IFT6390 Fondements de l'apprentissage machine

Professeur: Ioannis Mitliagkas

Devoir 2

- Ce devoir est à faire en équipe de 2 personnes. Assurez-vous d'avoir noté le nom de tous les coéquipiers en tête du rapport et en commentaires en tête de chacun des fichiers que vous remettrez.
- On demande, la remise d'un rapport en format électronique (.pdf). Tous les fichiers de code source que vous aurez créé ou adapté devront également être remis. La partie pratique est à faire en python (en utilisant les librairies numpy et matplotlib), et vous pouvez bien entendu fortement vous inspirer de ce qui a été fait pendant les labos.
- Vous pouvez remettre votre code python sous la forme d'un notebook ipython .ipynb. Pour produire un rapport avec des formules mathématiques vous pouvez utiliser le logiciel de votre choix: LaTeX; LyX; Word; voire même écrire directement les parties théoriques dans le notebook (en entrant les équations en format MathJaX pour qu'elles s'affichent). Dans tous les cas on vous demande d'exporter votre rapport en .pdf que vous remettrez.
- La remise doit se fait via StudiUM. Une seule remise par équipe (un seul des coéquipiers effectue la remise). Assurez-vous d'avoir noté le nom de tous les coéquipiers en tête du rapport. Si vous avez beaucoup de fichiers à remettre vous pouvez aussi (c'est peut-être plus pratique) en faire une archive (.zip ou .tar.gz) et téléverser le fichier d'archive.

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 prédire la valeur de la cible pour un nouveau point $\mathbf{x}$ (plus précisément elle va chercher à prédire l'espérance conditionnelle de la variable cible, conditionnelle à l'observation $\mathbf{x}$): $f(\mathbf{x}) \simeq \mathbb{E}[t|\mathbf{x}]$.

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.

5. On définit l'erreur du modèle sur un point d'entraînement (x, t) par $f(\mathbf{x}) - t$. Expliquez en Français la relation entre le gradient du risque empirique et les erreurs du modèle sur l'ensemble d'entraînement.

1.2 Régression linéaire régularisée ("ridge regression")

Nous allons maintenant considérer, plutôt que $\hat{R}$, un **risque empirique régularisé**: $\tilde{R} = \hat{R} + \lambda\mathcal{L}(\theta)$. Ici $\mathcal{L}$ 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.

3. Pour la régression linéaire (régularisée ou non), il se trouve qu'il existe une solution analytique à ce problème de minimisation ! En considérant que le terme de biais est nul (c.à.d. $b = 0$), exprimer le risque empirique et son gradient sous forme matricielle en fonction de la matrice $\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^{(1)} & \dots & \mathbf{x}_d^{(1)} \\ \vdots & \ddots & \vdots \\ \mathbf{x}_1^{(n)} & \dots & \mathbf{x}_d^{(n)} \end{pmatrix}$ et du vecteur $\mathbf{t} = \begin{pmatrix} t^{(1)} \\ \vdots \\ t^{(n)} \end{pmatrix}$.

4. Dérivez la solution analytique du problème de minimisation de la régression linéaire sous forme matricielle en mettant le gradient obtenu précédemment à zéro et en résolvant l'équation. Que se passe-t-il lorsque $N < d$ et $\lambda = 0$?

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 (ex. termes comme $x_i^{j_1} x_l^{j_2}$, pour $j_1 + j_2 \leq k$ et variables $i, l \leq d$). Pour $d = 2$, écrivez explicitement en fonction des 2 composantes de x, les transformations $\phi_{\text{poly}^1}(x)$, $\phi_{\text{poly}^2}(x)$, et $\phi_{\text{poly}^3}(x)$.
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. Ce programme peut prendre la forme d'un Jupyter Notebook (recommandé). Dans tous les cas, 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 que 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.

2. 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.

3. 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 $h(x)$, et la courbe de la fonction apprise avec le modèle de régression linéaire trouvée par descente de gradient (avec l'algorithme `regression_gradient`). **Remarque:** en principe la solution par descente de gradient devrait converger vers la droite passant au plus proche des 15 points (et vers la solution analytique): apprêtez-vous à devoir ajuster votre pas de gradient (suffisamment petit) et le nombre d'itérations (suffisamment grand).

4. Sur le même graphique, ajoutez les fonctions de prédiction trouvées pour une valeur intermédiaire de λ et pour une valeur extrême de λ. Précisez la valeur de λ utilisée dans la légende. Votre graphique devrait illustrer qualitativement ce qui se passe lorsqu'on augmente λ.

5. Générez un second ensemble de données D_{test} de 100 points en suivant la même procédure que pour D_n. Entraînez votre modèle linéaire par descente de gradient sur D_n avec les diffréentes valeurs de λ suivantes: $[0.0001, 0.001, 0.01, 0.1, 1, 10, 100]$. Pour chaque valeur de λ, mesurez l'erreur quadratique moyenne de votre modèle sur D_{test}. Produisez un graphique avec λ en abscisse et ces erreurs en ordonnée.

6. Employez la technique étudiée en 1.3 ci-dessus pour utiliser l'algorithme de régression linéaire régularisée pour obtenir une régression non-linéaire. Spécifiquement, effectuez le pré-traitement non-linéaire fixe ϕ_{poly^l} indiqué ci-dessus afin d'obtenir une régression polynômiale d'ordre l. Appliquez cette technique avec $\lambda = 0.01$, pour différentes valeurs de l. Générez un graphiques similaire à celui de la question 2.2 où figure chacune des fonctions de prédiction. Précisez les valeurs de l en légende. Représentez un nombre de fonctions approprié pour obtenir un graphique à la fois intéressant et lisible.

7. Commentez le phénomène observé quand on augmente l. Parlez notamment de l'évolution du risque empirique (erreur sur l'ensemble d'apprentissage D_n) et du vrai risque (erreur de généralisation sur de nouveaux points D_{test}).