CSCI 5521: Fall'20
Introduction To Machine Learning

Homework 4

(Due Fri, April 24, 11:59 pm)

1. (30 points) Let $\mathcal{X} = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$ with $\mathbf{x}^t \in \mathbb{R}^D, t = 1, \ldots, N$ be a given training set. Assume that the dataset is centered, i.e., $\sum_{t=1}^N \mathbf{x}^t = 0 \in \mathbb{R}^D$. We focus on performing linear dimensionality reduction on the dataset using PCA (principal component analysis). With PCA, for each $\mathbf{x}^t \in \mathbb{R}^D$, we get $\mathbf{z}^t = W\mathbf{x}^t$, where $\mathbf{z}^t \in \mathbb{R}^d, d < D$, is the low dimensional projection, and $W \in \mathbb{R}^{d \times D}$ is the PCA projection matrix. Let $\Sigma = \frac{1}{N}\sum_{t=1}^N \mathbf{x}^t(\mathbf{x}^t)^T$ be the sample covariance matrix. Further, let $\mathbf{v}^t = W^T\mathbf{z}^t$ so that $\mathbf{v}^t \in \mathbb{R}^D$.

   (a) (10 points) Professor HighLowHigh claims: $\mathbf{v}^t = \mathbf{x}^t$ for all $t = 1, \ldots, N$. Is the claim correct? Clearly explain and prove your answer with necessary (mathematical) details.

   (b) (20 points) Professor HighLowHigh also claims:

$$\sum_{t=1}^N \|\mathbf{x}^t\|_2^2 - \sum_{t=1}^N \|\mathbf{v}^t\|_2^2 = \sum_{t=1}^N \|\mathbf{x}^t - \mathbf{v}^t\|_2^2 \ ,$$

   where for a vector $\mathbf{a} = [a_1 \cdots a_D]$, $\|\mathbf{a}\|_2^2 = \sum_{k=1}^D (a_k)^2$. Is the claim correct? Clearly explain and prove your answer with necessary (mathematical) details.

2. (30 points) Let $\mathcal{Z} = \{(\mathbf{x}^1, \mathbf{r}^1), \ldots, (\mathbf{x}^N, \mathbf{r}^N)\}, \mathbf{x}^t \in \mathbb{R}^d, \mathbf{r}^t \in \mathbb{R}^k$ be a set of N training samples. We consider training a multilayer perceptron as shown in Figure 1. We consider a general setting where the transfer functions at each stage are denoted by g, i.e.,

$$z_h^t = g(a_h^t) = g\left(\sum_{j=1}^d w_{h,j}x_j^t + w_0\right) \qquad \text{and} \qquad y_i^t = g(a_i^t) = g\left(\sum_{h=1}^H v_{i,h}z_h^t + v_{i0}\right) \ ,$$

   where a_h^t, a_i^t respectively denote the input activation for hidden node h and output node i. Further, let $L(\cdot, \cdot)$ be the loss function, so that the learning focuses on minimizing:

$$E(W, V|\mathcal{Z}) = \sum_{t=1}^N \sum_{i=1}^k L(r_i^t, y_i^t) \ .$$

   (a) (10 points) Show that the stochastic gradient descent update for $v_{i,h}$ is of the form $v_{i,h}^{\text{new}} = v_{i,h}^{\text{old}} + \Delta v_{i,h}$, with the update

$$\Delta v_{i,h} = \eta \Delta_i^t z_h^t \ , \quad \text{where } \Delta_i^t = g'(a_i^t)\left(-\frac{\partial L(r_i^t, y_i^t)}{\partial y_i^t}\right) \ . \tag{1}$$

   (b) (20 points) Show that the stochastic gradient descent update for $w_{h,j}$ is of the form $w_{h,j}^{\text{new}} = w_{h,j}^{\text{old}} + \Delta w_{h,j}$, with the update

$$\Delta w_{h,j} = \eta \Delta_h^t x_j^t \ , \quad \text{where } \Delta_h^t = g'(a_h^t)\left(\sum_{i=1}^k \Delta_i^t v_{i,h}\right) \ . \tag{2}$$

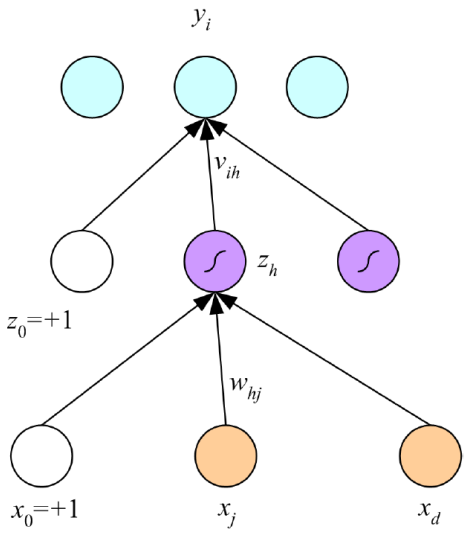

Figure 1: Two layer perceptron.

Programming assignment:

The next problem involves programming. For Question 3, we will be using the 2-class classification datasets from `Boston50` and `Boston75`. In particular, we will develop code for 2-class Support Vector Machines (SVMs) using gradient descent. The goal will be to modify your code for `MyLogisticReg2` from HW3.

3. (40 points) We will develop code for 2-class SVMs with parameters $(\mathbf{w}, w_0)$ where $\mathbf{w} \in \mathbb{R}^d, w_0 \in \mathbb{R}$. Assume a given dataset $\{(\mathbf{x}^t, y^t), t = 1, \ldots, N\}$, where $\mathbf{x}^t \in \mathbb{R}^d$ and $y^t \in \{-1, 1\}$. Recall from our discussion in class that training SVMs involves minimizing the following objective function:

$$f(\mathbf{w}, w_0) = \frac{1}{n}\sum_{i=1}^{n} \max\{0, 1 - y^t(\mathbf{w}^T\mathbf{x}^t + w_0)\} + \frac{\lambda}{2}\|\mathbf{w}\|^2 \ . \tag{3}$$

We will use $\lambda = 5$ in this assignment.

For reference, compare the objective function to that of regularized logistic regression which you recently worked on as part of HW3:

$$f(\mathbf{w}, w_0) = \frac{1}{n}\sum_{i=1}^{n} \{-y^t(\mathbf{w}^T\mathbf{x}^t + w_0) + \log(1 + \exp(\mathbf{w}^T\mathbf{x}^t + w_0))\} + \frac{\lambda}{2}\|\mathbf{w}\|^2 \ , \tag{4}$$

where we had used $\lambda = 0$ for the HW3 code.

We will develop code for `MySVM2` with corresponding `MySVM2.fit(X,y)` and `MySVM2.predict(X)` functions. Parameters for the model can be initialized following what you had done for `MyLogisticReg2`. In the `fit` function, the parameters will be estimated using *mini-batch stochastic gradient descent* with different mini-batch sizes $m \leq n$. In particular, you will

modify your `MyLogisticReg2` code by using gradients for the SVM objective in (3) instead of the logistic regression objective in (4). Further, you will have to add the mini-batch stochastic gradient descent (SGD) functionality which, for a pre-specified mini-batch size m, picks m unique points at random to do the gradient descent in each iteration. We will run experiments with different values of m.

We will compare the performance of `MySVM2` for different values of mini-batch size m with `LogisticRegression`[1] on two datasets: `Boston50` and `Boston75`. Recall that `Boston` has 506 data points, and a 5-fold cross-validation leaves $n \approx 400$ points for training in each fold.[2] For mini-batch SGD, we will consider three different values of m:

(i) $m = 40$, which is $\approx 10\%$ of the dataset in each fold for 5-fold cross-validation,

(ii) $m = 200$, which is $\approx 50\%$ of the dataset in each fold for 5-fold cross-validation, and

(iii) $m = n$, which is the full dataset in each fold for 5-fold cross-validation.

Note that $m = n$ uses the full dataset (available for that fold) in each iteration and hence corresponds to the usual gradient descent.[3]

Using `my_cross_val` with 5-fold cross-validation, report the error rates in each fold as well as the mean and standard deviation of error rates across all folds for the four methods: `MySVM2` with $m = 40, m = 200$, and $m = n$, and `LogisticRegression`, applied to the two 2-class classification datasets: `Boston50` and `Boston75`.

You will have to submit (a) **code** and (b) **summary of results**:

(a) **Code**: You will have to submit code for `MySVM2()` as well as a wrapper code `q3()`.
For developing `MySVM2()`, you are encouraged to consult the code for `MyLogisticReg2()` from HW3. You need to make sure you have `__init__`, `fit`, and `predict` implemented in `MySVM2`. `__init__(d,m)` will initialize the parameters and will take the data dimensionality d and mini-batch size m as input. You can add additional inputs such as the step size or the convergence threshold. `fit(X,y)` will take the data features X and labels y and will use mini-batch SGD to estimate the parameters $\mathbf{w}, w_0$. `predict(X)` will take a feature matrix corresponding to the test set and return the predicted labels. Your class `MySVM2()` will **NOT** inherit any base class in `sklearn`.

The wrapper code (main file) has no input and is used to prepare the datasets, and make calls to `my_cross_val(method,`X`,y,`k`)` to generate the error rate results for each dataset and each method. The code for `my_cross_val(method,`X`,y,`k`)` must be yours (e.g., code you made in HW1 with modifications as needed) and you cannot use `cross_val_score()` in `sklearn`. The results should be printed to terminal (not generating an additional file in the folder). Make sure the calls to `my_cross_val(method,`X`,y,`k`)`

[1] You should use `LogisticRegression` from `sklearn`, as we did for HW3. Note that Linear SVMs are implemented in `sklearn` as `LinearSVC`, but we will not use it since we have not discussed it in class. We will stick to `LogisticRegression` for comparisons.

[2] Note that we are denoting the number of training points available for training in each fold as n, which is smaller than the size of the full dataset.

[3] The exact value of n may differ mildly across the 5-folds since 506 cannot be exactly divided by 5. Your code for HW3 is already doing these splits, so this aspect should not need additional effort. In the code, the $m = n$ needs to be passed as a special option (say, $m = 10^6$ or $m =$"all") so the code knows it has use the the full dataset for that fold.

are made in the following order and add a print to the terminal before each call to show which method and dataset is being used:

i. `MySVM2` with $m = 40$ for `Boston50`;
ii. `MySVM2` with $m = 200$ for `Boston50`;
iii. `MySVM2` with $m = n$ for `Boston50`;
iv. `LogisticRegression` for `Boston50`;
v. `MySVM2` with $m = 40$ for `Boston75`;
vi. `MySVM2` with $m = 200$ for `Boston75`;
vii. `MySVM2` with $m = n$ for `Boston75`;
viii. `LogisticRegression` for `Boston75`.

*For the wrapper code, you need to make a `q3.py` file for it, and one should be able to run your code by calling "`python q3.py`" in command line window.

(b) **Summary of results**: For each dataset and each method, report the test set error rates for each of the $k = 5$ folds, the mean error rate over the k folds, and the standard deviation of the error rates over the k folds. Make a table to present the results for each method and each dataset (4 tables in total). Each column of the table represents a fold and add two columns at the end to show the overall mean error rate and standard deviation over the k folds. For example:

Error rates for MySVM2 with $m = 40$ for Boston50						
Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	SD
#	#	#	#	#	#	#

Additional instructions: Code can only be written in Python; no other programming languages will be accepted. One should be able to execute your code for Q3 directly from command prompt (e.g., "`python q3.py`") without the need to run Python interactive shell first. Test your code yourself before submission and suppress any warning messages that may be printed. Your code must be run on a CSE lab machine (e.g., csel-kh1260-01.cselabs.umn.edu). Please make sure you specify the full Python version you are using as well as instructions on how to run your program in the README file (must be readable through a text editor such as Notepad). Information on the size of the datasets, including number of data points and dimensionality of features, as well as number of classes can be readily extracted from the datasets in `scikit-learn`. Each function must take the inputs in the order specified in the problem and display the output via the terminal or as specified. For Q3, you can submit additional files/functions (as needed) which will be used by the main file. Please put comments in your code so that one can follow the key parts and steps in your code.

Extra Credit Problem:

EC1 (15 points) Consider Problem 2 with specific choices of the activation function $g(a)$. We will assume $L(r_i^t, y_i^t) = (r_i^t - y_i^t)^2$.

(a) (5 points) Let $g(u) = \max(0, u)$. What is the gradient $g'(a)$? What does the update in (1) look like with these specific choices of $g(\cdot)$ and $L(\cdot, \cdot)$? Clearly explain your answer and show details.

(b) (5 points) For some $\alpha \in [0, 1]$, let

$$g(a) = \max(0, a) + \alpha \min(0, a) \ . \tag{5}$$

What is the gradient $g'(a)$ in terms of α? Clearly explain your answer and show details.

(c) (5 points) For the activation function in (5), is there a specific choice of $\alpha \in [0, 1]$ which makes the two layer perceptron effectively a linear model, i.e., the predictions y_i^t are a linear function of the inputs x_j^t? Clearly explain your answer and show details.

Follow the rules strictly. If we cannot run your code, you will not get any credit.

- **Things to submit**

1. hw4.pdf: A document which contains the solution to Problems 1, 2, 3 (and extra credit problem) including the summary of results for Problem 3. This document must be in PDF format (no word, photo, etc., is accepted). If you submit a scanned copy of a hand-written document, make sure the copy is clearly readable, otherwise no credit may be given.
2. Python code for Problem 3 (must include the required `q3.py`).
3. README.txt: README file that contains your name, student ID, email, instructions on how to run your code, the full Python version (e.g., Python 3.5) you are using, any assumptions you are making, and any other necessary details. The file must be readable by a text editor such as Notepad.
4. Any other files, except the data, which are necessary for your code.