The Chinese University of Hong Kong, Shenzhen
School of Data Science

MDS5210 · Homework 3

Due: (23:59), April, 9

Instructions:

- Homework problems must be carefully and clearly answered to receive full credit. Complete sentences that establish a clear logical progression are highly recommended.
- You must submit your assignment in Blackboard. Please upload a file or a zip file. The file name should be in the format **last name-first name-hw3**.
- The homework must be written in English.
- Late submission will not be graded.
- Each student **must not copy** homework solutions from another student or from any other source.

Problem 1 (20pts). Fundamental Knowledge

(a) Explain the main insights of logistic regression including: What and why is the model? How to formulate the learning problem?

(b) Explain why convex problem is easier than nonconvex problems (consider the unconstrained optimization problem).

(c) Is it true that logistic regression is used for linear regression?

(d) Give the intuition why the Nestrov's accelerated gradient descent achieves faster convergence speed. (Hint: You may find the figure on the page 12 of slides 8 useful.)

(e) Derive the update for proximal gradient descent for Lasso in page 28 of lecture slides 8.

(f) Explain the main insights of SVM (linearly separable case) and how to learn the classifier.

(g) Explain the main insights of kernel methods.

Problem 2 (50pts). Regularization and Optimization Methods

In this exercise we will experiment the regularization techniques and optimization algorithms we learned in lecture. We consider a problem of predicting fetal health given Cardiotocograms result. You may refer to `https://www.kaggle.com/datasets/andrewmvd/fetal-health-classification` for more information about the dataset we are using.[1] We will use logistic regression with ℓ_1 regularization to do the classification.

[1]Ayres-de-Campos, D., Bernardes, J., Garrido, A., Marques-de-Sa, J., & Pereira-Leite, L. (2000). SisPorto 2.0: a program for automated analysis of cardiotocograms. Journal of Maternal-Fetal Medicine, 9(5), 311-318.

(a) **[10 points]** Design the proximal stochastic gradient descent, proximal stochastic gradient descent with momentum, and proximal accelerated gradient descent algorithm, i.e. write down their update. (Hint: Refer to the designs of proximal gradient descent and accelerated gradient descent in lecture slides)

(b) **[40 points]** Implement the ℓ_1-regularized logistic regression for this classification task. You should implement the proximal gradient descent, proximal accelerated gradient descent, proximal stochastic gradient descent, and proximal stochastic gradient descent with momentum for solving the learning problem, and

- Plot the training loss, test loss, and test accuracy of four optimization algorithms on three figures, respectively.
- For the proximal gradient descent algorithm, examine different lambda for l_2 regularization, compare the performance by plotting the training loss, test loss, and test accuracy.
- For the proximal gradient descent algorithm, examine different lambda for l_1 regularization, compare the performance by plotting the training loss, test loss, and test accuracy.
- Briefly summarize the phenomenon you observed.

Note: You cannot use any pre-built packages for implementing these algorithms

You may start with the provided code `p2.py` for data loading and pre-processing. The block marked by `#TODO` should be completed by yourself. If you choose to use this code, run it first to ensure all the packages are installed.

Problem 3 (30pts). Support Vector Machine

Scikit-learn Package. We will use a famous machine learning toolbox for this problem called scikit-learn. The official document can be founded at `https://scikit-learn.org/stable/`. For the installation, please check `https://scikit-learn.org/stable/install.html`.

Data. The dataset we use is MNIST. To download it, run the following command

```
from sklearn.datasets import fetch_openml
X, y = fetch_openml('mnist_784', version=1, return_X_y=True)
```

Here, `X` is of dimension (num_samples, 784), where each sample is represented by a 784 dimension vector, corresponding to a 28 image. `y` is of dimension (num_samples,), representing the label. To plot j_{th} image, we can use the following code

```
import matplotlib.pyplot as plt
plt.title('The jth image is a {label}'.format(label=int(y[j])))
plt.imshow(X[j].reshape((28,28)), cmap='gray')
plt.show()
```

There are approximatly 7000 samples for each class in total. You should split the dataset, with 4000 samples in training set and remainer for test set.

Task. Our task is to use SVM to do the multi-class classification. We will use scikit-learn to do this. You may refer `https://scikit-learn.org/stable/modules/svm.html#svm-mathematical-formulation` for the formulation of scikit-learn's implementation. In their formulation, they have a parameter C, which performs exactly the similar role to our λ. You can simply regard C as $\frac{1}{\lambda}$. You should use *cross-validation* to select C. Specifically, you should split the training set into 3000 for training and 1000 for validation. You may use the following code for generating candidate C

```
C_grid = np.logspace(-3, 3, 10)
```

which returns 10 numbers spaced evenly with respect to interval $[-3, 3]$ on a log-scale.
An example code for training linear SVM with $C = 1$ is

```
from sklearn import svm
clf = svm.SVC(C=1.0, kernel='linear')
clf.fit(X_train, y_train)
```

The error for the resulting classifier can be calculated as

```
Pe = 1 - clf.score(X_test, y_test)
```

(a) **[15 points]** Train an SVM using the kernels $k(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u}^\top \boldsymbol{v} + 1)^p, p = 1, 2$, that is, the inhomogeneous linear and quadratic kernel. To do this, you will want to set the parameters `kernel='poly'` and `degree=1` or `degree=2` in `svm.SVC` function.

For each kernel, report the best value of C (draw a figure where y-axis is the validation error, while x-axis is the value of C), the test error corresponding to this best C (do not forget to retrain on the whole training dataset after selecting the best value of C). Hand in your code.

(b) **[15 points]** Repeat the above using the radial basis function (RBF) kernel $k(\boldsymbol{u}, \boldsymbol{v}) = \exp(\|\boldsymbol{u} - \boldsymbol{v}\|_2^2)$ (by setting the parameters *kernel='rbf', gamma=gamma* in `svm.SVC` function. You will now need to determine the best value for both C and γ. Report the best value of C and γ (draw a figure where y-axis is the validation error, while x-axis is the value of C), the test error correspond to the best value of C and γ (do not forget to retrain on the whole training dataset after selecting the best value of C and γ). Hand in your code.