CSCI 5512: Artificial Intelligence II (Spring 2022)

Homework 4

(Due Tue, Apr 19, 11:59 pm central)

1. **(40 points)** The perceptron learning rule learns a linear model for classification problems by iteratively updating a weight vector via:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t(y_{i_t} - \mathbb{1}(\mathbf{w}_t^\top \mathbf{x}_{i_t} \geq 0))\mathbf{x}_{i_t}$$

   where t indexes the iteration, i_t is the data point used in iteration t (i.e., for N total data points $i_t \in \{1, \ldots, N\}$), $y_{i_t} \in \{0, 1\}$ is the label, $\mathbf{x}_{i_t} \in \mathbb{R}^d$ is the feature vector, $\alpha_t \in \mathbb{R}$ is the learning rate, and $\mathbb{1}(x)$ is the indicator function which returns 1 if the condition x is true and 0 otherwise. Implement the perceptron algorithm (from scratch) with $\alpha_t = 1/\sqrt{t}$ in file `hw4_q1.py`. Randomly iterate through all data points for 100,000 iterations. Using the scikit-learn wine dataset[1] with 80% of the data as training data and the remaining 20% as test data and only considering classes 0 and 1, apply your code to the dataset and compute the test set classification error. Do this for 10 repetitions, each time starting the algorithm from scratch and computing the classification error. Report the average classification error and standard deviation across the 10 repetitions on the test set. Your code must print these numbers out via the terminal; also report them in the hw4 writeup pdf. An example of how to call your code is: `python hw4_q1.py`.

2. **(40 points)** The k-nearest neighbor (knn) algorithm predicts the class of a test data point by computing the distance to all neighbors in the training set, selecting the k neighbors with smallest distance (i.e., the nearest neighbors), and predicting the class via majority vote among the neighbors' class labels. Implement the knn algorithm (from scratch) in file `hw4_q2.py` and use 5-fold cross validation to select the optimal value of $k = \{23, 51, 101\}$. Use the Euclidean distance to compute the distance between data points (i.e., $\|\mathbf{x}_a - \mathbf{x}_b\|_2 = \sqrt{\sum_{i=1}^{d}(x_a(i) - x_b(j))^2}$). Apply your code to the scikit-learn wine dataset (same as in problem 1). Report the average classification error on the test folds for each value of k. (In other words, for a given k value, compute the classification error on each of the 5 test folds, then average these values. Repeat for each value of k.) Your code must print these numbers out via the terminal; also report them in the hw4 writeup pdf. Explain which value of k you would choose to build your final model. An example of how to call your code is: `python hw4_q2.py`. You can use the cross validation[2] function `numpy.model_selection.KFold()` and Euclidean distance[3] function `numpy.linalg.norm()` to compute the cross validation splits and distances.

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html#sklearn.datasets.load_wine`

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html`

[3] `https://numpy.org/doc/stable/reference/generated/numpy.linalg.norm.html`

3. **(30 points)** The least squares linear regression algorithm learns a linear function $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$ which minimizes the squared error on the training data. Often we want to also minimize the complexity of the linear function (e.g., the magnitude of $\mathbf{w}$) using L_2 regularization to avoid overfitting. The closed-form solution for this regularized linear regression problem is

$$\begin{aligned}\mathbf{w}^* &= \text{argmin}_{\mathbf{w}\in\mathbb{R}^d} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2 \\ &= (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}\end{aligned}$$

where $\mathbf{w}$ is the weight vector, $\mathbf{y}$ is the target value vector of length N (for N data ponts), $\mathbf{X}$ is the design matrix of size $N \times d$ (recall, we stack the N data points, each which has length d, into a matrix to construct $\mathbf{X}$), $\mathbf{I}$ is the identity matrix of size $d \times d$ (i.e., a matrix with d rows and columns where the values on the diagonal are all 1s and the non-diagonal values are 0s), and $\lambda \in \mathbb{R}$ is the regularization parameter. Implement the least squares linear regression algorithm (from scratch) with L_2 regularization in file `hw4_q3.py` using the solution above. Apply your code to the scikit-learn diabetes dataset[4] and compute the average root mean squared error (RMSE) on the test folds for each value of λ. For a test fold with N_{test} datapoints, the RMSE is computed via

$$RMSE = \sqrt{\frac{1}{N_{\text{test}}}\sum_{i=1}^{N_{\text{test}}}(y_i - \mathbf{w}^\top\mathbf{x}_i)^2}.$$

Use 5-fold cross validation to select the optimal value of $\lambda \in \{0.1, 1, 10, 100\}$ and explain which value you would choose to build your final model. Your code must print out the RMSE for each value of λ via the terminal; also report them in the hw4 writeup pdf. An example of how to call your code is: `python hw4_q3.py`. You may want to use mathematical numpy functions such as `numpy.dot()`, `numpy.transpose()`, `numpy.linalg.inv()`, `numpy.eye()`, `numpy.linalg.norm()`, etc.

Extra Credit. *In order for extra credit solutions to be considered, you must provide reasonable solutions to all parts above. If you skip any part of a problem or do not provide a reasonable solution (i.e., make a real effort), we will not count any extra credit towards your grade.*

4. **Extra Credit (5 points)** In problem 3, we considered the least squares linear regression problem with L_2 regularization and the solution for the optimal $\mathbf{w}$ vector was given. Derive this optimal $\mathbf{w}$ vector by hand. For full credit, you must show all steps of your derivation. Hint, start with the optimization problem

$$\mathbf{w}^* = \text{argmin}_{\mathbf{w}\in\mathbb{R}^d} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2.$$

5. **Extra Credit (15 points)** In this problem, we will consider classifying the wine dataset (same as above) using the scikit-learn[5] machine learning library. (You do not need to implement the algorithms from scratch, use scikit-learn instead). Using the following algorithms, learn a model to classify the wine dataset:

[4] `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_diabetes.html#sklearn.datasets.load_diabetes`

[5] `https://scikit-learn.org/stable/`

- Logistic regression (set `max_iters = 3000`),
- Linear support vector machine (SVM) (set `kernel='linear'`),
- Random forest (RF) (set `criterion='entropy', max_depth=1`),
- Adaboost.

For each algorithm, use 5-fold cross validation (you may use the scikit-learn CV function[6] for this problem) to tune the following hyperparameters:

- Logistic regression[7]: $C \in [1e-5, 1e-4, 1e-3, 1e-2, 0.1, 1, 10, 100, 1000]$,
- SVM[8]: $C \in [1e-5, 1e-4, 1e-3, 1e-2, 0.1, 1, 10, 100, 1000]$,
- RF[9]: `n_estimators` $\in [1, 10, 20, 30, 40, 50, 100, 200]$,
- Adaboost[10]: `n_estimators` $\in [1, 10, 20, 30, 40, 50, 100, 200]$.

For each algorithm and hyperparameter, plot the mean classification **error rate** and standard deviation (as error bars) across the 5 test folds. For each algorithm, choose the 'best' hyperparameter and explain your choice. Submit a single python file named `hw4_ec.py` which takes no arguments and runs and saves the plots for each algorithm in the current working directory. Make sure to also include these plots in your hw4 writeup pdf.

Instructions

Code can only be written in Python 3.6+; no other programming languages will be accepted. One should be able to execute all programs from the Python command prompt or terminal. Each program must take the inputs in the order specified in the problem and display the textual output via the terminal and plots/figures should be included in the report. Please specify instructions on how to run your program in the README file.

For each part, you can submit additional files/functions as needed. You may use libraries for basic matrix computations and plotting such as numpy, pandas, and matplotlib. Put comments in your code so that one can follow the key parts and steps in your code.

Your code must be runnable on a CSE lab machine (e.g., csel-kh1260-01.cselabs.umn.edu). One option is to SSH into a machine. Learn about SSH, and other options, at these links: `https://cse.umn.edu/cseit/self-help-guides/secure-shell-ssh` and `https://cse.umn.edu/cseit/self-help-guides`.

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

- **Things to submit**

  1. hw4_sol.pdf: A document which contains solutions to all problems. This document must be in PDF format (doc, jpg, etc. formats are not 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 1,2, 3, and extra credit problem 5.

[6] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html`
[7] `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`
[8] `https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html`
[9] `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`
[10] `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html`

3. README.txt: README file that contains your name, student ID, email, assumptions you are making, other students you discussed the homework with, and other necessary details.
4. Any other files, except the data, which are necessary for your code (such as package dependencies like a requirements.txt or yml file).

Homework Policy. (1) You are encouraged to collaborate with your classmates on homework problems, but each person must write up the final solutions individually. You need to list in the README.txt which problems were a collaborative effort and with whom. (2) Regarding online resources, you should **not**:

- Google around for solutions to homework problems,
- Ask for help on online,
- Look up things/post on sites like Quora, StackExchange, etc.