PAO : machine learning et clicks prediction

Octave QUEFFELEC et Sandratra RASENDRASOA

11/06/2017

Part I
Introduction

> "We are drowning in information and starving for knowledge."
> Rutherford D. Roger[1]
>
> January 2009 Hal Varian, Google's Chief Economist, tells the McKinsey Quarterly: "I keep saying the sexy job in the next ten years will be statisticians. People think I'm joking, but who would've guessed that computer engineers would've been the sexy job of the 1990s? The ability to take data—to be able to understand it, to process it, to extract value from it, to visualize it, to communicate it—that's going to be a hugely important skill in the next decades... "[2]

Avec l'avènement des technologies de l'information, le Data Science (ou science des données) est une discipline ayant pour objectif de répondre au besoin d'analyser des immenses ensembles de données pour en extraire des informations -potentiellement- utiles. Les applications sont nombreuses, que ce soit dans le secteur privé (la recommandation de séries par Netflix, les moteurs de recherche, l'analyse financière), le domaine médical (l'aide aux diagnostics) et bien d'autres encore.

Le but de notre PAO est de s'initier aux méthodes et algorithmes issus du Machine Learning. Pour cela, nous avons à notre disposition un jeu de données issu de la base de données en ligne Kaggle, qui a été créé pour répondre à la problématique suivante : déterminer un modéle pour estimer le taux de clicks(ou CTR) qui est une unité de mesure importante pour évaluer les publicités en ligne. Nous avons utilisé comme base un projet de l'université de Washington, en réalisant la partie pratique du projet et en ayant une réunion toutes les deux

[1] T.Hastie, R. Tibshrani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction (Springer Verlag)

[2] Gil Press, "A Very Short History Of Data Science",Forbes

semaines avec nos responsables pédagogiques M.Gasso et M.Gauzere pour aborder ces différents points et les parties théoriques s'y référant.

L'ensemble du projet a été réalisé en langage Python qui est un des langages proposé dans le sujet de l'université de Washington. Nous avons notamment utilisé les librairies Numpy et Scipy pour nos algorithmes. Nous avons aussi dans un premier temps réalisé quelques algorithmes sous Matlab pour se familiariser avec les jeux de données.

Dans ce rapport, nous allons vous présenter tout d'abord les éléments théoriques nécessaires pour ce projet, puis l'ensemble des travaux pratiques que nous avons réalisé ainsi que les résultats qui y sont liés. Nous terminerons par une interprétation de ces résultats et une conclusion globale.

Part II
Partie théorique

1 Supervised learning

Il existe plusieurs algorithmes d'appprentissage automatique(ou machine learning) qui se différencient par le mode d'apprentissage qu'ils utilisent. Dans le cadre de cette initiation, nous ne verrons qu'un mode d'apprentissage, mais il nous semble important de mentionner que d'autres techniques existent pour répondre aux différents besoins en machine learning.
L'apprentissage supervisé (ou supervised learning) est la technique d'apprentissage la plus commune pour répondre à des problèmes de machine learning. L'objectif est le suivant :

> A partir des données $(x_i, y_i)\epsilon X \times Y, i = ..., N$, estimer les dépendances entre X et Y.

On parle ici d'apprentissage supervisé (en opposition à l'apprentissage non-supervisé) car les y_i (en pratique, il s'agira des cas déja traités et validés) permettent de guider le processus d'estimation. Un exemple concret pour différencier l'apprentissage supervisé de l'apprentissage non-supervisé est présenté dans la figure 1.

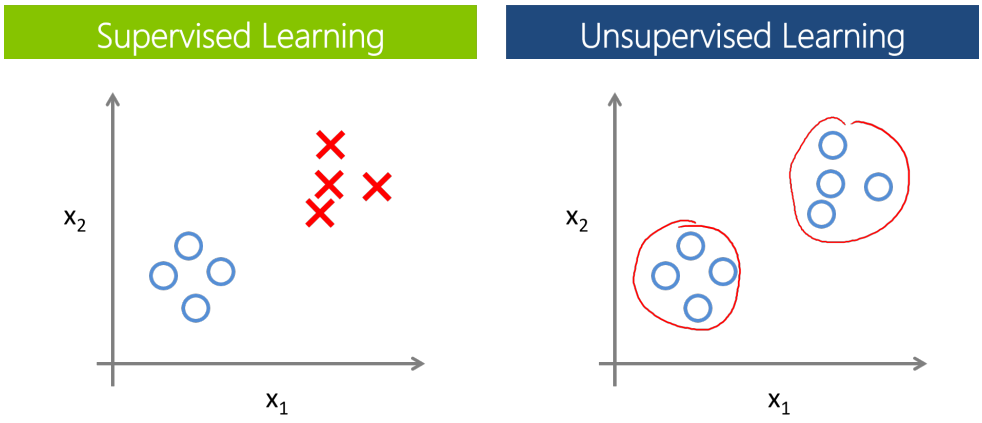

Figure 1: Apprentissage supervisé VS non-supervisé

En apprentissage supervisé, on sait d'avance qu'il y a exactement deux catégories(cercle bleu et croix rouge), tandis qu'en apprentisage non supervisé, on essaie de partitionner et classer les données dans des groupes homogènes (en anglais, on parle aussi de clustering). Dans le cadre de notre projet, nous travaillons en apprentissage supervisé, nous possédons ainsi plusieurs jeux de données correspondants à X et Y, que nous préciserons dans la partie pratique.

2 Fonction de coût

En apprentissage supervisé, l'objectif est de trouver une fonction $f : X \to Y$ qui permet d'estimer la valeur y associée à x. On peut alors se poser la question suivante : la fonction f que nous avons construite est-elle optimale pour résoudre notre problème ?
On introduit alors la notion de de coût $L(Y, f(X))$, dont l'objectif et d'évaluer la pertinence de la prédiction réalisée par f, et de pénaliser les erreurs. Dans l'idéal, on chercherait donc une fonction f qui prédit au mieux y; en d'autres termes, une fonction f qui minimise l'erreur entre la vérité et la valeur estimée. D'où la fonction f telle que :

$$R(f) = E_{X,Y}[L(Y, f(X))]$$

R(f) où R est appelé le risque moyen ou erreur de généralisation.
Un exemple de fonction de coût et de risque moyen associé est le coût quadratique (ou moindres carrés) :

$$L(Y, f(X)) = (Y - f(X))^2$$

$$R(f) = E[(Y - f(X))^2] = \int (y - f(x))^2 p(x, y) dx dy$$

La fonction de coût est associé à l'ensemble Y dont on souhaite prédire les valeurs, et on peut distinguer deux types de problèmes en fonction de cet ensemble Y.

1. Si l'ensemble Y est un sous-ensemble de R^d (donc continu), on traitera le problème comme étant un problème dit de régression, où les moindres carrés sont la fonction de coût usuelle.

2. Dans le cadre de notre projet, Y est un ensemble discret non-ordonné (clic ou non clic, 0 ou 1 dans le jeu de données), il s'agit d'un problème dit de classification. Ici, on cherchera plutôt à approcher la fonction de coût suivante : $\theta(-yf(x))$où θ est la fonction échelon.

3 Perceptron

Le perceptron est un classifieur lineaire de la forme :

$$f(x) = \begin{cases} 1 & si < w, x > +b > 0 \\ 0 & sinon \end{cases}, w \epsilon R^n$$

$w$le vecteur des poids, $b \in R$, le biais, $< w, x >$, le produit scalaire entre le vecteur de poids w et le vecteur d'entrée x. $< w, x > +b = 0$ est alors l'équation de l'hyperplan affine qui sépare l'espace en 2 classes. Le but est donc d'entrainer notre vecteur de poids w afin de connaître par la suite notre fonction f qui nous permettra de prédire la classe d'un vecteur d'entrée x donné.

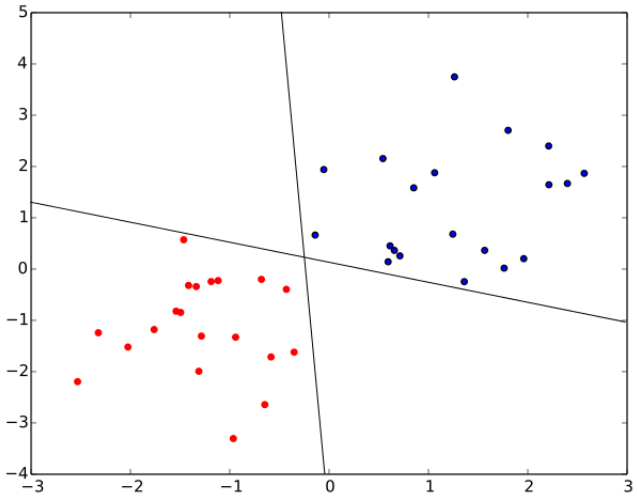

Figure 2: Perceptron

L'équation séparatrice des 2 classes n'est pas unique. Ici, 2 droites sont quasi-perpendiculaires l'une à l'autre mais le perceptron n'a aucun moyen d'en privilégier l'une au profit de l'autre.
Voici l'algorithme du perceptron, qui nous permet de mettre à jour à chaque iteration les poids w :

Algorithme d'apprentissage du Perceptron
Entrée : $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$, un échantillon complété linéairement séparable de $\mathbb{R}^{n+1} \times \{-1, 1\}$
$\mathbf{w}_0 = 0 \in \mathbb{R}^{n+1}, k = 0$
Répéter
  Pour $i = 1$ **à** l
    Si $y_i \langle \mathbf{w}_k, \mathbf{x}_i \rangle \leq 0$ **alors**
      $\mathbf{w}_{k+1} = \mathbf{w}_k + y_i \mathbf{x}_i$
      $k = k + 1$
  FinPour
Jusqu'à ce qu'il n'y ait plus d'erreurs
Sortie : $\mathbf{w}_k$

Figure 3: Algorithme Perceptron

Le perceptron peut etre vu comme un type de réseau de neurones simplifié. Nous avons donc voulu pour nous entrainer, implémenter en matlab puis en python l'algorithme.