Classification spatiale avec régresseurs : Exemple de la prédiction du prix du marché immobilier de Sacramento

Sur une proposition de M. Cornec

Victor CLUZEL
Édith DARIN
Noemie HAOUZI

Séminaire de Modélisation Statistique
M. Alquier

ENSAE ParisTech
Juin 2016

1 Introduction

Résumé de l'étude Le but de ce présent travail est de discuter l'article de Hallac, leskovec et Boyd [4] qui traite d'une méthode de regroupement ("clustering") et d'optimisation pour des graphes de grande taille : le Network Lasso. Ils s'inscrivent dans le cadre de l'optimisation convexe, problématique récurrente en analyse de données et plus particulièrement en apprentissage statistique ("machine learnging") et en exploration de données ("data mining").

L'idée principale est d'allier une modélisation globale à l'échelle de l'ensemble des données avec une précision de prédiction à l'échelle de subdivision en groupe de l'échantillon, autrement dit d'avoir une vision globale adaptable à un nombre de regroupement défini.

Par ailleurs, la grande particularité de leur nouveau outil, le Network Lasso (ou Lasso appliqué à des données de type réseau) est sa grande polyvalence, qui rend son utilisation pertinente, de la modélisation de situation réelle (comme les système de contrôle) à la construction de modèle statistique.

Motivation de l'étude Pour appréhender au mieux la force de cette approche, nous avons choisi de reposer notre exposé sur un exemple précis -la prédiction du prix sur le marché immobilier de Sacramento- afin d'éclairer au mieux les nombreux apports de cette méthode.

D'un point de méthodologique afin d'éviter la paraphrase de l'article, nous reprenons cet exemple déjà mentionné par les auteurs pour l'étoffer et expliquer en quoi le Network Lasso est une rupture d'avec les approche traditionnelle d'estimation des déterminants du prix.

Ainsi nous cherchons à apporter d'un côté une plus grande inscription de cet article en expliquant la faillite des modèles cherchant à représenter le prix du marché immobilier et de l'autre nous apporterons dans la partie théorique, une vision plus globale de l'univers dans lequel s'inscrit le Network Lasso.

2 Le marché immobilier de Sacramento

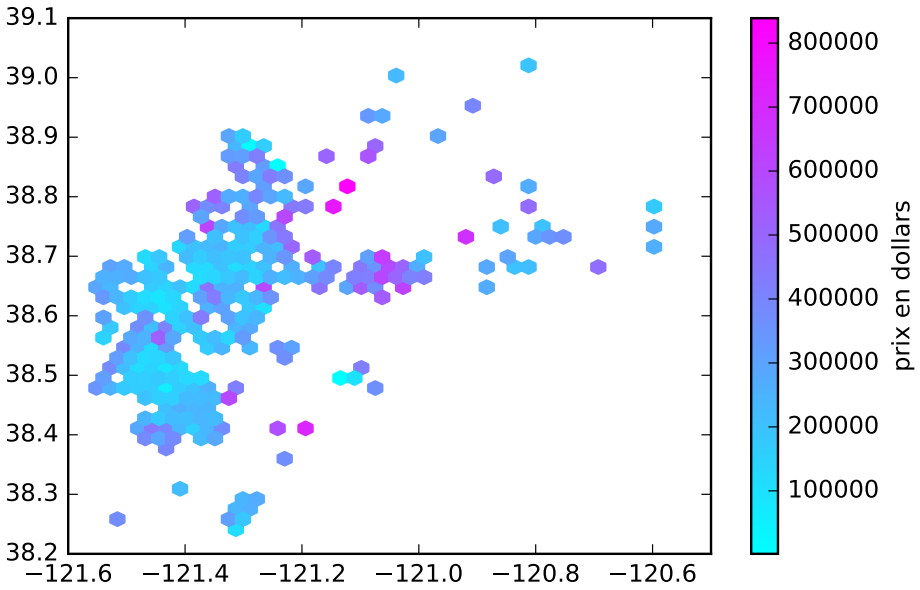
Un objet traditionnel Il existe une vaste littérature ([1], [3], pour n'en citer que deux) sur l'étude du marché immobilier qui s'inscrit à la fois dans une visée explicative mais aussi prédictive : à partir des caractéristiques d'une maison, peut-on savoir son prix ?

Il s'agit de saisir les déterminants structurelles de l'évolution du prix de l'immobiler, principalement par des méthodes de régressions linéaires de type moindre carrés ordinaires. Cependant il est nécessaire d'ajouter à ces modèles une dimension spatiale. En effet la localisation est primordiale dans la valeur qu'acquiert un bien immobilier. Avec l'apparition des systèmes d'informations géographiques et des données GPS, on dispose facilement de grandes bases de données contenant prix et coordonnées des maisons concernées. Le souci se situe à présent au niveau de la modélisation, comme le souligne la compétition rapportée dans cet article de Case, Clapp, Dubin et Rodriguez [2], qui compare différents modèle intégrant

cette caractéristique spatiale, soit par des variables instrumentales, soit par des indicatrices incluant la position géographique par quartier.

Un arbitrage entre un modèle général et une précision locale Nous disposons de données concernant 985 ventes ayant eu lieu dans la zone de Sacramento pendant cinq jours en mai 2008. La base contient comme variables pour chaque maison sa longitude, sa latitude, son nombre de chambres, de salles de bain, sa superficie et son prix de vente, variable d'intérêt. Or nous remarquons grâce à la carte de la Figure 1, que la localisation a une très forte importance dans la détermination des prix. Nous voulons donc aboutir à la construction de regroupements spatiaux, qui de par leur proximité, partageraient ainsi une même fixation du prix.

FIGURE 1 – Heatmap des prix du marché de l'Immobilier à Sacramento sur l'échantillon étudié



En abscisse la longitude et en ordonnée la latitude.

L'idée est donc de pouvoir adapter un modèle explicatif global à chaque zone résidentielle. En effet si l'on ne dispose que d'une représentation à l'échelle globale, on risque d'avoir une forte erreur de prédiction car il s'agira d'une moyenne entre les différentes zones d'habitation, tandis que si on décide d'avoir une régression par région on perd de l'information issue des autres régions et c'est peu robuste car dépendant de la zone géographique. On tombe dans un problème d'overfitting. Enfin cette analyse pêche par la nécessité de définir a priori les quartiers à partir de critères qu'il est difficile de quantifier.

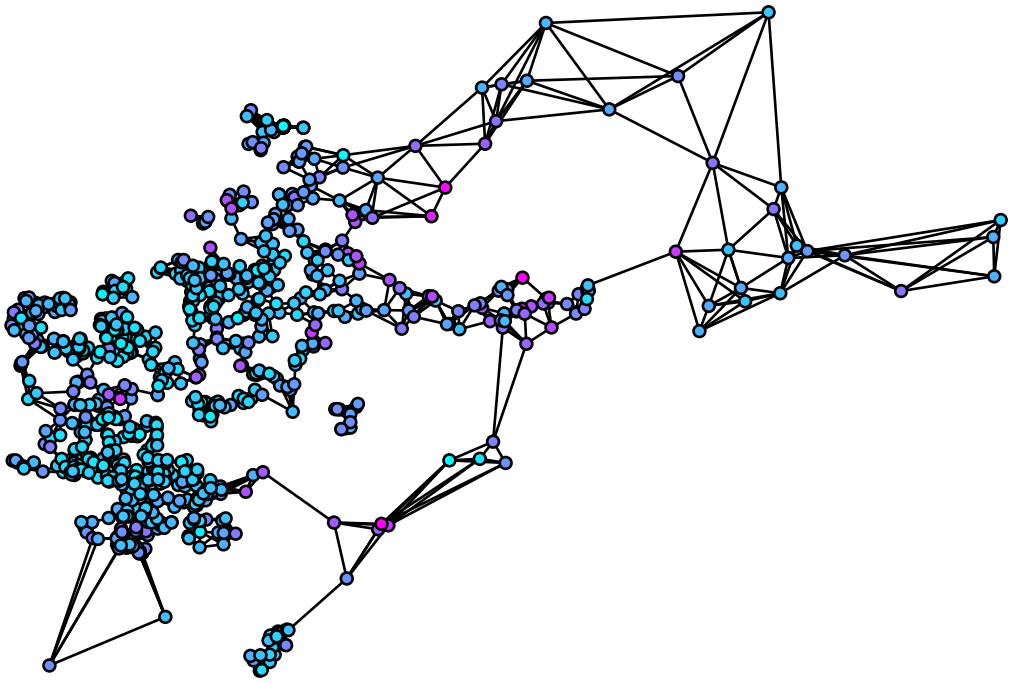
Enfin, à l'inverse, faire une régression par maison fait perdre de l'information issue des programmes d'optimisation des maisons voisines, qui partagent en définitive le même modèle sous-jacent.

3 La modélisation : adaptation du Network Lasso

Modélisation avec graphes

Afin d'appréhender la structure spatiale de notre problème, l'une des solutions est de représenter le tissu résidentiel de Sacramento sous la forme d'un graphe non orienté, où les maisons sont symbolisées par les sommets et reliées entre elles par les arêtes. Plus précisément il s'agit d'un modèle graphique, avec $\mathcal{V}$ l'ensemble des maisons et $\mathcal{E}$ l'ensemble des arêtes. Nous avons 985 maisons, donc $\dim(\mathcal{V}) = 985$. Nous construisons le graphique en reliant les 5 plus proches voisins de chaque maison : si la maison j fait partie des cinq plus proches voisins de i, on trace une arête peu importe que i appartienne elle-aussi aux cinq plus proches voisins de j. On obtient ainsi $\dim(\mathcal{E}) = 3023$.

FIGURE 2 – Représentation du réseau construit à partir des données de l'article



Les couleurs représentées sont les mêmes que celles utilisées pour représenter les prix sur la Figure 1

Pour construire ce réseau, nous avons tout d'abord importé les données de l'article dans Python, puis nous avons construit la matrice des distances euclidiennes entre les différentes maisons (ou nœuds) grâce aux latitudes et longitudes. Ensuite nous avons stocké dans une autre matrice les 5 plus proches voisins de chaque nœud à l'aide de la matrice des distances et d'un algorithme disponible en Annexe II. En utilisant le module *NetworkX* de Python, le graphe est ensuite construit à l'aide de ses nœuds et de ses arêtes. Pour ce qui est de la représentation graphique de la Figure 2 obtenue à la page précédente, il a fallu construire un dictionnaire des positions de chaque nœud pour qu'il possède des coordonnées spatiales. En Annexe I est disponible la même représentation avec les coordonnées et la répartition spatiale, et en Annexe II le code source en Python 2.7.

Notons que la méthode décrite par Hallac, leskovec et Boyd s'applique uniquement à des problèmes d'optimisation convexe préalablement représentés sous forme de réseau.

Introduction d'une contrainte et famille des régressions Lasso : les origines du *Network Lasso*

Nous cherchons donc à résoudre un problème d'optimisation simultanément à un problème de classification : trouver et une partition géographique pertinente du prix des maisons et une régression linéaire expliquant les déterminants de ces prix.

Lasso L'idée est d'utiliser l'optimisation sous contrainte initiée par Tibshirani [8], sous le nom de régression Lasso. En effet en introduisant une contrainte sur le nombre de paramètres, cette optimisation permet de sélectionner un nombre limité de variables. On obtient donc sous sa représentation lagrangienne, le programme suivant, avec y_i la variable d'intérêt pour l'observation i, X_i le vecteur des variables explicatives et f une fonction objectif convexe [1].

$$\hat{\beta} = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} f_{\beta}(X_i, y_i) + \lambda ||\beta_i||_1$$

Group Lasso A travers la modélisation selon un *Group Lasso*, cette approche a été complexifiée par Yuan et Lin [10], en introduisant la notion de groupe de variables qui permet de considérer la sélection de variables par groupe. Ainsi, soit $G = G_1, \cdots, G_K$, une partition des n variables en K groupes. On note β_G , pour $G \in \mathcal{G}$, le vecteur β restreint aux éléments du groupe G . L'objectif du Group-Lasso est :

$$\hat{\beta} = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} f_{\beta}(X_i, y_i) + \lambda \sum_{j=1}^{K} w_j \|\beta_{G_j}\|_2$$

avec $w_j > 0$, un poids associé au groupe G_j.

1. Tibshirani a premièrement écrit ce modèle à partir de la fonction linéaire : $||y - X\beta||_2^2$

Fused Lasso Le *Fused Lasso* permet, quant à lui, d'intégrer une dimension spatiale, le principe étant que les variables "proches" aient des coefficients estimés "proches". Cela est possible en pénalisant la norme ℓ_1 de la différence de deux coefficients successifs. De la même manière que pénaliser la norme ℓ_1 d'un coefficient a tendance à produire des coefficients égaux à 0, pénaliser la différence va favoriser l'égalité de deux coefficients successifs. L'objectif du Fused-Lasso est alors :

$$\hat{\beta} = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} f_\beta(X_i, y_i) + \lambda_1 ||\beta||_1 + \lambda_2 \sum_{j=2}^{p} ||\beta_j - \beta_{j-1}||_1$$

avec $\lambda_1 \geq 0$ et $\lambda_2 \geq 0$.

Convex Clustering Enfin on peut citer aussi la veine des travaux des méthodes de regroupement fondées sur des optimisations convexes sous contrainte (*Convex Clustering* : [6] [7] ou [5]), qui cherchent à résoudre un problème de classification non supervisée, c'est-à-dire à résoudre un problème d'optimisation convexe sans poser au préalable le nombre de partition recherchée. Ils utilisent ainsi également un terme de pénalisation, la somme des normes qui contrôle la pariton. En effet, l'idée est de minimiser la distance entre des centres, $\mu_1, \cdots, \mu_n$, pivots des futures partitions et les observations tout en y ajoutant un terme dit de régularisation qui pénalise le nombre total de partition. Autrement dit, voici le programme d'optimisation :

$$\arg\min_{\mu_1, \cdots, \mu_n} \sum_{j=1}^{n} ||x_j - \mu_j||_2^2 + \lambda \sum_{j=2}^{n} \sum_{i<j} ||\mu_i - \mu_j||_p$$

Avec l'introduction des coordonnées spatiales, on remarque qu'on s'approche de notre objectif de regroupement. Cependant cette méthode nécessit que la fonction-cible soit une certaine distance au carré, réduisant la portée générale de la modélisation.

Formalisation

Forme générale du *Network Lasso* On peut à présent comprendre au mieux l'apport à la littérature de l'article de Hallac, leskovec et Boyd. En effet réunissant la recherche sur l'optimisation sous contrainte, de type Lasso, les méthodes de partitionnement et l'analyse de réseaux et de graphes, ces auteurs proposent une méthode générale permettant de résoudre rapidement et de façon adaptée à la fois à l'échelle globale et à l'échelle des partitionnement, une optimisation convexe sur graphe. Plus précisément, en reprenant les notation du début concernant les graphes :

$$\arg\min \quad \underbrace{\sum_{i \in \mathcal{V}} f_i(x_i)}_{Objectif\ local} + \quad \underbrace{\lambda \sum_{(j,k) \in \mathcal{E}} w_{jk} ||x_j - x_k||_2}_{Objectif\ global} \tag{1}$$

avec $x_1, \ldots, x_p$, les variables explicatives, où p est le nombre de variables explicatives, $x_i \in \mathbb{R}^p$, les variables pour le sommet i, f_i une fonction de perte convexe. Il faut donc bien saisir que la première partie concerne le modèle pour le sommet/individu i, donc une vision à échelle individuelle, tandis que la seconde partie pénalise une trop grande différence entre les variables pour des sommets adjacents. λ quant à lui est un paramètre qui définit l'importance relative de l'objectif local vis-à-vis du modèle général.

On retrouve ainsi le terme de pénalisation d'un *Group Lasso* exprimé en fonction d'une norme de la différence et non de cette norme au carré ce qui oblige que x_j et x_k soient égaux et non juste proches. [2]

Application à notre étude de cas Le paramètre d'optimisation est pour le marché immobilier à Sacramento de la forme :

$$x_i = [a_i \quad b_i \quad c_i \quad d_i]^T$$

Et ainsi la prédiction du prix de chaque nœud est donnée par :

$$\overline{price}_i = a_i.Bedrooms + b_i.Bathrooms + c_i.SQFT + d_i$$

avec d_i un offset représentant le "prix de base". Enfin la fonction d'objectif à chaque nœud est :

$$f_i(x_i) = ||\overline{price}_i - price_i||_2^2 + \mu||\tilde{x}_i||_2^2, \quad \tilde{x}_i = [a_i \quad b_i \quad c_i]^T$$

Le programme d'optimisation devient donc :

$$\arg\min \quad \sum_{i \in \mathcal{V}} ||\overline{price}_i - price_i||_2^2 + \mu||\tilde{x}_i||_2^2 \quad + \quad \lambda \sum_{(j,k) \in \mathcal{E}} w_{jk}||x_j - x_k||_2 \tag{2}$$

avec w_{jk}, l'inverse de la distance entre les deux maisons.

Ainsi chaque maison résoud son propre problème d'optimisation représenté par f_i, fondé sur ses caractéristiques x_i et son prix $price_i$. Le second terme quant à lui force les maisons proches à àvoir le même modèle de régression et donc les mêmes paramètres. L'idée sous-jacente du *Network Lasso* est de déterminer pour chaque maison à quel voisinage elle appartient et d'apprendre de l'information à partir de ce voisinage afin d'améliorer sa propre optimisation. On obtient ainsi un partitionnement du marché immobilier, où chaque regroupement a son propre modèle de régression. On peut donc à partir de là inférer le prix d'une nouvelle maison en fonction de ses coordonnées spatiales, puisqu'on peut ainsi déterminer à quelle voisinage elle appartient.

2. En effet la norme au carré de la différence renvoie à une régularisation de type Laplace qui autorise de petites variations, lissant les différences ce qui réduit le pouvoir de classification du modèle.

4 Implémentation

Présentation de l'algorithme ADMM La méthode utilisée pour résoudre le problème numériquement est celle de l'"Alternating Direction Method of Multipliers" (ADMM). Cette méthode consiste en première approche à faire résoudre à chaque nœud sa propre fonction objectif et à la transmettre à ses voisins, et ainsi de suite jusqu'à ce que la totalité du réseau converge.

Formellement, est introduite une nouvelle variable z_{ij} qui est une copie de x_i au noeud ij. Le problème (2) devient alors :

$$\begin{aligned} \arg\min \quad & \sum_{i \in \mathcal{V}} f_i(x_i) + \quad \lambda \sum_{(j,k) \in \mathcal{E}} w_{jk} ||z_{jk} - z_{kj}||_2 \\ & subject \quad to \quad x_i = z_{ij}, \quad i = 1, ..., m, \quad j \in N(i) \end{aligned} \tag{3}$$

Ce qui amène à utiliser la Lagrangien augmenté, qui consiste en :

$$\begin{aligned} L_\rho(x, z, u) = & \sum_{i \in \mathcal{V}} f_i(x_i) \\ + & \sum_{(j,k) \in \mathcal{E}} \left(\lambda w_{jk} ||z_{jk} - z_{kj}||_2 - (\rho/2) \left(||u_{jk}||_2^2 + ||u_{kj}||_2^2 \right) + (\rho/2) \left(||x_j - z_{jk} + u_{jk}||_2^2 + ||x_k - z_{kj} + u_{kj}||_2^2 \right) \right. \end{aligned}$$

Où u est la variable duale normée, c'est-à-dire $u = (1/\rho)y$ où y est la variable duale pour $x = z$ ([9]). ρ est le paramètre de pénalité. A l'itération k, l'algorithme ADMM consiste à effectuer :

$$x^{k+1} = \arg\min_x L_\rho(x, z^k, u^k)$$

$$z^{k+1} = \arg\min_z L_\rho(x^{k+1}, z, u^k)$$

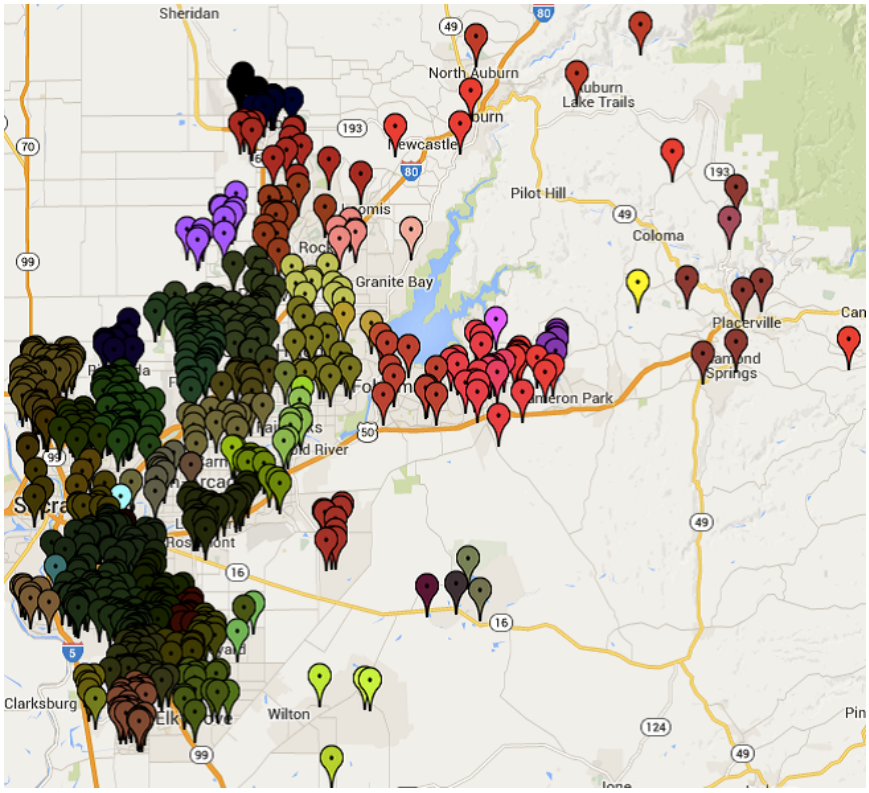
$$u^{k+1} = u^k + \left(x^{k+1} - z^{k+1} \right)$$

Il existe également une méthode dite "non convexe" qui en utilisant une fonction concave de la norme de $x_j - x_k$ qui permet d'empêcher que les différents clusters ne se "tirent" trop les uns vers les autres à travers leurs arêtes. En effet, la méthode présentée plus haut, dite "convexe" introduit une pénalité simplement proportionnelle à cette norme, et qui ne pénalise pas assez les différences juste supérieures à zéro.

Résultats Les résultats obtenus par l'algorithme ADMM dans le cadre de l'article étudié montrent que suivant la valeur du paramètre λ, la MSE obtenue varie grandement. En effet, quand $\lambda = 0$, alors le prix estimé est la médiane pondérée des prix des 5 voisins les plus proches, conduisant à une MSE de 0,6013 sur le test. Pour λ grand, c'est-à-dire supérieur à un $\lambda_{critique}$, on aboutit à un modèle commun pour toutes les maisons, cela a pour conséquence

que la MSE soit grande, à 0,8611. Les méthodes convexes et non convexes sont maximisées pour un λ environ égal à 5 et donnent des MSE respectivement de 0,4630 et 0,4539.
L'étude de l'évolution du problème en fonction de λ porte le nom de sentier de classification. En effet, comme nous venons de le voir, le comportement de les prédicteurs et des clusters constitués sera grandement différent en fonction des valeurs de λ. Les clusters optimaux sont trouvés pour des λ proches de 10, tout en gardant une MSE correctement faibles.

FIGURE 3 – Représentation des clusters pour $\lambda = 10$



Source : résultats de l'article étudié

Sont observées des anomalies, par exemple la maison jaune sur la partie droite de la carte (Figure 3). Elle est peu affectée par les modèles de ses voisins. Si l'on compare avec la Figure 2, on observera une maison assez éloignée de des voisins et d'un prix nettement plus élevé (très proche du violet) que ces dernières. Ainsi le *Network Lasso* permet également d'identifier les "outliers".

5 Portée et limites du *Network Lasso*

Network Lasso et module Python Dans le cadre du laboratoire de Stanford sur les analyses de réseaux (SNAP), les auteurs ont construit un module Python afin de pouvoir implémenter rapidement cette méthode à partir d'un graphe préalable. Il s'agit de snapVX. ce module est fondé sur l'installation préexistente de Snap.

Le code source utilise également le module CVXPY. Cependant, nous avons essayé à maintes reprises d'installer ce module en désinstallant et réinstallant tout Python (que ce soit sous forme pip, conda ou python(x,y)), cela ne fonctionnait jamais car cela retournait une erreur concernant Visual Basic (que nous avons également réinstallé). Nous avons donc choisi de nous concentrer sur l'implémentation de la représentation graphique du réseau en utilisant le module NetworkX.

Pouvoir explicatif du modèle Le *Network Lasso* propose une méthode rapide et adaptée à l'analyse en réseau permettant d'aboutir en parallèle à un partitionnement de l'espace des observations et à une application à chacune des regroupements d'un modèle spécifique. Nous n'avons pas spécialement insisté dessus au cours de notre exposé, mais, contrairement aux méthodes traditionnelles d'optimisation convexe (*Interior Point Methods*), la représentation sous forme de réseaux et l'application du Network lasso dans un second temps, permet de résoudre des optimisations sur de grands échantillons, avec une vitesse de résolution linéaire face à la taille du problème et non cubique. Ainsi il s'agit d'une méthode d'avenir pour les problèmes de large dimension qu'on rencontre actuellement en "machine learning". Par ailleurs, bien que notre approche ne permette pas de le mettre en lumière puisque nous nous sommes focalisés sur une unique application, la particularité de cette optimisation est sa grande portée : elle peut être utilisée dans de nombreux cas, à partir du moment où on peut envisager une représentation sous forme de graphe du problème.

Références

[1] Ayse Can. Specification and estimation of hedonic housing price models. *Regional science and urban economics*, 22(3) :453–474, 1992.

[2] Bradford Case, John Clapp, Robin Dubin, and Mauricio Rodriguez. Modeling spatial and temporal house price patterns : A comparison of four models. *The Journal of Real Estate Finance and Economics*, 29(2) :167–191, 2004.

[3] Allen C Goodman. Hedonic prices, price indices and housing markets. *Journal of Urban Economics*, 5(4) :471–484, 1978.

[4] David Hallac, Jure Leskovec, and Stephen Boyd. Network lasso : Clustering and optimization in large graphs. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 387–396. ACM, 2015.

[5] Toby Dylan Hocking, Armand Joulin, Francis Bach, and Jean-Philippe Vert. Clusterpath an algorithm for clustering using convex fusion penalties. In *28th international conference on machine learning*, page 1, 2011.

[6] Fredrik Lindsten, Henrik Ohlsson, and Lennart Ljung. Clustering using sum-of-norms regularization : With application to particle filter output computation. In *Statistical Signal Processing Workshop (SSP), 2011 IEEE*, pages 201–204. IEEE, 2011.

[7] Kristiaan Pelckmans, Joseph De Brabanter, JAK Suykens, and B De Moor. Convex clustering shrinkage. In *PASCAL Workshop on Statistics and Optimization of Clustering Workshop*, 2005.

[8] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society : Series B (Statistical Methodology)*, 67(1) :91–108, 2005.

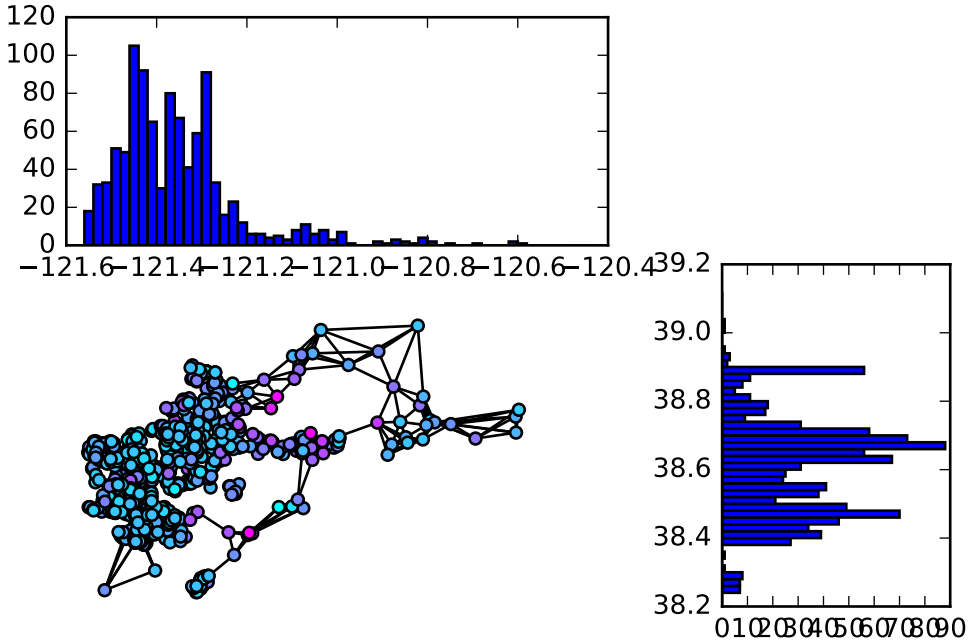
[9] Bo Wahlberg, Stephen Boyd, Mariette Annergren, and Yang Wang. An admm algorithm for a class of total variation regularized estimation problems. *arXiv preprint arXiv :1203.1828*, 2012.

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

6 Annexes

6.1 Annexe I - Représentation avec les coordonnées et la répartition spatiale

FIGURE 4 – Représentation du réseau et de sa répartition spatiale construite à partir des données de l'article



Les couleurs représentées sont les mêmes que celles utilisées pour représenter les prix sur la Figure 1. En abscisse la longitude et en ordonnée la latitude.

6.2 Annexe II - Code Python

```
$%$pylab inline
from networkx import *
import csv
import matplotlib.pyplot as plt
from pylab import *
from math import *
from numpy import*

#importation des données
lat = []
lon = []
price =[]
i=0
with open('C:/Users/Victor/Desktop/SemStat/Sacramentorealestatetransactions.csv', 'rU')
    reader = csv.reader(Coods, delimiter=',', dialect=csv.excel_tab)
    for row in reader:
        i=i+1
        if i>1:
            lat.append(float(row[10]))
            lon.append(float(row[11]))
            price.append(row[9])

unif = []
for row in price:
    unif.append(int(row))
unif

#représentation de la Heatmap
gridsize=40
plt.hexbin(lon, lat, C = unif, cmap=plt.cm.cool, gridsize=gridsize, bins = None)
plt.axis([-121.6,-120.5,38.2,39.1])
cb = plt.colorbar()
cb.set_label('prix en dollars')

plt.savefig('C:/Users/Victor/Desktop/SemStat/Heat.pdf')
plt.show()
```

```
#matrice des distances
mat = zeros([985,985])
for i in range(0,985):
        for j in range(0,985):
                mat[i,j]= sqrt((lon[i]-lon[j])**2+(lat[i]-lat[j])**2)

mat

#TEST de symétrie

s=0
for i in range(0,985):
        for j in range(0,985):
            if mat[i,j]!=mat[j,i]:
                s=s+1
s

#stockage des 5 plus proches voisins
#NOTA : l'algorithme ne fonctionnait pas en itérant sur j le ième plus proche vosin de i
#il a été choisi de faire cette itération "à la main"

matgen = lambda n, m: [[1000 for j in range(0,m)] for i in range(0,n)]
neigh = asmatrix(matgen(985,5))

for i in range(0,985):

    dist = 10000
    j = 0
    for k in range(0,985):
        if k!=i:
            if mat[i,k]<dist:
                neigh[i,j]=k
                dist=mat[i,k]

    j = 1
    dist = 10000
    for k in range(0,985):
        if (k!=i and k!=neigh[i,j-1]):
            if mat[i,k]<dist:
```

```
                neigh[i,j]=k
                dist=mat[i,k]
    j = 2
    dist = 10000
    for k in range(0,985):
        if (k!=i and k!=neigh[i,j-1] and k!=neigh[i,j-2]):
            if mat[i,k]<dist:
                neigh[i,j]=k
                dist=mat[i,k]

    j = 3
    dist = 10000
    for k in range(0,985):
        if (k!=i and k!=neigh[i,j-1] and k!=neigh[i,j-2] and k!=neigh[i,j-3]):
            if mat[i,k]<dist:
                neigh[i,j]=k
                dist=mat[i,k]

    j = 4
    dist = 10000
    for k in range(0,985):
        if (k!=i and k!=neigh[i,j-1] and k!=neigh[i,j-2] and k!=neigh[i,j-3] and k!=neig
            if mat[i,k]<dist:
                neigh[i,j]=k
                dist=mat[i,k]

neigh

#construction du graphe
G=nx.Graph()
G.add_nodes_from(range(0,985))

for i in range(0,985):
    for j in range(0,5):
        G.add_edge(i,neigh[i,j])

G.number_of_nodes()
G.number_of_edges()
```

```
#construction d'un dictionnaire des positions des nœuds
positions = dict.fromkeys(range(0, 984), 0)
for i in range(0,985):
    positions[i] = (lon[i],lat[i])
positions

#représentation du graphe
nx.draw(G, pos =positions, node_size = 20, node_color = price, cmap=plt.cm.cool, alpha=
plt.axis([-121.6,-120.5,38.2,39.1],'on')
plt.savefig('C:/Users/Victor/Desktop/SemStat/Net.pdf', bbox_inches='tight')
plt.show()

from mpl_toolkits.axes_grid1 import make_axes_locatable
axScatter = subplot(111)
nx.draw(G, pos =positions, node_size = 20, node_color = price, cmap=plt.cm.cool, alpha=
plt.axis([-121.6,-120.5,38.2,39.1],'on')
axScatter.axis([-121.6,-120.5,38.2,39.1],'on')


# create new axes on the right and on the top of the current axes.
divider = make_axes_locatable(axScatter)
axHistx = divider.append_axes("top", size=1.2, pad=0.1, sharex=axScatter)
axHisty = divider.append_axes("right", size=1.2, pad=0.6, sharey=axScatter)

# the scatter plot:
# histograms
binwidth = 0.02
bins1 = np.arange(-121.6, -120.5 + binwidth, binwidth)
bins2 = np.arange(38.2, 39.1 + binwidth, binwidth)
axHistx.hist(lon, bins = bins1)
axHisty.hist(lat, bins = bins2, orientation='horizontal')
plt.savefig('C:/Users/Victor/Desktop/SemStat/Tout.pdf', bbox_inches='tight')
plt.show()
```