Régression linéaire et polynomiale

Benjamin Monmege
benjamin.monmege@lsv.ens-cachan.fr

7 mars 2012

On considère un problème de régression linéaire pour lequel on fournit un ensemble de $N \in \mathbb{N}$ exemples $(x_i, y_i) \in (\mathbb{R}^d \times \mathbb{R})^N$. Il s'agit donc de trouver une fonction f linéaire de $\mathbb{R}^d$ dans $\mathbb{R}$ qui minimise l'erreur au sens des moindres carrés

$$\mathrm{Err}(f) = \frac{1}{2}\sum_{n=1}^{N}(f(x_n) - y_n)^2$$

Exercice 1. On s'intéresse dans un premier temps au cas où $d = 1$. On recherche donc un vecteur $\theta = (\theta_1, \theta_2)$ de paramètres qui minimisent $\mathrm{Err}(\theta) = \frac{1}{2}\sum_{n=1}^{N}(\theta_1 + \theta_2 x_n - y_n)^2$.

1. Rappeler la valeur du gradient de l'erreur $(\frac{\partial \mathrm{Err}}{\partial \theta_1}, \frac{\partial \mathrm{Err}}{\partial \theta_2})$.
2. Écrire une fonction `errorFunction` qui calcule la fonction Err ainsi que son gradient, en fonction de θ, $(x_n)_{1\leq n\leq N}$, $(y_n)_{1\leq n\leq N}$.
3. Implémenter une descente de gradient pour trouver une approximation du minimum de Err, en utilisant une recherche *backtracking line search* : plus précisément, il est demandé d'écrire une fonction `gradientDescent` qui prend en argument une fonction d'erreur (on l'appliquera avec la fonction `@(t)(errorFunction(t,X,y))`), le point de départ de la descente, les paramètres α et β pour le backtracking et l'erreur ε pour le critère d'arrêt ; elle renverra le paramètre θ^* optimal ainsi que la liste des valeurs d'erreur successivement prises (afin qu'on puisse la représenter si nécessaire).
4. Tester votre algorithme sur des données artificielles : $y = 3 + 2x$ sur $[0, 5]$, avec ou sans bruit gaussien (cf. la fonction `normrnd`), ainsi que $y = \sin x$ sur $[0, 5]$, avec ou sans bruit gaussien. Faire varier le nombre d'exemples n, l'intensité du bruit, et visualiser vos résultats en traçant les exemples, la fonction originelle ainsi que la droite de régression linéaire.

Exercice 2. On généralise ensuite à d'autres valeurs de d. On cherche donc un vecteur $\theta = (\theta_0\ \theta_1 \cdots \theta_d)^{\mathrm{T}} \in \mathbb{R}^{d+1}$ minimisant

$$\mathrm{Err}(\theta) = \frac{1}{2}\sum_{n=1}^{N}(\theta_0 + \theta_1 x_{n,1} + \ldots + \theta_d x_{n,d} - y_n)^2 \tag{1}$$

1. Mettre à jour si nécessaire votre fonction `errorFunction` de l'exercice précédent dans le cas général. Appliquer une méthode de gradient pour minimiser l'erreur. Afin de comparer les résultats, on pourra aussi utiliser la fonction MATLAB `fminunc` pour minimiser l'erreur automatiquement.
2. Tester votre algorithme sur des données artificielles.

Exercice 3. On s'intéresse à un problème de régression appliqué à l'estimation de biens immobiliers, en fonction de certains attributs : longueur et largeur du terrain, ainsi que nombre de pièces dans la maison. Après avoir entrainer un algorithme de régression sur un ensemble de 250 exemples, fournissant le prix de maisons vendues récemment, ainsi que la valeur des différents attributs, l'objectif est d'estimer au mieux une nouvelle maison dont on fournit les valeurs des attributs.

Récupérer le fichier `house.mat` situé à l'adresse `http://www.lsv.ens-cachan.fr/~monmege/teach/learning/house.mat`. Grâce à l'instruction `load`, charger son contenu dans MATLAB, à savoir une matrice X contenant les 250 exemples, chacun comptant 3 paramètres (longueur, largeur, nombre de pièces), une matrice y contenant les 250 valeurs de vente, ainsi que l'exemple de test x_0.

1. Appliquer un algorithme de régression linéaire. Êtes-vous satisfait du résultat ?
2. En vous servant de la signification des différents paramètres, créer de nouveaux attributs qui vous semblent plus appropriés pour appliquer une régression linéaire. Appliquer un algorithme de régression linéaire à ces nouveaux attributs et conclure.

Exercice 4. On se place à nouveau dans le cas de la régression linéaire à $d > 0$ dimensions.

1. On note X la matrice $(x_{n,j})_{1\leq n\leq N, 0\leq j\leq d}$, dans laquelle on a ajouté une colonne $x_{n,0} = 1$ aux colonnes de paramètres $x_{n,j}$ avec $1 \leq j \leq d$. Rappeler pourquoi le paramètre θ minimisant (1) vérifie l'équation $\theta = (X^T X)^{-1} X^T y$, en supposant que la matrice $X^T X$ est inversible.
2. En déduire une implémentation du problème de régression linéaire utilisant cette équation, en utilisant la commande `pinv` de MATLAB.

Exercice 5 (Régression polynomiale et surapprentissage)**.** On considère le problème de régression d'un ensemble de paires de points $\{(x_n, y_n)\}_{1\leq n\leq N}$ dans $\mathbb{R}^2$. On pourra par exemple considérer que x_n est uniformément réparti dans l'intervalle $[0; 1]$ et que $y_n = \sin(2\pi x_n) + \varepsilon$ avec ε suivant une loi Gaussienne de variance $\sigma^2 = 0,2$.

On cherche à approcher les données par des polynômes de degré M.

1. Fixer ici, $N = 10$ et appliquer une régression polynomiale sur les données (régression linéaire dans laquelle on génère les nouvelles données $x_n, x_n^2, x_n^3, \ldots$) simultanément avec les degrés $M \in \{0, 1, \ldots, 9\}$. Que remarquez-vous ?
2. Si $\mathrm{Err}(\theta)$ est la somme des erreurs au carré pour un modèle θ donné, on note $E_{\mathrm{RMS}} = \sqrt{2\mathrm{Err}(\theta)/N}$ (*root-mean-square error*). Pour les différentes valeurs de M, calculer et tracer les valeurs de E_{RMS} sur l'ensemble de données.
3. Une vertu attendue d'une fonction de régression est de pouvoir prédire la valeur y d'une nouvelle donnée x. Partant de cette idée, générer un nouvel ensemble de données, qu'on appellera ensemble de données de *validation* (contrairement à l'ensemble de données initial qu'on appelle ensemble de données *d'apprentissage*), comprenant 100 paires de points suivant la même loi que les points (x_n, y_n). Calculer, pour les valeurs de paramètres θ précédemment calculées, les valeurs des erreurs E_{RMS} de ce nouvel ensemble de données et les tracer sur le même graphique. À partir de ce graphique, quelle semble être la *meilleure* valeur de M.
4. Montrer (grâce à MATLAB !) que ce phénomène de surapprentissage (ou *overfitting*) s'estompe à mesure que le nombre N de paires de points de l'ensemble de données d'apprentissage croît.
5. En observant les coefficients des polynômes générés par l'algorithme sur l'ensemble de données avec $N = 10$, proposer une méthode permettant d'éliminer le surapprentissage : elle sera basée sur un terme de pénalité supplémentaire dans la fonction d'erreur $\mathrm{Err}(\theta)$, paramétré par une valeur $\lambda \geq 0$ ($\lambda = 0$ signifiera que le terme de pénalité n'est pas pris en compte). Implémenter votre proposition dans MATLAB. Pour $M = 9$, appliquer votre algorithme avec différentes valeurs pour le paramètre λ. Tracer en particulier les courbes de l'erreur E_{RMS} sur les ensembles de données d'apprentissage et de validation pour différentes valeurs de λ entre 0 et 1. À partir de ce graphique, quelle semble être la meilleure valeur de M.
6. On considère maintenant un *gros* ensemble de données. On peut alors partitionner cet ensemble en un ensemble d'apprentissage (60% des données environ) utilisé pour apprendre les paramètres θ pour diverses valeurs de M et λ, un ensemble de validation (20% des données environ) utilisé pour déterminer la meilleure valeur de λ pour chaque valeur de M étudiée, puis un ensemble de test (20% des données environ) utilisé pour déterminer la meilleur valeur de M. Appliquer cet algorithme pour un ensemble de données de taille 1000.
7. La partition d'un ensemble de données en 3 parties (apprentissage, validation et test) n'est possible que si l'ensemble de données est de taille suffisante. Une alternative possible est la validation croisée. Pour cela, on divise l'ensemble de données en S parties : $V_1, \ldots, V_S$. Pour chaque valeur de $s \in \{1, \ldots, S\}$, on peut lancer l'algorithme d'apprentissage sur $\bigcup_{i\neq s} V_i$, puis on valide la valeur de λ en utilisant l'ensemble V_s pour calculer l'erreur E_{RMS}. La sélection de la meilleure valeur de M peut alors se faire selon les différentes valeurs de s. Implémenter cet algorithme en MATLAB et le tester.