IFT-3395/6390 Fundamentals of Machine Learning
Professors: Guillaume Rabusseau / Ioannis Mitliagkas

Homework 2 - Practical component

Devoir 2 - Partie pratique

- This homework must be done and submitted to Gradescope in teams. You are welcome to discuss with students outside of your group but the solution submitted by a group must be its own. Note that we will use Gradescope's plagiarism detection feature. All suspected cases of plagiarism will be recorded and shared with university officials for further handling.
  Ce devoir doit être déposé sur Gradescope et peut être fait en équipe. Vous pouvez discuter avec des étudiants d'autres groupes mais les réponses soumises par le groupe doivent être originales. À noter que nous utiliserons l'outil de détection de plagiat de Gradescope. Tous les cas suspectés de plagiat seront enregistrés et transmis à l'Université pour vérification.

- The practical part should be coded in python (the only external libraries you can use are numpy and matplotlib) and all code will be submitted as a python file to Gradescope. To enable automated code grading you should work off of the template file given in this homework folder. Do not modify the name of the file or any of the function signatures of the template file or the code grading will not work for you. You may, of course, add new functions and any regular python imports.
  La partie pratique doit être codée en python (avec les librairies numpy et matplotlib), et envoyée sur Gradescope sous la forme d'un fichier python. Pour permettre l'évaluation automatique, vous devez travailler directement sur le modèle donné dans le répertoire de ce devoir. Ne modifiez pas le nom du fichier ou aucune des fonctions signatures, sinon l'évaluation automatique ne fonctionnera pas. Vous pouvez bien sûr ajouter de nouvelles fonctions et importations python

- Any graphing, charts, derivations, or other practical report parts should be submitted to Gradescope included at the end of your report for the theoretical part of the homework.

You are of course encouraged to draw inspiration from what was done in lab sessions.
Les figures, courbes et réponses aux questions (autre que du code) doivent être déposées sur Gradescope en les incluant dans le pdf que vous soumettez pour la partie théorique. Vous êtes bien sûr encouragés à vous inspirer de ce qui a été fait en TP.

One-versus-all, L2 loss SVM

This part consists of the implementation of a one-versus-all, L2 loss SVM, which is commonly used for multiclass classification. The L2 SVM loss is differentiable and imposes a bigger penalty on points that violate the margin. In the one-versus-all (OVA) approach, we train m binary classifiers, one for each class and during inference time, we select the class which classifies the test data with maximum margin.

Given a training set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^p$, $y_i \in \{1, .., m\}$ where p is the number of features and m is the number of classes, we would like to minimize the following objective function:

$$\frac{1}{2}\sum_{j'=1}^{m} \|\mathbf{w}^{j'}\|_2^2 + \frac{C}{n}\sum_{(\mathbf{x}_i,y_i)\in S}\sum_{j'=1}^{m} l(\mathbf{w}^{j'}; (\mathbf{x}_i, y_i))^2$$

where

$$l(\mathbf{w}^{j'}; (\mathbf{x}_i, y_i)) = \max\{0, 1 - (\langle \mathbf{w}^{j'}, \mathbf{x}_i\rangle)\mathbb{1}\{y_i = j'\}\}$$

and

$$\mathbb{1}\{y_i = j'\} = \begin{cases} 1 & \text{if } y_i = j' \\ -1 & \text{if } y_i \neq j' \end{cases}$$

In order to update the parameters $\mathbf{w}$ of the SVM, gradient descent techniques are used. (Note: in the dataset provided with this assignment, the last element of each row is a dummy element with the value 1. That means there is no separate bias parameter b; it is implicitly included in $\mathbf{w}$ as the weight for the dummy element.)

The training set for this part can be downloaded from https://www.dropbox.com/s/nf8zriqjgw02m1a/hw2-cifar-dataset.zip?dl=0.

Cette partie consiste à implémenter le SVM un-contre-tous avec pénalité L2, qui est couramment utilisé pour faire de la classification multi-classe.

La fonction de perte d'un SVM avec pénalité L2 est différentiable et impose une plus grande pénalité sur les points qui sont à l'intérieur de la marge maximale. Dans l'approche un-contre-tous (*one-versus-all* ou *OVA* en anglais), nous entraînons m classificateurs binaires, soit un pour chaque classe, et lors de la prédiction, nous sélectionnons la classe qui maximise la marge pour un point test.

Considérant un jeu d'entraînement $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, où $\mathbf{x}_i \in \mathbb{R}^p$, $y_i \in \{1, .., m\}$, p est le nombre de traits (ou attributs) et m est le nombre de classes, nous voulons minimiser la fonction objectif suivante:

$$\frac{1}{2}\sum_{j'=1}^{m} \|\mathbf{w}^{j'}\|_2^2 + \frac{C}{n}\sum_{(\mathbf{x}_i, y_i)\in S}\sum_{j'=1}^{m} l(\mathbf{w}^{j'}; (\mathbf{x}_i, y_i))^2$$

où

$$l(\mathbf{w}^{j'}; (\mathbf{x}_i, y_i)) = \max\{0, 1 - (\langle \mathbf{w}^{j'}, \mathbf{x}_i\rangle)\mathbb{1}\{y_i = j'\}\}$$

et

$$\mathbb{1}\{y_i = j'\} = \begin{cases} 1 & \text{if } y_i = j' \\ -1 & \text{if } y_i \neq j' \end{cases}$$

Afin de mettre à jour les paramètres $\mathbf{w}$ de notre fonction objectif, nous utiliserons des techniques de descente de gradient. (Note: Vous remarquerez que dans le jeu de données fourni pour cette partie, le dernier élément de chaque ligne est un 1. Cette notation permet de ne pas avoir de paramètre de biais b séparé, mais plutôt de l'inclure implicitement dans $\mathbf{w}$ comme étant un autre poids.)

Ce jeu de données peut être téléchargé à partir du lien suivant: `https://www.dropbox.com/s/nf8zriqjgw02m1a/hw2-cifar-dataset.zip?dl=0`.

1. Undergraduates 5 pts Graduates 5 pts
   Question. What is the derivative of the regularization term

   $$\frac{1}{2}\sum_{j'=1}^{m} \|\mathbf{w}^{j'}\|_2^2$$

   with respect to w_k^j (the kth weight of the weight vector for the jth class)? Show all your work and write your answer in the report.

Quelle est la dérivée du terme de régularisation de la fonction de perte

$$\frac{1}{2}\sum_{j'=1}^{m}\|\mathbf{w}^{j'}\|_2^2$$

par rapport à w_k^j? (le $k^{ième}$ poids du vecteur de poids pour la $j^{ième}$ classe)? Écrivez tous les étapes et mettez la réponse dans votre fichier PDF.

2. Undergraduates 10 pts Graduates 10 pts

Question. What is the derivative of the hinge loss term

$$\frac{C}{n}\sum_{(\mathbf{x}_i,y_i)\in S}\sum_{j'=1}^{m} l(\mathbf{w}^{j'};(\mathbf{x}_i,y_i))^2$$

with respect to w_k^j?

Express your answer in terms of $\mathbf{x}_{i,k}$ (the kth entry of the ith training example $\mathbf{x}_i$).

Assume that

$$\frac{\partial}{\partial a}\max\{0,a\} = \begin{cases} 1 & \text{if } a > 0 \\ 0 & \text{if } a \leq 0 \end{cases}$$

(This is not exactly true: at $a = 0$, the derivative is undefined. However, for this problem, it's OK to make this assumption.)

Quelle est la dérivée du terme appelé *hinge loss*,

$$\frac{C}{n}\sum_{(\mathbf{x}_i,y_i)\in S}\sum_{j'=1}^{m} l(\mathbf{w}^{j'};(\mathbf{x}_i,y_i))^2,$$

de la fonction de perte par rapport à w_k^j?

Exprimez votre réponse en termes de $\mathbf{x}_{i,k}$ (la $k^{ième}$ entrée du $i^{ième}$ exemple $\mathbf{x}_i$).

Assumez que

$$\frac{\partial}{\partial a}\max\{0,a\} = \begin{cases} 1 & \text{if } a > 0 \\ 0 & \text{if } a \leq 0 \end{cases}$$

(Cette dernière affirmation n'est pas exactement vraie: à a=0, la dérivée n'est pas définie. Cependant, pour ce problème, nous allons assumer qu'elle est correcte.)

3. Undergraduates 30 pts Graduates 30 pts

Question. Fill in the following in the code:
Complétez les méthodes suivantes dans le code:

(a) Undergraduates 5 pts Graduates 5 pts

SVM.make_one_versus_all_labels: Given an array of integer labels and the number of classes m, this function should create a 2-d array corresponding to the $\mathbb{1}\{y_i = j'\}$ term above. In this array, each row is filled with -1, except for the entry corresponding to the correct label, which should have the entry 1. For example, if the array of labels is $[1, 0, 2]$ and $m = 4$, this function would return the following array: $[[-1, 1, -1, -1], [1, -1, -1, -1], [-1, -1, 1, -1]]$. The inputs are y (a numpy array of shape (number of labels,)) and m (an integer representing the number of classes), and the output should be a numpy array of shape (number of labels, m). For this homework, m will be 10, but you should write this function to work for any $m > 2$.

SVM.make_one_versus_all_labels: Étant donné un tableau d'étiquettes qui sont des entiers et le nombre de classes m, cette fonction devrait retourner un tableau 2-d qui correspond au terme $\mathbb{1}\{y_i = j'\}$ défini plus haut. Dans ce tableau, chaque ligne contient des -1 à l'exception de l'élément qui correspond à la bonne classe et qui devrait être un 1. Par exemple, si le tableau que l'on donne en entrée est $[1, 0, 2]$ et que $m = 4$, la fonction retournera le tableau suivant:$[[-1, 1, -1, -1], [1, -1, -1, -1], [-1, -1, 1, -1]]$. Les entrées de la fonction sont y (un tableau numpy de dimension (nombre de classes,)) et m (un entier représentant le nombre de classes), et la sortie devrait être un tableau numpy de dimension (nombre d'exemples, m). Pour ce devoir, m sera égal à 10, mais vous devriez implémenter cette fonction pour qu'elle puisse fonctionner avec n'importe quel $m > 2$.

(b) Undergraduates 5 pts Graduates 5 pts

SVM.compute_loss: Given a minibatch of examples, this function should compute the loss function. The inputs are x (a numpy array of shape (minibatch size, 401)), y (a numpy array of shape (minibatch size, 10)), and the output should be the computed loss, a single float.

SVM.compute_loss : Étant donné un minibatch d'exemples, cette fonction devrait calculer la perte. Les entrées de la fonction sont x (un tableau numpy de dimension (minibatch size, 401)) et y

(un tableau numpy de dimension (minibatch size, 10)) et la sortie devrait être la perte calculée, un scalaire.

(c) Undergraduates 10 pts Graduates 10 pts
SVM.compute_gradient: Given a minibatch of examples, this function should compute the gradient of the loss function with respect to the parameters $\mathbf{w}$. The inputs are X (a numpy array of shape (minibatch size, 401)), y (a numpy array of shape (minibatch size, 10)), and the output should be the computed gradient, a numpy array of shape $(401, 10)$, the same shape as the parameter matrix $\mathbf{w}$. (Hint: use the expressions you derived above.)
Considérant un minibatch d'exemples, cette fonction devrait calculer le gradient de la fonction de perte par rapport au paramètre $\mathbf{w}$. Les entrées de la fonction sont X (un tableau numpy de dimension (minibatch size, 401)) et y (un tableau numpy de dimension (minibatch size, 10)) et la sortie devrait être le gradient calculé, un tableau numpy de dimension $(401, 10)$, soit la même dimension que celle du paramètre $\mathbf{w}$. (Indice: utilisez les expressions que vous avez dérivées précédemment.)

(d) Undergraduates 5 pts Graduates 5 pts
SVM.infer: Given a minibatch of examples, this function should infer the class for each example, i.e. which class has the highest score. The input is X (a numpy array of shape (minibatch size, 401)), and the output is *y_inferred* (a numpy array of shape (minibatch size, 10)). The output should be in the one-versus-all format, i.e. -1 for each class other than the inferred class, and $+1$ for the inferred class.
SVM.infer: Étant donné un minibatch d'exemples, cette fonction devrait prédire la classe de chaque exemple, c'est-à-dire la classe qui a le plus haut score. L'entrée de la fonction est X (un tableau numpy de dimension (minibatch size, 401)) et la sortie est *y_inferred* (un tableau numpy de dimension (minibatch size, 10)). La sortie devrait être en format un-contre-tous, c'est-à-dire -1 pour les classes qui ne sont pas prédites et $+1$ pour la classe prédite.

(e) Undergraduates 5 pts Graduates 5 pts
SVM.compute_accuracy: Given an array of inferred labels and

an array of true labels, this function should output the accuracy as a float between 0 and 1. The inputs are *y_inferred* (a numpy array of shape (minibatch size, 10)) and y (a numpy array of shape (minibatch size, 10)), and the output is a single float.
SVM.compute_accuracy: Étant donné un tableau de classes prédites et un tableau des vraies classes, cette fonction devrait retourner la proportion de classifications correctes, soit un scalaire entre 0 et 1. Les entrées de cette fonction sont *y_inferred* (un tableau numpy de dimension (minibatch size, 10)) et y (un tableau numpy de dimension (minibatch size, 10)) et la sortie est un scalaire.

4. Undergraduates 5 pts Graduates 5 pts

Question. The method SVM.fit uses the code you wrote above to train the SVM. After each epoch (one pass through the training set), SVM.fit computes the training loss, the training accuracy, the test loss, and the test accuracy.

Plot the value of these four quantities for every epoch for $C = 0.1, 1, 30, 50$. Use 200 epochs, a learning rate of 0.001, and a minibatch size of 5000.

You should have four plots: one for each quantity, with the curves for all four values of C. Include these four plots in your report.

La méthode SVM.fit utilise le code que vous avez écrit ci-dessus pour entraîner le SVM. Après chaque époque (après être passé à travers tous les exemples du jeu de données), SVM.fit calcule la perte et le taux de classifications correctes sur l'ensemble d'apprentissage et sur l'ensemble de test.

Faites le graphique de ces quatre quantités en fonction du nombre d'époques, pour $C = 0.1, 1, 30, 50$. Utilisez comme hyperparamètres 200 époques, un taux d'apprentissage de 0.001 et une taille de minibatch de 5000.

Vous devriez avoir 4 graphiques, soit un graphique pour chaque quantité, incluant les courbes pour les 4 valeurs de C. Ajoutez ces 4 graphiques dans votre rapport.