Département d'informatique et de recherche opérationnelle Automne 2008

IFT 6266: Algorithmes d'apprentissage

Yoshua Bengio et James Bergstra, pift6266@iro.umontreal.ca

Devoir #4, Donné le 28 octobre 2008, Dû le 12 novembre 2008

Ce devoir doit être fait complètement en latex. Vous pouvez utiliser l'énoncé source (tp4.tex) comme point de départ pour entrer vos réponses (en gardant l'énoncé des questions).

Ce devoir porte sur la régression à noyau. Comme la régression linéaire, la régression à noyau 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 hyper-paramètre de plus que la régresion linéaire L2, et ces deux hyper-paramètres contrôle le degré de complexité de la solution et 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 des hyper-paramètres et du nombre d'exemples d'apprentissage 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 à noyau

Nous avons pour chaque exemple un vecteur de caractéristiques x et une cible y 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. Plus précisément, le coût que l'on veut le plus faible possible sur un nouvel exemple (entrée=x,cible=y) est l'*erreur quadratique* $(f(x) - y)^2$. Pour obtenir une prédiction non-linéaire on va remplacer x par $\phi(x)$, avec $\phi(x)$ un vecteur de haute dimension (encore notée d), ce qui nous donne

$$f(x) = w'\phi(x)$$

avec w un vecteur (colonne) de **poids** de la même dimension d que le vecteur (colonne) $\phi(x)$ (pour simplifier on a pas de biais b ici). On va ensuite appliquer le **truc du noyau**: on va supposer que l'on connaît une fonction noyau $K(u, v)$ telle que

$$K(u, v) = \phi(u)'\phi(v).$$

On va ensuite réécrire la solution de la régression linéaire L2 en terme de K plutôt qu'en terme de ϕ, qui est intéressant quand la dimension d de $\phi(x)$ est plus grande que le nombre d'exemples d'apprentissage n. Cette fois-ci (revoir la notation du devoir 2) on va définir la matrice $\tilde{X}$ avec $\phi(x_t)'$ dans sa t-ième rangée, plutôt que $(1, x_t')$. Remarquez que dans ce cas la matrice $\tilde{X}'\tilde{X}$ $(d \times d)$ utilisée dans la solution de la régression n'est pas de rang plein (son rang est $n < d$), mais la matrice $\tilde{X}\tilde{X}'$ $(n \times n)$ l'est, et son rang est égal au nombre d'exemples n. On appelle $\tilde{X}\tilde{X}'$ la matrice de **Gram**, notée G ici. On va donc faire un changement de base de façon à passer d'une base avec trop d'éléments (les d dimensions de $\phi(x)$) à une base avec le bon nombre (les n exemples).

1. Vous allez montrez d'abord qu'il existe des α_t tels que l'on peut écrire la solution de la régression sous la forme
$$w = \sum_{t=1}^{n} \alpha_t \phi(x_t). \tag{1}$$
`Indices`: une manière (inefficace mais simple) d'obtenir w, est de faire de la descente de gradient, et on sait que sous certaines conditions sur la séquence des ϵ_τ, cette récursion converge vers la solution de la régression:
$$w^{\tau+1} = w^\tau - \epsilon_\tau \frac{\partial C(w^\tau)}{\partial w^\tau}$$
où w^τ est le vecteur de poids après une étape de l'optimisation par descente de gradient, et $C(w^\tau)$ est la valeur du critère d'apprentissage (erreur quadratique totale sur l'ensemble d'apprentissage, plus $\lambda||w^\tau||^2||$) quand les poids sont w^τ. Remarquez aussi que $\frac{\partial C(w^\tau)}{\partial w^\tau}$ peut s'écrire comme une combinaison linéaire des $\phi(x_t)$.

2. Deuxièmement, vous allez montrer comment utiliser l'eq. 1 pour réécrire la prédiction $f(x)$ sous la forme
$$f(x) = \sum_{t=1}^{n} \alpha_t K(x, x_t). \tag{2}$$

3. Finalement, vous allez réécrire le critère d'apprentissage C en terme d'α plutôt qu'en terme de w. Pour cela il suffit d'écrire et de développer l'énoncé $\frac{\partial C}{\partial \alpha} = 0$. Vous devriez trouver qu'on obtient le système linéaire
$$(G + \lambda I)\alpha = Y. \tag{3}$$
avec I la matrice identité $n \times n$, Y le vecteur colonne dont les éléments sont les cibles y_t, et G la matrice de Gram $n \times n$ avec $G_{s,t} = K(x_s, x_t) = \phi(x_s)'\phi(x_t)$ pour les paires d'exemples d'apprentissage x_s et x_t.

4. **Exercice pratique**

   Dans cet exercice je vous suggère d'utiliser le noyau gaussien
$$K_\gamma(x_s, x_t) = e^{-||x_s - x_t||^2/\gamma^2}$$

qui a un hyper-paramètre γ. Le nombre d'hyper-paramètres est donc 2 plutôt que 1: λ et γ. Pour explorer les valeurs de γ, vous pouvez utiliser le fait qu'il devrait être du même ordre que les distances euclidiennes entre deux exemples x_s et x_t pris au hasard, donc proportionnel à la racine carrée du nombre de dimensions de l'entrée.

(a) Vous allez valider votre code encore un fois en générant des données artificielles provenant d'une fonction f^* fixe, plus du bruit. Je vous suggère d'essayer avec des x en dimension 1 (facile à visualiser), et $f^*(x) = sin(5x)e^{-|2x|}$ la "vraie" fonction que l'apprentissage essaie d'approcher. Générez vos entrées x selon une $Normale(0, 1)$, et les cibles $y = f^*(x) + bruit$ avec du bruit Normale(0,σ^2) (je vous suggère $\sigma = 0.01$, 10 exemples d'apprentissage et 1000 exemples de test, mais vous pouvez jouer avec ces valeurs pour voir l'effet). Vous pouvez alors afficher les points (x, y) (d'apprentissage ou de test), la valeur qu'on aimerait que l'apprentissage découvre, au point $(x, f^*(x))$, et la valeur prédite par l'algorithme d'apprentissage, au point $(x, f(x))$. Remarquer comme la courbe prédite est généralement plus proches des cibles sur les exemples d'apprentissage par rapport aux exemples de test.

Remarquez que vous ne pourrez choisir $\lambda = 0$ car alors le système linéaire n'est pas bien conditionné. Il faut absolument régulariser car le nombre de degrés de liberté est plus grand que le nombre d'exemples.

Si γ et λ sont suffisamment petits, vous devriez encore trouver l'erreur moyenne d'apprentissage généralement inférieure à σ^2 (et approchant σ^2 au fur et à mesure qu'on entraîne avec plus de données). Quel que soit λ, vous devriez trouver une erreur moyenne de test généralement supérieure à σ^2 (notez que la moyenne sur un petit ensemble de test est plus sujette aux aléas du bruit). 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.

(b) Écrivez une fonction A, qui pour un ensemble d'apprentissage donné, choisit **automatiquement** les hyper-paramètres et retourne un objet f (une fonction, si vous utilisez un langage qui peut manipuler des fonctions, comme python) que l'on pourra appliquer sur un ensemble de test. La fonction A connaît une grille de valeurs des hyper-paramètres, et elle choisit un couple de valeurs (λ, γ) qui fonctionne le mieux sur *un ensemble de validation*, obtenu en prenant une fraction des données d'apprentissage (et en gardant le reste pour choisir les α). Une fois que le couple (λ, γ) est choisi, A ré-entraîne avec l'ensemble d'apprentissage complet. Testez cette fonction avec vos données artificielles de la question précédente.

(c) Vous allez tester A avec les données financières similaires à celles du devoir 3 (maintenant dans l'ordre chronologique, voir site du cours), mais en utilisant la validation séquentielle plutôt qu'une simple division train/valid/test. Cela veut dire que l'ordre des données est important. Pour le devoir vous avez 4878 examples. Votre validation séquentielle va entraîner d'abord avec les premiers 2078 exemples, et tester sur les 200

exemples suivants. Elle va ensuite entraîner avec les premiers 2278 exemples, et tester sur les 200 exemples suivants, etc. Jusqu'à entraîner avec tous les exemples sauf les 200 derniers, et tester sur les 200 derniers. On peut maintenant considérer la séquence des 2800 erreurs de test (pour chacun des exemples des 14 blocs de test de 200 chacun) et faire des statistiques dessus. Vous allez comparer la régression à noyau avec un prédicteur inconditionnel, qui prédit au temps t la moyenne des cibles précédentes y_s, $s \leq t$: $f(x_t) = \frac{1}{t} \sum_{s=1}^{t} y_s$.

Pour faire la comparaison, utilisez la formule de la variance des différences d'erreur qui tient compte de l'autocovariance des erreurs dans le temps, tel qu'expliqué à la fin du feuillet sur la sélection de modèle:

$$\text{var} = \frac{1}{T^2} \sum_{t=1}^{T} \sum_{s=1}^{T} \hat{\gamma}_{|t-s|}$$

où la covariance entre une différence d'erreur au temps t et au temps $t - \tau$ est estimée par

$$\text{covar}_\tau = \frac{1}{T - \tau - 1} \sum_{t=\tau+1}^{T} (\hat{e}_t - \bar{e})(\hat{e}_{t-\tau} - \bar{e}).$$

Est-ce que la régression à noyau prédit mieux (hors-échantillon) que le prédicteur inconditionnel (au seuil de 5%)?