CS 412 — Introduction to Machine Learning — Fall 2019

Assignment 3

Department of Computer Science, University of Illinois at Chicago

Instructor: Xinhua Zhang

Due: Nov 15, 2019 by 5 PM (CST)

Instructions

This is an **individual** assignment**.** Your mark will be out of **130**, and it contributes 8% to your final course score.

What and where to submit:

1. The solution to the written questions (Q1 – Q2) should be submitted to **Gradescope**. You can submit PDF or image. It can be either typed using WORD or latex, or scanned copies of handwritten submissions provided that the handwriting is neat and legible. CamScanner is a great app for scanning on your smartphone with OCR. Please specify clearly which pages correspond to which question.

2. The solution to the programming questions must be submitted to **Blackboard** as a single zipped file (.zip or .tar) **named**

$\boxed{\text{Surname\_UIN.zip} \quad \text{or} \quad \text{Surname\_UIN.tar}}$

where Surname is your last name and UIN is your UIC UIN.

This zip/tar file should include the following files:

a) A single Portable Document Format (PDF) file, which answers Question 3. The file name should be $\boxed{\text{Surname\_UIN.pdf}}$, where Surname is your last name and UIN is your UIC UIN. It must be typed using WORD or latex.

b) **Code**: All the code you wrote – in a form that the TA can run it. Please include plenty of comments. Put all code in one folder named Code/

c) **ReadMe**: A ReadMe file that explains to your TA how to run your code. Name the file as ReadMe under the Code/ folder.

How to submit on Blackboard: Go to the class web site, and folder **Assessment**. The entry "Assignment 3 (coding)" in this folder will have a link through which you can upload your abovementioned zip/tar file for Q3 (programming question). **Do NOT submit your solutions to Q1-Q2 on Blackboard; they should go to Gradescope**.

Late policy: Late submissions will not be accepted in any case, unless there is a documented personal emergency. Arrangements must be made with the instructor as soon as possible after the emergency arises, preferably well before the deadline.

Resubmission: Both Blackboard and Gradescope accept resubmission (upload a new version) until the deadline. Grading will be based on the last version uploaded.

Programming Language: **Python only**.

Do NOT submit a Jupyter Notebook. Please submit your code in a standalone form so that the TA can run it directly.

The grading will be based on the output of executing your code on some test examples, along with the performance of algorithms that you plot in the assignment submission. Clarify and readability of your code will also count.

Cheating and Plagiarism: All assignments must be done individually. Remember that plagiarism is a university offence and will be dealt with according to university procedures. Please refer to the corresponding UIC policies: http://dos.uic.edu/docs/Student%20Disciplinary%20Policy.pdf

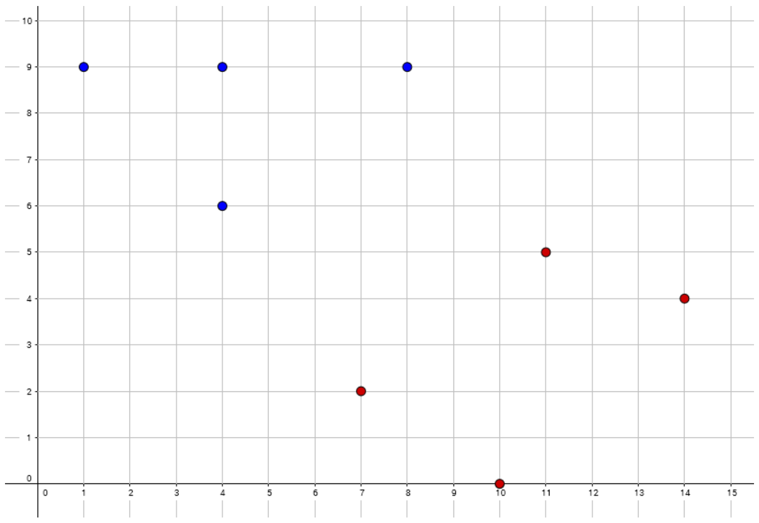
Latex primer: http://ctan.mackichan.com/info/lshort/english/lshort.pdf

Problem 1. Decision Trees (25 points) Consider the following data set comprised of three binary input attributes (A_1, A_2, and A_3) and one binary output:

Example	A_1	A_2	A_3	Output y
$\mathbf{x}_1$	1	0	0	0
$\mathbf{x}_2$	1	0	1	0
$\mathbf{x}_3$	0	1	0	0
$\mathbf{x}_4$	1	1	1	1
$\mathbf{x}_5$	1	1	0	1

Use the algorithm in Figure 9.3 (page 219) to learn a decision tree for these data. Show the computations made to determine the attribute to split at each node.
Refer to the decision tree question in Tutorial 3 for an example. Also read the decision tree example on Blackboard, under "Lecture Notes".

Problem 2. Hard Margin Support Vector Machine (50 points)



Consider the dataset above where $\mathbf{x}_{1,\cdots,4} = \{(7,2),(10,0),(11,5),(14,4)\}$ belong to the red class (positive) and $\mathbf{x}_{5,\cdots,8} = \{(1,9),(4,6),(4,9),(8,9)\}$ belong to the blue class (negative).
SVM tries to find a hyperplane:

$$w_0 + w_1x_1 + w_2x_2 + \ldots + w_kx_k = 0,$$

which has the maximum margin. For two dimensional data, the hyperplane can also be written as

$$w_0 + w_1x_1 + w_2x_2 = 0.$$

We want to find the hard margin SVM solution for the problem, i.e., the value of parameters w_0, w_1 and w_2.

(a) Draw the data points above, and sketch the maximum-margin hyperplane and also the marginal hyperplanes (the hyperplanes parallel to the maximum-margin hyperplane which pass the nearest points to the maximum-margin hyperplane).
Write down the margin ρ (i.e., the distance from the decision boundary to the margin boundary). **(15 points)**

(b) Choose three support vectors, write out the system of equations for those support vector data points: $y_i(w_0 + w_1 x_{i,1} + w_2 x_{i,2}) = 1$, and solve for w_0, w_1, and w_2. **(15 points)**

(c) Will the solution change if we remove the following point(s) from the dataset?: **(10 points)**

(i) $\mathbf{x}_1 = (7, 2)$ **(2 point)**

(ii) $\mathbf{x}_7 = (4, 9)$ **(2 point)**

(iii) $\mathbf{x}_1 = (7, 2)$ and $\mathbf{x}_2 = (10, 0)$ **(2 point)**

(iv) $\mathbf{x}_1 = (7, 2)$ and $\mathbf{x}_3 = (11, 5)$ **(2 point)**

(v) $\mathbf{x}_3 = (11, 5)$ and $\mathbf{x}_6 = (4, 6)$ **(2 point)**

(d) Will the solution change if we add the following point(s) to the dataset? **(10 points)**

(i) $(5, 0)$ with class = red **(2 points)**

(ii) $(2, 0)$ with class = red **(2 points)**

(iii) $(0, 3)$ with class = blue **(2 points)**

(iv) $(9, 3)$ with class = red, and $(5, 8)$ with class = blue **(2 points)**

(v) $(8, 4)$ with class = red, and $(5, 6)$ with class = blue **(2 points)**

Problem 3. Logistic Regression Classification (Programming) (55 points)

In this problem, we will attempt to identify spam e-mail using the Logistic Regression model. You will need to download `Programming.zip` from Blackboard, which contains a sample code `logreg.py`, along with `spambase-train.csv` and `spambase-test.csv` datasets. This data originally comes from https://archive.ics.uci.edu/ml/datasets/Spambase with some modifications.

Unlike the previous sms spam dataset which contains the sequence of words, the spambase dataset consists of continuous variables/features extracted from email data. These include the frequency of certain words and characters.

Let i enumerate/index different samples and let j enumerate different features/variables. Let $\mathbf{w}$ be the parameter of a Logistic Regression model. In Logistic Regression, the probability can be calculated as follows:

$$P(Y = 0|\mathbf{x}, \mathbf{w}) = \frac{1}{1 + \exp(w_0 + \sum_j w_j x_j)}, \quad P(Y = 1|\mathbf{x}, \mathbf{w}) = \frac{\exp(w_0 + \sum_j w_j x_j)}{1 + \exp(w_0 + \sum_j w_j x_j)} \tag{1}$$

For simplicity, we ignore w_0 in our formulation. Instead, we add a variable that is always 1 to our data. Therefore, we can use the following equivalent notation:

$$P(Y = 0|\mathbf{x}, \mathbf{w}) = \frac{1}{1 + \exp(\mathbf{w} \cdot \mathbf{x})}, \quad P(Y = 1|\mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w} \cdot \mathbf{x})}{1 + \exp(\mathbf{w} \cdot \mathbf{x})} \tag{2}$$

Logistic Regression tries to maximize the log-likelihood of data:

$$\ell(\mathbf{w}) = \ln \prod_i P(y_i|\mathbf{x}_i, \mathbf{w}) = \sum_i \left[y_i \ln\left(\frac{\exp(\mathbf{w} \cdot \mathbf{x}_i)}{1 + \exp(\mathbf{w} \cdot \mathbf{x}_i)}\right) + (1 - y_i) \ln\left(\frac{1}{1 + \exp(\mathbf{w} \cdot \mathbf{x}_i)}\right)\right] \tag{3}$$

$$= \sum_i \left[y_i \left(\mathbf{w} \cdot \mathbf{x}_i\right) - \ln\left(1 + \exp(\mathbf{w} \cdot \mathbf{x}_i)\right)\right] \tag{4}$$

We will try to find the maximum likelihood estimation above using gradient ascent optimization. The gradient can be computed as follows:

$$\frac{\partial \ell(\mathbf{w})}{\partial w_j} = \sum_i \left[y_i x_{i,j} - \frac{x_{i,j}\exp(\mathbf{w}\cdot\mathbf{x}_i)}{1+\exp(\mathbf{w}\cdot\mathbf{x}_i)} \right] = \sum_i x_{i,j}\left(y_i - \frac{\exp(\mathbf{w}\cdot\mathbf{x}_i)}{1+\exp(\mathbf{w}\cdot\mathbf{x}_i)} \right), \tag{5}$$

where $x_{i,j}$ is the j-th feature of $\mathbf{x}_i$. We start from an arbitrary $\mathbf{w}$ (typically a vector of straight 0). Then, in every iteration of gradient ascent, we update $\mathbf{w}$ using the following rule:

$$w_j \leftarrow w_j + \eta \frac{\partial \ell(\mathbf{w})}{\partial w_j}, \tag{6}$$

where η is the learning rate we set.

(a) Download the file `Programming.zip` from Blackboard, unzip it, and find `logreg.py`. It contains hints for implementing gradient ascent, and the learning rate was set to 10^{-3}. Complete the code by adding the gradient and update calculations to this methods. Train the Logistic Regression classifier using different learning rates $\{10^0, 10^{-2}, 10^{-4}, 10^{-6}\}$ using `spambase-train.csv` and evaluate its predictions using `spambase-test.csv`, both included in `Programming.zip`.

For each learning rate, plot one figure with two curves, showing how the training and testing accuracy of the model (y-axis) evolve as a function of the number of iterations taken so far (x-axis). You need to produce four figures in total. Explain the effect of choosing different learning rates. Submit your source code. **(27 points)**

(b) In practice, regularization is usually employed when building Logistic Regression models to avoid overfitting. In this part we will use L2 regularization. Our objective function is modified to be the log-likelihood minus regularization term:

$$\ell(\mathbf{w}) - \frac{1}{2}\lambda\|\mathbf{w}\|^2 = \sum_i \left[y_i\left(\mathbf{w}\cdot\mathbf{x}_i\right) - \ln\left(1+\exp(\mathbf{w}\cdot\mathbf{x}_i)\right) \right] - \frac{1}{2}\lambda\|\mathbf{w}\|^2 \tag{7}$$

Note that the partial derivative of the regularization term can be computed as the following:

$$\frac{\partial(\frac{1}{2}\lambda\|\mathbf{w}\|^2)}{\partial w_j} = \lambda w_j \tag{8}$$

Implement the regularized logistic regression model. Compare the performance of regularized logistic regression model using different regularization coefficients λ i.e. $\lambda = 2^k$ where $k \in \{-8. -6, -4, -2, 0, 2\}$. Provide a plot of training and testing accuracy (two curves) where the x axis represent the value of k, i.e. $\{-8, -6, -4, -2, 0, 2\}$. Compare your training/testing accuracy with non-regularized logistic regression model. Submit your code. **(28 points)**

Notes

If you want to check whether your code computes the gradient correctly, you can leverage a tool called scipy.optimize.check_grad
`https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.check_grad.html`
It checks the correctness of a gradient function by comparing it against a (forward) finite-difference approximation of the gradient. The example in the above link clearly explains its usage.
Since it can be costly to run finite-difference, you may want to extract a subset of the dataset in order to run check_grad. Say just use 4 features and 4 data points.