IFT3395/6390 Fondements de l'apprentissage machine

(Automne 2011)

Professeur : Pascal Vincent

Devoir 2

– On préférera, dans la mesure du possible, la remise d'un rapport en format électronique (`rapport.pdf`). Pour un rapport avec des équations correctement typographiées, vous pouvez par exemple utiliser le logiciel Lyx. Tous les fichiers de code source que vous aurez créé ou adapté, et les fichiers de résultat produits (ex. courbes) devront également être remis. **Veillez à clairement écrire votre nom dans votre rapport et dans tous les fichiers remis.**
– La remise se fait via le site web du cours sur le système STUDIUM.

1 Régression linéaire et non linéaire régularisée (50 pts)

1.1 Régression linéaire

Soit un problème de régression pour lequel on dispose d'un ensemble de données d'entraînement D_n comportant n exemples sous forme (entrée, cible) :

$$D_n = \{(\mathbf{x}^{(1)}, t^{(1)}), \ldots, (\mathbf{x}^{(n)}, t^{(n)}\}$$

Avec $\mathbf{x}^{(i)} \in \mathbb{R}^d$, et $t^{(i)} \in \mathbb{R}$

Le modèle de régression linéaire suppose une forme paramétrée pour une fonction f qui va prédir la valeur de la cible pour un nouveu point $\mathbf{x}$ (plus précisément elle va chercher à prédire l'espérance conditionnelle de la variable cible, conditionnelle à l'observation $\mathbf{x}$) : $f(\mathbf{x}) \simeq \mathbb{E}[\mathbf{t}|\mathbf{x}]$.

La forme paramétrée consiste en une transformation linéaire (ou plus précisément *affine*) de $\mathbf{x}$:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

1. Indiquez quel est l'ensemble θ des paramètres de ce modèle, ainsi que la nature et la dimensionalité de chacun.
2. La fonction de perte (ou de coût) habituellement utilisée pour la régression linéaire est l'erreur quadratique :

$$L((\mathbf{x},t),f) = (f(\mathbf{x})-t)^2$$

   On définit ici le **risque empirique** $\hat{R}$ sur l'ensemble D_n comme étant la **somme** des pertes sur l'ensemble D_n (plutôt que la moyenne des pertes comme on le définit parfois). Donnez précisément l'expression mathématique de ce risque empirique.
3. Selon le principe de minimisation du risque empirique, on va chercher la valeur des paramètres qui donne le moins d'erreur sur l'ensemble d'entraînement, donc qui minimise le risque empirique. Exprimez ce problème de minimisation.
4. Une manière générique pour tenter de résoudre ce problème d'optimisation est par une technique de descente de gradient. Exprimez le gradient du risque empirique.

1.2 Régression linéaire régularisée ("ridge regression")

Nous allons maintenant considérer, plutôt que $\hat{R}$, un **risque empirique régularisé** : $\tilde{R} = \hat{R} + \lambda \mathscr{L}(\theta)$. Ici $\mathscr{L}$ calcule une pénalité scalaire en fonction des paramètres, plus ou moins importante selon une préférence à priori qu'on veut encourager sur les valeurs des paramètres. λ est un **hyper-paramètre** (scalaire, positif ou nul) qui contrôle le compromis entre trouver des valeurs des paramètres qui minimisent le risque empirique ou qui minimisent cette pénalité. Remarquez qu'on retombe sur le risque empirique ordinaire (non régularisé) quand $\lambda = 0$. On va considérer ici une régularization de type "ridge" ou "weight decay", quadratique qui pénalise la norme carrée (norme L2) des poids (mais pas le biais) : $L(\theta) = \|\mathbf{w}\|^2 = \sum_{k=1}^{d} \mathbf{w}_k^2$. On veut en fait minimiser le risque régularisé $\tilde{R}$ plutôt que $\hat{R}$.

1. Exprimez le gradient du risque régularisé. En quoi diffère-t-il du gradient du risque empirique non régularisé ?
2. Écrivez le pseudo-code détaillé de l'algorithme d'entraînement qui cherchera les paramètres optimaux qui minimisent le risque empirique régularisé $\tilde{R}$ par descente de gradient **batch.** Pour simplifier les choses, on utilisera un pas de gradient η fixe.

Pour le cas particulier de la régression linéaire (régularisée ou non), il se trouve qu'il existe une solution analytique à ce problème de minimisation. Cette solution est donnée par la formule matricielle suivante :

$$\begin{pmatrix} b^* \\ \mathbf{w}^* \end{pmatrix} = \left(\check{X}^T\check{X} + \lambda\check{I}\right)^{-1}\check{X}^T\mathbf{t} \tag{1}$$

où $\check{X} = \begin{pmatrix} 1 & \mathbf{x}_1^{(1)} & \dots & \mathbf{x}_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \mathbf{x}_1^{(n)} & \dots & \mathbf{x}_d^{(n)} \end{pmatrix}$, $\mathbf{t} = \begin{pmatrix} t^{(1)} \\ \vdots \\ t^{(n)} \end{pmatrix}$, le ^{T} indique la transposée et le $^{-1}$ indique l'inverse matriciel,
et $\check{I}$ est une matrice $(1+d)\times(1+d)$ telle que $\check{I}_{1,1} = 0$, $\check{I}_{i,i} = 1$ pour $i \geq 2$, et $\check{I}_{i,j} = 0$ pour $i \neq j$. Autrement dit, ajouter $\lambda\check{I}$ à une matrice revient à ajouter λ à chacun des éléments sur sa diagonale à l'exception de l'élément à la position $1,1$ (le coin en haut à gauche).

1.3 Régression avec un pré-traitement non-linéaire fixe

On peut construire un algorithme de régression non linéaire en utilisant une non-linéarité fixe : une fonction $\phi(\mathbf{x})$ qui transforme $\mathbf{x}$ de manière non-linéaire en un $\tilde{\mathbf{x}}$ de plus haute dimension.

Par ex. si on est en dimension 1 : $x \in \mathbb{R}$, on peut considérer la transformation polynômiale :

$$\tilde{x} = \phi_{\text{poly}^k}(x) = \begin{pmatrix} x \\ x^2 \\ \vdots \\ x^k \end{pmatrix}$$

On peut alors "entraîner" un régresseur non pas sur les $(x^{(i)}, t^{(i)})$ de l'ensemble de donnée d'entraînement d'origine D_n mais sur un ensemble transformé , les $(\phi(x^{(i)}), t^{(i)})$. Cet entraînement trouve les paramètres d'une fonction affine f.

La prédiction pour un point test x est alors obtenue non pas par $f(x)$ mais par $\widetilde{f}(x) = f(\phi(x))$.

1. Écrivez de manière détaillée la forme paramétrique qu'on obtient pour $\tilde{f}(x)$ dans le cas une dimension $(x \in \mathbb{R})$ si on utilise $\phi = \phi_{\text{poly}^k}$
2. Précisez quels sont les paramètres et leur dimensionalité.

3. En dimension $d \geq 2$ une transformation polynômiale devrait aussi contenir, en plus des exposants d'ordre $\leq k$ des composantes individuelles de x, tous les termes d'interaction d'ordre k et moins entre plusieurs composantes de l'entrée. Pour $d = 2$, écrivez explicitement en fonction des 2 composantes de x, les transformations $\phi_{\text{poly}^1}(x)$, $\phi_{\text{poly}^2}(x)$, $\phi_{\text{poly}^3}(x)$, et $\phi_{\text{poly}^4}(x)$.
4. Quelle sera la dimensionalité de $\phi_{\text{poly}^k}(x)$, en fonction de d et k.

2 Partie pratique (50 pts)

Vous devez remettre les fichiers python ayant servi à produire vos résultats, incluant un programme principal qui produit les courbes demandées, les unes après les autres, avec une pause entre chaque. On doit pouvoir reproduire vos résultats ! Indiquez dans votre rapport comment exécuter votre programme.

1. Implémentez en python l'algorithme de régression linéaire régularisée, par descente de gradient batch. Nommons cet algorithme `regression_gradient`. Notez qu'à la différence de ces précédents algorithmes, nous avons ici à la fois des paramètres à entraîner ($\mathbf{w}$ et b) sur l'ensemble d'entraînement, et un *hyper*-paramètre de contrôle de capacité : λ, ainsi que des hyper-paramètres pour l'optimisation : le pas de gradient η et possiblement le nombre d'itérations. Indication : ceci est très proche de ce que vous avez fait lors du TP7 sur "classifieurs et descente de gradient" avec un coût quadratique, vous pouvez donc partir du même code.
2. Implémentez en python l'algorithme de régression linéaire régularisée en utilisant la solution analytique Eq. 1 donnée ci-dessus. Appelons cet algorithme `regression_analytique`.
3. Soit la fonction $h(x) = \sin(x) + 0.3x - 1$. Générez un ensemble de données D_n constitué de paires $(x, h(x))$ comportant $n = 15$ points où x est tiré alétoirement de manière uniforme dans l'intervalle $[-5, 5]$. Assurez-vous d'utiliser le **même** ensemble D_n pour **tous** les graphiques ci-dessous.
4. Avec $\lambda = 0$, produisez un graphique (avec une légende claire) sur lequel vous afficherez dans l'intervalle $[-10, 10]$: les points de l'ensemble D_n, la courbe $h(x)$, la prédiction obtenue sur l'intervalle avec le modèle de régression linéaire trouvé par la solution analytique (l'algorithme `regression_analytique`), et dans une couleur différente la solution obtenue par descente de gradient (avec l'algorithme `regression_gradient`). **Remarque :** en rincipe la solution par descente de gradient devrait converger vers la solution analytique : apprêtez-vous à devoir ajuster votre pas de gradient (suffisamment

petit) et le nombre d'itérations (suffisamment grand). Dans votre rapport indiquez les valeurs numériques des paramètres trouvés par les 2 algorithmes.

5. Produisez un graphique similaire pour une valeur intermédiaire de λ et un autre pour une valeur extrême de λ (vos graphiques devraient indiquer dans leur titre la valeur de λ utilisée). Ils devraient illustrer qualitativement ce qui se passe lorsqu'on augmente λ. Dans votre rapport indiquez les valeurs numériques des paramètres trouvées par les 2 algorithmes pour chaque cas.
6. Employez la technique étudiée en 1.3 ci-dessus (déjà employée lors des démos) pour utiliser l'algorithme de régression linéaire régularisée pour obtenir une régression non-linéaire. Spécifiquement, effectuez le pré-traitement non-linéaire fixe ϕ_{poly^k} indiqué ci-dessus afin d'obtenir une régression polynômiale d'ordre k. Appliquez cette technique avec $\lambda = 0$, pour différentes valeurs de k, et générez les graphiques correspondants à chaque cas que vous joindrez les à votre rapport. Dans votre rapport indiquez les valeurs numériques des paramètres trouvées par les 2 algorithmes (analytique et descente de gradient) pour chaque cas. Commentez le phénomène observé quand on augmente k. Parlez notamment de l'évolution du risque empirique (erreur sur l'ensemble d'apprentissage) et du risque espéré (erreur de généralisation sur de nouveaux points).
7. Refaites les mêmes expériences qu'à la question précédente, mais pour chaque valeur de k considérée, faites varier λ. Pour cette question, vous pouvez si vous voulez vous contenter d'utiliser l'algorithme `regression_analytique`. Commentez l'influence de l'hyper-paramètre λ que vous observez, et illustrez-le par des graphiques représentatifs.
8. Appliquez votre algorithme de `regression_analytique` sur le problèmes de classification 2D "ellipse.txt" vu lors du TP7 avec les prétraitements 2D $\phi_{\text{poly}^1}(x)$, $\phi_{\text{poly}^2}(x)$, $\phi_{\text{poly}^3}(x)$, et $\phi_{\text{poly}^4}(x)$ de la question 1.3. Pour chacun de ces prétraitements, joignez deux graphiques de région de décision obtenus, l'un avec $\lambda = 0$ et l'autre avec une valeur plus intéressante de λ qui illustre l'effet de la régularisation.