

Cours de Data Science

Marc Tommasi

7 octobre 2020

Outline

1 Introduction du cours

2 Introduction du ML

3 Fondamentaux

4 Compromis Biais-Complexité

5 Régression linéaire

Outline

1 Introduction du cours

2 Introduction du ML

3 Fondamentaux

4 Compromis Biais-Complexité

5 Régression linéaire

Préambule

- évaluation en CCI
- étudiants salariés hors apprentissage ?
- projet de communication prologin :
 - ▶ concours de programmation depuis 1992
 - ▶ 20 ans et moins
 - ▶ épreuve régionale à Lille ?
 - ▶ centre d'examen
 - ▶ amphi
 - ▶ salles machines
 - ▶ un samedi
 - ▶ modulo covid

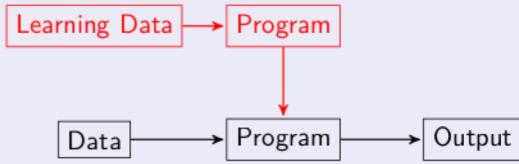
Objectifs

- Présentation de quelques méthodes, algorithmes fréquemment rencontrés en science des données
- Introduction à l'apprentissage machine
- Introduction de quelques notions théoriques du domaine
- Aspects pratiques
- Vous ne serez pas des experts à la fin du cours !

Programmation



IA et Machine Learning



- **IA Classique** systèmes experts, systèmes à base de règles, représentation des connaissances, raisonnement logique,...
- **Machine Learning** : Approches complètement dirigées par les données

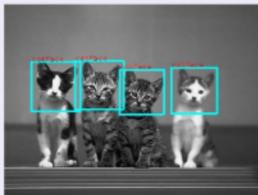
Machine Learning

- ML provides a computer with the ability to do certain tasks **without being explicitly programmed** for it (Arthur Samuel, 1959)
- This is done by learning from **data**
- Multidisciplinary field : computer science, statistics, optimization
- ML is fueling the current progress in AI
- Processus d'apprentissage est automatisable

Secteurs d'activité

- Services : banques, assurances, commerce et distribution,
 - ▶ des tableaux de bord à la prise décision
 - ▶ généralisation à de nouveaux services
- Agriculture : encore peu développé, nombreux efforts initiatives actuellement
- Industrie : incitations avec l'industrie 4.0

Example applications



Economic growth has slowed down in recent years.
Das Wirtschaftswachstum hat sich in den letzten Jahren verlangsamt.
Economic growth has slowed down in recent years.
La croissance économique s'est ralentie ces dernières années.

- Requires **large amounts of data** (potentially *sensitive, personal data*)

Types de données et Applications

- Images : astronomie, agriculture, météo, archéologie, authentification (reconnaissance faciale), médecine (IRM, ...), OCR, voitures autonomes,...
- Texte : NLU, Génération, Spam, textes médicaux, traduction,
- Son : MIR, STT, Chatbots, sous-titrage vidéo, traduction, reconnaissance et authentification,...
- Données génétiques
- Données de capteurs : transport, robotique, maintenance prédictive, météo,...
- Jeux : Go, échecs, jeux vidéo, ...
- Données du web : recommandation, etc...

- Dès l'apparition de l'ordinateur !
- Articles importants dans les années 50 et 60
- Période un peu froide à partir de la fin des années 60
- Systèmes experts dans les années 70-80
- Essor du machine learning, apprentissage statistique à partir des années 90
- Succès dans la vision par ordinateur, les jeux, ... par les techniques d'apprentissage profond

Difficultés de l'IA et du ML en particulier

- attaques
- robustesse
- données personnelles
- équité
- confiance
- interprétation
- cadre légal
- acceptation sociale et peurs
- différence ML et automatisation

Conclusion

- Progrès assez spectaculaires récents
- Passage récent dans l'industrie et le grand public,
- Technologies pas toujours mûres, en constante évolution,
- Besoin de personnes qualifiées, qui s'adapteront aux évolutions certaines
- De nombreuses questions de société.

Outline

- 1 Introduction du cours
- 2 Introduction du ML
- 3 Fondamentaux
- 4 Compromis Biais-Complexité
- 5 Régression linéaire

Introduction

- Machine Learning : convertir expérience en connaissance et expertise.

- Citation de Mitchell :

A computer program is said to learn from experience E with respect to some task T and some performance measure P if its performance on task T , measured by P , improves with experience E .

- entrée : l'expérience est représenté par des données d'apprentissage

- sortie : l'expertise est un autre programme qui réalise une tâche

- Questions :

- ▶ Quelles données en entrée ?
- ▶ Comment automatiser l'apprentissage ?
- ▶ Quand dire qu'on apprend, comment évaluer le succès ?

Apprentissage par cœur et généralisation

- Détection du spam : on peut mémoriser tous les spam qui ont été étiquetés comme tels. A-t-on appris ?
- besoin de **généralisation**, par exemple repérer des mots qui sont témoins d'un spam
- Attention aux faux amis (les pigeons de Skinner qui renforcent un comportement n'ayant rien à voir avec la récompense (superstition))
- Importance d'information a priori (les rats qui ont appris à distinguer bonne/mauvaise nourriture n'apprennent pas s'ils reçoivent un stimulus négatif différent d'une nausée. Ils ignorent des faits corrélés dans le temps mais non déterminants)
- No free lunch theorem : on n'apprend pas sans un certain biais

Quand passer à de l'apprentissage automatique ?

- Quand la tâche est routinière mais pas explicitement définissable facilement.
Trop difficile à programmer directement comme conduire une voiture, reconnaître une image, ou traduire une parole en texte.
- Quand les données sont trop massives et la combinatoire trop importante : prédire la météo, analyser des données génomiques, ... retrouver des motifs dans les données massives en règle générale.
- Quand on doit s'adapter. Les programmes écrits « à la main » sont trop rigides. Penser à tous les programmes qui demandent une personnalisation (reco de la parole,...) ou doivent évoluer vite (spam).

Type d'apprentissage

- Expérience : données d'entraînement pour l'apprentissage ; Mesure de la performance sur le test.
- Supervisé, non supervisé, par renforcement :
 - ▶ En non supervisé train et test n'apportent pas plus d'information,
 - ▶ supervisé et par renforcement, le train a plus d'information que le test
- Actif/passif : l'apprenant influence ou pas l'environnement avec lequel il interagit.
- Avec teatcher ou pas : comme l'école ou comme un scientifique face à la nature. NB : le teatcher peut être un adversaire
- Online ou batch : quand doit on prendre la décision ?
- Basé sur un modèle ou sur les instances des données (paramétrique, non paramétrique)

Relation avec les autres sciences

- Optim, stats, théorie de l'information, théorie des jeux, et bien-sûr l'informatique
- Branche de l'IA, mais on n'essaye pas d'imiter, mais on utilise l'information comme outil supplémentaire/complémentaire à l'intelligence

ML et Stats

- Stats plus souvent sur la vérification/validité d'une hypothèse, ML va chercher ces hypothèses.
- En stat on s'intéresse plus à des résultats asymptotiques. En ML, on s'intéresse à le faire sur machine, contraintes de temps et mémoire.
- En stats on pose hypothèses sur des lois. En ML en général assez peu d'hypothèses sur les distributions ;

Défis

- Obtenir des données
- Obtenir de bonnes données
 - ▶ biais de sampling
 - ▶ outliers
 - ▶ données non représentatives
- Overfitting
- Underfitting
- Évaluation de l'erreur
- Sélection de modèle ou d'algorithme, mais « no free lunch » !

Outline

- 1 Introduction du cours
- 2 Introduction du ML
- 3 Fondamentaux
- 4 Compromis Biais-Complexité
- 5 Régression linéaire

Exemples de problèmes de classification supervisée

- spam : présence des mots « sex », « offer »
- papaye bonne ou pas selon couleur, moelleux, ...

Formalisation

- Données et espace de description $x \in \mathcal{X}$
- Cible $y \in \mathcal{Y}$
- Un échantillon $S \subseteq \mathcal{X} \times \mathcal{Y}$
- sortie de l'apprentissage : une règle de prédiction $f : \mathcal{X} \rightarrow \mathcal{Y}$

Vocabulaire

- espace de description : features, attributs, (voire champs)
- échantillons ou jeu de données, dataset
- composé d'instance, records, exemples, enregistrements
- cible, (étiquette ou classe dans certains cas),
- données d'entraînement
- sortie : fonction, règle(s) ou modèle

Hypothèses classiques pour la classification supervisée

- Les données de S sont générées selon une distribution de probabilités \mathcal{D} , fixée et inconnue.
- On suppose qu'il existe une telle fonction cible f et que les données sont générées par \mathcal{D} puis étiquetées par f .
- Expression de la perte, ou **erreur réelle** pour une hypothèse h comme

$$L_{\mathcal{D},f}(h) = \mathcal{D}(\{x \mid h(x) \neq f(x)\}).$$

- L'apprenant ne connaît pas \mathcal{D} donc
- le calcul de $L_{\mathcal{D},f}(h)$ ne peut pas être fait par l'apprenant !

ERM : Principe de minimisation du risque empirique

- La chose qu'il peut calculer est le **risque ou perte empirique**

$$L_S(h) = \frac{|\{i \in [m] \mid h(x_i) \neq y_i\}|}{m}.$$

Limite du modèle

- Problème d'apprentissage par cœur et d'overfitting. La règle suivante est parfaite si on minimise le risque empirique :

$$h(x) = \begin{cases} y_i & \text{si il existe } i \text{ tq } x = x_i \\ 0 & \text{sinon} \end{cases}$$

- On peut arriver au même constat avec des classes de fonctions (comme des polynômes)
- En généralisation, on ne fait pas mieux que random.

ERM avec un biais

- on fixe une classe d'hypothèses \mathcal{H} , a priori, avant de voir les données, c'est une connaissance antérieure

$$\text{ERM}_{\mathcal{H}}(S) = h_S \in \underset{h \in \mathcal{H}}{\operatorname{argmin}} L_S(h)$$

- Dans l'exemple du spam ou des papayes,
 - ▶ possibilité : des rectangles dans le plan fonctionne avec le principe ERM (limite overfit)
 - ▶ une classe plus large permet l'overfit
- Question fondamentale du ML : quelle classe choisir pour ne pas avoir de l'overfitting avec ce principe ?

ERM et calcul d'erreurs

- L'*Hypothèse de réalisabilité* dit que \mathcal{H} contient une fonction h^* qui réalise une perte nulle selon la distribution $L_{\mathcal{D},f}(h^*) = 0$.
- Cela entraîne que l'erreur empirique $L_S(h^*) = 0$
- Et si on suit le principe ERM, alors $L_S(h_S) = 0$ avec probabilité 1 si S est tiré selon \mathcal{D} .
- Mais on veut calculer l'*erreur réelle* $L_{\mathcal{D},f}(h_S)$ et non pas l'erreur empirique,
- d'où l'importance de du tirage selon \mathcal{D} selon l'hypothèse iid.

Big data, small data

- Rappel : des données, de bonnes données, etc...
- quid dans les cas très déséquilibrés, distributions difficiles,...

Confiance et approximation

- $L_S(h_S)$ est bien une variable aléatoire car le choix de h_S dépend de S .

Confiance

- On a un résultat en probabilité car on ne peut garantir d'éviter des tirages extrêmes (toujours la même valeur de y, \dots).
- On note souvent δ la probabilité d'avoir un échantillon non représentatif. Alors $(1 - \delta)$ est la confiance.

Approximation

- Le calcul de h peut n'être qu'approximatif (on commet quelques erreurs).
- On peut les tolérer jusqu'à un certain seuil de précision, souvent noté ϵ : on cherche à avoir $L_{D,f}(h_S) \leq \epsilon$
- On se trompe pour tous les échantillons S tels que $L_{D,f}(h_S) > \epsilon$

Classe de taille finie

Theorem (Les classes finies sont apprenables sous l'hypothèse de réalisabilité)

Si \mathcal{H} est de taille finie, pour tout $\epsilon > 0$ et $\delta \in [0, 1]$, pour toute fonction cible f de l'apprentissage, pour toute distribution \mathcal{D} telle que il existe $h \in \mathcal{H}, L_{\mathcal{D}, f}(h) = 0$, alors avec probabilité $1 - \delta$ sur le tirage iid d'un échantillon S de taille supérieure à $\frac{\log(|\mathcal{H}|/\delta)}{\epsilon}$, l'erreur réelle de la minimisation du risque empirique (ERM) est plus petite que ϵ

$$L_{\mathcal{D}, f}(h_s) \leq \epsilon.$$

- On considère que \mathcal{H} est de taille finie, on prend $m \geq \frac{\log(|\mathcal{H}|/\delta)}{\epsilon}$ exemples alors $\mathcal{D}^m(\{S \mid L_{\mathcal{D}, f}(h_S) > \epsilon\}) \leq \delta$.

Calcul de l'erreur réelle

- Comment borner la probabilité de tirer un échantillon qui entraîne un échec ?
Quel est le poids selon \mathcal{D}^m de cet échantillon de taille m qui calcule une hypothèse erronée ?

$$\mathcal{D}^m(\{S \mid L_{\mathcal{D},f}(h_S) > \epsilon\})$$

Des exemples non suffisamment informatifs

- On va construire une mauvaise hypothèse h_S si les exemples font croire que h_S est correcte : $L_{\mathcal{D},f}(h_S) > \epsilon$ et $L_S(h_S) = 0$ car on suppose l'*Hypothèse de réalisabilité*.
- On note M cet ensemble d'échantillons pour lesquels on peut construire une mauvaise hypothèse :

$$M = \bigcup_{h \text{ s.t. } L_{\mathcal{D},f}(h) > \epsilon} \{S' \mid L_{S'}(h) = 0\},$$

et S appartient donc à cet ensemble M donc

$$\mathcal{D}^m(\{S \mid L_{\mathcal{D},f}(h_S) > \epsilon\}) \leq \mathcal{D}^m(M).$$

Union Bound

- On utilise souvent cette inégalité en probabilités, appelée inégalité de Boole ou **union bound** en anglais : $\mathcal{D}(\bigcup_i A_i) \leq \sum_i \mathcal{D}(A_i)$.
- Appliqué ici, on trouve

$$\mathcal{D}^m(\{S \mid L_{\mathcal{D}, f}(h_S) > \epsilon\}) \leq \sum_{h \text{ s.t. } L_{\mathcal{D}, f}(h) > \epsilon} \mathcal{D}^m(\{S' \mid L_{S'}(h) = 0\}).$$

Borner le membre droit

- $L_{S'}(h) = 0$ si pour tout exemple x_i de S' on a $h(x_i) = f(x_i)$. Les exemples sont tirés de façon iid, donc

$$\mathcal{D}^m(\{S' \mid L_{S'}(h) = 0\}) = \prod_i^m \mathcal{D}(\{x_i \mid h(x_i) = f(x_i)\}).$$

- Comme $L_{\mathcal{D},f}(h) = \mathcal{D}(\{x \mid f(x) \neq h(x)\})$ on a

$$\mathcal{D}(\{x_i \mid h(x_i) \neq f(x_i)\}) = 1 - L_{\mathcal{D},f}(h).$$

- Pour un h pour lequel l'erreur est plus grande que ϵ on a

$$\mathcal{D}(\{x_i \mid h(x_i) \neq f(x_i)\}) \leq 1 - \epsilon,$$

et

$$\mathcal{D}^m(\{S' \mid L_{S'}(h) = 0\}) \leq (1 - \epsilon)^m.$$

Résultat

- En combinant

$$\mathcal{D}^m(\{S \mid L_{\mathcal{D},f}(h_S) > \epsilon\}) \leq \sum_{h \text{ s.t. } L_{\mathcal{D},f}(h) > \epsilon} \mathcal{D}^m(\{S' \mid L_{S'}(h) = 0\})$$

et

$$\mathcal{D}^m(\{S' \mid L_{S'}(h) = 0\}) \leq (1 - \epsilon)^m.$$

on obtient en posant $\mathcal{H}_B = \{h \text{ s.t. } L_{\mathcal{D},f}(h) > \epsilon\}$ l'ensemble des mauvaises hypothèses

$$\mathcal{D}^m(\{S \mid L_{\mathcal{D},f}(h_S) > \epsilon\}) \leq |\mathcal{H}_B|(1 - \epsilon)^m \leq |\mathcal{H}_B|e^{-\epsilon m} \leq |\mathcal{H}|e^{-\epsilon m}.$$

- Apprentissage Probablement Approximativement Correct

Definition (Modèle PAC)

Une classe d'hypothèses \mathcal{H} est PAC apprenable si il existe $m_{\mathcal{H}} : (0, 1)^2 \rightarrow \mathbb{N}$ et un algorithme tels que pour tout $\epsilon, \delta \in (0, 1)$, pour toute distribution \mathcal{D} sur \mathcal{X} , pour tout étiquetage $f : \mathcal{X} \rightarrow \{0, 1\}$, si l'*Hypothèse de réalisabilité* est vraie, alors l'algorithme avec $m > m_{\mathcal{H}}(\epsilon, \delta)$ exemples en entrée tirés iid selon \mathcal{D} produit h tq la probabilité que $L_{\mathcal{D}, f}(h) \leq \epsilon$ est au moins $1 - \delta$.

Complexité en nombre d'exemples

- La fonction $m_{\mathcal{H}}$ détermine le nombre d'exemple nécessaires pour apprendre.
- On cherche la fonction $m_{\mathcal{H}}$ la plus petite.

Interprétation

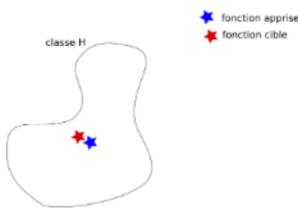


FIGURE – Avec forte probabilité,
l'erreur est faible

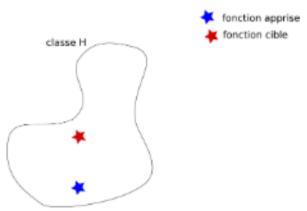


FIGURE – Avec faible probabilité,
l'erreur est forte

- l'apprentissage doit être probablement (avec probabilité $> 1 - \delta$) approximativement (avec une erreur bornée par ϵ) correct
- pour toute distribution et valeur de ϵ et δ .
- Existe-t-il des classes PAC apprenables ?

Classes finies

Les classes finies sont PAC apprenables

Types de problèmes et fonction de perte

- Problème binaire \mathcal{Y} est de taille 2 (e.g. spam/non spam)
- Problèmes multiclass : \mathcal{Y} est de taille finie, on prédit une des valeurs (e.g. prix élevé, moyen ou bas)
- Problèmes multi étiquettes : \mathcal{Y} est de taille finie, on prédit une à plusieurs étiquettes : (e.g. catégories d'un billet de blog)
- Problèmes de régression : $\mathcal{Y} = \mathbb{R}$ (e.g. prédire une vente)

fonctions de perte

- La perte 0,1 utilisée dans le cas binaire ou multiclass.
- La perte quadratique $(f(x) - y)^2$ en régression, ou $|f(x) - y|$ ou matrice de coût...

Relâcher l'hypothèse de réalisabilité

- Illusoire de penser que les valeurs à prédire sont une fonction des descriptions.

PAC Agnostique

- \mathcal{D} est maintenant une distribution sur la donnée jointe des descriptions et des valeurs à prédire : $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$
- On considère une fonction de perte $\ell : \mathcal{H} \times \mathcal{Z} \rightarrow \mathbb{R}_+$
- Similaire à PAC, mais on cherche alors le nombre d'exemples pour être à une distance au plus ϵ du classifieur optimal dans la classe \mathcal{H} .

$$L_{\mathcal{D}}(h) = \mathbb{E}_{z \sim \mathcal{D}}[\ell(h, z)] \leq \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \epsilon$$

- L'erreur empirique est

$$L_S(h) = \frac{|\{(x, y) \in S \mid h(x) \neq y\}|}{|S|}$$

Interprétation



FIGURE – Avec forte probabilité,
l'erreur est faible

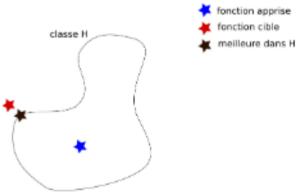


FIGURE – Avec faible probabilité,
l'erreur est forte

- On n'a plus \mathcal{D} sur \mathcal{X} puis une fonction f qui associe une étiquette dans \mathcal{Y} à toute donnée de \mathcal{X} mais directement une distribution \mathcal{D} sur $\mathcal{X} \times \mathcal{Y}$.
- On mesure avec une fonction de perte ℓ qui s'applique sur une fonction de prédiction et un couple (x, y)
- La fonction cible peut être hors de la classe
- On regarde la perte par rapport à la meilleure fonction dans la classe

Outline

- 1 Introduction du cours
- 2 Introduction du ML
- 3 Fondamentaux
- 4 Compromis Biais-Complexité
- 5 Régression linéaire

Pas d'algorithme PAC universel

- Tout algo d'apprentissage d'une tâche binaire a au moins une distribution pour lequel il va échouer.
- Un biais est nécessaire, une connaissance a priori
- Le biais peut être une restriction sur la distribution, sur les propriétés de la classe de fonctions

Theorem (No free lunch)

Soit A un algorithme d'apprentissage pour une tâche de classification binaire, avec une perte de type 0-1 sur un domaine \mathcal{X} . Soit une taille d'échantillon $m \leq |\mathcal{X}|/2$. Alors il existe une distribution \mathcal{D} sur $\mathcal{X} \times \{0, 1\}$ telle que

- il existe une fonction $f : \mathcal{X} \rightarrow \{0, 1\}$ telle que $L_{\mathcal{D}}(f) = 0$
- avec probabilité au moins $1/7$ sur le choix de S tiré iid selon \mathcal{D}^m , on a $L_{\mathcal{D}}(A(S)) \geq 1/8$.

Décomposition de l'erreur

- Découpage de l'erreur entre approximation et estimation.

Erreurs d'approximation

- due au choix d'une classe \mathcal{H} : biais inductif.
- cette erreur est nulle si on a l'hypothèse de réalisabilité
- NB : dans le cadre agnostique, l'erreur comprend l'erreur de Bayes (qu'on pourrait soustraire)

Erreurs d'estimation

- Erreur empirique due à l'échantillon
- Dans le cas de \mathcal{H} finie, cela croît en log selon la taille de \mathcal{H} et décroît avec la taille de S .

Compromis

- on augmente la taille de H on baisse l'erreur d'approximation mais augmente la complexité de la classe et le risque de sur apprentissage.



FIGURE – Une classe « simple/petite », un biais fort, il est facile de trouver une fonction proche de la meilleure. Grande erreur d'approximation, faible erreur d'estimation. Sujet à underfitting.

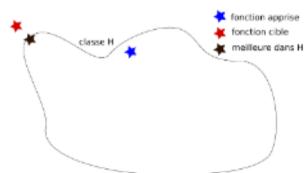


FIGURE – Une classe plus « large/complexe », moins de biais, il est plus difficile de trouver une fonction proche de la meilleure. Faible erreur d'approximation, grande erreur d'estimation. Sujet à overfitting.

Outline

- 1 Introduction du cours
- 2 Introduction du ML
- 3 Fondamentaux
- 4 Compromis Biais-Complexité
- 5 Régression linéaire

Idées informelles

- Utilisé pour expliquer comme pour prédire. Exemple : expliquer la consommation en fonction du poids d'une la voiture.
- On prend la classe des droites $y = ax + b$. On cherche a et b étant donné des poids (x) et la consommation observée (y).
- On appelle aussi b *intercept* (l'ordonnée à l'origine) et a est un coefficient (qui définit la pente de la droite).
- Si on sait que les données x sont centrées en 0 alors l'intercept est nul.
- Généralisation à la dimension n . Exemple : prédire la consommation en fonction du poids de la voiture et de la puissance du moteur, de la météo, de la vitesse, etc.
- On dispose de m exemples de la forme $((x_1, x_2, \dots, x_d), y)$
- On prend une classe \mathcal{H} de fonctions linéaires : on cherche les w_i tels que $y = w_0 + w_1 x_1 + \dots + w_d x_d$
- Remarque : on peut poser $x_0 = 1$ et résoudre $y = w_0 x_0 + w_1 x_1 + \dots + w_d x_d = \mathbf{w}^\top \mathbf{x} = \langle \mathbf{w}, \mathbf{x} \rangle$

ERM

- $\mathcal{X} \subseteq \mathbb{R}^d$, $\mathcal{Y} \subseteq \mathbb{R}$ et $\mathcal{H} = \{\mathbf{x} \rightarrow \langle \mathbf{w}, \mathbf{x} \rangle \mid \mathbf{w} \in \mathbb{R}^d\}$.
- On utilise la perte quadratique : $\ell(h, (\mathbf{x}, y)) = (h(\mathbf{x}) - y)^2$

ERM, fonction objectif

- On cherche à minimiser l'erreur quadratique pour chaque exemple et la fonction de perte, fonction d'objectif est

$$\operatorname{argmin}_{\mathbf{w}} \frac{1}{m} \sum_i (\langle \mathbf{w}, \mathbf{x}_i \rangle - y_i)^2$$

(MSE : Mean Square Error, moyenne des erreurs quadratiques)

- Notation matricielle si toutes les lignes de \mathbf{X} sont les données et les colonnes sont les attributs

$$\operatorname{argmin}_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$$

(on a viré le $\frac{1}{m}$)

- l'objectif est convexe : il y a un minimum global
- on peut le calculer analytiquement.

Dérivées et gradient

- La dérivée f' d'une fonction $f : \mathbb{R} \rightarrow \mathbb{R}$ évaluée en x est définie par

$$f'(x) = \lim_{\epsilon \rightarrow 0} \frac{f(x + \epsilon) - f(x)}{\epsilon}$$

- La dérivée s'annule pour un x qui minimise f .
- Pour $f : \mathbb{R}^d \rightarrow \mathbb{R}$, on peut considérer la dérivée par rapport à chaque composante de $\mathbf{x} = (x_1, \dots, x_d)$. C'est noté $\frac{\partial f}{\partial x_i}$.
- Le vecteur de dimension d de toutes ces fonctions dérivées est le gradient $\nabla f(\mathbf{x}) = (\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_d})$.
- NB : Pour une fonction $f : \mathbb{R}^d \rightarrow \mathbb{R}^m$, on peut considérer la matrice $J_{\mathbf{x}}(f)$ de m lignes qui sont les gradients $\nabla f_i(\mathbf{x})$
- la dérivée de $f(x) = x^n$ est $f'(x) = nx^{n-1}$ et la dérivée d'une constante est nulle;

Bilan

- On peut trouver une solution analytique qui sera $(X^T X)^{-1} X^T \mathbf{y}$
- Informellement, on cherche $X\mathbf{w} = \mathbf{y}$, qu'on peut réécrire $X^T X \mathbf{w} = X^T \mathbf{y}$, puis $\mathbf{w} = (X^T X)^{-1} X^T \mathbf{y}$.
- Remarque
 - ▶ La matrice $(X^T X)$ peut ne pas être inversible, on peut calculer plutôt $X^\dagger \mathbf{y}$ où X^\dagger est la Moore-Penrose pseudo-inverse, qui elle se calcule par une SVD (Si $X = U\Sigma V^T$ alors $X^\dagger = V\Sigma^\dagger U^T$ et Σ^\dagger est calculée en prenant l'inverse des valeurs sauf pour les 0 qui restent 0).

Complexité

- Le calcul analytique est coûteux pour de très grands jeux de données
- En $O(n^3)$ pour la plupart des algos (voire $O(n^{2.4})$ pour les meilleurs).
- Une approche par descente de gradient permet d'avoir un algorithme quadratique qui marche plutôt bien en pratique.