de la représentation initiale des données. Cette fonction est appelée un noyau :

$$\begin{aligned} &k : \mathbb{R}^d \to \mathbb{R} \\ &k(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle \end{aligned} \tag{4.41}$$

L'utilisation de la fonction noyau permet d'apprendre un modèle dans $\mathcal{F}$ sans jamais avoir besoin de calculer la représentation des exemples dans cet espace. On dit que l'espace $\mathcal{F}$ est induit par le noyau k. Certaines fonctions noyaux permettent par exemple d'apprendre un modèle dans un espace induit de dimension infinie.

Noyau linéaire On parle de noyau linéaire quand on apprend le modèle sans transformer les données d'entrée. Le noyau se résume donc au produit scalaire dans l'espace de départ :

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle \tag{4.42}$$

Noyau RBF Le noyau RBF (*Radial Basis Function*) est très populaire et est utilisé quand on n'a pas d'a priori sur les données. Il correspond à une mesure de similarité entre exemples :

$$k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2} \tag{4.43}$$

où γ est un paramètre qui contrôle la complexité du modèle. Si γ est élevé, seuls les exemples dans un voisinage proche de $\mathbf{x}$ auront une forte influence sur la prévision $\hat{f}(\mathbf{x})$. Ainsi le modèle pourra apprendre beaucoup de propriétés locales des données, et sera plus complexe. Si γ est très faible, une grande partie des exemples auront une influence sur $\mathbf{x}$, le modèle étant donc plus global et contraint, donc dans un sens moins complexe.

L'espace $\mathcal{F}$ induit par ce noyau est de dimension infinie. Une des propriétés de ce noyau est de pénaliser toutes les dérivées de $\hat{f}$, ce qui permet d'estimer une fonction très régulière (lisse), ce qui est important pour diminuer l'erreur de généralisation. Cette propriété est très intéressante en pratique et n'est pas triviale, car apprendre une fonction plane dans un certain espace à haute dimension, ne garantit pas nécessairement qu'elle corresponde à une fonction simple dans un espace de faible dimension.

4.6 Régression k-NN

La formulation la plus simple de la prédiction k-NN est la suivante :

$$\hat{y} = \hat{f}(\mathbf{x}) = \frac{1}{k} \sum_{\mathbf{x}_i \in \mathcal{N}_k(\mathbf{x})} y_i \tag{4.44}$$

où $\mathcal{N}_k(x)$ est le voisinage de taille k du point x, c.à.d. l'ensemble contenant les k exemples d'entraînement les plus proches du point x. La tâche de prédiction consiste donc à identifier les entrées x_i appartenant à ce voisinage, puis de produire une prédiction en calculant la moyenne des sorties y_i correspondantes.

Quel est x dans le cas des séries temporelles ?
Dans le cas univarié :

$$\begin{cases} y = z(t+h) \\ \mathbf{x} = (z(t), \ldots, z(t-d)) \end{cases} \tag{4.45}$$

Dans le cas multivarié :

$$\begin{cases} y = (z_1(t+h), \ldots, z_N(t+h)) \\ \mathbf{x} = (z_1(t), \ldots, z_1(t-d), \ldots, z_N(t), \ldots, z_n(t-d)) \end{cases} \tag{4.46}$$

Dans le cas on a sélectionné p variables spatio-temporelles pertinentes pour prédire le futur d'une série z_i :

$$\begin{cases} y = z_i(t+h) \\ \mathbf{x} = \big(z_{i_1}(t-\tau_1), \ldots, z_{i_p}(t-\tau_p)\big) \end{cases} \tag{4.47}$$

avec i_j et τ_j correspondant respectivement à l'identifiant de la série temporelle et au décalage temporel de la j^{e} variable spatio-temporelle sélectionnée.

Identification de $\mathcal{N}_k(\mathbf{x})$ La détermination d'un voisinage pour l'exemple $\mathbf{x}$ nécessite la définition d'une mesure de distance ou de similarité dans l'espace d'entrée. Le choix le plus classique consiste à utiliser une distance euclidienne :

$$d(\mathbf{x}, \mathbf{x}_i) = \|\mathbf{x} - \mathbf{x}_i\|_2 \tag{4.48}$$

Dans le cadre de la prédiction de séries temporelles, les composantes du vecteur $\mathbf{x}$ correspondent à des observations à différents instants. Lors du calcul de distance, on peut souhaiter pondérer les composantes pour accorder plus d'importance aux observations récentes par exemple.

Calcul de la prédiction Selon l'équation (4.44), on identifie le voisinage de $\mathbf{x}$, puis la prédiction est simplement la moyenne des sorties y_i correspondant aux entrées $\mathbf{x}_i \in \mathcal{N}_k(\mathbf{x})$. Cette approche accorde la même importance aux k voisins de $\mathbf{x}$.

Il est possible d'imaginer d'autres manières de combiner ces sorties. Par exemple, on peut souhaiter pondérer cette moyenne en utilisant la distance $d(\mathbf{x}, \mathbf{x}_i)$, pour accorder plus d'influence sur la prédiction aux exemples les plus proches de $\mathbf{x}$. Une façon de faire est de pondérer par l'inverse de la distance :

$$w_i = \frac{1}{d(\mathbf{x}_i, \mathbf{x})}, \quad \mathbf{x}_i \in \mathcal{N}_k(\mathbf{x}) \tag{4.49}$$

Pour le cas particulier où certaines distances $d(\mathbf{x}_i, \mathbf{x})$ sont nulles, on fixe le poids à un pour ces exemples, et à zéro pour les autres.

Pour simplifier l'écriture, on normalise les poids pour que leur somme soit égale à un :

$$w'_i = \frac{w_i}{\sum w_j} \tag{4.50}$$

ce qui donne l'équation de prédiction suivante :

$$\hat{y} = \hat{f}(\mathbf{x}) = \sum_{\mathbf{x}_i \in \mathcal{N}_k(\mathbf{x})} w'_i y_i \tag{4.51}$$

4.7 ARIMA

Le modèle ARIMA est un modèle de référence pour décrire le comportement d'un processus stationnaire ou qui peut être rendu stationnaire par une opération nommée différenciation. Ce modèle est une généralisation du modèle ARMA, qui est lui-même la combinaison du modèle autorégressif (AR) et du modèle moyenne mobile (MA). La lettre « I » pour *integrated* fait référence au fait que la série peut être différenciée pour être rendue stationnaire.

4.7.1 Modèle autorégressif AR(p)

Le modèle autorégressif suppose que la valeur actuelle de la série peut être régressée à partir des ses p valeurs passées. On parle de modèle autorégressif d'ordre p. Il se formule de la façon suivante :

$$\begin{aligned} z(t) &= c + \phi_1 z(t-1) + \cdots + \phi_p z(t-p) + \epsilon(t) \\ &= c + \sum_{i=1}^{p} \phi_i z(t-i) + \epsilon(t) \end{aligned} \tag{4.52}$$

avec c une constante jouant le même rôle que l'ordonnée à l'origine dans un modèle de régression classique.

4.7.2 Modèle moyenne mobile MA(q)

Le modèle moyenne mobile permet d'exprimer le terme d'erreur comme une combinaison linéaire des termes d'erreurs passés. Cela permet notamment de modéliser l'impact d'un choc imprévu sur l'évolution du système.

Ce modèle s'exprime de la façon suivante :

$$\begin{aligned} z(t) &= \mu + \epsilon(t) + \theta_1\epsilon(t-1) + \cdots + \theta_q + \epsilon(t-q) \\ &= \mu + \epsilon(t) + \sum_{i=1}^{q} \theta_q \epsilon(t-i) \end{aligned} \tag{4.53}$$

4.7.3 ARMA(p,q)

Il existe une équivalence théorique entre les modèles autorégressifs et les modèles moyenne mobile. En effet, pour un modèle autorégressif d'ordre p fini, il existe une formulation équivalente de ce modèle sous la forme d'un modèle moyenne mobile d'ordre infini. De même, pour tout modèle moyenne mobile d'ordre fini q, il existe une formulation équivalente sous forme de modèle autorégressif d'ordre infini.

En pratique, on souhaite apprendre un modèle avec un ordre faible, pour limiter le nombre de paramètres à estimer et ainsi respecter le principe de parcimonie. Parfois, il est difficile de trouver un modèle AR(p) ou MA(q) d'ordre faible qui décrit correctement les données. Mais en combinant les deux approches, il est possible d'apprendre un modèle ARMA(p,q) plus parcimonieux, ce qui est très utile en pratique. Le modèle ARMA(p,q) est formulé comme suit :

$$z(t) = c + \epsilon(t) + \sum_{i=1}^{p} \phi_i z(t-i) + \sum_{i=1}^{q} \theta_q \epsilon(t-i) \tag{4.54}$$

Opérateur retard Pour faciliter l'écriture et la combinaison de modèles appartenant à la famille ARIMA, il est courant d'utiliser l'opérateur retard L. Cet opérateur associe à chaque élement d'une série temporelle, l'observation précédente. On aura donc :

$$Lz(t) = z(t-1) \tag{4.55}$$

L'opérateur retard peut être appliqué plusieurs fois :

$$L(Lz(t)) = L^2 z(t) = z(t-2) \tag{4.56}$$

On peut ainsi ré-écrire le modèle ARMA avec cet opérateur :

$$z(t) = c + \epsilon(t) + \sum_{i=1}^{p} \phi_i L^i z(t) + \sum_{i=1}^{q} \theta_q L^i \epsilon(t) \tag{4.57}$$

ou encore, sous une forme factorisée qui fait apparaître des polynômes :

$$(1 - \phi_1 L - \cdots - \phi_p L^p) z(t) = c + (1 + \theta_1 L + \cdots + \theta_q L^q)\epsilon(t) \tag{4.58}$$

4.7.4 ARIMA(p,q,d)

Difference Lorsque la série n'est pas stationnaire, il est parfois possible de lui appliquer une transformation pour la rendre stationnaire. Une opération souvent utilisée est d'appliquer l'opérateur différence Δ, qui renvoie la différence entre deux valeurs successives de la série :

$$\Delta z(t) = z(t) - z(t-1) \tag{4.59}$$

que l'on peut réécrire avec l'opérateur retard :

$$\Delta z(t) = (1 - L)z(t) \tag{4.60}$$

Comme pour l'opérateur retard, on peut l'appliquer plusieurs fois :

$$\begin{aligned} \Delta^2 z(t) &= \Delta(\Delta z(t)) \\ &= (1-L)(1-L)z(t) \\ &= (1 - 2L + L^2)z(t) \\ &= z(t) - 2z(t-1) + z(t-2) \end{aligned} \tag{4.61}$$

ARIMA(p,d,q) Le modèle ARIMA consiste à appliquer une ou plusieurs fois l'opérateur différence à la série d'origine (en pratique, rarement plus de deux fois) afin de la rendre stationnaire, puis d'apprendre un modèle ARMA(p,q) pour cette nouvelle série. Le paramètre d détermine le nombre de fois où l'on différencie la série d'origine. Le modèle s'énonce donc comme un modèle ARMA en remplaçant $z(t)$ par $\Delta^d z(t)$

$$(1 - \phi_1 L - \cdots - \phi_p L^p)\Delta^d z(t) = c + (1 + \theta_1 L + \cdots + \theta_q L^q)\epsilon(t) \tag{4.62}$$

Estimation du modèle Si les hyperparamètres (p, d, q) sont fixés, alors la série $\Delta^d z(t)$ est calculée, puis le modèle ARMA(p,q) est appris, le plus souvent, par une estimation exacte du maximum de vraisemblance par filtre de Kalman [30].

Originellement, le choix des hyperparamètres (p, d, q) se faisait par l'observation des courbes de la série temporelle, de ses fonctions d'autocorrélation et d'autocorrélation partielle et, si besoin, de la série différenciée une ou plusieurs fois. Cette méthodologie proposée par Box et Jenkins [8] présente l'inconvénient de faire beaucoup intervenir la subjectivité du modélisateur, et d'être peu adaptée lorsque le nombre de séries à modéliser est important.

Une procédure pour l'apprentissage automatique des hyperparamètres est proposée par Hyndmann et Khandakar [46, 45]. Tout d'abord, le nombre

de différences d est estimé par une séquence de tests de racine-unitaire KPPS [58]. Si le premier test est positif, on différencie la série et on effectue un nouveau test sur la série obtenue. On arrête de différencier quand le test est négatif. Pour choisir le nombre de termes auto-régressifs p et de moyennes mobiles q, une approche itérative est proposée, pour choisir le meilleur modèle selon le critère d'information d'Akaike corrigé (AICc)[11]. Plusieurs modèles simples sont évalués, puis le meilleur est conservé, et plusieurs variations de ce modèle sont à nouveau évaluées (en ajoutant ou soustrayant 1 au paramètre p ou q ; en ajoutant ou retirant un terme constant c) et la meilleure variation est conservée. La recherche s'arrête quand aucune variation n'améliore l'AICc.

4.8 Modèle Vecteur Autorégressif (VAR(p))

Le modèle vecteur autorégressif est une généralisation multivariée du modèle autorégressif, pour pouvoir modéliser le comportement d'un ensemble de séries temporelles et leurs interactions. Pour désigner la série temporelles multivariée constituée de N séries temporelles, on utilisera la notation suivante :

$$\mathbf{z}(t) = \begin{pmatrix} z_1(t) \\ \vdots \\ z_N(t) \end{pmatrix} \tag{4.63}$$

Le modèle VAR(p) s'écrit alors de la manière suivante :

$$\mathbf{z}(t) = \mathbf{c} + \mathbf{A}_1\mathbf{z}(t-1) + \cdots + \mathbf{A}_p\mathbf{z}(t-p) + \boldsymbol{\epsilon}(t) \tag{4.64}$$

où $\mathbf{c}$ est un vecteur de valeurs constantes, les $\mathbf{A}_i$ sont des matrices $N \times N$ et $\boldsymbol{\epsilon}(t)$ est un vecteur de termes d'erreur :

$$\mathbf{c} = \begin{pmatrix} c_1 \\ \vdots \\ c_N \end{pmatrix}, \quad \mathbf{A}_i = \begin{pmatrix} a^i_{11} & \cdots & a^i_{1N} \\ a^i_{21} & \cdots & a^i_{2N} \\ \vdots & \ddots & \vdots \\ a^i_{N1} & \cdots & a^i_{NN} \end{pmatrix}, \quad \boldsymbol{\epsilon}(t) = \begin{pmatrix} \epsilon_1(t) \\ \vdots \\ \epsilon_N(t) \end{pmatrix}. \tag{4.65}$$

Le vecteur d'erreur $\boldsymbol{\epsilon}(t)$ doit respecter les conditions suivantes :

- tous les termes d'erreur ont une moyenne nulle : $\mathbb{E}[\boldsymbol{\epsilon}(t)] = 0$;
- il existe une covariance entre les termes d'erreur à l'instant t, capturée par la matrice de covariance $\Sigma = \mathbb{E}[\boldsymbol{\epsilon}(t)\boldsymbol{\epsilon}^\intercal(t)]$;
- il n'y pas de corrélation dans le temps : $\forall k \neq 0$, $\mathbb{E}[\boldsymbol{\epsilon}(t)\boldsymbol{\epsilon}^\intercal(t-k)] = 0$.

Stationnarité Avant d'apprendre un modèle VAR(p), il faut déterminer si les séries temporelles $(z_1, \ldots, z_N)$ sont stationnaires, où si elles ont besoin d'être différenciées. Comme pour le modèle ARIMA, cette vérification peut être effectuée en utilisant des tests de racine-unitaire, comme le test KPSS [58] ou le test augmenté de Dickey-Fuller (ADF) [88]. Si toutes les variables sont stationnaires, on peut apprendre un modèle VAR(p). Si toutes les variables sont intégrées de degré d (i.e. non-stationnaires, mais qu'on peut rendre stationnaires par d différences successives), la spécification du modèle est plus compliquée et fait intervenir la notion de cointégration [109], qui ne sera pas traitée dans ce manuscrit.

Choix du modèle Comme recommandé par Lütkepohl [66], si l'objectif final est la prédiction, l'utilisation de l'AIC [3, 4] est une bonne manière de choisir la valeur de l'ordre d'autorégression p. On fixe une valeur p_{max}, puis on évalue les modèles pour $p \in \{1, \ldots, p_{max}\}$ en calculant l'AIC et on conserve le meilleur modèle.

Dans cette section, nous avons décrit les différentes approches de prévision que l'on compare dans l'étude expérimentale. Nous avons souligné quelques propriétés de ces méthodes, comme le principe de parcimonie, l'importance de contrôler la complexité du modèle et l'utilité de pouvoir modéliser la non-linéarité dans le cas de la prévision de trafic.

Dans la section suivante, nous introduisons les approches de sélection de variables, qui permettent de fournir en entrée de ces modèles de prévision l'information la plus pertinente possible, afin d'accélérer l'apprentissage et d'améliorer la performance de prévision.

Chapitre 5

Sélection de variables

Sommaire

5.1 `TiGraMITe` **67**
5.2 **Apprendre un graphe avec Lasso** **73**
5.3 **Sélection à partir du graphe de dépendance** . . . **74**

La sélection de variables consiste à sélectionner un sous-ensemble de variables du jeu de données sur lequel on applique une méthode d'apprentissage. Le grand principe motivant la sélection de variables est qu'utiliser plus de données et donc d'information n'est pas toujours profitable dans une application d'apprentissage artificiel.

Dans les applications récentes, on est passé de quelques dizaines de variables à plusieurs centaines, voire milliers dans certains cas. Les principaux problèmes liés à cette quantité de données sont les variables redondantes et les variables non pertinentes, ainsi que le fléau de la dimensionnalité qui en est une conséquence.

Les approches de sélection de variables permettent de mieux comprendre les données, de réduire le temps de calcul des modèles, d'éviter le fléau de la dimensionnalité et ainsi d'améliorer la performance de prévision des modèles.

Dans le contexte applicatif de la prévision de trafic, la sélection de variables permet d'identifier un voisinage spatio-temporel, c.-à-d. d'identifier pour un capteur donné quelles sont les valeurs dans le passé de quels capteurs et avec quel décalage temporel qui ont une influence sur son futur. Au delà de la prévision, l'identification d'un tel voisinage peut permettre de mieux comprendre le phénomène de trafic étudié.

Les méthodes de sélection de variables peuvent être classées en trois grandes catégoriées : les approches de type *filter*, *wrapper*, et *embedded* [13].

Approches *filter* Les approches de type *filter* consistent à sélectionner les variables en amont de la tâche d'apprentissage (ici de régression). Le choix du sous-ensemble sélectionné est donc indépendant du modèle de régression utilisé. Ces approches s'appuient sur une mesure d'intérêt permettant d'évaluer la pertinence d'une variable d'entrée (ou d'un ensemble de variables) pour la tâche de régression. La corrélation entre la variable d'entrée et la variable cible, ou l'information mutuelle sont des exemples de ces mesures d'intérêt. Les approches de type *ranking* sont un sous-ensemble des approches *filter* dans lesquelles les variables sont triées selon leur score d'intérêt. D'autres approches ne considèrent pas uniquement le lien entre une variable d'entrée et la variable cible, mais s'intéressent à des ensembles de variables d'entrées dont la synergie peut aider à prédire la variable cible.

L'algorithme de `TiGraMITe`, que l'on présente dans la section suivante appartient à cette famille d'approche. En effet, on peut l'utiliser en amont d'une quelconque méthode de prévision. Cependant, nous verrons qu'il est également possible de l'utiliser comme une famille de type wrapper.

Approches *wrapper* Les approches de type *wrapper* s'appuient sur la performance de prévision comme critère de choix du meilleur sous-ensemble de variables. L'algorithme de prévision est considéré comme une boîte noire à laquelle on fournit les données d'un sous-ensemble de variables et dont on observe l'erreur de prévision pour évaluer la pertinence du sous-ensemble proposé. L'approche par force brute consistant à tester tous les sous-ensembles possibles ne passe pas à l'échelle car il existe un nombre exponentiel de sous-ensembles possibles (2^d avec d le nombre de variables). Il faut donc utiliser des procédures sous-optimales pour pouvoir éliminer les variables redondantes ou non pertinentes en un temps raisonnable.

Les approches de type *forward* (ou *backward*) consistent respectivement à commencer avec un ensemble vide (ou toutes les variables) et à ajouter (enlever) progressivement des variables, en évaluant si cette action permet ou non d'améliorer la prévision.

D'autres approches heuristiques, comme les algorithmes génétiques, peuvent être utilisées pour explorer l'ensemble des sous-ensembles de variables.

L'algorithme de `TiGraMITe`, présenté comme une méthode *filter* dans le paragraphe précédent, possède néanmois un paramètre I_{seuil} qui va guider la sélection. Ce paramètre peut être fixé en amont, mais il peut également être optimisé par validation croisée. On s'approche alors d'une méthode de type *wrapper*, car les performances de prévisions sont utilisées pour fixer la valeur de ce paramètre.

Approches *embedded* Les approches de type *embedded* tentent d'éviter le coût de calcul des approches *wrapper* causé par les nombreuses évaluations de sous-ensembles, chacune nécessitant de réapprendre le modèle de prévision. Dans ce but, l'idée est d'intégrer la sélection de variables au cœur même de l'algorithme d'apprentissage. L'approche principale est de modifier la fonction objectif optimisée lors de l'apprentissage pour contraindre l'optimisation à converger vers des modèles n'utilisant qu'un sous-ensemble des variables. C'est notamment la stratégie implémentée dans la régression Lasso, où un terme de régularisation ℓ_1 est ajouté à la fonction coût de la régression linéaire classique.

5.1 TiGraMITe

`TiGraMITe` [86] est une bibliothèque python d'analyse de la causalité pour les séries temporelles. Dans cette bibliothèque sont implémentés des algorithmes récemment proposés dans le domaine de la physique, pour l'analyse des systèmes complexes dont les dynamiques peuvent être fortement non-linéaires [84, 85]. Ces outils ont notamment été utilisés pour l'analyse de phénomènes climatiques comme El Niño [87]. Le résultat fourni par ces algorithmes peut également être utilisé pour sélectionner des variables en entrée d'un algorithme de prévision [83].

L'idée principale au cœur de ces algorithmes est de représenter les dépendances statistiques entre les différentes séries temporelles par un graphe. Les modèles graphiques probabilistes [55] sont des modèles permettant de représenter la distribution de probabilité jointe de plusieurs variables aléatoires par un graphe. Les nœuds de ce graphe représentent les variables aléatoires, et les arcs permettent d'exprimer la structure de dépendance conditionnelle entre ces variables. Ces modèles se séparent en deux grandes catégories : si le graphe est non-orienté, on parle de champs aléatoires de Markov [55], et si le graphe est orienté acyclique, on parle de réseaux bayésiens [48, 55].

Les modèles graphiques probabilistes peuvent être utilisés de différentes manières. Dans certains cas, la structure de dépendance entre les variables est connue (par exemple par les experts du problème étudié), on peut alors chercher à apprendre les poids des arcs, qui représenteront les valeurs des différentes probabilités conditionnelles, afin de les utiliser ensuite pour inférer la probabilité de nouveaux évènement. Dans d'autres cas, on ne dispose pas de connaissances à priori sur les données, et on souhaite donc découvrir la structure de dépendance entre les variables, c.-à-d. apprendre la structure du graphe. Dans le cadre de la thèse, on s'intéresse à ce problème d'apprentissage de la structure de dépendance, car celle-ci permet d'identifier le voisinage

spatio-temporel pertinent pour la prévision.

Des travaux ont permis d'étendre la notion de modèles graphiques probabilistes pour l'analyse des relations dynamiques entre les variables pour le cas de séries temporelles multivariées [26]. L'algorithme d'apprentissage de la structure de dépendance entre séries temporelles implémenté dans `TiGraMITe` est une adaptation de la méthode d'apprentissage de structure qui fait référence pour les réseaux bayésiens : l'algorithme PC [92] (du nom de ses créateurs).

Mesure de la dépendance : information mutuelle L'information mutuelle entre deux variables X et Y est une mesure de la dépendance statistique entre ces variables. Dans l'introduction aux statistiques, nous avons présenté le coefficient de corrélation, qui permet de mesurer une dépendance strictement linéaire entre les deux variables. L'information mutuelle, notée $I(X, Y)$ est plus générale et permet de mesurer toute dépendance statistique, qu'elle soit linéaire ou non. Ainsi, l'information mutuelle entre deux variables est nulle si et seulement si ces variables sont indépendantes. Formellement, cette mesure est définie comme suit :

$$I(X,Y) = \int_{\mathbb{R}} \int_{\mathbb{R}} f_{X,Y}(x,y) \log \left(\frac{f_{X,Y}(x,y)}{f_X(x) f_Y(y)} \right) d_x d_y \tag{5.1}$$

où $f_{X,Y}$ est la densité de probabilité jointe entre X et Y. Les fonction f_X et f_Y sont les densités de probabilités marginales respectives de X et Y.

Le choix de l'information mutuelle dans l'algorithme de `TiGraMITe` est dicté par l'objectif de pouvoir modéliser des dépendances complexes et non-linéaires entre les séries temporelles. Cette propriété est également importante pour l'analyse des séries temporelles de trafic urbain. En effet, différents travaux ont montré que la non-linéarité était un phénomène souvent observé dans les mesures de débit de trafic [106].

De façon informelle, on peut dire que l'information mutuelle $I(X, Y)$ est la quantité moyenne d'information sur X gagnée en observant Y. On peut remarquer que la formule de l'information mutuelle est symétrique, ainsi $I(X, Y)$ est également la quantité moyenne d'information sur Y gagnée en observant X.

Dans le contexte de la prévision de séries temporelles, les variables correspondent à une observation à un instant t. Mesurer l'information mutuelle entre deux variables $Z_i(t)$ et $Z_j(t+1)$ correspond à déterminer la quantité moyenne d'information sur la valeur qu'on observera sur le capteur n° j à l'instant $t+1$ qui sera gagnée en observant la valeur du capteur n° i à l'instant t.

Information mutuelle conditionnelle L'information mutuelle conditionnelle permet de mesurer l'information contenue dans deux variables X et Y qui n'est pas contenue dans une troisième variable Z. Elle est définie formellement comme suit :

$$I(X,Y|Z) = \int \left(\int \int f_{X,Y|Z}(x,y|z) \log \frac{f_{X,Y|Z}(x,y|z)}{f_{X|Z}(x|z) f_{Y|Z}(y|z)} \, d_x d_y \right) p_Z(z) \, d_z \tag{5.2}$$

On retrouve dans le terme entre parenthèses une expression similaire à l'information mutuelle $I(X,Y)$, mais dont les densités de probabilités sont devenues des densités conditionnelles.

Mesurer l'information mutuelle conditionnelle $I(X,Y|Z)$ revient à poser la question suivante : « sachant que l'on a déjà observé la valeur prise par Z, quelle quantité d'information nouvelle sur Y va-t-on découvrir en moyenne en observant aussi X ? » Comme X et Y jouent un rôle symétrique, on peut les inverser dans cette phrase, de façon équivalente.

L'information mutuelle conditionnelle permet donc d'étudier la redondance de l'information entre les différentes variables. En effet, imaginons que l'on observe les conditions suivantes :

$$\begin{cases} I(X,Y) > 0 \\ I(X,Y|Z) = 0 \end{cases}$$

Si l'objectif est de prédire la variable Y, la première équation nous indique que X contient de l'information utile pour prédire Y. Cependant, la deuxième équation indique que toute l'information sur Y contenue dans X est également contenue dans Z. Ainsi, la variable X est redondante, et l'on pourrait se passer de l'observer, sans perdre en précision de prévision de la variable Y.

L'information mutuelle conditionnelle permet également d'étudier les relations de causalité entre les variables. Pour tendre vers une interprétation purement causale, on doit pouvoir mesurer l'influence d'une variable sur une autre, mais également exclure d'autres influences extérieures. Mais la plupart du temps, il est impossible d'exclure toutes les autres influences, surtout si l'on ne peut pas manipuler expérimentalement le système étudié. La « causalité » évoquée ici doit donc être comprise relativement au système étudié (ici l'ensemble des mesures des différents capteurs). L'un des plus gros problèmes pour l'interprétation causale est le cas des causalités trompeuses (*spurious causality*) qui peuvent être dues à des influences indirectes ou à un facteur causal commun. Prenons l'exemple d'une interaction causale de la forme $X \to Z \to Y$. Une étude uniquement bivariée, basée sur l'information

mutuelle donnerait les résultats suivants :

$$\begin{cases} I(X,Z) \neq 0 \\ \boldsymbol{I(X,Y) \neq 0} \\ I(Y,Z) \neq 0 \end{cases}$$

Sur la base de cette analyse, à partir de la ligne en gras, il serait possible de conclure à tort que X a une influence directe sur Y. Cependant, si l'on s'appuie sur une analyse de l'information mutuelle conditionnelle, on pourrait observer :

$$\begin{cases} I(X,Y|Z) = 0 \\ I(Y,Z|X) \neq 0 \end{cases}$$

On peut déduire de la première équation que toute l'information sur Y contenue dans X est également contenue dans Z. De la deuxième équation, on apprend que Z contient de l'information sur Y qui n'est pas contenue dans X. D'une certaine manière, on peut conclure que toute l'influence que X a sur Y passe par Z. Son influence sur Y est donc indirecte. Ainsi, dans le graphe représentant la distribution de probabilité jointe de X, Y et Z (c.-à-d. le graphe de dépendance), il n'y aura pas d'arc entre X et Y.

Versions multivariées Les mesures d'information mutuelle (conditionnelle) utilisées dans l'algorithme de `TiGraMITe` sont des versions multivariées. Ainsi, on pourra étudier l'information mutuelle entre deux ensembles de variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ et $\mathbf{Y} = \{Y_1, \ldots, Y_m\}$, ou conditionnée par rapport à un troisième ensemble de variable $\mathbf{Z}$. Dans l'algorithme, les tests effectués sont souvent de forme :

$$I(X,Y|\mathbf{Z})$$

où X et Y sont deux variables, et $\mathbf{Z}$ un ensemble de variable. L'objectif est de tester si l'on doit conserver un arc entre X et Y dans le graphe. On va donc vérifier que X contient de l'information sur Y qui n'est pas contenu par les variables de l'ensemble $\mathbf{Z}$.

Dans notre application, les arcs sont forcément orientés, car les variables dépendent du temps. Ainsi un arc entre deux variables $Z_1(t-1)$ et $Z_2(t)$ sera forcément de la forme $Z_1(t-1) \rightarrow Z_2(t)$, car il est dirigé du passé vers le futur.

Une méthode d'estimation de l'information mutuelle multivariée par une approche des k plus proches voisins a été proposée en 2004 [56], puis étendue en 2007 pour estimer l'information mutuelle conditionnelle multivariée [28]. C'est cette dernière méthode qui est utilisée dans `TiGraMITe` pour mesurer la dépendance entre les variables.

Graphe de dépendance pour les séries temporelles Avant d'expliquer le fonctionnement de l'algorithme permettant d'apprendre un graphe de dépendance entre les différentes séries temporelles mesurée par les capteurs, il faut définir plus en précisément la notion de graphe de dépendance.

Soit $\mathbf{Z}$ un processus stochastique multivarié à temps discret stationnaire, et $Z_1, \dots, Z_N$ les sous-processus qui le composent. Dans notre application, chaque sous-processus Z_i correspond à l'activité observée par le capteur n$^{\text{o}}$ i et le processus $\mathbf{Z}$ correspond donc à l'activité mesurée sur l'ensemble du réseau. La valeur à l'instant t du processus est notée $\mathbf{Z}(t)$ et celle d'un sous-processus est notée $Z_i(t)$. Leurs passés sont notés :

$$\begin{cases} \mathbf{Z}^p(t) = (\mathbf{Z}(t-1), \mathbf{Z}(t-2), \dots) \\ Z_i^p(t) = (Z_i(t-1), Z_i(t-2), \dots) \end{cases}$$

Chaque nœud dans le graphe représente une variable aléatoire correspondant à un sous-processus Z_i à un instant t. Deux nœuds $Z_i(t-\tau)$ et $Z_j(t)$ sont connectés dans le graphe par un arc $Z_i(t-\tau) \to Z_j(t)$ allant vers l'avant dans le temps, si et seulement si :

$$\begin{cases} \tau > 0 \\ I\left(Z_i(t-\tau), Z_j(t) \,|\, \mathbf{Z}^p(t) \setminus \{Z_i(t-\tau)\}\right) > 0 \end{cases} \tag{5.3}$$

Il s'agit donc de l'information mutuelle entre $Z_i(t-\tau)$ et $Z_j(t)$, conditionnée par l'ensemble de variables $\mathbf{Z}^p(t) \setminus \{Z_i(t-\tau)\}$. Cet ensemble contient toutes les variables appartenant au passé de $\mathbf{Z}(t)$ sauf $Z_i(t-\tau)$. Si cette valeur est strictement positive, cela signifie donc que $Z_i(t-\tau)$ contient une information unique sur $Z_j(t)$, qui n'est présente dans aucune autre variable appartenant au passé. Ceci est indiqué par un arc dans le graphe, qui signifie une influence directe.

Étant donnée la variable $Z_j(t)$, on peut définir l'ensemble $\mathcal{P}_{Z_j(t)}$ des parents de cette variable dans le graphe. Cet ensemble est constitué de toutes les variables possédant un arc sortant dirigé vers $Z_j(t)$. Formellement, on a :

$$\mathcal{P}_{Z_j(t)} = \{Z_i(t-\tau) \in \mathbf{Z}^p(t),\ Z_i(t-\tau) \to Z_j(t)\}$$

On peut réécrire l'équation 5.3 en remplaçant $\mathbf{Z}^p$ par les parents :

$$I\left(Z_i(t-\tau), Z_j(t) \,|\, \mathcal{P}_{Z_j(t)} \setminus \{Z_i(t-\tau)\}\right) > 0$$

Cette formulation donne un graphe équivalent, car l'ensemble des parents $\mathcal{P}_{Z_j(t)}$ est le sous-ensemble de $\mathbf{Z}^p(t)$ constitué des variables contenant de l'information unique sur la variable $Z_j(t)$. Ainsi, rajouter une autre variable n'appartenant pas aux parents dans le terme de droite de l'information mutuelle conditionnelle n'a donc aucune influence sur sa valeur, qui est déjà minimale.

L'algorithme d'apprentissage du graphe de dépendance L'idée reprise de l'algorithme PC est de découvrir de façon itérative les arcs du graphe en testant l'indépendance conditionnelle entre toutes les paires possibles de nœuds, en conditionnant progressivement à un nombre croissant de variables, et en testant toutes les combinaisons possibles de ces variables. Cependant, l'algorithme est adapté pour éviter un maximum de tests, en supprimant le plus tôt possible les arcs qui ne doivent pas être présents dans le graphe de dépendance final.

Initialisation : pour chaque variable $Z_i(t) \in \mathbf{Z}(t)$, on souhaite initialiser l'ensemble de ses possibles parents, noté $\widetilde{\mathcal{P}}_{Z_i(t)}$. Au moment de son initialisation, $\widetilde{\mathcal{P}}_{Z_i(t)}$ est un sur-ensemble des vrais parents $\mathcal{P}_{Z_i(t)}$. Au fur et à mesure, on retirera les variables correspondant à des liens indirects pour converger vers l'ensemble des parents. L'initialisation se fait en calculant l'information mutuelle classique (non-conditionnelle) :

$$\widetilde{\mathcal{P}}_{Z_i(t)} = \{Z_j(t-\tau) \,:\, Z_j \in \mathbf{Z},\, 0 < \tau \leq \tau_{\max},\, I(Z_j(t-\tau), Z_i(t)) > 0\}$$

Pour une valeur fixée de $\tau_{\max}$, on calcule l'information mutuelle entre $Z_i(t)$ et toutes les variables mesurées du temps $(t-1)$ au temps $(t-\tau_{\max})$. Toutes les variables pour lesquelles l'information mutuelle est nulle n'ont aucun lien, direct ou indirect, avec la variable cible $Z_i(t)$, et ne peuvent donc pas appartenir à $\mathcal{P}_{Z_i(t)}$. Les variables pour lesquelles l'information mutuelle n'est pas nulle ont donc un lien avec la variable cible, et sont candidates pour faire partie de ses parents. Elle sont donc inclues dans $\widetilde{\mathcal{P}}_{Z_i(t)}$. Les étapes suivantes de l'algorithme permettront de vérifier que ce lien est direct et non indirect. On trie ensuite $\widetilde{\mathcal{P}}_{Z_i(t)}$ par ordre d'informations mutuelles décroissantes, ce qui sera utile pour la suite de l'algorithme.

Mise à jour de $\widetilde{\mathcal{P}}_{Z_i(t)}$: Dans cette étape, nous allons mesurer l'information mutuelle conditionnelle (IMC), en augmentant itérativement le nombre de variables dans la condition. Le nombre de conditions sera noté c, et on commence donc avec $c = 1$.

Pour cette valeur de c, nous allons itérer sur toutes les variables candidates $Z_j(t-\tau) \in \widetilde{\mathcal{P}}_{Z_i(t)}$, pour tester grâce à l'IMC si ces liens sont indirects, auquel cas elle sont annotées pour être retirées de $\widetilde{\mathcal{P}}_{Z_i(t)}$ ultérieurement.

Pour chaque variable $Z_j(t-\tau)$, il faut mesurer son IMC en conditionnant à tous les ensembles de taille c, et garder la valeur minimale, qui correspond à avoir conditionné avec les variables qui ont le plus d'information en commun avec $Z_j(t-\tau)$ sur $Z_i(t)$. On a donc :

$$I_{\min} = \min_{\mathcal{P}} I(Z_j(t-\tau), Z_i(t)|\mathcal{P}), \quad \text{avec } \mathcal{P} \subset \widetilde{\mathcal{P}}_{Z_i(t)} \setminus Z_j(t-\tau),\, |\mathcal{P}| = c$$

Pour décider si la variable $Z_j(t-\tau)$ doit être retirée de $\widetilde{\mathcal{P}}_{Z_i(t)}$, on fixe un seuil minimal I_{seuil} pour la valeur de l'IMC. On note la variable comme étant à retirer si $I_{\min} < I_{\text{seuil}}$. En réalité, on ne teste pas tous les sous-ensembles de taille c car cela serait trop coûteux. Comme $\widetilde{\mathcal{P}}_{Z_i(t)}$ est trié par ordre décroissant d'IMC, on énumère $\mathcal{P}$ en respectant cet ordre, jusqu'à un nombre maximum de valeurs testées noté p_{max}. C'est une bonne approximation, car les ensembles les plus informatifs sont testés en premier, diminuant ainsi progressivement la probabilité que les ensembles suivants permettent de rejeter la variable si les premiers ne l'ont pas permis. A chaque $\mathcal{P}$ testé, $I_{\min}$ est mise à jour et comparée à I_{seuil} ; si la variable $Z_j(t-\tau)$ peut être rejetée, on stoppe l'énumération de $\mathcal{P}$ et on passe à la variable suivante.

Après avoir testé toutes les variables, $\widetilde{\mathcal{P}}_{Z_i(t)}$ est mis à jour en supprimant les variables précédemment annotées, et est trié par valeur d'IMC décroissante. On incrémente le nombre de conditions c et on réitère sur les tests d'IMC sur les variables restantes.

Ce procédé itératif s'arrête quand le nombre de conditions c devient plus grand que le nombre de variables contenues dans $\widetilde{\mathcal{P}}_{Z_i(t)}$.

Notons que l'augmentation progressive du nombre de conditions, permet d'éliminer tôt de nombreux arcs sans avoir à évaluer l'IMC en grande dimension. C'est important, car l'estimation de cette mesure est plus fiable en faible dimension, à cause du fléau de la dimensionnalité.

5.2 Apprendre un graphe avec Lasso

Comme présenté dans le chapitre précédent sur les méthodes de prévision, la régression Lasso est une adaptation de la méthode de régression linéaire classique. La différence réside dans le terme de régularisation (pénalité L_1) dans la fonction coût. La présence de ce terme conduit à un apprentissage durant lequel de nombreux coefficients convergent vers zéro. Ainsi, les variables d'entrées associées à ces coefficients nuls ne sont pas utilisées dans la fonction de prévision du modèle. À ce titre, le modèle de régression Lasso est une méthode de prévision avec un mécanisme de sélection de variables intégré.

Les variables retenues par la méthode Lasso sont sélectionnées pour obtenir le modèle de régression linéaire avec la meilleure performance estimée de généralisation. Ainsi, le choix de ces variables est contraint par le fait qu'elles servent d'entrée à un modèle linéaire. Un modèle d'une nature différente pourrait peut-être bénéficier de l'ajout de certaines autres variables en entrée, qu'il pourrait exploiter, là où le modèle linéaire ne peut pas le faire. En ce sens, l'ensemble des variables sélectionnées est spécifique à la méthode

de prévision utilisée.

De prime abord, il semble donc peu adapté d'utiliser l'ensemble de variables sélectionnées par Lasso en entrée d'un modèle très différent. Notamment, une approche comme la régression k-NN ne fait pas d'hypothèse sur la nature de la dépendance. Pour cette raison, un algorithme comme TiGraMITe, qui partage avec k-NN cette absence d'a priori sur la forme de la dépendance, semble plus adapté pour l'étape de sélection. Cependant, les variables retenues par Lasso sont au moins utiles pour la prévision avec un modèle linéaire, ce qui indique déjà qu'elles contiennent de l'information utile sur la variable que l'on souhaite prédire, et sont donc probablement pertinentes. Il est possible que le modèle sous-exploite l'information contenue par ces variables si la dépendance n'est pas totalement linéaire. Ainsi, reprendre cet ensemble de variables comme base et utiliser un autre modèle de prévision peut parfois donner des résultats intéressants. La simplicité du Lasso est un avantage, car il permet à faible coût d'avoir un ensemble candidat de variables sélectionnées, qui sans être optimal de façon garantie, a des chances d'être plus pertinent que de ne pas faire de sélection.

Construction du graphe Comme le paramètre $\tau_{\max}$ dans TiGraMITe, on fournit en entrée de la régression Lasso des variables jusqu'à un certain moment du passé. En apprenant un modèle Lasso pour prédire une variable $Z_i(t)$, un ensemble de variables $Z_j(t-\tau)$ est conservé car leurs coefficients sont non-nuls. Cet ensemble de variables définit les arcs présents dans le graphe, de la forme $Z_j(t-\tau) \to Z_i(t)$ et joue donc un rôle équivalent à l'ensemble des parents $\mathcal{P}_{Z_i(t)}$ dans TiGraMITe. Avec cette approche, on n'a pas de garantie sur l'élimination de liens indirects. Seul les résultats expérimentaux sur le jeu de test permettent d'évaluer la pertinence de l'utilisation de ce graphe pour sélectionner les variables en entrée d'un algorithme en particulier. Nous verrons dans la section suivante comment on utilise ce graphe pour effectuer la sélection.

5.3 Sélection à partir du graphe de dépendance

Nous avons vu dans les sections précédentes comment apprendre un graphe de dépendance à partir d'un ensemble de séries temporelles. Ce graphe permet de décrire les interactions entre les différentes variables, et ainsi de déterminer quelles variables contiennent de l'information pertinente pour prédire une variable donnée. Nous allons maintenant présenter comment utiliser ce graphe en pratique pour sélectionner les variables en entrée d'un algorithme de prévision.

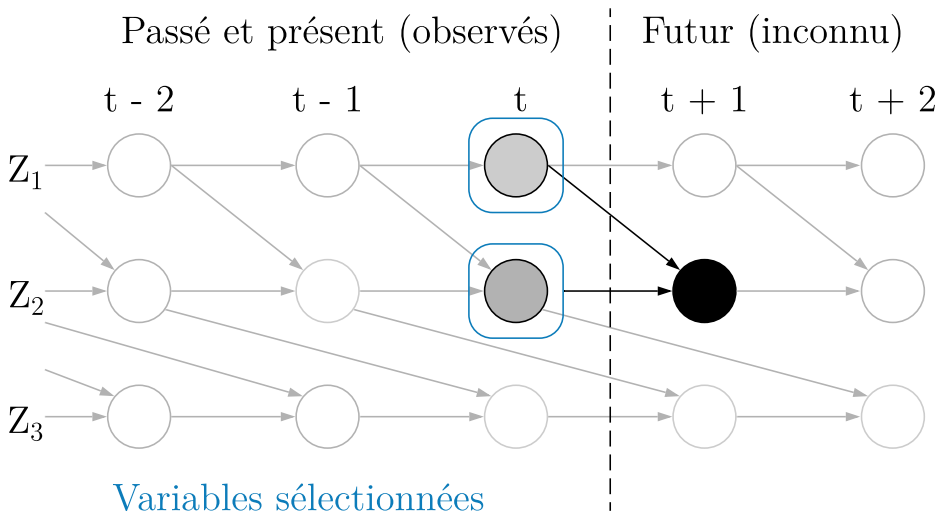


FIGURE 5.1 – Sélection de variables avec le graphe de dépendance (cas n° 1). On souhaite prédire la variable $Z_2(t+1)$ (en noir). L'ensemble de ses parents (en gris) est $\mathcal{P}_{Z_2(t+1)} = \{Z_1(t); Z_2(t)\}$. Les deux parents sont des variables associées à l'instant t déjà observé. On peut donc les sélectionner comme entrées d'un algorithme de prévision. L'ensemble des variables sélectionnées (en bleu) est donc dans ce cas identique à l'ensemble des parents.

Cas n° 1 : utilisation des parents Dans la figure 5.1, on présente un exemple de graphe de dépendance à partir d'un jeu de données constitué de trois séries temporelles Z_1, Z_2 et Z_3. On dispose des données jusqu'à l'instant t compris, et on souhaite prédire la valeur de $Z_2(t+1)$. D'après l'information portée par le graphe, les variables contenant de l'information unique sur la variable cible sont ses parents dans le graphe $\mathcal{P}_{Z_2(t+1)} = \{Z_1(t); Z_2(t)\}$. Comme on se trouve à l'instant t au moment d'effectuer la prévision, on dispose bien des observations pour ces deux variables. L'ensemble des variables à sélectionner est donc dans ce cas identique aux parents de la variable $Z_2(t+1)$. Nous verrons dans l'exemple suivant que ce n'est pas toujours le cas.

Cas n° 2 : itérer sur les parents des parents Dans la figure 5.2, on considère le même graphe de dépendance que pour le cas n° 1. Cette fois-ci, on souhaite prédire la variable $Z_3(t+2)$. Comme précédemment, on commence par s'intéresser aux parents $\mathcal{P}_{Z_3(t+2)} = \{Z_2(t); Z_3(t+1)\}$. Cette fois-ci, un des parents $(Z_3(t+1))$ se trouve encore dans le futur non observé, puisque l'on se place à l'instant t pour effectuer la prévision. Il ne sera donc

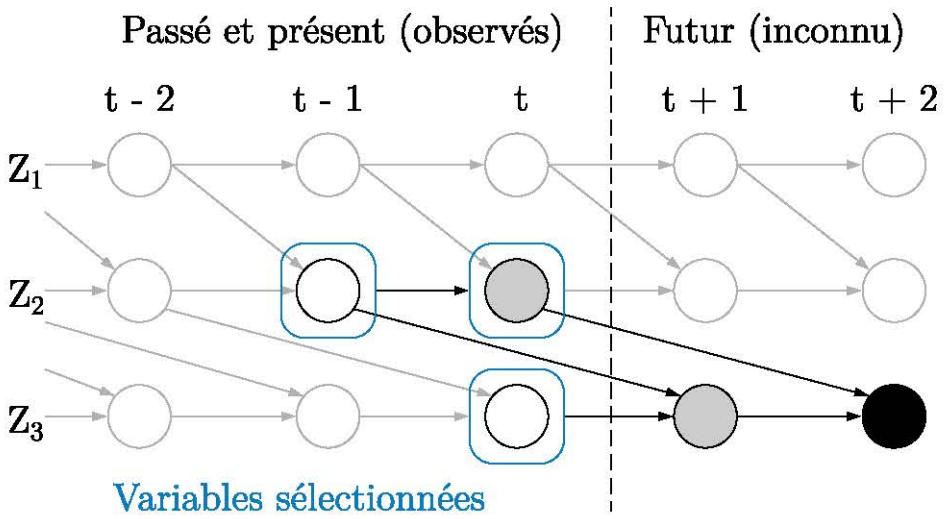


FIGURE 5.2 – Sélection de variables avec le graphe de dépendance (cas n° 2). On souhaite prédire la variable $Z_3(t+2)$ (en noir). L'ensemble de ses parents (en gris) est $\mathcal{P}_{Z_3(t+2)} = \{Z_2(t); Z_3(t+1)\}$. Le parent $Z_3(t+1)$ n'est pas encore observé à l'instant t, or c'est à cet instant qu'on souhaite produire la prévision. On va donc remplacer cette variable par ses parents $\mathcal{P}_{Z_3(t+1)} = \{Z_2(t-1); Z_3(t)\}$. Ces variables étant déjà observées au moment où l'on souhaite produire une prévision, elle peuvent être sélectionnées. L'ensemble final des variables sélectionnées (en bleu) est donc constitué des trois variables $\{Z_2(t-1); Z_2(t); Z_3(t)\}$. Dans ce cas, l'ensemble des variables sélectionnées est différent de l'ensemble des parents.

pas possible de s'appuyer sur l'information contenue dans cette variable. Cependant, le graphe nous permet de rechercher les variables contenant le plus d'informations en commun avec $Z_3(t+1)$, c.-à-d. ses propres parents. On va donc la remplacer par ses parents $\mathcal{P}_{Z_2(t+1)} = \{Z_1(t); Z_2(t)\}$ dans les variables candidates à la sélection. Cette fois-ci ses parents sont tous deux observés à l'instant t et peuvent donc être sélectionnés. Si une de ces variables avait été encore dans le futur, il aurait fallu répéter récursivement l'opération jusqu'à n'obtenir que des variables observables à l'instant t ou antérieurement. Nous obtenons ainsi l'ensemble des variables sélectionnées suivant : $\{Z_2(t-1); Z_2(t); Z_3(t)\}$.

Influence de l'horizon de prévision Si l'on se place à l'instant t pour prédire la valeur d'une variable $Z_i(t+h)$, nous avons vu précédemment qu'il

faut chercher ses parents, puis si ces derniers sont dans le futur, effectuer une recherche récursive de parents jusqu'à l'obtention d'un ensemble de variables toutes observables à l'instant t. Plus l'horizon de prévision h est élevé, plus on a de chances que les parents, les grands-parents, etc. de la variable soient dans le futur.

Il faut noter qu'à chaque fois que l'on remplace une variable par ses parents dans le graphe, si celle-ci a plus d'un parent, on augmente le nombre de variables potentiellement sélectionnées. Ce phénomène est logique car plus un phénomène est loin dans le futur, plus ses causes passées peuvent être multiples, et l'information nécessaire pour le prédire est diluée entre toutes ces variables.

Cependant, cela souligne l'importance d'éviter au maximum, lors de l'apprentissage du graphe, d'ajouter des arcs non significatifs. En effet, ces derniers seront d'autant plus problématiques que l'on souhaitera prédire loin dans le futur, car ils feront sélectionner un nombre trop important de variables non pertinentes.

Couplage entre apprentissage du graphe et sélection de variable Depuis le début de cette section, on présente l'approche de prévision comme si elle se déroulait en plusieurs étapes distinctes : l'apprentissage du graphe de dépendance, puis la sélection des variables à partir de ce graphe, et enfin la prévision à partir des variables sélectionnées. En réalité, il s'agit plutôt d'un pipeline intégré de prévision que l'on optimise dans son entièreté.

Dans le cas de la régression `Lasso`, nous avons vu que le nombre de variables dont le coefficient tend vers zéro dépend de la valeur du paramètre de régularisation. Or, la valeur de ce paramètre est déterminée par validation croisée. On utilise donc les résultats de prévision sur les jeux de validation pour produire une estimation de l'erreur de généralisation et choisir la valeur du paramètre qui minimise cette erreur. On voit donc que la sélection des variables a été optimisée dans le même temps que les paramètres internes du modèle de régression. C'est pour cela que l'on parle d'un pipeline de prévision intégré.

Dans le cas de `TiGraMITe`, le paramètre I_{seuil} détermine la valeur minimale de l'IMC à partir de laquelle on rejette un lien du graphe de dépendance. Ce paramètre détermine ainsi la densité du graphe, et donc les variables sélectionnées pour la prédiction d'une variable à un certain horizon. Dans la même logique que pour la régression `Lasso`, la valeur de ce paramètre est fixée par validation croisée.

Contribution à la sélection de voisinage pour la prévision de trafic
Une contribution importante de la thèse est d'appliquer pour la première fois `TiGraMITe` à des données de trafic. La méthode a été retenue pour ses propriétés théoriques qui sont en adéquation avec les défis identifiés dans la littérature du trafic : `TiGraMITe` est conçue pour apprendre la structure de dépendance de phénomènes complexes dont les dynamiques sont non-linéaires. La méthode est prometteuse, comme l'attestent les nombreuses publications dans des revues prestigieuses de physique, pour la description mathématique de la méthode mais aussi pour son application à l'étude de phénomènes climatiques complexes.

Au début des travaux de la thèse, l'implémentation de `TiGraMITe` ne proposait qu'une interface graphique pour interagir avec la méthode. Un travail de la thèse a été d'adapter le code de la méthode pour pouvoir scripter les expérimentations, exporter et importer les résultats dans le mécanisme de sélection de variable et de paralléliser la méthode afin de passer à l'échelle. Dans le même temps, les développeurs de la méthode ont amélioré le logiciel pour fournir une API offrant les mêmes possibilités et d'autres nouveautés.

L'autre contribution pour la sélection de variables est d'appliquer pour la première fois la sélection Lasso sur des données d'un réseau urbain. En effet, si la méthode a déjà été appliquée avec succès sur des données d'autoroute, il reste une interrogation sur son adéquation avec des données urbaines, notamment à cause de l'hypothèse de linéarité sur laquelle elle repose.

Chapitre 6

Résultats expérimentaux

Sommaire

6.1 Données 80
6.1.1 Jeu de données de Lyon 80
6.1.2 Jeu de données de Marseille 83
6.1.3 Différences entre trafic urbain et autoroutier 84
6.2 Mesures de performance 85
6.3 Comparaison des approches de prévision 87
6.3.1 Comparaison des méthodes de régression linéaire . 88
6.3.2 Comparaison des variantes SVR 89
6.3.3 Comparaison des variantes MLP 90
6.3.4 Comparaison des variantes k-NN 90
6.3.5 Comparaison des meilleures méthodes 94
6.4 Étude de l'impact de la résolution temporelle . . 99

Dans ce chapitre, nous présentons les résultats expérimentaux obtenus en appliquant les méthodes présentées précédemment sur les données de Lyon et de Marseille. La section 6.1 présente les deux jeux de données, ainsi que les particularités des infrastructures sur lesquelles sont collectées les données. Les mesures de performance de prévision utilisées dans cette étude sont introduites dans la section 6.2. La section 6.3 présente les résultats des méthodes de prévision sur les deux jeux de données et une analyse de ces résultats selon deux problématiques de la thèse : évaluer l'utilité des méthodes de sélection de variables pour la prévision et mettre en lumière les différences entre les données de milieu urbain et les données d'autoroute, et les différences de choix de méthodes que cela implique. Enfin, dans la section 6.4, nous décrivons les résultats de l'étude de l'impact de la résolution temporelle des données sur la qualité de la prévision.

6.1 Données

Dans cette section nous présentons les jeux de données utilisés dans les expérimentations. Nous disposons d'un jeu de données du réseau urbain de la Métropole de Lyon, qui est partenaire du projet à l'origine de la thèse. Le deuxième jeu de données décrit le trafic sur une portion d'autoroute urbaine de Marseille.

6.1.1 Jeu de données de Lyon

Les données collectées au début de la thèse proviennent de la Métropole de Lyon. Elles nous ont été fournies par le PC CRITER, qui est le poste de gestion centralisée du trafic de la Métropole de Lyon chargé de gérer en temps réel le trafic routier de l'agglomération.

Afin de pouvoir surveiller le trafic, les axes principaux sont équipés de capteurs (boucle inductives) qui détectent le passage des véhicules. Il est donc possible de compter le nombre de véhicules ayant emprunté un tronçon pendant une période donnée, et d'en déduire un débit en véhicules par heure. Cette valeur est mesurée par les capteurs avec un pas d'agrégation temporel de 6 minutes, c.-à-d. une mesure correspond au débit moyen observé sur les 6 dernières minutes.

La figure 6.1 présente la courbe d'évolution du débit pour un capteur du réseau, sur une durée d'une semaine (du lundi au dimanche). On observe une tendance sur les jours de semaine (du lundi au vendredi) avec un pic de débit les matins et en fin d'après-midi, correspondant aux déplacement des automobilistes se rendant sur leur lieu de travail. On constate également que les journées du samedi et du dimanche sont très différentes, correspondant à la différence d'activités des usagers lors des jours non travaillés. Enfin, on observe les importantes fluctuations à haute fréquence, dont la modélisation est un enjeu majeur pour la prévision.

Nous disposons de l'historique de près de 600 capteurs dont les positions sont données dans la figure 6.2, pour une durée allant de quelques mois pour les plus récents, à deux ans pour les plus anciens. Cependant, le jeu de données contient de nombreuses valeurs manquantes, parfois sur de longues plages temporelles. Ces valeurs manquantes peuvent être causées par une panne du capteur, par une intervention de maintenance ou encore par des travaux sur la voirie. Il peut également contenir des données aberrantes, par exemples lorsqu'un véhicule est garé sur un capteur. Cela pose problème pour de nombreuses méthodes de prévision, qui nécessitent un jeu de données nettoyé.

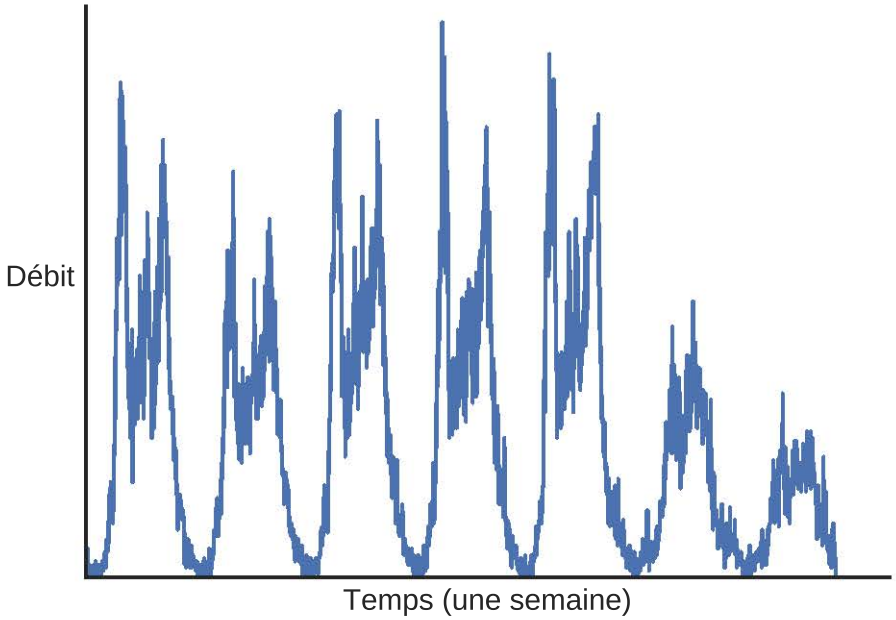


FIGURE 6.1 – Lyon – Mesures de débit sur un capteur de trafic pour une semaine (du lundi au dimanche)

Jeu de données nettoyé Nous avons fait le choix d'exclure les capteurs avec des trop longues plages de valeurs manquantes (plus de 5 pas de temps consécutifs) et d'imputer les valeurs manquantes restantes par une interpolation linéaire. Une possibilité pour l'imputation des longues plages de données aurait été d'utiliser les valeurs moyennes ou médianes sur l'historique (pour le même moment de la journée). Cependant, nous voulions éviter de biaiser l'évaluation des méthodes de prévision et leur comparaison, par les choix d'imputation effectués avant l'apprentissage. Ainsi, nous avons préféré nous restreindre à un ensemble de capteurs avec peu de valeurs manquantes.

Le jeu de données ainsi conservé contient 44 capteurs dont les positions sont données dans la figure 6.3. Ces capteurs sont repartis en trois quartiers, assez espacés les uns des autres. Cette configuration présente un intérêt pour évaluer les approches de sélection de variables. Étant donnée la nature physique du phénomène de trafic, les valeurs mesurées par deux capteurs proches et connectés par le réseau ont logiquement une forte dépendance, avec un décalage temporel faible. Les valeurs mesurées par capteurs éloignés auront une relation plus diffuse et atténuée, avec un décalage temporel plus fort. Ainsi, on souhaite observer si les variables retenues par les approches de sélection

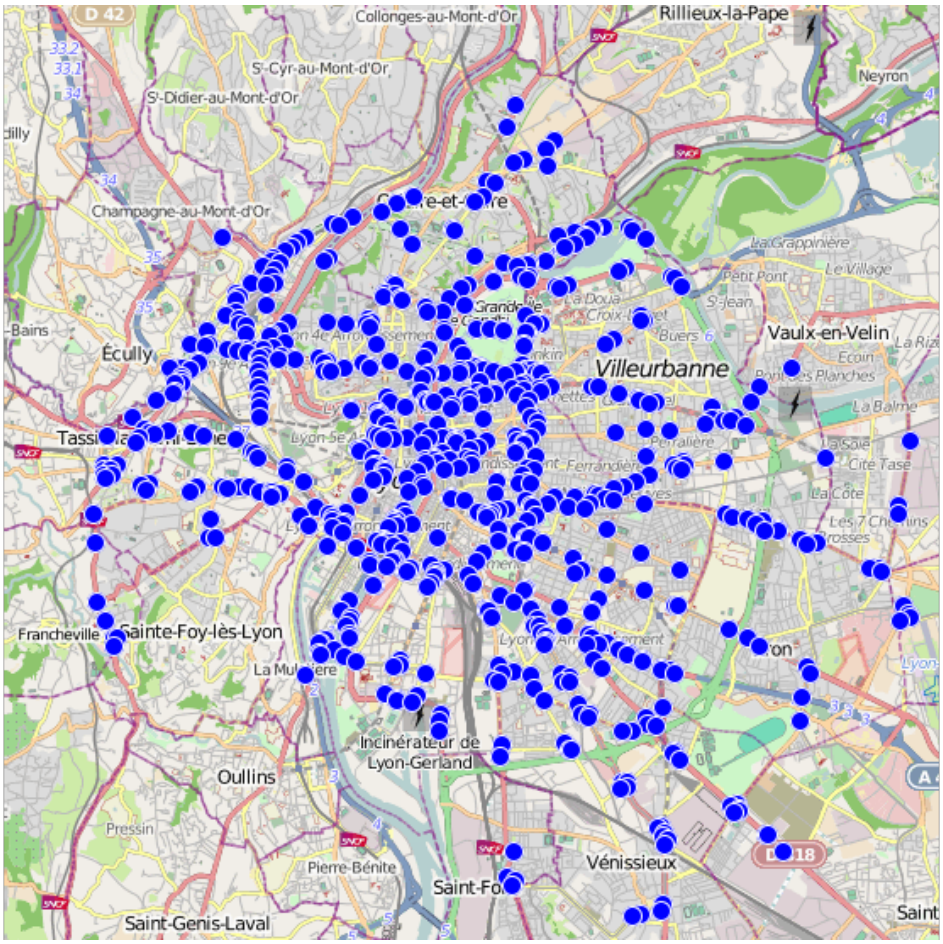


FIGURE 6.2 – Disposition de tous les capteurs pour la ville de Lyon

reflètent cette topologie et son exploitation par la méthode.

Un jeu d'apprentissage a été constitué à partir de l'historique du mois de janvier 2013 pour ces capteurs. Les dynamiques de trafic étant différentes les jours fériés, ainsi que les week-ends, nous avons écarté ce type de journée du jeu de données. Cette approche est courante dans la littérature du trafic, car les modèles ne sont souvent pas conçus de façon à capturer ces différences de dynamiques.

Le jeu de test est composé de quatre journées de semaine, hors vacances, début février.

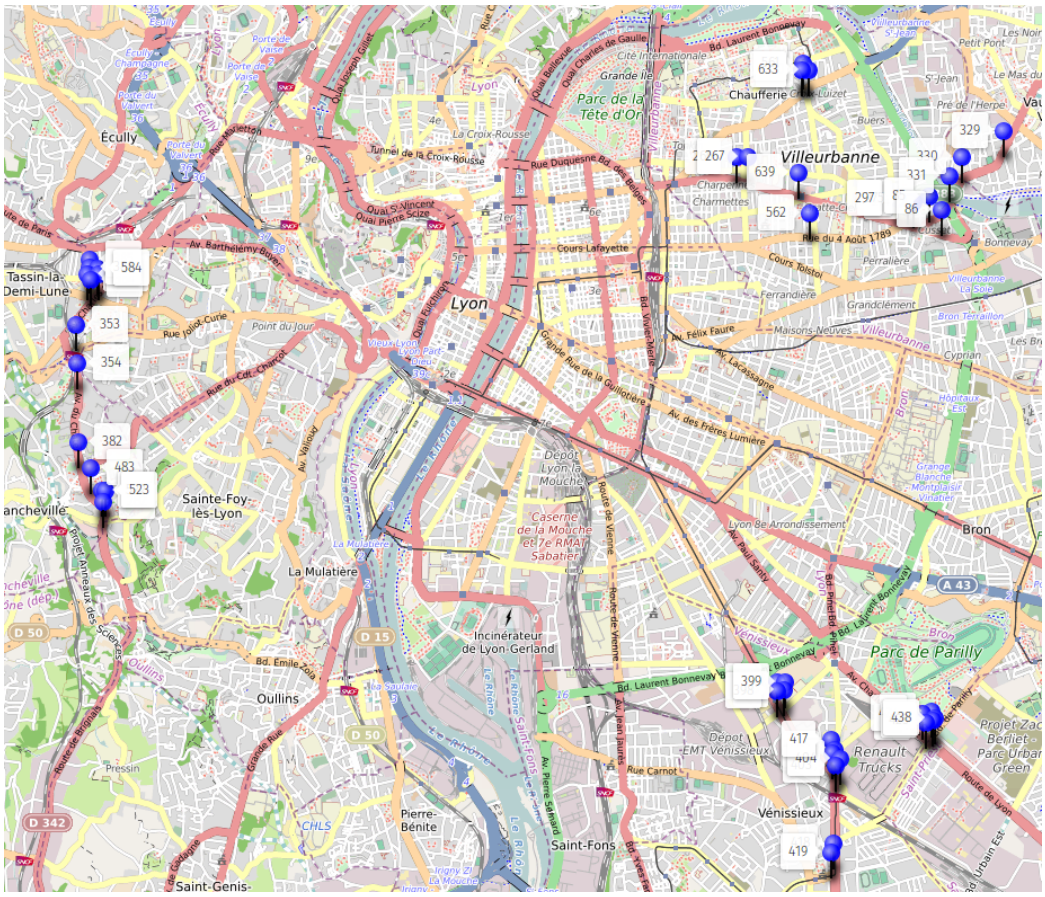


FIGURE 6.3 – Disposition des capteurs pour le sous-ensemble des données nettoyées pour la ville de Lyon

6.1.2 Jeu de données de Marseille

En addition du jeu de données de Lyon, nous disposons également de données pour la ville de Marseille, d'une nature légèrement différente. Alors qu'à Lyon, les capteurs sont installés en grande partie sur le réseau urbain, à l'intérieur de la ville, les données sont ici collectées sur l'autoroute A50, en périphérie immédiate de la ville de Marseille.

Les capteurs (boucles inductives) sont installés pour les deux sens de circulations. On dispose de 24 capteurs sur une portion d'environ 15 kilomètres représentée dans la figure 6.4. Lors de la collecte, le passage de chaque véhicule est enregistré avec l'horodatage associé. A partir de ces enregistrements, il est possible de reconstruire des mesures de débit pour chaque capteur, et de choisir le pas d'agrégation temporelle.

Le jeu d'apprentissage est composé de 20 jours de semaine des mois de

février et mars 2010. Le jeu de test n° 1 est composé de 4 jours de semaine début avril, et le jeu de test n° 2 de quatre jours de vacances mi-avril. Toutes les expérimentations ont été effectuées sur les deux jeux de données. Les résultats étant tous stables entre les deux jeux de données, seuls les résultats du jeu de test n° 1 sont détaillés dans la suite sans que cela ne modifie les conclusions, afin de rendre plus lisible les comparaisons entre le jeu de Lyon et le jeu de Marseille.

FIGURE 6.4 – Disposition des capteurs pour la ville de Marseille

6.1.3 Différences entre trafic urbain et autoroutier

Il existe des différences entre le trafic urbain et autoroutier, aussi bien en terme de structure du réseau que de dynamiques présentes dans les données.

Structure du réseau On représente souvent la structure du réseau routier par un graphe, mais cette représentation peut être multiple. Le premier graphe auquel on peut s'intéresser est une représentation fidèle de la topologie physique du réseau, où chaque noeud du graphe représente une intersection et chaque arc est une route qui relie deux intersections. Ce graphe correspond à l'expérience que l'on a de la lecture d'une carte routière.

Le deuxième graphe que l'on peut considérer va représenter la dynamique du réseau routier, c.-à-d. le déplacement des véhicules sur le réseau. On représente donc un flux d'information qui se propage sur le réseau physique. Lors de la tâche de la prévision de trafic, c'est cette représentation du réseau qui sera la plus utile. Ce flux d'information est mesuré par les capteurs de trafic, et le graphe étudié a donc pour nœuds les tronçons équipés de capteurs, et des arcs relient deux nœuds si l'information peut passer directement d'un tronçon à l'autre, sans tronçon équipé entre les deux.

Or, il est rare de disposer de suffisamment de capteurs pour couvrir l'entièreté des routes sur un réseau urbain. Sur le réseau de Lyon, même avant nettoyage des données, de nombreuses routes ne sont pas équipées de capteur. On mesure donc l'activité de trafic sur les axes principaux, mais pas sur les petites routes à l'intérieur de chaque quartier. Ainsi, à chaque carrefour, certains véhicules peuvent passer d'une section équipée de capteurs à une section non équipée. Il y a donc une partie de l'information du trafic mesuré qui peut s'évaporer et réapparaître dans le graphe sans avoir traversé une séquence d'arcs connectés dans le graphe. Cette disparition d'une partie de l'information peut poser des problèmes lors de l'apprentissage de modèle, car ce qui n'est pas mesuré ne peut donc pas être directement utilisé pour produire une prévision.

A la différence, les tronçons d'autoroutes sont généralement bien équipés, et l'on s'intéresse souvent à une portion linéaire d'autoroute avec quelques entrées et sorties équipées de capteurs. L'information mesurée est donc plus fidèle à la dynamique du trafic, et l'on peut faire l'analogie avec un flux de liquide qui s'écoulerait dans des tuyaux, sans fuite.

Différence d'environnements Alors qu'une autoroute est un environnement plutôt fermé, un réseau urbain comporte de nombreuses sources de variabilité : les habitudes de trafic sont plus complexes, d'autres usagers que les automobilistes (piétons, cyclistes, trottinettes) partagent avec eux une partie de l'espace public et leur comportement, moins prévisible et difficile à mesurer a un impact sur le trafic routier. On peut noter l'influence des livraisons en ville, sur l'apparition de congestions, ainsi que la présence de cycles de feu, parfois dynamiques avec des priorités pour certains modes comme le tramway à Lyon. Ces éléments sont autant de sources de variabilité que l'on va retrouver dans les données mais qui sera difficile à modéliser et à quantifier, ce qui va rendre la tâche de prévision plus complexe.

6.2 Mesures de performance

RMSE La racine carrée de l'erreur quadratique moyenne, RMSE (*Root Mean Squared Error*) est une mesure classique en prévision. Pour une prévision à l'horizon h, elle est donnée par la formule suivante :

$$\mathrm{RMSE} = \sqrt{\frac{1}{T_{\mathrm{test}} - h} \sum_{t=1}^{T_{\mathrm{test}}-h} \left(\hat{x}(t+h) - x(t+h)\right)^2}$$

avec T_{test} le nombre d'exemples dans le jeu de test. La racine carrée permet d'obtenir une mesure dans la même unité que la variable prédite. Cependant, minimiser cette erreur est équivalent à minimiser l'erreur quadratique moyenne, qui est souvent plus simple à manipuler. Le fait d'élever chaque erreur au carré signifie que l'on va pénaliser plus fortement les grandes erreurs que les petites.

MAE L'erreur absolue moyenne, MAE (*Mean Absolute Error*) est une autre mesure fréquemment utilisée. Pour une prévision à l'horizon h, elle est donnée par la formule suivante :

$$\text{MAE} = \frac{1}{T_{\text{test}} - h} \sum_{t=1}^{T_{\text{test}}-h} |\hat{x}(t+h) - x(t+h)|$$

Cette mesure diffère de la RMSE principalement par l'utilisation de la valeur absolue à la place du carré. Cela signifie qu'elle pénalise moins les grandes erreurs que la RMSE. Cela signifie qu'elle est moins sensible à la présence de données aberrantes.

MASE Les séries temporelles de débit mesurées par les capteurs peuvent avoir des amplitudes très différentes, causées par les différences de volume de trafic entre les différentes routes. C'est particulièrement vrai pour un jeu de données de trafic urbain, avec les différents types de rues, de la petite rue aux grandes avenues, parfois même le périphérique. Si l'on souhaite obtenir une mesure de performance à l'échelle du réseau, il n'est pas possible d'utiliser une mesure classique telle que la RMSE (ou la MAE) en la moyennant sur l'ensemble des capteurs, car les différences de volumes de trafic vont influer sur l'amplitude de la RMSE pour la prévision d'un capteur. C'est pourquoi on s'intéresse à une mesure ne présentant pas cette particularité.

La MASE (*Mean Absolute Scaled Error*), proposée par Hyndman et Koehler [44] est une version normalisée de la MAE. Elle est donnée par le ratio entre la MAE de la méthode évaluée (sur le jeu de test) et la MAE de la méthode de prévision naïve consistant à répéter la dernière valeur observée (évaluée, pour plus de robustesse, sur le jeu d'entraînement qui comporte souvent plus d'exemples). Cette mesure est donc sans unité et ne dépend pas de l'amplitude des séries prédites. Le coefficient de normalisation est défini comme suit :

$$Q = \frac{1}{T_{\text{train}} - 1} \sum_{t=2}^{T_{\text{train}}} |x(t) - x(t-1)|$$

où T_{train} est le nombre d'exemples dans le jeu d'entraînement. Pour une prévision à l'horizon h, la MASE est donnée par la formule suivante :

$$\text{MASE} = \frac{1}{T_{\text{test}} - h} \sum_{t=1}^{T_{\text{test}}-h} \frac{|\hat{x}(t+h) - x(t+h)|}{Q}$$

Utiliser le même coefficient de normalisation Q pour chaque horizon de prévisions permet d'évaluer l'évolution de la performance d'une méthode par rapport à l'horizon de prévision. Pour une prévision à un pas de temps ($h = 1$), la MASE peut-être vue comme une comparaison entre la méthode évaluée et la méthode naïve. Une valeur supérieure à un indique que la méthode est moins bonne que la méthode naïve. Une valeur inférieure représente une meilleure performance. C'est une approximation, car Q est calculé sur le jeu d'apprentissage et non de test. Pour les autres horizons de prévision, on ne peut pas directement interpréter une valeur de MASE, mais simplement comparer des valeurs entre elles.

RMSE normalisée On peut préférer l'utilisation de la RMSE à celle de la MAE dans l'évaluation de la prévision d'une série, par exemple quand on souhaite pénaliser fortement les grandes erreurs. Dans ce cas, nous proposons d'adopter une approche similaire à la MASE. On aura alors le coefficient de normalisation suivant :

$$Q = \frac{1}{T_{\text{train}} - 1} \sum_{t=2}^{T_{\text{train}}} (x(t) - x(t-1))^2$$

et la formule de la RMSE normalisée :

$$\text{RMSEN} = \frac{1}{T_{\text{test}} - h} \sum_{t=1}^{T_{\text{test}}-h} \frac{(\hat{x}(t+h) - x(t+h))^2}{Q}$$

Mesure présentée La mesure de performance utilisée pour les résultats présentés dans cette section est la MASE. Les figures présentées ont aussi été générées pour la RMSEN. Le comportement observé pour ces deux mesures est quasi-identique, et l'ordre des méthodes est toujours conservé.

6.3 Comparaison des approches de prévision

Dans cette section sont présentés les résultats expérimentaux sur les deux jeux de données. Dans un premier temps, les mesures de performances utilisées pour évaluer les résultats sont présentées. Puis nous comparons tout

(a) Lyon (b) Marseille

FIGURE 6.5 – Comparaison méthodes linéaires

d'abord entre elles les différentes variantes de méthodes appartenant à une même famille d'approche (régression linéaire, machine à vecteur support (SVR), réseau de neurones (MLP) ou régression k-NN). Ensuite, les meilleurs variantes de chacune de ces familles sont comparées entre elles. Enfin, les résultats sont analysés pour répondre aux problématiques de la thèse : étudier l'utilité des approches de sélection de variables et montrer les différences entre données de trafic urbain et autoroutier, en déterminant quel type d'approche correspond le mieux à chaque contexte.

6.3.1 Comparaison des méthodes de régression linéaire

Dans cette sous-section, nous allons comparer les performances des trois méthodes de régression linéaire retenues : la régression Lasso, le modèle vecteur auto-régressif (VAR), et le modèle ARIMA. La figure 6.5 illustre les performances de ces méthodes sur les données de Lyon et de Marseille.

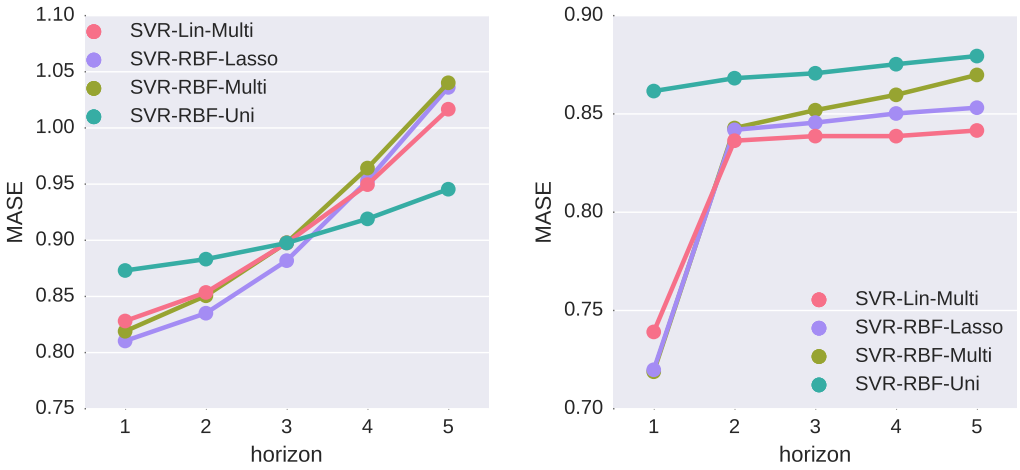
Lyon La méthode ARIMA est largement dominée par les autres, ce qui s'explique par le fait qu'elle est univariée. La régression Lasso donne des résultats légèrement meilleurs que la méthode VAR. Cela peut s'expliquer par le fait qu'elle permet d'apprendre un modèle légèrement plus parcimonieux.

Marseille Sur ce jeu de données également, ARIMA n'est pas compétitive. On observe de nouveau une supériorité de la régression Lasso sur le modèle vecteur autorégressif. Cette supériorité augmente avec l'horizon de prévision

et est nettement plus marquée sur ce jeu de donnée que sur les données urbaines de Lyon. Cela s'explique par le fait que Lasso arrive à apprendre un graphe de dépendance pertinent dans ce cas plus simple, et à l'utiliser pour rendre le modèle parcimonieux.

6.3.2 Comparaison des variantes SVR

Dans cette sous-section, nous allons comparer quatre variantes de la régression à vecteur support. Trois de ces variantes utilisent un noyau gaussian (RBF) : une version univariée (`SVR-RBF-Uni`), une version multivariée sans sélection de variable (`SVR-RBF-Multi`) et une version avec sélection de variables utilisant le graphe de dépendance appris par Lasso (`SVR-RBF-Lasso`). Enfin, on teste également une version multivariée sans sélection de variable avec le noyau linéaire (`SVR-Lin-Multi`). Les résultats sont présentés dans la figure 6.6. Notons que les valeurs étant trop différentes pour les deux jeux de données, pour des raisons de lisibilité, les deux figures côte à côte n'ont pas la même échelle verticale.



(a) Lyon (b) Marseille

FIGURE 6.6 – Comparaison SVR

Lyon On observe tout d'abord que les trois versions multivariées (`SVR-RBF-Multi`, `SVR-RBF-Lasso`, `SVR-Lin-Multi`) ont des meilleures performances que la version univariée (`SVR-RBF-Uni`) pour les courts horizons, mais que la tendance s'inverse pour les horizons plus grands. En réalité, si l'on compare avec les autres familles d'approches, la méthode `SVR` a de mauvais résultats pour

les grands horizons sur ce jeu de données. La méthode univariée est moins affectée. Les résultats ne permettent pas de mettre en avant une supériorité d'un noyau sur l'autre et la sélection de voisinage apporte peu d'amélioration.

Marseille Les méthodes multivariées obtiennent de meilleurs résultats que la version univariée. Il est de nouveau difficile de conclure à la supériorité d'un noyau sur l'autre. On n'observe pas la même dégradation des performances avec l'horizon de prévision que sur les données de Lyon.

6.3.3 Comparaison des variantes MLP

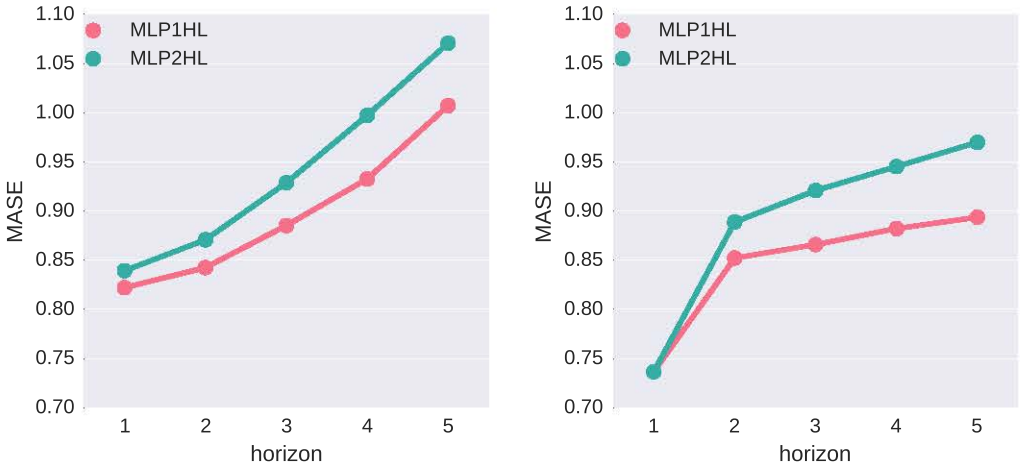
Dans cette sous-section, nous allons comparer deux variantes du perceptron multicouche : une version avec une couche cachée (`MLP1HL`) et une version avec deux couches cachées (`MLP2HL`). Ces deux variantes sont des versions multivariées (la couche d'entrée correspond aux p dernières valeurs observées sur l'ensemble des capteurs). Plusieurs paramétrages ont été testés pour chacune de ces méthodes, en faisant notamment varier la régularisation et le nombre d'époques. Seules le meilleur paramétrage a été retenu pour cette étude. Les résultats sont présentés dans la figure 6.7.

On observe sur les deux jeux de données que la version à une couche cachée est plus performante que la version plus complexe à deux couches cachées. L'écart de performance entre les deux méthodes augmente avec l'horizon de prévision. Même s'il semble contre-intuitif qu'un réseau à deux couches cachées soit moins performant qu'un réseau à une couche cachée, il possède beaucoup plus de paramètres et il est donc possible que le jeu de données ne soit pas assez fourni pour pouvoir converger vers un bon modèle, c.-à-d. que l'on se trouve dans une situation de sur-apprentissage.

6.3.4 Comparaison des variantes k-`NN`

Dans cette sous-section, nous allons comparer le comportement des différentes approches de la famille k-`NN`. On distinguera les versions univariée (`KNN-Uni`), multivariée sans sélection de variable (`KNN-Multi`), multivariée avec sélection de variables lasso (`KNN-Lasso`) et multivariée avec sélection de variables `TiGraMITe` (`KNN-Tig`).

Avant de comparer ces variantes entre elles, nous allons tout d'abord étudier l'impact du choix de la méthode de combinaison des prédictions sur la qualité de la régression k-`NN`.



(a) Lyon (b) Marseille

FIGURE 6.7 – Comparaison MLP

Combinaison des prédictions : rôle de la pondération

Comme présenté en section 4.6, la méthode de régression k-NN est composée de l'identification des k séquences les plus proches, qui permettent de déterminer k possibles prévisions, et de la combinaison de ces k prévisions pour produire une unique prévision. On a ensuite présenté deux manières d'effectuer cette combinaison. La première consiste simplement à choisir la moyenne des k valeurs candidates, comme présenté dans l'équation (4.44). La deuxième approche consiste à donner plus d'importance, parmi les k voisins, à ceux qui ressemblent le plus à la séquence pour laquelle on cherche une prévision. Cela se fait en introduisant des poids dans le calcul de la moyenne ; ici, ces poids correspondent à l'inverse de la distance entre les voisins et la séquence courante, comme présenté dans l'équation (4.49).

On souhaite évaluer l'intérêt d'utiliser l'approche pondérée dans le cas de la prévision de trafic. Pour les différentes versions de la régression k-NN, nous allons comparer sur le jeu de test la performance de prévision de la moyenne classique et pondérée.

Pour les deux jeux de tests du jeu de données de Marseille, les résultats sont similaires et les conclusions identiques. On présente donc uniquement les résultats du jeu de test n° 1. On observe que l'approche pondérée est systématiquement plus performante que l'approche classique, comme illustré dans le tableau 6.1.

Pour le jeu de données de Lyon, cette conclusion est moins évidente. Les résultats sont présentés dans le tableau 6.2. Pour un horizon de deux

pas de temps (12 min) ou moins, la pondération semble utile ; mais pour les horizons plus grands, la moyenne classique semble plus adaptée pour les méthodes ayant une bonne performance de prévision. On peut noter que la régression k-NN univariée (qui est toujours moins bonne que les autres méthodes) est toujours améliorée par l'introduction de la pondération. Dans les résultats présentés par la suite, on conservera l'approche pondérée pour les deux premiers pas de temps, et l'approche classique pour les pas de temps suivants.

TABLE 6.1 – Marseille - MASE

méthode	horizon	classique	pondérée
KNN-Lasso	1	0.7716	**0.7675**
KNN-Tig	1	0.8163	**0.8108**
KNN-Multi	1	0.8257	**0.8203**
KNN-Uni	1	0.8860	**0.8782**
KNN-Tig	2	0.8792	**0.8728**
KNN-Lasso	2	0.8794	**0.8732**
KNN-Multi	2	0.8845	**0.8780**
KNN-Uni	2	0.9231	**0.9139**
KNN-Lasso	3	0.9224	**0.9132**
KNN-Tig	3	0.9224	**0.9132**
KNN-Multi	3	0.9229	**0.9145**
KNN-Uni	3	0.9636	**0.9524**
KNN-Multi	4	0.9548	**0.9456**
KNN-Lasso	4	0.9568	**0.9458**
KNN-Tig	4	0.9568	**0.9458**
KNN-Uni	4	1.0032	**0.9888**
KNN-Multi	5	0.9792	**0.9698**
KNN-Lasso	5	0.9818	**0.9697**
KNN-Tig	5	0.9818	**0.9697**
KNN-Uni	5	1.0440	**1.0290**

Comparaison des différentes variantes k-NN

On compare dans cette sous-section les différentes variantes de k-NN : `KNN-Uni`, `KNN-Multi`, `KNN-Lasso`, `KNN-Tig`. Ces variantes diffèrent par la façon

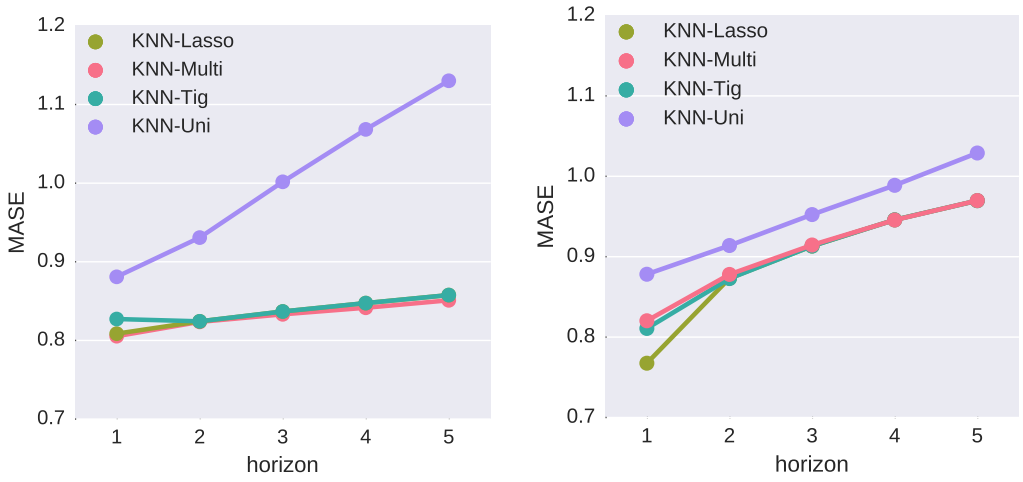
TABLE 6.2 – Lyon - MASE

méthode	horizon	classique	pondérée
KNN-Multi	1	0.8063	**0.8054**
KNN-Lasso	1	0.8103	**0.8086**
KNN-Tig	1	0.8293	**0.8272**
KNN-Uni	1	0.8883	**0.8809**
KNN-Multi	2	0.8238	**0.8236**
KNN-Lasso	2	0.8247	**0.8243**
KNN-Tig	2	0.8247	**0.8243**
KNN-Uni	2	0.9399	**0.9308**
KNN-Multi	3	**0.8332**	0.8335
KNN-Lasso	3	**0.8368**	0.8368
KNN-Tig	3	**0.8368**	0.8368
KNN-Uni	3	1.0017	**0.9925**
KNN-Multi	4	**0.8415**	0.8425
KNN-Lasso	4	**0.8475**	0.8484
KNN-Tig	4	**0.8475**	0.8484
KNN-Uni	4	1.0684	**1.0590**
KNN-Multi	5	**0.8510**	0.8525
KNN-Lasso	5	**0.8577**	0.8593
KNN-Tig	5	**0.8577**	0.8593
KNN-Uni	5	1.1302	**1.1215**

de choisir les variables en entrée du modèle. Les résultats sont présentés dans la figure 6.8.

Lyon Pour ce jeu de données, on observe tout d'abord que la méthode univariée (`KNN-Uni`) n'est pas compétitive face aux alternatives multivariées. Cela semble cohérent avec la nature du trafic en milieu urbain, pour lequel on s'attend à trouver de l'information utile à la prévision dans le voisinage spatial du capteur que l'on prédit.

Ensuite, on peut noter que la variante multivariée sans sélection de variables (`KNN-Multi`) domine les autres, c.-à-d. qu'elle n'est jamais battue. Cela signifie que les différentes stratégies de sélection de variables (Lasso ou `TiGraMITe`) ne permettent pas d'améliorer la prévision. On constate même que la sélection avec `TiGraMITe` dégrade la prévision pour un horizon à très court-terme.



(a) Lyon (b) Marseille

FIGURE 6.8 – Comparaison k-NN

Pour les horizons plus grands, les résultats des trois méthodes multivariées (KNN-Multi, KNN-Lasso, KNN-Tig) convergent, car les graphes de dépendances sélectionnés et utilisés pour la sélection de variables sont tellement denses que la quasi-totalité des variables est conservée. Ainsi, à partir d'un certain horizon, toutes ces approches reviennent finalement à une unique méthode : utiliser toutes les variables pour effectuer la prévision.

Marseille Sur les données de Marseille, on observe également que la méthode univariée est la moins bonne. La variante avec sélection lasso (KNN-Lasso) est la plus performante pour la prédiction à un pas de temps, et a des performances équivalente à la variante sans sélection (KNN-Multi) pour les autres horizons. Cela suggère que le graphe de dépendance appris avec le Lasso est pertinent pour la sélection de variables. Au contraire, la méthode TiGraMITe n'est, une nouvelle fois, pas en mesure de sélectionner un bon sous-ensemble de variables et le graphe obtenu est de nouveau très dense.

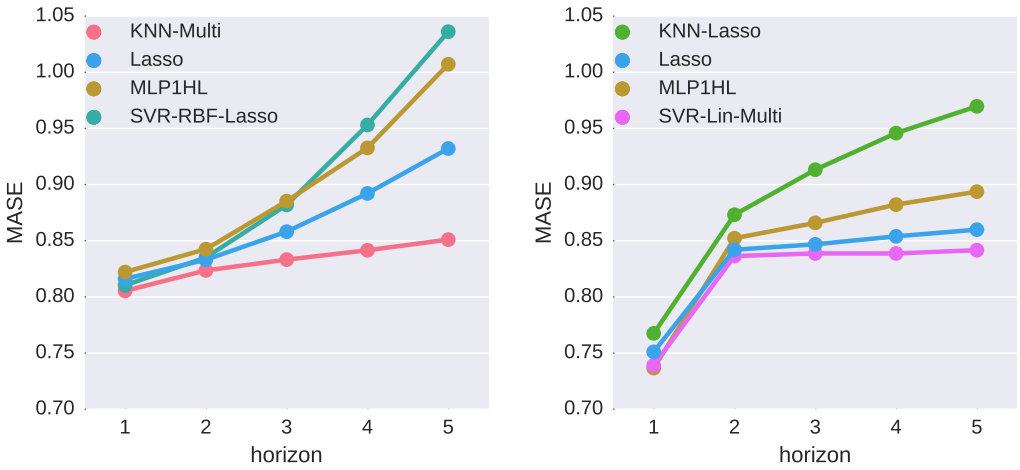
6.3.5 Comparaison des meilleures méthodes

Dans cette sous-section, pour rendre les figures plus lisibles, nous avons à chaque fois conservé uniquement le meilleur représentant pour les différentes familles de méthodes (k-NN, modèles classiques de régression, régression à vecteurs support et perceptron multicouches). Les résultats sont présentés dans la figure 6.9. Les résultats sur les deux jeux de données sont présen-

tés, puis analysé selon deux axes : l'utilité de la sélection de variable et les différences entre prévision de trafic urbain et autoroutier.

Lyon Sur ce jeu de données de la ville de Lyon, on observe que la meilleure méthode est la prévision k-NN multivariée (`KNN-Multi`), suivie par la régression `Lasso`, puis par le réseau de neurones (`MLP1HL`) et la régression à vecteur support (`SVR-RBF-Lasso`). Il est intéressant de noter que la qualité de la prévision du modèle SVR et MLP se dégrade fortement avec l'horizon de prévision.

Marseille Sur ce jeu de données d'autoroute, on observe que les méthodes de prévisions paramétriques sont bien meilleures que la méthode des k-NN. La qualité de prévision k-NN se dégrade très rapidement avec l'horizon de prévision, à la différence des méthodes paramétriques.



(a) Lyon (b) Marseille

FIGURE 6.9 – Comparaison des meilleures méthodes

Incertitude liée à l'horizon de prévision On observe que l'écart de performance entre les méthodes a tendance à augmenter avec l'horizon de prévision, alors que les résultats sont assez proches pour un pas de temps petit. Cette observation est cohérente avec le fait que l'incertitude associée à une prévision est d'autant plus grande que l'on souhaite prédire loin dans le futur. On peut dire que l'horizon le plus petit est plutôt facile à prédire, car il dépend de façon très directe et contrainte de ce qui vient d'être observé.

A mesure que l'on s'éloigne dans le temps, les erreurs de modélisations sont plus pénalisées, car elles ne permettent plus d'extrapoler correctement.

Bilan sur la sélection de variables

Sélection sur données de trafic urbain Sur le jeu de données de Lyon, la régression Lasso est légèrement meilleure que le modèle vecteur auto-régressif. Cela suggère que dans le cas où l'on doit choisir parmi ces deux modèles de régression linéaire, la pénalisation de Lasso permet d'apprendre un modèle plus simple. Cependant, le graphe de dépendance appris par Lasso et utilisé pour faire de la sélection de variables en entrée du k-NN multivarié (KNN-Lasso) n'apporte pas d'amélioration par rapport au modèle k-NN multivarié sans sélection (KNN-Multi). En effet, on constate que le graphe de dépendance est très dense et ne permet donc presque pas d'éliminer de variable. La différence que l'on observe entre VAR et Lasso est donc plutôt imputable à la contrainte sur la taille des coefficients imposée par le terme de régularisation du Lasso, qui contraint tout de même le modèle à être plus simple et contrôle donc le sur-apprentissage. Mais cette régularisation permet donc uniquement de simplifier le modèle, sans réellement être capable d'éliminer complètement des variables. Elle est donc utile lors de l'apprentissage d'un modèle linéaire, mais ne peut pas être utilisée pour intégrer de l'information à priori dans d'autres types de modèle. On a constaté également que le graphe appris avec TiGraMITe est également très dense et ne permet pas de sélection efficace.

Graphe de dépendance sur données urbaines La difficulté d'apprendre un graphe de dépendance utile pour les données de Lyon peut surprendre, notamment au vue de la nature des phénomènes de trafic urbain. En effet, on s'attend à ce que les capteurs proches contiennent de l'information utile, alors que les capteurs très éloignés semblent peu pertinents et devraient être écartés. Une explication possible tient dans le fait que la relation de dépendance entre deux tronçons de route dépend de leurs états de trafic respectif : un tronçon avec un trafic fluide va avoir une influence causale sur les tronçons situés en aval, alors qu'un tronçon présentant de la congestion vont avoir une influence sur les tronçons en amont. En effet, les ondes de congestions se propagent dans le sens inverse de la circulation. Cette propriété est difficile à capturer si on apprend un unique graphe de dépendance, car il ne pourra pas tenir compte des différentes combinaisons possibles d'états de trafic sur l'ensemble des capteurs.

Ainsi, en s'appuyant sur les connaissances des experts du trafic, nous avons donc essayé de valider cette intuition en décomposant la journée en périodes pendant lesquelles on peut supposer que le sens de cette influence

causale ne change pas. Cela revient notamment à isoler les périodes de forte congestion. On obtient alors une segmentation de la journée en cinq régimes distincts (nuit, pics du matin et de l'après-midi, et régimes intermédiaires entre les deux pics et le soir) [51]. Cette segmentation répond à l'objectif d'apprendre la structure du graphe dans des conditions les plus homogènes possibles. Cependant, cela n'a pas permis d'améliorer l'apprentissage de la structure de dépendance. Cet échec est probablement dû au fait qu'en milieu urbain, l'état du trafic à l'échelle du réseau est composé des conditions de trafic, localement à chaque tronçon. Différents tronçons peuvent passer d'un état fluide à un état congestionné à des pas de temps différents, et il n'y pas de raisons que cette transition se passe simultanément sur chaque tronçon. Cela signifie, que durant la journée, à chaque pas de temps, il est possible que certains tronçons passent d'un régime à l'autre, ce qui impacte localement la structure de dépendance avec les tronçons voisins. Il serait donc nécessaire à chaque fois d'utiliser un graphe de dépendance spécifique à ce nouvel état global. Cela permet d'expliquer pourquoi, même avec une segmentation en cinq régimes distincts identifiés par les experts du trafic, il reste difficile de capturer cette structure de dépendance très instable. Ces dynamiques de trafic contribuent à la grande difficulté de la prévision en milieu urbain.

Importance d'un graphe parcimonieux Comme évoqué précédemment, les performances de `KNN-Tig` et `KNN-Lasso` tendent vers celles de `KNN-Multi` quand l'horizon augmente. Cela peut s'expliquer par le fait que quand l'horizon augmente, le nombre de parents (dans le graphe de dépendance) utilisés pour la prédiction augmente. Cela nous rapproche donc du cas multivarié sans sélection, où les variables utilisées sont en fait tous les nœuds du graphe de dépendance.

Par exemple, pour prédire la valeur de x_i à l'horizon $h = 3$, on cherche dans le graphe les parents directs de la variable $X_i(t+3)$, qui seront notés $\mathcal{P}_{X_i(t+3)}$. Ces derniers devraient être sélectionnés comme variables prédictives, mais certaines de ces variables peuvent encore appartenir au futur non-observé – par exemple, si on a $X_j(t+1) \in \mathcal{P}_{X_i(t+3)}$ – et l'information qu'elles contiennent n'est donc pas disponible à l'instant t. Ainsi, pour obtenir une partie de cette information, nous remplaçons ces variables par leurs parents, par exemple $\mathcal{P}_{x_j(t+1)}$. Cette procédure est répétée de façon récursive, jusqu'à ce que les variables sélectionnées soient toutes évaluées à l'instant t ou à un instant antérieur. Cette méthode permet d'obtenir l'ensemble des variables sélectionnées, noté $P(i, 3)$, qui sera utilisé pour prédire $X_i(t+3)$. À chaque étape de la récursion, une variable est remplacée par l'ensemble de ses parents, qui peut contenir de nombreuses variables – d'autant plus

quand le graphe est dense. C'est cette raison qui explique l'augmentation du nombre de variable sélectionnées quand l'horizon de prévision augmente. Cela se comprend intuitivement : pour prédire le futur très proche, on va s'intéresser uniquement aux tronçons très proches dans le réseau routier. Mais quand on s'éloigne dans le futur, l'information utile pour la prévision est diluée dans les différentes parties du réseau qui sont connectées au tronçon étudié. L'impact de la densité du graphe de dépendance souligne la nécessité de garder le graphe parcimonieux, autant que possible, c.-à-d. en limitant au maximum les arêtes du graphe qui ne contribuent pas à améliorer la prévision.

Comparaison entre trafic urbain et autoroutier

Autoroute : méthodes linéaires performantes Le fait que les méthodes linéaires comme la régression `Lasso` ou le SVR avec un noyau linéaire (`SVR-Lin-Multi`) donnent de bons résultats sur les données de Marseille indique qu'une approche paramétrique simple est probablement plus adaptée pour modéliser le trafic autoroutier qu'une approche basée sur des critères de théorie de l'information qui font très peu d'hypothèses sur la nature de la dépendance. Dans le cas de lasso, le paramètre de régularisation fait tendre de nombreux coefficients vers zéro, et on obtient un nombre de paramètres plus faible. Cette parcimonie du modèle permet de mieux généraliser, en limitant le sur-apprentissage. Le SVR dispose également d'un mécanisme contre le sur-apprentissage, à travers les paramètres C et ϵ qui contrôlent le nombre de vecteurs support et la complexité du modèle. Une bonne valeur de ces paramètres est obtenue par validation croisée. L'usage du noyau linéaire dans ce cas permet également de limiter la taille de la famille d'hypothèse considérée. Rappelons qu'une famille de modèle contrainte par de bonnes hypothèses sur la structure de dépendance permet de converger plus rapidement (en terme du nombre d'observations nécessaires) vers un bon modèle final.

Urbain : flexibilité du k-`NN` La bonne performance de la méthode k-NN peut s'expliquer par sa flexibilité. On n'apprend pas vraiment un modèle qui doit être pertinent sur l'ensemble des données. Pour chaque prévision, seul un petit nombre d'exemples du jeu de données sont utilisés. Les prévisions peuvent donc être très spécifiques à la situation observée, sans être contaminées par les exemples correspondant à des situations trop différentes. Or dans un cadre aussi dynamique que la ville, il est difficile d'apprendre un modèle avec seulement quelques paramètres qui rendent réellement compte de la dynamique. L'hypothèse de dépendance linéaire entre les variables explicatives et la variable expliquée est certainement loin d'être respectée. La méthode k-NN présente l'avantage de ne pas faire d'hypothèse sur la forme

de la dépendance, et se concentre sur le futur des situations similaires contenues dans l'historique. Le risque théorique de sur-apprentissage est fort pour ce type d'approche, mais il semble que dans ce cas, la méthode marche en pratique mieux que ses concurrentes.

6.4 Étude de l'impact de la résolution temporelle

Dans cette section, nous proposons une évaluation de l'impact de la résolution temporelle des données sur la performance des méthodes de prévision.

Comme mentionné dans le chapitre 3, le choix de la résolution temporelle des données a un impact sur la qualité de prévision des méthodes. Cependant, il n'existe pas de méthode de référence pour choisir la meilleure résolution, ce choix étant laissé à l'appréciation du modélisateur.

Nous avons présenté le compromis associé à ce choix : plus la résolution temporelle est fine, plus les données comportent du bruit et des fluctuations importantes qui peuvent perturber l'apprentissage des modèles ; moins la résolution est fine, plus les données sont lissées et le bruit éliminé, mais au risque de perdre de l'information importante pour la prévision.

Sur le jeu de données de Lyon, le pas d'agrégation temporelle est fixé à 6 minutes dès la collecte des données et ne peut pas être modifié. Nous avons identifé cela comme une limite, car cette résolution ne peut pas capturer les phénomènes très dynamiques du trafic urbain comme les cycles de feux de signalisation, par exemple. Cette information, qui pourrait être utile à la prévision, se retrouve dans les données sous la forme d'un bruit qui ne peut pas être exploité par les méthodes de prévision.

Sur le jeu de données de Marseille, les données collectées sont fournies avec l'horodatage exact du passage d'un véhicule sur le capteur, et nous pouvons donc reconstruire les séries temporelles de débit en choisissant la résolution temporelle de l'agrégation. Ceci est donc une opportunité pour étudier l'impact du pas d'agrégation temporel sur la qualité de la prévision.

Méthodologie Nous avons choisi de comparer les résultats de prévision pour des données agrégées par pas de 2, 3 et 6 minutes afin de déterminer la résolution la plus adaptée. Les méthodes de prévision ne peuvent pas fournir des prévisions avec une résolution plus fine que les données sur lesquelles elles sont entraînées. Ainsi, les méthodes entraînées avec un pas de temps de 6 minutes ne pourront fournir des prévisions que par tranches de 6 minutes,

ce qui sera le facteur limitant dans notre cas d'étude. Nous avons choisi de comparer la performance de prévision des méthodes pour des prévisions par tranches de 6 minutes jusqu'à un horizon maximal de 30 minutes.

Pour les méthodes entraînées sur les jeux de données avec un pas d'agrégation de 2 ou 3 minutes, une adaptation est nécessaire pour pouvoir comparer les performances avec celles des méthodes entraînées sur les données avec un pas de 6 minutes.

Prenons l'exemple des données avec un pas d'agrégation de 3 minutes. La prévision du débit moyen pour les 3 prochaines minutes correspond à prédire à un pas de temps dans le futur. Le prévision du débit moyen prédit entre 3 et 6 minutes dans le futur correspond à la prévision à un horizon de 2 pas de temps. Pour produire une prévision du débit moyen pour les 6 prochaines minutes, il faut donc moyenner les prévisions à un et deux pas de temps.

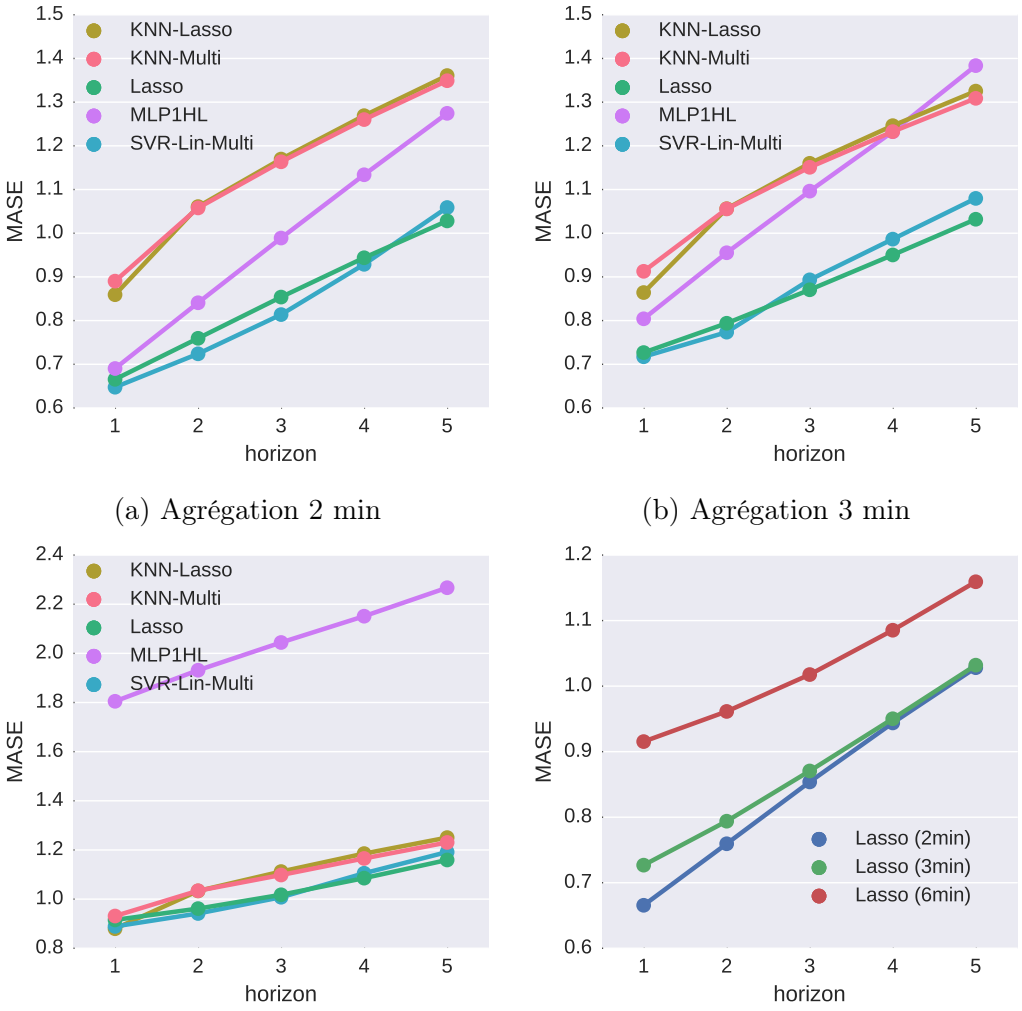
Résultats Les résultats expérimentaux sont présentés dans la figure 6.10. Les méthodes ont été entraînées sur des données avec un pas d'agrégation de 2, 3 et 6 minutes. Les prévisions ont ensuite été effectuées par tranches de 6 minutes jusqu'à un horizon maximum de 30 minutes. Ainsi, pour toutes les sous-figures, l'horizon 1 correspond à 6 minutes et l'horizon 5 correspond à 30 minutes. Les échelles verticales des sous-figures ne sont pas les mêmes, car les résultats diffèrent trop d'une sous-figure à l'autre.

Les trois premières sous-figures présentent les performances des méthodes pour une valeur du pas d'agrégation fixée. La dernière sous-figure présente les performances de la meilleure méthode de prévision (`Lasso`) selon le pas d'agrégation des données sur lesquelles on l'a entraînée.

La première observation intéressante est que l'ordre de performance des méthodes est peu modifié lorsque le pas d'agrégation change. On observe que les modèles paramétriques linéaires (`Lasso`, `SVR-Lin-Multi`) sont toujours les plus performants pour ces données d'autoroutes de Marseille. On observe toutefois une très mauvaise performance du réseau de neurones `MLP1HL` pour le pas d'agrégation de 6 minutes. Cela est dû à la diminution du nombre d'exemples d'entraînement causée par l'agrégation (pour la même période, on aura deux fois moins d'exemples pour un pas d'agrégation de 6 minutes que pour un pas d'agrégation de 3 minutes). Les réseaux de neurones ayant besoin de beaucoup de données pour converger vers un bon modèle, cela explique la mauvaise performance pour les données les plus agrégées.

Enfin, l'observation la plus importante pour notre étude se trouve dans la sous-figure qui compare les performances de la méthode `Lasso` pour les différents pas d'agrégation. On observe que plus la résolution est fine (2min), plus la performance est bonne. Cette tendance est plus importante pour les

horizons de prévision à très court-terme (moins de 12 minutes). Ce résultat est logique, car pour les prévisions à plus long terme, l'impact des fluctuations à haute fréquence est lissé et la tendance générale est plus importante pour effectuer une prévision.



(a) Agrégation 2 min

(b) Agrégation 3 min

(c) Agrégation 6 min

(d) Comparaison de l'agrégation

FIGURE 6.10 – Étude de l'impact du pas d'agrégation sur la prévision

Discussion Les résultats présentés dans cette section apportent un élément de réponse à la question de la meilleure résolution pour les données de trafic. On a montré pour ces données d'autoroute qu'une résolution fine (2 minute) permet de produire de meilleures prévisions, même pour un horizon

de prévision élevé (30 min). Cela vient confirmer l'hypothèse formulée dans certains travaux [106] que le lissage des données est contraire à la nature du phénomène de trafic, et que de l'information cruciale peut être capturée dans les fluctuations à haute fréquence des données de débit.

Ces observations nous confortent également dans l'idée que les données de trafic urbain de Lyon agrégées par pas de 6 minutes sont insuffisantes pour capturer les dynamiques à très-court terme du trafic, et que la performance serait probablement améliorée par l'utilisation de données avec une résolution temporelle plus fine.

Chapitre 7

Discussion

7.1 Synthèse

L'étude de la littérature sur la prévision de trafic à court-terme à permis de dégager plusieurs problématiques que nous avons souhaité traiter dans les travaux de la thèse. Tout d'abord, de nombreux types d'approches ont été proposés pour prédire le trafic, le plus souvent pour des données d'autoroute et plus rarement pour des données urbaines. Nous avons souhaité évaluer la performance des méthodes sur ces deux types de réseau, afin de pouvoir mettre en lumière les différences de dynamiques entre ces deux cadres applicatifs et de souligner les avantages et faiblesses des différentes approches pour ces deux problèmes.

Nous avons également identifié la nécessité de pouvoir sélectionner, parmi toute l'information collectée sur le réseau, le voisinage spatio-temporel pertinent pour la prévision. Pour contribuer à la résolution de cette question, nous avons comparé deux approches aux propriétés différentes : la sélection `Lasso` et `TiGraMITe`.

Enfin, la problématique du choix de la résolution temporelle des données est une question ouverte de la littérature sur la prévision à court-terme. Nous avons proposé une étude de l'impact de la résolution temporelle en reconstruisant les séries temporelles de débit avec différents pas d'agrégation et en comparant les performances des méthodes pour ces différentes résolutions temporelles.

Une contribution importante de la thèse a été l'application de l'approche de sélection de variable `TiGraMITe` dans le contexte de l'étude du trafic. Cette méthode se montrait prometteuse pour de nombreuses raisons. Tout d'abord, l'introduction théorique de cette méthode et son application à des

cas d'études complexes comme les phénomènes climatiques extrêmes ont donné lieu à de nombreuses publications dans des revues prestigieuses de physique. Ces travaux démontraient le potentiel de cette méthode pour l'étude de phénomènes spatio-temporels complexes partageant des similarités avec le phénomène de trafic. Nous avons identifié dans la littérature la nécessité de prendre en compte la nature non-linéaire du trafic, surtout dans le cadre urbain, et constaté que l'approche de sélection ayant jusqu'à présent donné les meilleurs résultats, la sélection Lasso, était basée sur une hypothèse de linéarité et n'avait été testée jusqu'à présent que sur des données d'autoroute. Notre étude expérimentale a permis de confirmer l'intérêt de la sélection Lasso pour les données d'autoroute, mais également de démontrer son incapacité à identifier la structure de dépendance pour les données urbaines, c.-à-d. en présence de non-linéarité. Nous avons également démontré que l'approche TiGraMITe, malgré sa capacité théorique à capturer des dépendances non-linéaires, n'a pas permis d'apprendre un graphe de dépendance qui améliore la prévision. Cela met en évidence la réelle difficulté d'identifier un voisinage spatio-temporel pertinent dans un réseau urbain pour lequel des changements d'états de trafic apparaissent très fréquemment sur les différents segments de routes, changeant ainsi dynamiquement la structure de dépendance entre les capteurs tout au long de la journée. Nous avons également montré qu'un découpage simple de la journée en différentes périodes ne permet pas d'homogénéiser suffisamment les conditions pour atténuer ce problème.

Une volonté forte qui a guidé les travaux de la thèse était de sortir du cloisonnement entre les différents types d'approches appliquées à la prévision de trafic. Nous avons souhaité que notre étude expérimentale puisse refléter la diversité des familles de méthodes de prévision afin de pouvoir comparer ces approches sur les mêmes jeux de données. Ces jeux de données devaient également être représentatifs des différents réseaux de transports, à savoir les réseaux urbains et les réseaux d'autoroute. Le projet de thèse ayant été proposé en partenariat avec la Métropole de Lyon, nous souhaitions identifier les caractéristiques des méthodes permettant de prédire le trafic urbain. Nous avons démontré la pertinence d'utiliser des méthodes paramétriques linéaires (la régression Lasso et la régression à vecteurs de support avec noyau linéaire) pour capturer la dynamique du trafic sur autoroute et le prédire efficacement. Nous avons également confirmé les limitations de ces approches pour modéliser le trafic urbain. Nous avons montré qu'une méthode non-paramétrique comme la régression k-NN, est la plus adaptée pour le trafic urbain, grâce à sa flexibilité et sa capacité à retrouver dans l'historique les situations très similaires à la situation courante. L'étude expérimentale a également confirmé

la pertinence du passage des approches classiques de séries temporelles à des approches d'apprentissage automatique.

Enfin, nous avons proposé une réponse à la question du choix de la résolution temporelle des données qu'on donne en entrée aux modèles. L'absence de méthodologie de référence face à ce problème provient du compromis suivant : une résolution trop fine entraîne d'importantes fluctuations à haute fréquence, qui peuvent compliquer l'apprentissage des modèles, mais une résolution trop faible permettant de filtrer le bruit risque d'éliminer également de l'information utile pour la prévision. Nous avons montré dans cette étude qu'une résolution très fine (2 min) fournit les meilleurs résultats de prévision même pour un horizon lointain (30 min). Nous confirmons donc l'hypothèse formulée par Vlahogianni et al. [106] que les fluctuations à haute fréquence sont une partie essentielle de la dynamique de trafic, et qu'il est nécessaire de les conserver pour assurer une bonne performance de prévision. Nous émettons également l'hypothèse que cette conclusion est encore plus importante pour le trafic urbain, comportant de nombreuses dynamiques à très court-terme.

7.2 Limites et perspectives

Une des limites identifiées dans la thèse est de ne pas avoir étudié les méthodes sur un jeu de données comportant les mesures collectées par un nombre plus important de capteurs. C'est une perspective envisageable mais qui présente deux défis importants : le passage à l'échelle des expérimentations pour effectuer la comparaison d'un nombre important de méthodes de prévision sur ce jeu de données conséquent et principalement l'effort de nettoyage des données pour pouvoir constituer un tel jeu de données. En effet, les sources de valeurs manquantes et aberrantes sont nombreuses : erreur lors de l'enregistrement des mesures, panne du capteur physique, véhicule garé sur le capteur. Pour de nombreux capteurs, on est en présence de longues plages de valeurs manquantes qui ne peuvent pas être imputée avec les mêmes méthodes que pour des valeurs manquantes isolées. La littérature de l'imputation de valeurs manquantes souligne également la nécessité de distinguer les valeurs manquantes selon le processus qui les a généré (valeurs manquantes complètement au hasard, valeurs manquantes au hasard, valeur ne manquant pas au hasard) [82]. Enfin, il est parfois délicat de distinguer les valeurs aberrantes causées par des erreurs de mesures et les valeurs extrêmes mais réelles du trafic.

Une autre limite du jeu de données est le mécanisme de collecte des mesures. Tout d'abord, la résolution temporelle des données de Lyon : un pas d'agrégation de 6 minutes ne permet pas de capturer les dynamiques à très court-terme, comme l'impact des feux de signalisation par exemple. Il serait intéressant d'étudier des données de trafic urbain avec une résolution temporelle plus fine, comme l'a démontré notre étude sur les données de Marseille. Enfin, on ne disposait pour ces travaux que de données issues de capteurs de type boucle inductive installés sous la route. Même si le fait de pouvoir produire soi-même et gérer les données est un enjeu majeur de gouvernance pour les métropoles, il faut souligner les performances de prévision obtenues par les géants de la donnée, notamment à partir des données collectés par les dispositifs de navigation utilisés par les conducteurs (GPS, smartphone). La problématique de fusion des données issues de différentes méthodes de collecte est une perspective importante pour la recherche en prévision de trafic [27].

La volonté dans la thèse de représenter la diversité des approches de prévision présente également une limite : il n'a pas été possible d'explorer les méthodes les plus sophistiquées appartenant à chaque famille d'approche. Par exemple, bien que l'étude présente des résultats pour des réseaux de neurones simples (MLP) qui ont été mis en avant dans la littérature pour leur performance, nous n'avons pas pu creuser dans la direction de l'apprentissage profond qui obtient des résultats impressionnants dans de nombreux domaines. Ce compromis dans la thèse est toutefois assumé, et nous soulignons que l'intérêt du travail ne réside pas uniquement dans la performance de prévision de la meilleure méthode mais également dans les enseignements que l'on tire sur les tâches de prévision de trafic en contexte urbain et autoroutier. Enfin, la présentation des méthodes moins sophistiquées de chaque famille d'approche présente un intérêt pour le gestionnaire d'infrastructure qui doit résoudre une problématique opérationnelle en prenant en compte différents facteurs : le coût d'implémentation de la méthode de prévision, l'explicabilité du modèle appris, le coût de calcul et de paramétrage des modèles, l'importance de pouvoir former et de conserver l'expertise en interne afin de pouvoir surveiller et diagnostiquer le comportement du modèle.

La limite d'une étude comparative d'un nombre important de méthodes réside dans le fait que l'expérimentateur ne peut pas être expert de toutes les méthodes. On dit ainsi parfois que la comparaison mesure plus le niveau de compétence de l'expérimentateur dans chacune des méthodes que leurs performances relatives. Il faut nuancer ce propos en soulignant qu'on a limité le

niveau de sophistication des méthodes étudiées afin de pouvoir garantir une comparaison équitable des méthodes en consacrant au paramétrage de chacune une attention complète. Cependant, il est certain que la communauté de recherche sur la prévision de trafic a besoin de permettre aux spécialistes de chaque méthode de faire valoir tous les atouts des méthodes qu'ils proposent et de fournir une comparaison équitable avec les autres approches.

Les dernières années ont été marquées par l'émergence de plate-formes en ligne hébergeant des compétitions d'apprentissage artificiel, permettant aux participants de soumettre leur modèles et d'être évalués sur la performance de prévision de ces modèles. Les données d'entraînement (entrées et sorties) sont fournies à tous les participants, ce qui leur permet d'entraîner les modèles. Un jeu de test est également fourni (entrées uniquement). A partir de ces entrées de test, les participants calculent la sortie du modèle et soumettent le résultat à la plate-forme, sur laquelle est calculée l'erreur de prévision à partir des sorties du jeu de test (qui ne sont pas rendues publiques). Le nombre de soumission par participant est limité pour qu'on ne puisse pas apprendre indirectement sur le jeu de test en utilisant le score fourni par la plate-forme. Souvent, dans une dernière phase, les modèles retenus pour les meilleurs participants sont réévalués sur un nouveau jeu de test, inutilisé jusqu'à ce moment.

Nous pensons qu'un effort collectif de la communauté de recherche sur la prévision de trafic à court-terme pourrait permettre de créer une collection de jeux de données de différentes caractéristiques (type de réseau, type de variable de trafic, résolution temporelle, volume de données) qui servirait de base à l'organisation de compétitions de prévision permettant aux experts de mettre en valeur les performances de leurs approches.

Cette proposition est d'autant plus importante que la question de la reproductibilité des travaux est une problématique majeure pour le futur de la recherche. Disposer d'un ensemble de jeu de données de référence, et permettre aux participants de publier leur code ainsi qu'un rapport décrivant les modèles utilisés et leur implémentation est un pas ambitieux, mais nécessaire, pour permettre de dégager de réels enseignements de la masse toujours croissante de travaux dans le domaine.

Bibliographie

[1] Mohammed S. Ahmed and Allen R. Cook. *Analysis of freeway traffic time-series data by using Box-Jenkins techniques*. Number 722. 1979.

[2] AiLing Ding, XiangMo Zhao, and LiCheng Jiao. Traffic flow time series prediction based on statistics learning theory. In *Proceedings. The IEEE 5th International Conference on Intelligent Transportation Systems*, pages 727–730, September 2002.

[3] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6) :716–723, December 1974.

[4] Hirotogu Akaike. Information Theory and an Extension of the Maximum Likelihood Principle. In Emanuel Parzen, Kunio Tanabe, and Genshiro Kitagawa, editors, *Selected Papers of Hirotugu Akaike*, Springer Series in Statistics, pages 199–213. Springer New York, New York, NY, 1998.

[5] N. S. Altman. An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician*, 46(3) :175–185, August 1992.

[6] Masako Bando, Katsuya Hasebe, Akihiro Nakayama, Akihiro Shibata, and Yuki Sugiyama. Dynamical model of traffic congestion and numerical simulation. *Physical review E*, 51(2) :1035, 1995.

[7] Jaime Barceló and Jordi Casas. Dynamic network simulation with aimsun. In *Simulation approaches in transportation analysis*, pages 57–98. Springer, 2005.

[8] George EP Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. *Time series analysis : forecasting and control*. John Wiley & Sons, 2015.

[9] Pinlong Cai, Yunpeng Wang, Guangquan Lu, Peng Chen, Chuan Ding, and Jianping Sun. A spatiotemporal correlative k-nearest neighbor model for short-term traffic multistep forecasting. *Transportation Research Part C : Emerging Technologies*, 62 :21–34, January 2016.

[10] Manoel Castro-Neto, Young-Seon Jeong, Myong-Kee Jeong, and Lee D. Han. Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions. *Expert Systems with Applications*, 36(3) :6164–6173, April 2009.

[11] Joseph E. Cavanaugh. Unifying the derivations for the Akaike and corrected Akaike information criteria. *Statistics & Probability Letters*, 33(2) :201–208, April 1997.

[12] Srinivasa Ravi Chandra and Haitham Al-Deek. Predictions of Freeway Traffic Speeds and Volumes Using Vector Autoregressive Models. *Journal of Intelligent Transportation Systems*, 13(2) :53–72, May 2009.

[13] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1) :16–28, January 2014.

[14] H. Chang, Y. Lee, B. Yoon, and S. Baek. Dynamic near-term traffic flow prediction : systemoriented approach based on past experiences. *IET Intelligent Transport Systems*, 6(3) :292–305, September 2012.

[15] Chenyi Chen, Yin Wang, Li Li, Jianming Hu, and Zuo Zhang. The retrieval of intra-day trend and its influence on traffic prediction. *Transportation research part C : emerging technologies*, 22 :103–118, 2012.

[16] Tao Cheng, James Haworth, and Jiaqiu Wang. Spatio-temporal autocorrelation of road network data. *Journal of Geographical Systems*, 14(4) :389–413, October 2012.

[17] Axel Cleeremans, David Servan-Schreiber, and James L McClelland. Finite state automata and simple recurrent networks. *Neural computation*, 1(3) :372–381, 1989.

[18] Yuliang Cong, Jianwei Wang, and Xiaolei Li. Traffic Flow Forecasting by a Least Squares Support Vector Machine with a Fruit Fly Optimization Algorithm. *Procedia Engineering*, 137 :59–68, 2016.

[19] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3) :273–297, September 1995.

[20] M Cremer and J Ludwig. A fast simulation model for traffic flow on the basis of boolean operations. *Mathematics and Computers in Simulation*, 28(4) :297–303, 1986.

[21] Carlos F. Daganzo. The cell transmission model : A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research Part B : Methodological*, 28(4) :269–287, August 1994.

[22] Davis Gary A. and Nihan Nancy L. Nonparametric Regression and Short-Term Freeway Traffic Forecasting. *Journal of Transportation Engineering*, 117(2) :178–188, March 1991.

[23] P. Dell'Acqua, F. Bellotti, R. Berta, and A. De Gloria. Time-Aware Multivariate Nearest Neighbor Regression Methods for Traffic Flow Prediction. *IEEE Transactions on Intelligent Transportation Systems*, 16(6) :3393–3402, December 2015.

[24] Qing Yan Ding, Xi Fu Wang, Xiu Yuan Zhang, and Zhan Quan Sun. Forecasting Traffic Volume with Space-Time ARIMA Model. *Advanced Materials Research*, 156-157 :979–983, October 2010.

[25] Mark Dougherty. A review of neural networks applied to transport. *Transportation Research Part C : Emerging Technologies*, 3(4) :247–260, August 1995.

[26] Michael Eichler. Graphical modelling of multivariate time series. *Probability Theory and Related Fields*, 153(1) :233–268, June 2012.

[27] Nour-Eddin El Faouzi, Henry Leung, and Ajeesh Kurian. Data fusion in intelligent transportation systems : Progress and challenges–a survey. *Information Fusion*, 12(1) :4–10, 2011.

[28] Stefan Frenzel and Bernd Pompe. Partial mutual information for coupling analysis of multivariate time series. *Physical Review Letters*, 99(20) :204101, November 2007.

[29] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA :, 2001.

[30] G. Gardner, Andrew C. Harvey, and Garry DA Phillips. Algorithm AS 154 : An algorithm for exact maximum likelihood estimation of autoregressive-moving average models by means of Kalman filtering. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 29(3) :311–322, 1980.

[31] M. Gentili and P. B. Mirchandani. Locating sensors on traffic networks : Models, challenges and research opportunities. *Transportation Research Part C : Emerging Technologies*, 24 :227–255, October 2012.

[32] Nikolas Geroliminis, Jack Haddad, and Mohsen Ramezani. Optimal perimeter control for two urban regions with macroscopic fundamental diagrams : A model predictive approach. *IEEE Transactions on Intelligent Transportation Systems*, 14(1) :348–359, 2012.

[33] B. Ghosh, B. Basu, and M. O'Mahony. Multivariate Short-Term Traffic Flow Forecasting Using Time-Series Analysis. *IEEE Transactions on Intelligent Transportation Systems*, 10(2) :246–254, June 2009.

[34] Xiaoyan Gong and Feiyue Wang. Three improvements on KNN-NPR for traffic flow forecasting. In *Proceedings. The IEEE 5th International Conference on Intelligent Transportation Systems*, pages 736–740, September 2002.

[35] Mohammad M. Hamed, Hashem R. Al-Masaeid, and Zahi M. Bani Said. Short-term prediction of traffic volume in urban arterials. *Journal of Transportation Engineering*, 121(3) :249–254, 1995.

[36] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning : data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2) :83–85, 2005.

[37] James Haworth. *Spatio-temporal forecasting of network data*. PhD thesis, UCL (University College London), 2014.

[38] James Haworth, John Shawe-Taylor, Tao Cheng, and Jiaqiu Wang. Local online kernel ridge regression for forecasting of urban travel times. *Transportation Research Part C : Emerging Technologies*, 46 :151–178, September 2014.

[39] Simon Haykin. *Neural networks : a comprehensive foundation*. Prentice Hall PTR, 1994.

[40] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8) :1735–1780, 1997.

[41] Wei-Chiang Hong. Traffic flow forecasting by seasonal SVR with chaotic simulated annealing algorithm. *Neurocomputing*, 74(12) :2096–2107, June 2011.

[42] Wei-Chiang Hong, Yucheng Dong, Feifeng Zheng, and Chien-Yuan Lai. Forecasting urban traffic flow by SVR with continuous ACO. *Applied Mathematical Modelling*, 35(3) :1282–1291, March 2011.

[43] Serge P Hoogendoorn and Piet HL Bovy. State-of-the-art of vehicular traffic flow modelling. *Proceedings of the Institution of Mechanical Engineers, Part I : Journal of Systems and Control Engineering*, 215(4) :283–303, 2001.

[44] R. Hyndman and A. Koehler. Another look at measures of forecast accuracy. *International journal of forecasting*, 22(4) :679–688, 2006.

[45] Rob J. Hyndman and George Athanasopoulos. *Forecasting : principles and practice*. OTexts, 2018.

[46] Rob J. Hyndman and Yeasmin Khandakar. Automatic Time Series Forecasting : The forecast Package for R. *Journal of Statistical Software*, 27(1) :1–22, July 2008.

[47] Anahita Jamshidnejad, Ioannis Papamichail, Markos Papageorgiou, and Bart De Schutter. A mesoscopic integrated urban traffic flow-emission model. *Transportation Research Part C : Emerging Technologies*, 75 :45–83, 2017.

[48] Finn V Jensen et al. *An introduction to Bayesian networks*, volume 210. UCL press London, 1996.

[49] T. Ji, Q. Pang, and X. Liu. Study of Traffic Flow Forecasting Based on Genetic Neural Network. volume 1, pages 960–965, October 2006.

[50] Rui Jiang, Qingsong Wu, and Zuojin Zhu. Full velocity difference model for a car-following theory. *Physical Review E*, 64(1) :017101, 2001.

[51] Y. Kamarianakis, W. Shen, and L. Wynter. Real-time road traffic forecasting using regime-switching space-time models and adaptive lasso. *Applied stochastic models in business and industry*, 28(4) :297–315, 2012.

[52] Yiannis Kamarianakis and Poulicos Prastacos. Forecasting traffic flow conditions in an urban network : Comparison of multivariate and univariate approaches. *Transportation Research Record*, 1857(1) :74–84, 2003.

[53] Yiannis Kamarianakis and Poulicos Prastacos. Space–time modeling of traffic flow. *Computers & Geosciences*, 31(2) :119–133, March 2005.

[54] Boris S Kerner, Sergey L Klenov, Gerhard Hermanns, and Michael Schreckenberg. Effect of driver over-acceleration on traffic breakdown in three-phase cellular automaton traffic flow models. *Physica A : Statistical Mechanics and its Applications*, 392(18) :4083–4105, 2013.

[55] Daphne Koller and Nir Friedman. *Probabilistic graphical models : principles and techniques*. MIT press, 2009.

[56] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical Review E*, 69(6) :066138, June 2004.

[57] Kranti Kumar, M. Parida, and V. K. Katiyar. Short Term Traffic Flow Prediction for a Non Urban Highway Using Artificial Neural Network. *Procedia - Social and Behavioral Sciences*, 104 :755–764, December 2013.

[58] Denis Kwiatkowski, Peter C. B. Phillips, Peter Schmidt, and Yongcheol Shin. Testing the null hypothesis of stationarity against the alternative of a unit root : How sure are we that economic time series have a unit root ? *Journal of Econometrics*, 54(1) :159–178, October 1992.

[59] Pierre-Antoine Laharotte. *Contributions à la prévision court-terme, multi-échelle et multi-variée, par apprentissage statistique du trafic routier*. PhD thesis, 2016.

[60] Chi-Sen Li and Mu-Chen Chen. Identifying important variables for predicting travel time of freeway with non-recurrent congestion with neural networks. *Neural Computing and Applications*, 23(6) :1611–1629, November 2013.

[61] Ming-Wei Li, Wei-Chiang Hong, and Hai-Gui Kang. Urban traffic flow forecasting using Gauss–SVR with cat mapping, cloud model and PSO hybrid algorithm. *Neurocomputing*, 99 :230–240, January 2013.

[62] Lighthill Michael James and Whitham Gerald Beresford. On kinematic waves II. A theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 229(1178) :317–345, May 1955.

[63] Shu Lin, Bart De Schutter, Yugeng Xi, and Hans Hellendoorn. Fast model predictive control for urban road networks via milp. *IEEE Transactions on Intelligent Transportation Systems*, 12(3) :846–856, 2011.

[64] Xingwei Liu, Xuming Fang, Zhenhua Qin, Chun Ye, and Miao Xie. A short-term forecasting algorithm for network traffic based on chaos theory and svm. *Journal of network and systems management*, 19(4) :427–447, 2011.

[65] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. Traffic flow prediction with big data : A deep learning approach. *IEEE Trans. Intelligent Transportation Systems*, 16(2) :865–873, 2015.

[66] Helmut Lütkepohl. *New Introduction to Multiple Time Series Analysis*. Springer-Verlag, Berlin Heidelberg, 2005.

[67] Xiaolei Ma, Zhimin Tao, Yinhai Wang, Haiyang Yu, and Yunpeng Wang. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C : Emerging Technologies*, 54 :187–197, 2015.

[68] Xiaolei Ma, Haiyang Yu, Yunpeng Wang, and Yinhai Wang. Large-Scale Transportation Network Congestion Evolution Prediction Using Deep Learning Theory. *PLOS ONE*, 10(3) :e0119044, March 2015.

[69] Tiep Mai, Bidisha Ghosh, and Simon Wilson. Multivariate short-term traffic flow forecasting using Bayesian vector autoregressive moving average model. Technical report, 2012.

[70] Stéphane Mallat. Cours au collège de france : Science des données. https ://www.college-de-france.fr/site/stephane-mallat/index.htm.

[71] X. Min, J. Hu, and Z. Zhang. Urban traffic network modeling and short-term traffic flow forecasting based on GSTARIMA model. In *13th International IEEE Conference on Intelligent Transportation Systems*, pages 1535–1540, September 2010.

[72] Xinyu Min, Jianming Hu, Qi Chen, Tongshuai Zhang, and Yi Zhang. Short-term traffic flow forecasting of urban network based on dynamic STARIMA model. In *Intelligent Transportation Systems, 2009. ITSC'09. 12th International IEEE Conference on*, pages 1–6. IEEE, 2009.

[73] Kai Nagel and Michael Schreckenberg. A cellular automaton model for freeway traffic. *Journal de physique I*, 2(12) :2221–2229, 1992.

[74] Cheol Oh and Seri Park. Investigating the effects of daily travel time patterns on short-term prediction. *KSCE Journal of Civil Engineering*, 15(7) :1263, September 2011.

[75] Bei Pan, Ugur Demiryurek, and Cyrus Shahabi. Utilizing real-world transportation data for accurate traffic prediction. In *2012 IEEE 12th International Conference on Data Mining*, pages 595–604. IEEE, 2012.

[76] Harold J Payne. Model of freeway traffic and control. *Mathematical Model of Public System*, pages 51–61, 1971.

[77] Phillip E. Pfeifer and Stuart Jay Deutsch. A STARIMA Model-Building Procedure with Application to Description and Regional Forecasting. *Transactions of the Institute of British Geographers*, 5(3) :330–349, 1980.

[78] Ilya Prigogine and Robert Herman. Kinetic theory of vehicular traffic. Technical report, 1971.

[79] Fengxiang Qiao, Xin Wang, and Lei Yu. Optimizing aggregation level for intelligent transportation system data based on wavelet decomposition. *Transportation Research Record*, 1840(1) :10–20, 2003.

[80] Hesham Rakha and Brent Crowther. Comparison of greenshields, pipes, and van aerde car-following and traffic stream models. *Transportation Research Record*, 1802(1) :248–262, 2002.

[81] Paul I. Richards. Shock Waves on the Highway. *Operations Research*, 4(1) :42–51, February 1956.

[82] Donald B Rubin. Inference and missing data. *Biometrika*, 63(3) :581–592, 1976.

[83] Jakob Runge, Reik V. Donner, and Jürgen Kurths. Optimal model-free prediction from multivariate time series. *Physical Review E*, 91(5) :052909, May 2015.

[84] Jakob Runge, Jobst Heitzig, Norbert Marwan, and Jürgen Kurths. Quantifying causal coupling strength : A lag-specific measure for multivariate time series related to transfer entropy. *Physical Review E*, 86(6) :061121, December 2012.

[85] Jakob Runge, Jobst Heitzig, Vladimir Petoukhov, and Jürgen Kurths. Escaping the Curse of Dimensionality in Estimating Multivariate Transfer Entropy. *Physical Review Letters*, 108(25) :258701, June 2012.

[86] Jakob Runge, Peer Nowack, Marlene Kretschmer, Seth Flaxman, and Dino Sejdinovic. Detecting causal associations in large nonlinear time series datasets. *arXiv :1702.07007 [physics, stat]*, February 2017. arXiv : 1702.07007.

[87] Jakob Runge, Vladimir Petoukhov, Jonathan F. Donges, Jaroslav Hlinka, Nikola Jajcay, Martin Vejmelka, David Hartman, Norbert Marwan, Milan Paluš, and Jürgen Kurths. Identifying causal gateways and mediators in complex spatio-temporal systems. *Nature Communications*, 6 :8502, October 2015.

[88] Said E. Said and David A. Dickey. Testing for Unit Roots in Autoregressive-Moving Average Models of Unknown Order. *Biometrika*, 71(3) :599–607, 1984.

[89] Helge Siegel and Dennis Belomestnyi. Stochasticity of road traffic dynamics : Comprehensive linear and nonlinear time series analysis on high resolution freeway traffic records. *arXiv preprint physics/0602136*, 2006.

[90] Brian L. Smith, Billy M. Williams, and R. Keith Oswald. Comparison of parametric and nonparametric models for traffic flow forecasting. *Transportation Research Part C : Emerging Technologies*, 10(4) :303–321, 2002.

[91] Smith Brian L. and Demetsky Michael J. Traffic Flow Forecasting : Comparison of Modeling Approaches. *Journal of Transportation Engineering*, 123(4) :261–266, July 1997.

[92] Peter Spirtes, Clark N Glymour, Richard Scheines, David Heckerman, Christopher Meek, Gregory Cooper, and Thomas Richardson. *Causation, prediction, and search*. MIT press, 2000.

[93] Anthony Stathopoulos and Matthew G. Karlaftis. A multivariate state space approach for urban traffic flow modeling and prediction. *Transportation Research Part C : Emerging Technologies*, 11(2) :121–135, April 2003.

[94] Antony Stathopoulos, Loukas Dimitriou, and Theodore Tsekeris. Fuzzy Modeling Approach for Combined Forecasting of Urban Traffic Flow.

Computer-Aided Civil and Infrastructure Engineering, 23(7) :521–535, 2008.

[95] Shiliang Sun and Qiaona Chen. Kernel Regression with a Mahalanobis Metric for Short-Term Traffic Flow Forecasting. In Colin Fyfe, Dongsup Kim, Soo-Young Lee, and Hujun Yin, editors, *Intelligent Data Engineering and Automated Learning – IDEAL 2008*, Lecture Notes in Computer Science, pages 9–16. Springer Berlin Heidelberg, 2008.

[96] Sun Shiliang, Huang Rongqing, and Gao Ya. Network-Scale Traffic Modeling and Forecasting with Graphical Lasso and Neural Networks. *Journal of Transportation Engineering*, 138(11) :1358–1367, November 2012.

[97] J. Tang, F. Liu, Y. Zou, W. Zhang, and Y. Wang. An Improved Fuzzy Neural Network for Traffic Speed Prediction Considering Periodic Characteristic. *IEEE Transactions on Intelligent Transportation Systems*, 18(9) :2340–2350, September 2017.

[98] Yongxue Tian and Li Pan. Predicting short-term traffic flow by long short-term memory recurrent neural network. In *2015 IEEE international conference on smart city/SocialCom/SustainCom (SmartCity)*, pages 153–158. IEEE, 2015.

[99] Lykourgos Tsirigotis, Eleni I. Vlahogianni, and Matthew G. Karlaftis. Does information on weather affect the performance of short-term traffic forecasting models ? *International Journal of Intelligent Transportation Systems Research*, 10(1) :1–10, 2012.

[100] J. W. C. van Lint, S. P. Hoogendoorn, and H. J. van Zuylen. Accurate freeway travel time prediction with state-space neural networks under missing data. *Transportation Research Part C : Emerging Technologies*, 13(5) :347–369, October 2005.

[101] L. Vanajakshi and L. R. Rilett. A comparison of the performance of artificial neural networks and support vector machines for the prediction of traffic speed. In *IEEE Intelligent Vehicles Symposium, 2004*, pages 194–199, June 2004.

[102] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Information Science and Statistics. Springer-Verlag, New York, 2 edition, 2000.

[103] Eleni Vlahogianni and Matthew Karlaftis. Temporal aggregation in traffic data : implications for statistical characteristics and model choice. *Transportation Letters*, 3(1) :37–49, 2011.

[104] Eleni I. Vlahogianni, John C. Golias, and Matthew G. Karlaftis. Short-term traffic forecasting : Overview of objectives and methods. *Transport Reviews*, 24(5) :533–557, September 2004.

[105] Eleni I. Vlahogianni, Matthew G. Karlaftis, and John C. Golias. Optimized and meta-optimized neural networks for short-term traffic flow prediction : A genetic approach. *Transportation Research Part C : Emerging Technologies*, 13(3) :211–234, June 2005.

[106] Eleni I Vlahogianni, Matthew G Karlaftis, and John C Golias. Statistical methods for detecting nonlinearity and non-stationarity in univariate short-term time-series of traffic volume. *Transportation Research Part C : Emerging Technologies*, 14(5) :351–367, 2006.

[107] Eleni I. Vlahogianni, Matthew G. Karlaftis, and John C. Golias. Spatio-Temporal Short-Term Urban Traffic Volume Forecasting Using Genetically Optimized Modular Networks. *Computer-Aided Civil and Infrastructure Engineering*, 22(5) :317–325, 2007.

[108] Eleni I. Vlahogianni, Matthew G. Karlaftis, and John C. Golias. Short-term traffic forecasting : Where we are and where we're going. *Transportation Research Part C : Emerging Technologies*, 43 :3–19, June 2014.

[109] Mark W Watson. Vector autoregressions and cointegration. *Handbook of econometrics*, 4 :2843–2915, 1994.

[110] G. C. K Wong and S. C Wong. A multi-class traffic flow model – an extension of LWR model with heterogeneous drivers. *Transportation Research Part A : Policy and Practice*, 36(9) :827–841, November 2002.

[111] C.-H. Wu, J.-M. Ho, and D.T. Lee. Travel-Time Prediction With Support Vector Regression. *IEEE Transactions on Intelligent Transportation Systems*, 5(4) :276–281, December 2004.

[112] Cheng-Ju Wu, Thomas Schreiter, and Roberto Horowitz. Multiple-clustering ARMAX-based predictor and its application to freeway traffic flow prediction. In *2014 American Control Conference*, pages 4397–4403, Portland, OR, USA, June 2014. IEEE.

[113] Shanhua Wu, Zhongzhen Yang, Xiaocong Zhu, and Bin Yu. Improved k-nn for Short-Term Traffic Forecasting Using Temporal and Spatial Information. *Journal of Transportation Engineering*, 140(7) :04014026, July 2014.

[114] Yuankai Wu, Huachun Tan, Lingqiao Qin, Bin Ran, and Zhuxi Jiang. A hybrid deep learning based traffic flow prediction method and its understanding. *Transportation Research Part C : Emerging Technologies*, 90 :166–180, May 2018.

[115] Dawen Xia, Huaqing Li, Binfeng Wang, Yantao Li, and Zili Zhang. A MapReduce-based nearest neighbor approach for big-data-driven traffic flow prediction. *IEEE access*, 4 :2920–2934, January 2016.

[116] Dawen Xia, Binfeng Wang, Huaqing Li, Yantao Li, and Zili Zhang. A distributed spatial-temporal weighted model on MapReduce for short-term traffic flow forecasting. *Neurocomputing*, 179 :246–226, February 2016.

[117] Chenyun Yu and Ka Chi Lam. Applying multiple kernel learning and support vector machine for solving the multicriteria and nonlinearity problems of traffic flow prediction : MKL AND SVM-BASED MODEL FOR TRAFFIC FLOW PREDICTION. *Journal of Advanced Transportation*, 48(3) :250–271, April 2014.

[118] Ning Zhang, Yunlong Zhang, and Haiting Lu. Seasonal autoregressive integrated moving average and support vector machine models : prediction of short-term traffic flow on freeways. *Transportation Research Record : Journal of the Transportation Research Board*, (2215) :85–92, 2011.

[119] Yunlong Zhang and Yuanchang Xie. Forecasting of Short-Term Freeway Volume with v-Support Vector Machines. *Transportation Research Record*, 2024(1) :92–99, January 2007.

[120] Zheng Zhao, Weihai Chen, Xingming Wu, Peter CY Chen, and Jingmeng Liu. Lstm network : a deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems*, 11(2) :68–75, 2017.

[121] Zuduo Zheng and Dongcai Su. Short-term traffic volume forecasting : A k-nearest neighbor approach enhanced by constrained linearly sewing principle component algorithm. *Transportation Research Part C : Emerging Technologies*, 43 :143–157, June 2014.

[122] Ming Zhong, Satish Sharma, and Pawan Lingras. Short-Term Traffic Prediction on Different Types of Roads with Genetically Designed Regression and Time Delay Neural Network Models. *Journal of Computing in Civil Engineering*, 19(1) :94–103, January 2005.

FOLIO ADMINISTRATIF

THESE DE L'UNIVERSITE DE LYON OPEREE AU SEIN DE L'INSA LYON

NOM : SALOTTI

DATE de SOUTENANCE : 24/09/2019

Prénoms : Julien, Simon, Jean

TITRE : Méthodes de sélection de voisinage et de prévision à court-terme pour l'analyse du trafic urbain

NATURE : Doctorat

Numéro d'ordre : 2019LYSEI077

Ecole doctorale : InfoMaths

Spécialité : Informatique

RESUME : Dans le contexte de la ville intelligente, le besoin d'informer, d'anticiper, et d'agir sur l'état du réseau routier est à l'origine du développement de nombreuses méthodes de prévision de trafic. L'augmentation de nos capacités à stocker et à traiter des données, notamment en temps réel, ainsi que le nombre croissant de segments de routes équipés de capteurs sont de nouveaux éléments à considérer lors du choix d'une méthode de prévision. Malgré de nombreux travaux de recherche, nous ne disposons toujours pas d'une compréhension claire des critères permettant de prédire efficacement à l'échelle d'un réseau routier.
Dans cette thèse, nous nous appuyons sur deux jeux de données réelles, collectés respectivement sur le réseau urbain de la Métropole de Lyon, et sur les autoroutes urbaines de Marseille. Nous étudions la performance de différentes méthodes issues de la littérature statistiques des séries temporelles (méthodes autorégressives) et de la littérature de l'apprentissage artificiel (machine à vecteurs de support, réseaux de neurones). Nous étudions également l'apport de différentes stratégies de sélection de voisinage (sélection d'un sous-ensemble de capteurs utiles pour la prévision d'un capteur en particulier) pour améliorer la qualité de la prévision, tout en diminuant la complexité des modèles appris. Nous comparons ainsi une approche classique (la sélection Lasso) et testons pour la première fois sur des données de trafic une méthode issue de la théorie de l'information, ayant de très bons résultats sur des problèmes similaires de physique (tigramite). Nos résultats expérimentaux confirment l'utilité de mécanismes de sélection de voisinage et illustrent la complémentarité des approches de prévisions, selon le type de réseau (urbain, autoroute) et l'horizon de prévision (de 6 à 30 minutes).

MOTS-CLÉS : Prévision de trafic ; apprentissage supervisé ; régression ; sélection de variables ; knn ; lasso ; SVR ; réseaux de neurones ; séries temporelles ; modèles graphiques probabilistes

Laboratoire (s) de recherche : LIRIS ; LICIT

Directeurs de thèse: Christine Solnon (LIRIS), Nour-Eddin El Faouzi (LICIT)

Président de jury :

Composition du jury : Vincent Aguiléra, Azedine Boulmakoul, Nour-Eddin El Faouzi, Serge Fenet, Elisa Fromont, Christine Solnon, Peter Sturm