Programming Assignment 2

CS6510: Applied Machine Learning
IIT-Hyderabad
Aug-Nov 2017

Max Marks: 15
Due: 17 Nov 2017 11:59 pm

This homework is intended to cover programming exercises in the following topics:

- Regression, Dimensionality Reduction

Instructions

- Please use Google Classroom to upload your submission by the deadline mentioned above. Your submission should comprise of a single file (PDF/ZIP), named `<Your_Roll_No>_PA2`, with all your solutions.
- For late submissions, 10% is deducted for each day (including weekend) late after an assignment is due. Note that each student begins the course with 8 grace days for late submission of assignments (with a max of 4 per submission). Late submissions will automatically use your grace days balance, if you have any left. You can see your balance on the CS6510 Marks and Grace Days document under the course Google drive.
- You should use PYTHON for the programming assignments.
- Please read the department plagiarism policy. Do not engage in any form of cheating - strict penalties will be imposed for both givers and takers. Please talk to instructor or TA if you have concerns.

1 Questions

1. **Principal Component Analysis (PCA) and t-SNE:** *(4 + 4 = 8 marks)*
   - For this problem, you will implement Principal Component Analysis on the Iris dataset. The Iris dataset contains classifications of iris plants based on four features: sepal length, sepal width, petal length, and petal width. There are three classes of iris plants on this dataset: Iris Setosa, Iris Versicolor, and Iris Virginica. You can download the dataset from `http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data`. Implement Principal Component Analysis in Python on your own using only NumPy functions (no `sklearn`). Use the first and second principal components to plot this dataset in 2 dimensions. Use different colors for each of the three classes in your plot. Share your observations in your report.

- Now, use `sklearn`'s inbuilt `t-SNE` function to visualize the same data. Include these in your plots, and compare the plots with the PCA plots. Do you see any differences/similarities? Share them in your report.
- Now, compare PCA and t-SNE on the SwissRoll dataset (you can use the inbuilt `sklearn` function to generate this dataset using `http://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_swiss_roll.html`. Now, what do you see? Share your observations and inferences in your report.

  Deliverables:

  (a) Your well-documented/verbatose code for PCA implementation. *(4 marks)*

  (b) Report with the required plots and inferences: (i) PCA for Iris; (ii) PCA vs t-SNE for Iris; and (iii) PCA vs t-SNE for SwissRoll dataset. *(1 + 1 + 2 = 4 marks)*

2. **Ridge Regression and LASSO:** *(4 + 3 = 7 marks)* Start by creating a design matrix for regression with $m = 150$ examples, each of dimension $d = 75$. We will choose a true weight vector θ that has only a few non-zero components:

   - Let $X \in \mathbb{R}^{m \times d}$ be the "design matrix" where the ith row of X is $x_i \in \mathbb{R}^d$. Construct a random design matrix X using `numpy.random.rand()` function.
   - Construct a true weight vector $\theta \in \mathbb{R}^{d \times 1}$ as follows: Set the first 10 components of θ to 10 or -10 arbitrarily, and all the other components to zero.
   - Let $y = (y_1, \cdots, y_m)^T \in \mathbb{R}^{m \times 1}$ be the response. Construct the vector $y = X\theta + \epsilon$, where ϵ is an $m \times 1$ random noise vector generated using `numpy.random.randn()` with mean 0 and standard deviation 0.1.
   - Split the dataset by taking the first 80 points for raining, the next 20 points for validation, and the last 50 points for testing.

   Note that we are not adding an extra feature for the bias in this problem. By construction, the true model does not have a bias term.
   By construction, we know that our dataset admits a sparse solution. We now want to evaluate the performance of ridge regression (i.e. L_2-regularized linear regression) on this dataset.

   - Run ridge regression on this dataset. Choose the λ that minimizes the square loss on the validation set. For the chosen λ, examine the model coefficients. Report on how many components with true value 0 have been estimated to be non-zero, and vice-versa (don't worry if they are all nonzero). Now choose a small threshold (say 10^3 or smaller), count anything with magnitude smaller than the threshold as zero, and repeat the report. (For running ridge regression, you can use `scipy.optimize.minimize`. For debugging purposes, you can compare your results to what you get from `sklearn.linear_model.Ridge`, but not use it directly.)
   - Now, run the inbuilt `sklearn.linear_model.Lasso` function on the same setup (ensure no randomness bias creeps in), and compare the same. What do you observe?

     Deliverables:

     (a) Your well-documented/verbatose code for Ridge Regression implementation. *(4 marks)*

     (b) Report with the results of ridge regression as required, and comparison with the results from Lasso. *(3 marks)*