CSCI 5525: Advanced Machine Learning (Spring 2023)

Homework 2

(Due Thu, Feb. 16, 11:59 PM CST)

1. **(15 points)** We consider logistic regression for a 2-class classification setting. Let $\mathcal{X} = \{(\mathbf{x}_i, y_i), \ldots, (\mathbf{x}_n, y_n)\}$ be a dataset for 2-class classification, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{0, 1\}$. For any $a \in \mathbb{R}$, let $\sigma(a) = \frac{1}{1+\exp(-a)}$. For each sample $(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \{0, 1\}$, the conditional log-likelihood of logistic regression is

$$L(y_i \mid \mathbf{x}_i, \mathbf{w}) = y_i \log(\sigma(\mathbf{w}^\top \mathbf{x}_i)) + (1 - y_i) \log(\sigma(-\mathbf{w}^\top \mathbf{x}_i)).$$

   Derive the gradient of $L(y_i \mid \mathbf{x}_i, \mathbf{w})$ with respect to w_j (the j-th coordinate of $\mathbf{w}$), i.e., $\frac{\partial}{\partial w_j} L(y_i \mid \mathbf{x}_i, \mathbf{w})$. (You must clearly mention the derivative properties used.)

2. **(30 points)** In this problem, we will use logistic regression to classify the provided dataset `hw2_q2_q4_dataset.csv`. We denote the given dataset as $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^d$ are the feature vectors and $y_i \in \{0, 1\}$ are the binary labels. Write Python code to implement logistic regression from scratch in the class `MyLogisticRegression`. To train your model, minimize the average loss using full gradient descent (not SGD). Start with a $\mathbf{w}_0$ vector chosen uniformly at random in $[-0.01, 0.01]^d$. Convergence of gradient descent can be determined by checking the difference in the loss function value between subsequent iterates $\mathbf{w}_{t-1}$ and $\mathbf{w}_t$ and making sure the change is below a pre-specified threshold (such as 10^{-6}). You can also specify `max_iters`, the maximum number of iterations gradient descent can run, and set it to a suitably large value such as 1000.

   For logistic regression using gradient descent, the hyperparameter is the learning rate η used in the update equation

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla f(\mathbf{w}_t).$$

   Use 10-fold cross validation and the error rate to tune η with the parameter range given in the code (see below). You may use your own cross validation function `my_cross_val` from homework 1 or any of the sklearn cross validation functions such as `KFold`[1] or `cross_val_score`[2].

   Report the error rate in each fold as well as the mean and standard deviation across folds. Which value of η is optimal? Using the optimal value of η, train a single model using all the training data and then predict using the test data. Report the error rate on the test data. In addition to `MyLogisticRegression.py` (details below), add your code to `hw2_q2.py` and use it to test.

   `MyLogisticRegression.py` is a class which must have the following:

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

```
class MyLogisticRegression:

   def __init__(self, d, max_iters, eta_val):
       ...
   def fit(self, X, y):
       ...
   def predict(self, X):
       ...
```

Your class `MyLogisticRegression` **should not** inherit any base class. The `fit` method does not return anything and the `predict` method should return a list of predictions of length equal to the number of rows in `X`. Note, for your `fit` function, you may need to clip the values after computing the sigmoid function to avoid Inf/NaN – use `np.clip`[3].

Please write up your results and submit them in a PDF document. For each value of η, report the validation set error rate for each of the $k = 10$ folds, the mean error rate over the k folds, and the standard deviation of the error rate over the k folds. Make a table to present the results. Include a column in the table for each 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 Logistic Regression											
F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	Mean	SD
#	#	#	#	#	#	#	#	#	#	#	#

3. **(20 points)** We consider support vector machines (SVMs) for a 2-class classification setting. Let $\mathcal{X} = \{(\mathbf{x}_i, y_i), \ldots, (\mathbf{x}_n, y_n)\}$ be a dataset for 2-class classification, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$. Here we consider the SRM perspective of SVMs which define the objective function as

$$f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n} \max\left(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\right).$$

Derive the gradient of $f(\mathbf{w})$ with respect to w_j (the j-th coordinate of $\mathbf{w}$), i.e., $\frac{\partial}{\partial w_j} f(\mathbf{w})$. (Hint: separately consider the cases when the hinge loss is 0 and otherwise.)

4. **(35 points)** In this problem, we will use SVMs to classify the dataset `hw2_q2_q4_dataset.csv`. We denote the given dataset as $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^d$ are the feature vectors and $y_i \in \{-1, 1\}$ are the binary labels. Write Python code to implement an SVM from scratch in the class `MySVM`. To train your model, minimize the average loss using full gradient descent (not SGD). Start with a $\mathbf{w}_0$ vector chosen uniformly at random in $[-0.01, 0.01]^d$. Convergence of gradient descent can be determined by checking the difference in the loss function value between subsequent iterates $\mathbf{w}_{t-1}$ and $\mathbf{w}_t$ and making sure the change is below a pre-specified threshold (such as 10^{-6}). You can also specify `max_iters`, the maximum number of iterations gradient descent can run, and set it to a suitably large value such as 1000.

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

For SVM using gradient descent, there are two hyperparameters: the learning rate η used in the update equation

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla f(\mathbf{w}_t)$$

and the regularization parameter C. Use 10-fold cross validation and the error rate to tune η and C with the parameter range given in the code (see below). You may use your own cross validation function `my_cross_val` from homework 1 or any of the sklearn cross validation functions such as `KFold`[4] or `cross_val_score`[5].

Report the error rate in each fold as well as the mean and standard deviation across folds for each combination of η and C. Which values of η and C are optimal? Using the optimal values of η and C, train a single model using all the training data and then predict using the test data. Report the error rate on the test data. In addition to `MySVM.py` (details below), add your code to `hw2_q4.py` and use it to test.

`MySVM.py` is a class which must have the following:

```
class MySVM:

   def __init__(self, d, max_iters, eta_val, C):
       ...
   def fit(self, X, y):
       ...
   def predict(self, X):
       ...
```

Your class `MySVM` **should not** inherit any base class. The `fit` method does not return anything and the `predict` method should return a list of predictions of length equal to the number of rows in `X`.

Please write up your results and submit them in a PDF document. For each method and value of η and C, report the validation set error rate for each of the $k = 10$ folds, the mean error rate over the k folds, and the standard deviation of the error rate over the k folds. Make a table to present the results. Include a column in the table for each 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 SVM											
F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	Mean	SD
#	#	#	#	#	#	#	#	#	#	#	#

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

Instructions

You must complete this homework assignment individually. You may discuss the homework at a high-level with other students but make sure to include the names of the students in your README file. You may not use any AI tools (like GPT-3, ChatGPT, etc.) to complete the homework. 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. Make sure to include a requirements.txt, yaml, or other files necessary to set up your environment. Please specify instructions on how to run your program in the README file.

Each function must take the inputs in the order specified in the problem and display the textual output via the terminal and plots/figures, if any, should be included in the PDF report.

In your code, you can only use machine learning libraries such as those available from scikit-learn as specified in the problem description. 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.

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

- **Things to submit**

  1. `[YOUR_NAME]_hw2_solution.pdf`: A document which contains solutions to all problems.
  2. `hw2_q2.py` and `MyLogisticRegression.py`: Code for Problem 2.
  3. `hw2_q4.py` and `MySVM.py`: Code for Problem 4.
  4. README.txt: README file that contains your name, student ID, email, instructions on how to run your code, any assumptions you are making, and any other necessary details.
  5. Any other files, except the data, which are necessary for your code.

Homework Policy. (1) You are encouraged to collaborate with your classmates on homework problems at a high level only. 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. Please refer to the syllabus for more details. (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.