CSCI 5521: Fall'20
Introduction To Machine Learning

Homework 3

(Due Fri, November 20, 11:59 pm)

1. (25 points) Let $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ be a given training set where $\mathbf{x}_i \in \mathbb{R}^d, y_i \in \{0, 1\}$. We consider the following regularized logistic regression objective function:

$$f(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \left\{ -y_i \mathbf{w}^T \mathbf{x}_i + \log(1 + \exp(\mathbf{w}^T \mathbf{x}_i)) \right\} + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 \ ,$$

   where $\lambda > 0$ is a constant. Let $\mathbf{w}^*$ be the global minimizer of the objective, and let $\|\mathbf{w}^*\|_2 \leq c$, for some known constant $c > 0$.

   (a) (10 points) Clearly show and explain the steps of the projected gradient descent algorithm for optimizing the regularized logistic regression objective function.[1] The steps should include an exact expression for the gradient.

   (b) (5 points) Is the objective function strongly convex? Clearly explain your answer by stating and using the definition of strong convexity.[2]

   (c) (5 points) Is the objective function smooth? Clearly explain your answer by stating and using the definition of smoothness.[2]

   (d) (5 points) Let $\mathbf{w}_T$ be the iterate after T steps of the projected gradient descent algorithm with a suitably chosen step size depending on the properties of the function. What is a bound on the difference $f(\mathbf{w}_T) - f(\mathbf{w}^*)$? Clearly explain all quantities in the bound.

2. (25 points) Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}, \mathbf{x}_i \in \mathbb{R}^d$ be a set of n samples drawn i.i.d. from a mixture of k multivariate Gaussian distributions in $\mathbb{R}^d$. For component $G_h, h = 1, \ldots, k$, let π_h, μ_h, Σ_h respectively denote the prior probability, mean, and covariance matrix of G_h. We will focus on the expectation maximization (EM) algorithm for learning the mixture model, in particular for estimating the parameters $\{(\pi_h, \mu_h, \Sigma_h), h = 1, \ldots, k\}$ as well as the posterior probabilities $p(G_h|\mathbf{x}_i)$.

   (a) (10 points) In your own words, describe the EM algorithm for mixture of Gaussians, highlighting the two key steps (E- and M-), illustrating the methods used in the steps on a high level, and what information they need.

   (b) (10 points) Assuming the posterior probabilities $p(G_h|x_i)$ are known, show the estimates of the component prior, mean, and covariance $\pi_h, \mu_h, \Sigma_h, h = 1, \ldots, k$ given by the M-step (you do not need to show the derivation).

   (c) (5 points) Assuming the component prior, mean, and covariance $\pi_h, \mu_h, \Sigma_h, h = 1, \ldots, k$ are known, show how the posterior probabilities $p(G_h|x_i)$ are computed in the E-step.

[1] The projection here will be to the set $\{\mathbf{w} : \|\mathbf{w}\|_2 \leq c\}$ since $\mathbf{w}^*$ belongs to that set.

[2] Since the function is twice differentiable, you can answer the question by considering the second derivative. Please see the notes posted in canvas.

Programming assignments:

The next two problems involve programming. For Question 3, we will be using the 2-class classification datasets from `Boston50` and `boston25`, and for Question 4, we will be using the the `Digits` dataset. For Q3, we will develop code for 2-class logistic regression with only one set of parameters $(\mathbf{w}, w_0)$. For Q4, we will develop code for PCA.

3. (25 points) We will develop code for 2-class logistic regression with one set of parameters $(\mathbf{w}, w_0)$ where $\mathbf{w} \in \mathbb{R}^d$, $w_0 \in \mathbb{R}$. Assuming the two classes are $\{0, 1\}$, and the data $\mathbf{x} \in \mathbb{R}^d$, the posterior probability of class C_1 is given by

$$P(1|\mathbf{x}) = \frac{\exp(\mathbf{w}^T\mathbf{x} + w_0)}{1 + \exp(\mathbf{w}^T\mathbf{x} + w_0)} ,$$

and $P(0|\mathbf{x}) = 1 - P(1|\mathbf{x})$. f We will develop code for `MyLogisticReg2` with corresponding `MyLogisticReg2.fit(X,y)` and `MyLogisticReg2.predict(X)` functions. Parameters for the model can be initialized following suggestions in the textbook. In the `fit` function, the parameters will be estimated using gradient descent as described in the textbook and in class.

We will compare the performance of `MyLogisticReg2` with `LogisticRegression`[3] on two datasets: `Boston50` and `boston25`. 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 two methods: `MyLogisticReg2` and `LogisticRegression`, applied to the two 2-class classification datasets: `Boston50` and `boston25`.

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

(a) **Code**: You will have to submit code for `MyLogisticReg2()` as well as a wrapper code `q3()`.

For `MyLogisticReg2()`, you are encouraged to consult the code for `MultiGaussClassify()` from HW2. You need to make sure you have `__init__`, `fit`, and `predict` implemented in `MyLogisticReg2`. `__init__(d)` will initialize the parameters $(\mathbf{w}, w_0)$ and will take the data dimensionality d as input. `fit(X,y)` will take the data features X and labels y and will use gradient descent to estimate the parameters $\mathbf{w}, w_0$. Convergence of gradient descent can be determined by checking the difference between subsequent iterates $(\mathbf{w}^{t-1}, w_0^{t-1})$ and $(\mathbf{w}^t, w_0^t)$ and making sure the change is below a pre-specified (small) threshold. You can also specify $t_{\max}$, the maximum number of iterations gradient descent can run, and set it to a suitably large value. `predict(X)` will take a feature matrix corresponding to the test set and return the predicted labels. Your class `MyLogisticReg2()` will **not** inherit any base class in `sklearn`.

The wrapper code (main file `q3.py`) has no input and is used to prepare the datasets, and make calls to `my_cross_val(method,`X`,`$\mathbf{y}$`,`k`)` to generate the error rate results for each dataset and each method. The code for `my_cross_val(method,`X`,`$\mathbf{y}$`,`k`)` can be either yours from hw1 or you can 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`,`$\mathbf{y}$`,`k`)` 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:

[3]You should use `LogisticRegression` from `scikit-learn`, similar to HW1 and HW2.

(1) `MyLogisticReg2` with `Boston50`;

(2) `MyLogisticReg2` with `Boston25`;

(3) `LogisticRegression` with `Boston50`;

(4) `LogisticRegression` with `Boston25`.

One should be able to run your wrapper code `q3.py` by "`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 MyLogisticReg2 with Boston50						
Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	SD
#	#	#	#	#	#	#

4. (25 points) We will modify the provided ipython notebook `generate_digits.ipynb` to add our own code for the parts where PCA is getting used. The notebook shows how to learn and subsequently generate data which look like the data from the `Digits` dataset. Let the data matrix be denoted as $X \in \mathbb{R}^{D\times n}$ where D is the number of features and n is the number of samples. Each column of X corresponds to a data point $\mathbf{x}_j \in \mathbb{R}^D$. For `Digits`, we have $D = 64$ and $n \approx 1800$. The feature covariance matrix of the data can be computed as:

$$\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^T \ , \tag{1}$$

where $\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$ is the mean of the data points. The notebook has the following steps:

(a) Compute a PCA projection $Z \in \mathbb{R}^{d\times n}, d \leq D$ of the original data $X \in \mathbb{R}^{D\times n}$ so that $\alpha\%$ of the variance is preserved in the projected space. The notebook has details for $\alpha = 90$, where $d = 21$ was sufficient. The notebook also has details for $\alpha = 99$, where $d = 41$ was sufficient. The projection can be represented by a matrix $W \in \mathbb{R}^{d\times D}$.

In the homework, instead of using sklearn's functionality for determining d and W, we will develop our own function `myPCA` (see below) using `numpy` and `scipy` as needed. This will be the only major change you will have to do in the notebook.

(b) Consider the dataset $\mathcal{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ of n points in $\mathbb{R}^d$, and fit a mixture of Gaussians (MoG) model where the number of mixture components is determined by a suitable model selection technique. The notebook already implements this part, and we will not change it.

(c) Sample a set of m new points $\tilde{\mathcal{Z}} = \{\tilde{\mathbf{z}}_1, \ldots, \tilde{\mathbf{z}}_m\}, \tilde{z}_i \in \mathbb{R}^d$ from the MoG model in the projected space. Let $\tilde{Z} \in \mathbb{R}^{d\times m}$ be the matrix corresponding to the sampled data $\tilde{\mathcal{Z}}$, where the columns are $\tilde{\mathbf{z}}_i$. The notebook already implements this part with $m = 100$, and we will not change it.

(d) Inverse transform the new points $\tilde{Z}$ in the d-dimensional space to the original D-dimensional space. The notebook currently uses sklearn's functionality for doing the inverse transformation. Instead we will use our own code to do the inverse transformation. In particular, the inverse transformation can be computed using W and $\hat{\boldsymbol{\mu}}$ which we will get as outputs from `myPCA` (see below). The inverse transform is a simple linear algebraic operation (say, 1-2 lines of code), so we will not write a separate function for this, but just inline the code.

The main function we will write is `myPCA(`X`,`α`)`, which takes in the original data $X \in \mathbb{R}^{D \times n}$ and the percentage $\alpha \in [0, 100]$ of variance to be preserved, and returns three things as a tuple:

(i) the PCA projection dimensionality d which is sufficient to preserve $\alpha\%$ of the original variance in the projected space,

(ii) the projection matrix $W \in \mathbb{R}^{d \times D}$, and

(iii) the estimated mean $\hat{\mu} \in \mathbb{R}^D$ of the original data.

Add `myPCA` as a function in the notebook and update the rest of the notebook to use `myPCA` instead of `sklearn`'s PCA. Please make sure the current functionality in the notebook is maintained. In particular, your notebook should not `import PCA` but still maintain the same functionality. You will have to submit your notebook `my_generate_digits.ipynb`.

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. Code for Q4 will be a modification of the provided notebook `generate_digits.ipynb`.

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

- **Things to submit**

1. hw3.pdf: A document which contains the solution to Problems 1, 2, 3 and 4 including the summary of results for 3 and 4. 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 Problems 3 and 4 (must include the required `q3.py` and `my_generate_digits.ipynb`).

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) your 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.