CSCI 5521: Introduction to Machine Learning (Spring 2020)[1]

Homework 0

Problem 1: Linear regression learns a linear function of feature variables to predict the response. In this problem, you will derive the closed-form solution of the least-square formulation of linear regression.

1. The standard least-square problem is to minimize the following objective function,

$$\underset{\boldsymbol{w}}{\text{minimize}} \quad ||\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}||^2,$$

   where $\boldsymbol{X} \in \mathbb{R}^{n \times m}$ $(n \geq m)$ represents the feature matrix, $\boldsymbol{y} \in \mathbb{R}^{n \times 1}$ represents the response vector and $\boldsymbol{w} \in \mathbb{R}^{m \times 1}$ is the vector variable of the linear coefficients. This is a convex objective function of $\boldsymbol{w}$. Derive the optimal $\boldsymbol{w}$ by setting the derivative of the function wrt w to zero to minimize the objective function.

2. In practice, a L_2-norm regularizer is often introduced to the least-square objective function, called Ridge Regression, to overcome the ill-posed problems where the hessian matrix is not positive definite. The objective function of ridge regression is defined as

$$\underset{\boldsymbol{w}}{\text{minimize}} \quad ||\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}||^2 + \lambda||\boldsymbol{w}||^2,$$

   where $\lambda > 0$. This objective function is strictly convex. Derive the solution of the ridge regression problem to find the optimal $\boldsymbol{w}$.

Problem 2: Consider a coin with probability of heads equal to $\Pr(\text{H}) = p$ and probability of tails $\Pr(\text{T}) = 1 - p$. You toss it 5 times and get outcomes H,H,T,T,H.

1. What is the probability of observing the sequence H,H,T,T,H in five tosses. Also give the formula for the natural logarithm of this probability. Your formulas should be a function of p.

2. You have a box containing exactly 2 coins, one fair with $p = 1/2$ and one biased with $p = 2/3$. You choose one of these two coins at random with equal probability, toss it 5 times and get the outcome H,H,T,T,H.

   (a) Give the *joint* probability that the coin chosen was the fair coin ($p = 1/2$) *and* the outcome was H,H,T,T,H.

   (b) Give the *joint* probability that the coin chosen was the biased coin ($p = 2/3$) *and* the outcome was H,H,T,T,H.

[1]Instructors: Rui Kuang (kuang@umn.edu). TAs: Tianci Song (song0309@umn.edu) and Ruyuan Wan (wanxx199@umn.edu)

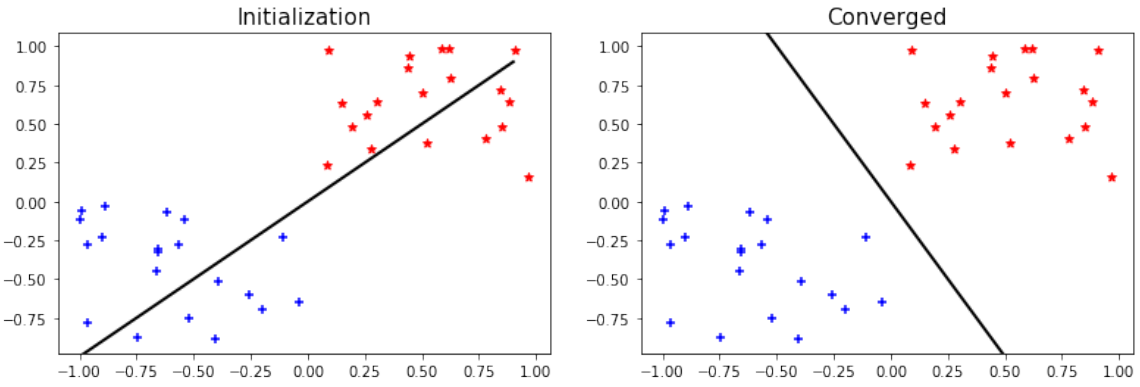
3. What should the bias $p = \Pr(\text{H})$ be to maximize the probability of observing H,H,T,T,H, and what is the corresponding probability of observing H,H,T,T,H (i.e., what is the maximum likelihood estimate for p), assuming p were unknown? Show the derivation. Hint: maximize the *log* of the function.

Problem 3: Below is the pseudo-code of perceptron algorithm for binary classification that we discussed in the class,

```
1 w = w_0.
2 Do Iterate until convergence (no mistakes)
3     For each sample (x^t, r^t)
4         If(⟨w, x^t⟩ r^t ≤ 0)
5             w = w + r^t x^t
```

where the dataset is $X = \{(\boldsymbol{x}^t, r^t)\}_{t=1,2,\ldots,N}$ and $\boldsymbol{w}_0$ is a random initialization.

1. Implement the perceptron algorithm and test it on the provided data. To begin, load the file `data1.mat` into Python. $\boldsymbol{X} \in \mathbb{R}^{40\times 2}$ is the feature matrix of 40 samples in 2 dimensions and $y \in \mathbb{R}^{40\times 1}$ is the label vector $(+1/-1)$. Initialize $\boldsymbol{w}$ to be vector $[1, -1]$. Visualize all the samples (use different colors for different classes) and plot the decision boundary defined by the initial w.
   Now, run your perceptron algorithm on the given data. How many iterations does it take to converge? Plot the decision boundary defined by the $\underline{\text{w}}$ returned by the perceptron program.
   Hint: To load data in MATLAB format, you may consider to use the function *loadmat*, which is included in the `io` module of `scipy` package. To visualize the samples you could use the function *scatter()*, which is included in the `pyplot` module of `matplotlib` package. Plotting the boundary is equivalent to plot the line $\langle \boldsymbol{w}, \boldsymbol{x} \rangle = 0$. Therefore, you could first generate a vector a to be your x-axis, then compute the y-axis vector b as $b = -\frac{w(1)a}{w(2)}$. Once the plot is generated you could use *xlim()* and *ylim()* functions in `pyplot` module of `matplotlib` package to make sure your axes are in a right range. When you are done your plots will look like the following figures:



2. The perceptron algorithm assumes that the samples from the two classes are linearly separable. Now let's look at a linearly non-separable case. Load the file `data2.mat` into Python and run your perceptron algorithm with $\boldsymbol{w} = [1, -1]$. Can the perceptron algorithm converge? Explain why. To solve the non-separable case, we can introduce a "soft" linear classifier to tolerate errors. It turns out we can solve the following Linear Programming problem:

$$\begin{aligned} \underset{\boldsymbol{w},\xi^t}{\text{minimize}} \quad & \sum_t \xi^t \\ \text{subject to} \quad & r^t(\boldsymbol{w}^T\boldsymbol{x}^t) \geq 1 - \xi^t \\ & \xi^t \geq 0. \end{aligned}$$

where ξ^t is the error to tolerate which also needs to be minimized. The function *linprog()* from `optimize` module of `scipy` package can be used for the problem. Now, run the following Python code to solve the LP problem on `data2.mat`.

```
import numpy as np
from scipy.optimize import linprog
m, n = np.shape(X)
X = np.hstack((X, np.ones((m, 1))))
n = n + 1
f = np.append(np.zeros(n), np.ones(m))
A1 = np.hstack((X*np.tile(y, (n, 1)).T, np.eye(m)))
A2 = np.hstack((np.zeros((m,n)), np.eye(m)))
A = -np.vstack((A1, A2))
b = np.append(-np.ones(m), np.zeros(m))
x = linprog(f,A,b)
w = x['x'][0:n]
```

Apply this algorithm to `data2.mat`, visualize the data and plot the boundary by the w returned by LP.

Submission

- **Things to submit:**

  1. hw0_sol.pdf: a document contains all the derivations of Problem 1&2 and the three plots asked by Problem 3.
  2. MyPerceptron.py: a Python function defined as *MyPerceptron*(X, y, $w0$) with w and *step* returned, where X is the feature matrix, y is a label vector and $w0$ is the initialization of the parameter w. In the output, w is the parameter found by perceptron and *step* represents the number of steps the algorithm takes to converge. The function should also display the plot of samples and boundary. Note that only `numpy` pacakge and functions mentioned above are allowed to use in this assignment.
  3. Zip all the files into a single zipped file and name it as your name.

- **Submit**: All material must be submitted electronically via Canvas. This homework will not be graded but required as a proof of satisfying the prerequisites for taking the class.