Département d'informatique et de recherche opérationnelle Automne 2008

IFT 6266: Algorithmes d'apprentissage

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

Devoir #3, Donné le 6 octobre 2008, Dû le 20 octobre 2008

1. (4 points) Vous allez prouver ici (dans un cas simple où les entrées et les sorties sont réellement linéairement dépendentes), étape par étape, qu'il existe toujours une pénalisation de la norme des paramètres qui améliore la généralisation obtenue par une régression. Nous allons supposer que les données (x, y) sont générées par le modèle de la régression classique:

$$Y = wX + \epsilon$$

où toutes les quantités sont scalaires, w est inconnu, $X = x$ est connu (fixé par l'expérimentateur), et ϵ suit une loi d'espérance 0 et de variance σ^2, et ϵ est indépendent de X. Comme d'habitude on va aussi supposer que tous les exemples (x_t, y_t) $(t = 1 \cdots T)$ sont générés indépendemment de cette manière. Appelons y_1^T la séquence des sorties désirées ainsi générées (c'est la seule quantité considérée aléatoire). On choisit le paramètre $\hat{w}$ qui minimise l'erreur quadratique pénalisée:

$$C = \lambda\, \hat{w}^2 + \sum_{t=1}^{T} 0.5(\hat{w}\, x_t - y_t)^2$$

Dans le développement qui suit, nous allons considérer ce qui se passe quand λ est très près de zero, ce qui nous permet certaines simplifications.

   (a) Écrivez la solution $\hat{w}$ qui minimise C.

   (b) En supposant les entrées fixes (données), écrivez l'espérance et la variance de $\hat{w}$ (la seule quantité aléatoire est $\epsilon_t = y_t - w\, x_t$).

   (c) Nous avons vu que l'erreur de généralisation totale est la somme de trois termes: bruit, biais au carré, et variance. Le premier ne dépend pas du choix de $\hat{w}$. Écrivez une expression pour le biais au carré (étant donné un x de test fixe: $(E_{y_1^T}[\hat{w}\, x - w\, x])^2$; et une expression pour la variance: $E[(\hat{w}\, x - w\, x)^2]$ qui est approximativement la même chose que $Var[\hat{w}\, x]$ (cette dernière quantité nous suffira pour l'argument qui suit).

   (d) Calculez la dérivée du biais au carré par rapport à λ ainsi que la dérivée de $Var[\hat{w}\, x]$ par rapport à λ.

(e) Comparez ces deux dérivées: la première est positive (le biais augmente quand on augmente λ) et la seconde est négative (la variance diminue quand on augmente λ). Montrez que si $\sigma > 0$ alors il existe une valeur positive de λ qui donne une meilleure généralisation que $\lambda = 0$. Notez que plus w (le vrai poid) est petit et plus la variance descend vite par rapport à l'augmentation du biais.

2. (6 points) Vous allez implanter l'algorithme d'apprentissage pour un réseau de neurones à une couche cachée similaire à ce qui a été décrit en classe, pour un problème de régression. Vous allez aussi comparer ce que cet algorithme donne avec les prédiction d'une fenêtre de Parzen. Pour comparer les deux algorithmes, vous allez considérer une division de l'ensemble de données en 3 parties: apprentissage (pour estimer les paramètres, ou placer les centres des Gaussiennes de la fenêtre de Parzen), validation (pour choisir les hyper-paramètres), et test (pour comparer les deux algorithmes). Dans tous les cas on va mesurer l'erreur quadratique de prédiction $L(f(x), y) = (f(x) - y)^2$.

   Les données seront fournies sur le site web du cours.

   Parzen

   Le prédicteur pour la régression de Parzen est très simple:

$$f_{D,\sigma}(x) = \frac{\sum_{t=1}^{n} y_t K_\sigma(x, x_t)}{\sum_{t=1}^{n} K_\sigma(x, x_t)}$$

   avec

$$K_\sigma(u, v) = e^{-0.5||u-v||^2/\sigma^2}$$

   et un unique hyper-paramètre σ. Géométriquement, ce modèle de régression correspond à des "bosses" Gaussiennes dans l'espace $\mathcal{X}$ avec $x \in \mathcal{X}$. Le valeur $y_t \in \mathcal{Y}$ est la hauteur de chaque bosse, et σ est sa largeur.

   Réseau de Neurones

   Le prédicteur par réseau de neurones a la forme suivante:

$$f_\theta(x) = c + \sum_{i=1}^{h} w_i \tanh(b_i + \sum_{j=1}^{d} v_{ij} x_j)$$

   avec paramètres $\theta = (c, w, b, v)$. On va le régulariser avec la pénalité L1, comme dans le devoir 2, avec une pénalité

$$R(\theta) = \lambda \sum_i (|b_i| + \sum_j |v_{ij}|)$$

   pour chaque exemple. Le coût par exemple est donc $C(x, y, \theta) = L(f_\theta(x), y) + R(\theta)$.

- L'entraînement de ce modele est fait par la descente de gradient stochastique (voir devoir 2). Itérativement, après chaque exemple, on prend un pas de gradient de taille ϵ qui va dans la direction $-\frac{\partial C(x,y,\theta)}{\partial \theta}$. Vous pouvez peut-être ré-utiliser la structure de votre code du devoir 2.
- Vous allez utiliser la méthode de l'arrêt prématuré pour choisir le nombre d'itérations d'apprentissage (on fait un grand nombre d'itérations de descente de gradient mais on garde le θ qui fonctionne le mieux sur l'ensemble de validation). Quand ϵ est petit, θ ne change pas rapidement, et alors ça ne sert a rien de l'évaluer après chaque mise à jour (et surtout, c'est tres coûteux en calculs). Donc on vous conseille d'évaluer θ (en mesurant les erreurs d'apprentissage et de validation) seulement après chaque passe à travers l'ensemble d'entrainement.

Que faut-il faire?

Pour cette question, vous allez comparez ces deux algorithmes: *Parzen windows* et *Réseau de neurones à une couche cachée*. Lequel est meilleur pour les données fournies? Pour y répondre, que faut-il faire?

(a) (Avec les données d'apprentissage) Démontrez que votre implantation de Parzen Windows est correcte. Ca vous prendra un petit paragraphe et une courbe de l'erreur d'apprentissage vs. σ. (Indice, ca ne devrait pas être une courbe en U!)

(b) (Avec les données d'apprentissage) Démontrez que votre implantation de réseau de neurones est correcte.

- Écrire un pseudo-code pour le calcul de $f(x)$ et $L(f(x), y)$ et $C(x, y, \theta)$, étant donné θ, x, y.
- Écrire les équations et le pseudo-code pour le calcul de $\frac{\partial C(x,y,\theta)}{\partial \theta}$.

Les hyper-paramètres sont maintenant nombreux: h, λ, ϵ (et implicitement le nombre d'itérations d'apprentissage, auto-sélectionné par l'arrêt prématuré). Montrez que pour quelques valeurs d'hyperparametres, le critère C est bien optimisé (C ne devrait pas remonter d'une passe à l'autre!), et que l'erreur d'apprentissage se comporte de façon raisonnable pendant que vous optimisez C (elle descend!) (Indice: une courbe de l'erreur moyenne en fonction du nombre d'iterations aiderait ici.) Ultimement on s'intéresse à l'erreur de généralisation, alors il n'est pas important que C soit vraiment minimisé (il n'est pas nécessaire d'atteindre le minimum, avec $\frac{\delta C}{\delta \theta} = 0$); si on voit que l'erreur d'apprentissage ne descend plus, ce n'est pas nécessaire de continuer. Appliquez votre jugement ici.

(c) (En ajoutant les données de validation) Démontrez que vous avez bien encapsulé les algorithmes de Parzen et réseau de neurones comme algorithmes d'apprentissage– c'est a dire, comme procédures qui retourne des fonctions ($\mathcal{X} \rightarrow \mathcal{Y}$) étant donné des données d'apprentissage et des hyperparamètres.

Pour l'algo de Parzen, discutez et montrez le relation entre l'erreur de validation et σ. Quel est la meilleur σ?

Pour l'algo RdN, expliquez comment vous optimisez les trois hyper-paramètres. Essayez de trouver une valeur des trois hyper-paramètres qui semble approximativement au minimum de l'erreur de validation. Étant donné la longueur des calculs, un choix approximatif est suffisant (e.g. multiplier par 3 ou diviser par 3 l'hyper-paramètre n'améliore pas l'erreur). Vérifiez que pour chacun des trois hyper-paramètres vous avez des courbes en U de l'erreur de validation autour de cette valeur optimale. Il se peut que vos courbes soient 'bruitées'. Comme d'habitude vous pouvez obtenir des courbes plus lisses en faisant la moyenne sur plusieurs simulations (ici il suffit de considérer des seeds différents pour l'initialisation aléatoire, mais on pourrait aussi refaire l'expérience avec des sous-ensemble différents des données).

Voilà! Quand on inclus la procédure pour sélectionner les hyperparamètres, on a deux algorithmes d'apprentissage qui prennent des données (apprentissage + validation) et sortent une fonction $\mathcal{X} \to \mathcal{Y}$. On peut finalement les comparer directement.

(d) (En ajoutant les données de test) Vous pouvez maintenant utiliser l'ensemble de test pour comparer les deux algorithmes. Vous utilisez le modèle obtenu pour le meilleur choix d'hyper-paramètre, pour chacun des deux algorithmes, et vous mesurez leur erreur moyenne de test. Pour estimer l'incertitude sur la différence μ entre ces deux moyennes, vous allez calculer l'erreur-type de cette différence, c'est à dire l'écart-type de la moyenne, avec la formule suivante:

$$s = \sqrt{\frac{1}{(n-1)n}\sum_{t=1}^{n}(A_t - B_t - \mu)^2}.$$

avec A_t et B_t l'erreur quadratique de l'algorithme A et de l'algorithme B, respectivement, pour l'exemple de test t. On considère traditionnellement que la différence est significative si $|\mu| > 2\sigma$. Essayez donc de conclure! Souvent les différences entre deux algorithmes ne sont pas significatives, n'ayez pas peur de conclure ce que vos résultats vous affirment!