Fall 2020: Machine Learning
Homework Set 1

Instructor: Lu Sheng
College of Software, Beihang University

Instructions

- Email the TAs (contact them in the QQ/WeChat group of this course) a zipped file named as `yourname_id.zip`, which includes
  (1) an electronic version (saved as `name_id.pdf`, can be printed or handwritten) of your answer,
  (2) a Jupyter Notebook file (saved as `name_id.ipynb`) includes the codes, results, and/or discussions if required. For more details about python and Jupyter Notebook, as well as some useful packages (such as `numpy` and `matplotlib`), please check the following link.
- The deadline is 23:59 pm, December 14, 2020.

Section A: Background (Answering this section is optional; if you do submit answers to these questions, there will be extra credits up to 30 points)

1. Write a python function `birthday_prob(n)` that computes the probability that, in a group of n people at least two share the same birthday (assume that $n \geq 2$). Using this function, compute the minimum size of the group such that this probability is at least 0.5.

2. Let $X_1, X_2, \ldots, X_n$ be independent random variables, each with a Uniform distribution over (0,1):
$$f(x) = \begin{cases} 1 & 0 < x < 1 \\ 0 & \text{otherwise} \end{cases}$$
Find (a) $\mathbb{E}[\max(X_1, X_2, \ldots, X_n)]$ and $\mathbb{E}[\min(X_1, X_2, \ldots, X_n)]$.

3. *Monty Hall problem.* You are a contestant in the Monty Hall game show. In this game, there are three doors. Behind one door is a brand new Corvette; there are goats behind the other two doors. Monty Hall asks you to select a door; after you have chosen one, he opens one of the other two doors to reveal a goat behind it. You can now choose to stick with you original choice, or to switch to the remaining door.
   (a) What should you do? Will it make a difference if your switch?
   (b) Run a thousand trials of this experiment in python. Can you explain the results?

4. (a) Let $\mathbf{x}$ be a random vector following multivariate Gaussian distribution with
$$\mathbb{E}[\mathbf{x}] = \begin{bmatrix} 10 \\ 5 \end{bmatrix}, \quad \text{cov}(\mathbf{x}) = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$$
Write an expression for PDF of $\mathbf{x}$ that does not use matrix notation, *i.e.*, if $\mathbf{x} = [x_1, x_2]^\top$, write the joint PDF as $p(x_1, x_2)$.

(b) Let $\mathbf{A}, \mathbf{B}$ be $p \times q$ matrices and $\mathbf{x}$ be a random $q \times 1$ vector. Prove that

$$\text{cov}(\mathbf{Ax}, \mathbf{Bx}) = \mathbf{A}\text{cov}(\mathbf{x})\mathbf{B}^\top$$

where $\text{cov}(\mathbf{u}, \mathbf{v}) = \mathbf{E}[(\mathbf{u} - \mathbb{E}[\mathbf{u}])(\mathbf{v} - \mathbb{E}[\mathbf{v}])^\top]$ is the cross-covariance matrix between random vectors $\mathbf{u}$ and $\mathbf{v}$, while $\text{cov}(\mathbf{u}) = \mathbf{E}[(\mathbf{u} - \mathbb{E}[\mathbf{u}])(\mathbf{u} - \mathbb{E}[\mathbf{u}])^\top]$ is the covariance matrix for $\mathbf{u}$.

Section B: (Answering this section is compulsory, 100 points in total)

1. (a) Consider two nonnegative numbers a and b, show that if $a \leq b$, then $a \leq (ab)^{1/2}$.
   (b) Using the result in (a) to show that, if the decision regions of a two-class classification problem are chosen to minimize the probability of error, this probability will satisfy

$$p(\text{error}) \leq \int \{p(\mathbf{x}, \omega_1)p(\mathbf{x}, \omega_2)\}^{1/2} \text{d}\mathbf{x},$$

2. Suppose that we have the option either to assign the pattern to one of c classes, or to reject it as being unrecognizable. Let

$$\lambda(\alpha_i|\omega_j) = \begin{cases} 0 & i = j, i, j = 1, \ldots, c, \\ \lambda_r & i = c + 1 \\ \lambda_s & \text{otherwise,} \end{cases}$$

   where λ_r is the loss incurred for choosing the $(c + 1)$-th action (rejection).
   (a) Derive the decision rule in order to minimize the risk.
   (b) What happens if $\lambda_r = 0$? What happens if $\lambda_r > \lambda_s$?

3. Implement a linear regression. Start by implementing the following functions:

   - a function `theta = linear_regress(y, X)` where `X` and `y` are $n \times d$ and $n \times 1$ matrices respectively. This function trains a linear regression model on a training set of n samples, each of which is a d-dimensional vector. The ground truth values for the examples are in `y` and are scalar values. The function should return `theta` as the final model parameter. Note that we are NOT explicitly including the offset parameter but instead rely on the feature vectors to provide a constant component, *i.e.*, $y = \boldsymbol{\theta}^\top \mathbf{x}$.
   - a function `y_hat = linear_pred(theta, X_test)` that predicts the `y_hat` from the test data `X_test`.

   For this problem, we have provided you a custom-created dataset named `p3.mat`, which can be loaded by `scipy.io.loatmat()`.

   (a) Load data from `p3.mat`, which should include load the matrices `y_noisy`, `y_true`, `X_in`. The `y` vectors are $n \times 1$ while `X_in` is a $n \times 3$ matrix with each row corresponding to a point in $\mathbb{R}^3$. The y_{true} vectors correspond to the ideal y values, generated directly from the "true" model (whatever it may be) without any noise. In contrast, the y_{noisy} vectors are the actual, noisy observations, generated by adding Gaussian noise to the y_{true} vectors. You should use y_{noisy} for any estimation. y_{true} is provided only to make it easier to evaluate the error in your predictions. You would not have y_{true} in any real task.
   (b) As introduced in the lecture notes, the feature mapping can substantially affect the regression results. We will consider two possible feature mappings:

$$\boldsymbol{\phi}_1(x_1, x_2, x_3) = [1, x_1, x_2, x_3]^\top \tag{1}$$

$$\boldsymbol{\phi}_2(x_1, x_2, x_3) = [1, \log x_1^2, \log x_2^2, \log x_3^2]^\top \tag{2}$$

Implement a python function `feature_mapping` to transform the input data matrix into a matrix

$$\mathbf{X} = \begin{bmatrix} \boldsymbol{\phi}(\mathbf{x}_1)^\top \\ \boldsymbol{\phi}(\mathbf{x}_2)^\top \\ \vdots \\ \boldsymbol{\phi}(\mathbf{x}_n)^\top \end{bmatrix}. \tag{3}$$

For example, `X = feature_mapping(X_in, 1)` would get you the first feature representation. Using your completed linear regression functions, compute the mean squared prediction error for each feature mapping (two numbers).

4. Implement a Perceptron classifier. Start by implementing the following functions:

   - a function `perceptron_train(X, y)` where `X` and `y` are $n \times d$ and $n \times 1$ matrices respectively. This function trains a Perceptron classifier on a training set of n samples, each of which is a d-dimensional vector. The labels for the examples are in `y` and are 1 or -1. The function should return `[theta, k]`, the final classification vector and the number of updates performed, respectively. You may assume that the input data provided to your function is linearly separable. No offset is used for simplicity.
   - a function `perceptron_test(theta, X_test, y_test)` where `theta` is the classification vector to be used. `X_test` and `y_test` are $m \times d$ and $m \times 1$ matrices respectively, corresponding to m test examples and their true labels. The function should return `test_err`, the fraction of test examples which were misclassified.

   For this problem, we have provided you two custom-created datasets. The dimension d of both the datasets is 2, for ease of plotting and visualization.

   (a) Load data from `p4a.mat` (it contains `p4aX` and `p4ay`), and train your Perceptron classifier on it. Using the function `perceptron_test`, ensure that your classifier makes no errors on the training data. What is the angle between `theta` and the vector $(1, 0)^\top$? What is the number of updates k_a required before the Perceptron algorithm converges?

   (b) Repeat the above steps for data loaded from `p4b.mat` (it contains `p4bX` and `p4by`). What is the angle between `theta` and the vector $(1, 0)^\top$? What is the number of updates k_b?

   (c) For parts (a) and (b), compute the geometric margins (the minimum distance of samples in one class to the decision hyperplane), γ^a_{geo} and γ^b_{geo}, of your classifiers with respect to their corresponding training datasets. Recall that the distance of a point $\mathbf{x}_t$ from the line $\boldsymbol{\theta}^\top \mathbf{x} = 0$ is $|\frac{\boldsymbol{\theta}^\top \mathbf{x}_t}{\|\boldsymbol{\theta}\|}|$.

   (d) Plot the data (as points in the X-Y plane) from part (a), along with decision boundary that your Perceptron classifier computed. Create another plot, this time using data from part (b) and the corresponding decision boundary. Your plots should indicate the class of each point (*e.g.*, by choosing different colors or symbols to mark the points from the two classes).

Feedback? (Optional)

Please help us make the course better. If you have any feedback for this assignment, we'd love to hear it!