Département d'informatique et de recherche opérationnelle Automne 2008

IFT 6266: Algorithmes d'apprentissage

Yoshua Bengio, Yoshua.Bengio@umontreal.ca
James Bergstra, bergstrj@iro.umontreal.ca

Devoir #1

Donné le 8 septembre 2008, Dû le 22 septembre 2008

1. Cet exercice est destiné à se familiariser avec la notion d'hyper-paramètre, et revoir la régression linéaire dans un contexte d'apprentissage machine.

   La régression linéaire avec régularisation L2 est un algorithme d'apprentissage où l'optimisation des paramètres peut se faire analytiquement, et l'implantation est plus simple que pour d'autres algorithmes d'apprentissage, si on utilise une librairie existante pour la résolution d'équations linéaires. Il a un unique hyper-paramètre qui contrôle le degré de complexité de la solution, et qui permet de contrôler le compromis entre erreur d'apprentissage et "optimisme" du prédicteur (différence entre l'erreur d'apprentissage et l'erreur de test). L'exercice consiste à implanter et tester l'algorithme, en observant l'effet de l'hyper-paramètre sur plusieurs quantités importantes en apprentissage machine: l'erreur d'apprentissage, l'erreur de test, la différence entre les deux.

   Concepts sous-jacents à l'algorithme de régression L2

   Soit x un vecteur (par convention colonne) dont les d dimensions correspondent à d différentes caractéristiques de chaque exemple. Soit y un nombre qui est la cible que l'on voudrait associer à x, avec une fonction $f(x)$ qui représente notre prédiction de y étant donné x. On voudrait choisir f pour que $f(x)$ soit proche de y pour des paires (x, y) tirées de la distribution inconnue P qui génère nos données. Comme pour la régression linéaire, le coût que l'on veut le plus faible possible sur un nouvel exemple (entrée=x,cible=y) est l'*erreur quadratique* $(f(x) - y)^2$. On va utiliser le prédicteur affine, i.e., de la forme $f(x) = w'x + b$, avec w un vecteur (colonne) de **poids**, de la même dimension d que le vecteur (colonne) x, et b un scalaire appelé **biais**. Les paramètres sont donc les éléments du vecteur $\theta = (b, w_1, w_2, ...w_d)$. L'erreur d'apprentissage totale est

   $$\sum_{(x_t,y_t)\in D_{train}} (f(x_t) - y_t)^2$$

   où D_{train} est l'ensemble d'apprentissage. On peut définir de manière similaire l'erreur totale pour l'ensemble de test D_{test}, ainsi que les erreurs moyennes correspondantes (on divise par

la taille de l'ensemble). Dans la régression linéaire ordinaire, on minimise seulement l'erreur quadratique totale. Dans la régression régularisée, pour choisir θ, on va minimiser l'erreur d'apprentissage totale plus un terme de **régularisation** qui donne une préférence aux solutions telles que les w sont petits. Avec la régularisation L2, on préfère les solutions dont la norme L2 de w est petit. Le terme de régularisation est donc

$$\lambda||w||^2 = \lambda \sum_{i=1}^{d} w_i^2$$

avec λ un scalaire que nous appelons hyper-paramètre et qui contrôle le degré de régularisation L2 que l'on impose à la solution. Notons que $||w||$ est la norme L2 du vecteur w, d'où le nom de régression L2 (aussi appelée 'ridge regression' en anglais). Notre critère d'apprentissage est

$$C = \lambda||w||^2 + \sum_{(x_t,y_t)\in D_{train}} (f(x_t) - y_t)^2$$

On remarquera que C est convexe en θ, ce qui veut dire qu'il y a un seul point critique de C (là où $\frac{\partial C}{\partial \theta}$ s'annulle) et que ce point critique est forcément le minimum global de C. Pour choisir θ on donc va simplement chercher la solution de l'équation

$$\frac{\partial C}{\partial \theta} = 0.$$

Algorithme d'apprentissage pour régression L2

Soit X la matrice qui contient dans chaque ligne un vecteur d'entrée (la partie x d'un exemple, vue comme un vecteur ligne cette fois, et la t-ième rangée de X). Soit Y le vecteur colonne qui contient les valeurs cibles correspondantes (avec la partie y de l'exemple à la position t de Y). Soit $\tilde{X}$ la matrice qui contient des 1 dans sa première colonne et X dans les autres colonnes: la t-ième rangée de la matrice $\tilde{X}$ contient $(1, x_{t1}, x_{t2}, ...x_{td})$), où l'on note x_{ti} le i-ème élément du t-ième exemple.

Exercice théorique (3 points)

Montrez d'abord que $\frac{\partial C}{\partial \theta} = 0$ est équivalent au système d'équations suivant:

$$(\tilde{X}'\tilde{X} + \lambda I_d)\theta = \tilde{X}'Y \tag{1}$$

La matrice I_d est une matrice de dimension $(d+1) \times (d+1)$ contenant seulement des 0 sauf des 1 aux positions (i, i) pour i de 2 à $d+1$ (cette matrice est donc la matrice identité dont l'élément (1,1) est remplacé par 0).

L'algorithme d'apprentissage peut être vu comme une fonction qui prend D_{train} en argument et retourne une fonction de prédiction (sous la forme des paramètres θ). Il consiste donc à construire les matrices $\tilde{X}$, Y, I_d et à résoudre le système d'équations linéaires (1), afin de trouver $\theta = (b, w)$.

Exercice pratique

S'il vous plaît mettez votre code source (dans le langage de votre choix) en appendice de votre devoir, ou bien faites le parvenir électroniquement à *pift6266@iro.umontreal.ca* (une remise électronique du devoir lui-même est aussi préférée).

(a) (4 points) Valider votre code en générant des données aléatoires. Pour cela vous devez choisir des paramètres (b^*, w^*) d'un modèle linéaire générateur. Je vous suggère de prendre $b^* = 0$ et w^* un vecteur dont les éléments sont choisis aléatoirement selon une Normale(0,1). Ensuite vous pouvez utiliser un modèle générateur $y = w^{*\prime}x^* + b^* + bruit$, avec les x choisis selon une Normale(0,1), et le bruit selon Normale(0,σ^2). Je vous suggére σ autour de 0.5 ou 1. Je vous suggère de varier le nombre d'exemples d'apprentissage entre 2 et 20, et de prendre un assez grand nombre d'exemples de test (100). Si vous travaillez en python (avec la librairie numpy et matplotlib) regardez les fonctions `numpy.random.randn` et `matplotlib.plot`.

Commencez avec $\lambda = 0$, c'est à dire aucune régularisation. Vous devriez trouver un b proche de 0 et w proche de w^*. Pour $\lambda = 0$, l'erreur moyenne d'apprentissage devrait être généralement inférieure à σ^2 (et approchant σ^2 au fur et à mesure qu'on entraîne avec plus de données), c'est à dire meilleure que l'erreur de généralisation la plus basse possible (celle du modèle générateur). Qu'arrive-t-il à la différence entre l'erreur d'apprentissage moyenne et σ^2 au fur et à mesure que le nombre d'exemples augmente?

Augmentez la régularisation $\lambda > 0$, et variez encore le nombre d'exemples e.g. $1, 2, 5, 20$. Comment est l'erreur moyenne d'apprentissage reliée à σ^2? Qu'est-ce qui se passe aux erreurs moyenne d'apprentissage et de test quand vous augmentez σ?

Notez que la moyenne sur un petit ensemble de test est plus sujette aux aléas du bruit. Vous pouvez répéter une expérience sur un tirage aléatoire de données différentes, pour en avoir le coeur net.

Notez aussi que la différence entre l'erreur d'apprentissage et l'erreur de test devrait en moyenne (sur différents ensembles d'apprentissage et de test) diminuer au fur et à mesure qu'on entraîne avec plus de données. Vous pouvez aussi vérifier que les deux erreurs augmentent généralement quand on augmente σ.

On peut interpréter λ comme l'importance que l'on accorde à avoir des w petits, donc pour spécifiquer que l'on préfère des w petits. Qu'arrive-t-il si on prends $\lambda < 0$, et qu'est-ce que cela signifie?

(b) (3 points) Vous trouverez à
`http://www.iro.umontreal.ca/~pift6266/A08/documents/polygon_examples` un fichier d'images binaires 10×10 pixels, où chaque image est précédé de sa catégorie (0 pour les triangles et 1 pour les rectangles). Vous pouvez alternativement lire les fichiers
`http://www.iro.umontreal.ca/~pift6266/A08/documents/shapesX.mat`
`http://www.iro.umontreal.ca/~pift6266/A08/documents/shapesY.mat`
pour la matrice X, avec une rangée par image (obtenue en concaténant toutes les rangées

de l'image dans un grand vecteur), et pour le vecteur colonne Y, avec les classes correspondantes. Séparer l'ensemble de données en deux parties D_{train} et D_{test} de la même taille.

i. Préparer une série d'expériences dans lesquelles on va varier λ, et on va regarder l'erreur moyenne de test en fonction de λ. On devrait observer une courbe en U (avec un minimum d'erreur pour un certain choix de $\lambda > 0$). Valeurs suggérées de λ pour votre courbe: $(1e-3, .01, .1, 1, 3, 10, 30, 100, 300, 1000, 10000)$. En général on peut explorer la quantité de régularisation à imposer, λ, par sauts multiplicatifs.

ii. En plus de mettre sur votre graphe l'erreur moyenne de test en fonction de λ, mettez-y l'erreur moyenne d'apprentissage et la différence entre les deux.
Y a-t-il un meilleur choix de λ? Comment pourrait-on le choisir? Y a-t-il des facteurs qui influencent ce choix? Essayez de réfléchir à cette question, en utilisant votre cadre expérimental pour vous aider à y répondre, et faites nous part de votre réflexion.

iii. On remarquera que dans ce problème les cibles prennent des valeurs discrètes (ici, 0 ou 1) correspondant à des classes (triangle ou rectangle, respectivement). On pourrait vouloir utiliser la prédiction numérique $f(x) = w'x + b$ pour produire aussi une réponse catégorique $g(x) \in \{0, 1\}$, qui représente la classe prédite. Par exemple on pourrait prendre $g(x) = 1$ quand $f(x) \geq 0.5$ et 0 autrement, i.e., $g(x) = 1_{f(x) \geq 0.5}$. L'*erreur de classification* pour l'exemple (x, y) est donc 0 si $g(x) = y$, et 1 autrement, i.e. $1_{g(x)=y}$. Refaites vos courbes d'erreur (en fonction de $\log(\lambda)$, pour données d'apprentissage et de test) en mesurant l'erreur de classification plutôt que l'erreur quadratique.