Machine Learning (CS771A)
Homework 1 (Due date: Aug 31, 2016, 11:59pm)

Instructions

- Each late submission will receive a 10% penalty per day for up to 5 days. No submissions will be accepted after the 5th late day.
- We will only accept electronic submissions and the main writeup must be as a PDF file. If you are handwriting your solutions, please scan the hard-copy and convert it into PDF. Your name and roll number should be clearly written at the top. In case you are submitting multiple files, all files must to be zipped and **submitted as a single file** (named: your-roll-number.zip). Please do not email us your submissions. Your submissions have to be uploaded in a common directory (link to the upload website and details of upload procedure will be sent soon).

Problem 1 (10 marks)

(Distance from Means) Given training data $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ for a classification problem (assuming binary labels $y_n \in \{-1, +1\}$), show that, for the $y = \text{sign}[f(\boldsymbol{x})]$ decision rule in the "distance from means" classification model, $f(\boldsymbol{x})$ can be written as $f(\boldsymbol{x}) = \sum_{n=1}^N \alpha_n \langle \boldsymbol{x}_n, \boldsymbol{x} \rangle + b$, where $\boldsymbol{x}$ denotes the feature vector of the test example. For this form of the decision rule, give the expressions for the α_n's and b.

Problem 2 (15 marks)

(Classes as Gaussians) Consider a model for binary classification where data in class "+1" and class "-1" is modeled using D-dimensional Gaussian distributions $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_+, \boldsymbol{\Sigma})$ and $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_-, \boldsymbol{\Sigma})$, respectively. For simplicity, let's assume the covariance matrix $\boldsymbol{\Sigma}$ to be the same for both the classes. Suppose we have already learned the means $\boldsymbol{\mu}_+$, $\boldsymbol{\mu}_-$ and the covariance matrix $\boldsymbol{\Sigma}$ from some training data. Now, given a test example $\boldsymbol{x}$, we wish to predict its class. To do so, we will use the following rule: assign $\boldsymbol{x}$ to the class under which it has a higher probability. Show that this rule can be written as $y = \text{sign}[f(\boldsymbol{x})]$ where $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$ and give the expressions of $\boldsymbol{w}$ and b. Under what condition this rule reduces to a "distance from means" classifier?

Problem 3 (10 marks)

(Importance-Weighted Linear Regression) The standard linear regression model (and its ridge variant) gives equal importance to each training example and the model's error on each example gets penalized equally. Suppose you assign each example $(\boldsymbol{x}_n, y_n)$ an "importance weight" or a "cost" c_n (assumed non-negative) so that the model gets penalized on some examples more than the others. Write down the objective function for this importance-weighted linear regression model and derive the closed-form expression for the regression weight vector that will be learned by the model. Using a regularizer is optional.

Problem 4 (15 marks)

(Noise as Regularizer) We have seen regularized models where large values of weights are penalized by imposing some penalty on the norm (e.g., squared ℓ_2 norm) of the weight vector. There is actually another way to accomplish the same effect *without* explicit regularization: by adding some noise to each of inputs.

Consider the standard linear regression model $y = \boldsymbol{w}^\top \boldsymbol{x}$ with sum-of-squared-errors loss function

$$L(\boldsymbol{w}) = \sum_{n=1}^{N} \{y_n - \boldsymbol{w}^\top \boldsymbol{x}_n\}^2$$

Suppose we add Gaussian noise ϵ_n with zero mean and variance σ^2 to each *input* $\boldsymbol{x}_n \in \mathbb{R}^D$ (so it becomes $\boldsymbol{x}_n + \epsilon_n$) and call the new loss function $\tilde{L}(\boldsymbol{w})$. Show that minimizing the *expectation* of the "noisy" loss function , i.e., $\mathbb{E}[\tilde{L}(\boldsymbol{w})]$ is equivalent to minimizing the original sum-of-squared-errors error $L(\boldsymbol{w})$ for noise-free inputs, plus an ℓ_2 regularizer on $\boldsymbol{w}$. The expectation $\mathbb{E}$ is w.r.t. the Gaussion noise distribution for which the following properties hold: $\mathbb{E}[\epsilon_n] = 0$ and $\mathbb{E}[\epsilon_m \epsilon_n] = \delta_{mn}\sigma^2\mathbf{I}$ (where $\delta_{mn} = 1$ if $m = n$, and 0 otherwise).

Problem 5 (10 marks)

(Decision Trees for Regression) When constructing Decision Trees for classification, we prefer to split on features that divide a node such that the set of labels at the children nodes is as "pure" as possible, i.e., we want each child node to consist of examples such that one label dominates the other label(s). Entropy/information-gain can quantify the purity in the classification setting. Suggest a good criteria to choose a feature to split on if we were doing *regression* instead of classification (so the labels are real-valued instead of discrete labels in classification)? Your criteria should somehow quantify the homogeneity/diversity of the set of *real-valued* labels of the examples at each node. You may define it using equations or state it in words (but be precise).

Problem 6 (40 marks)

For this problem, the task is to implement a multi-class extension of the "distance from means" classifier (lecture-2) and apply it on the MNIST digits classification problem (10 classes, representing digits 0-9). You will be working on a small subset of the data, provided in the `mnist_hw1.mat` file. Each image is represented as a 784-dimensional vector of pixel intensities (between 0-255). Loading the data in MATLAB will show:

- `dataX`: Cell array containing the training data. Within this, `dataX{1}` denotes the feature matrix (each row is an example) of all training images for digit 0, `dataX{2}` denotes the same for digit 1, and so on.
- `X_test`: This is the feature matrix of test data with each row denoting the feature vector of a test image.
- `Y_test`: This is a vector of containing the ground truth class of each of the test images. You will use it to test the accuracy of your implementation.

Implement the distance from means classifier (using Euclidean distance) and experiment with the following setting: Vary the number of training examples per class from 50 to 2000 with increments of 50. For each case, apply the learned model on the test data and compute the classification accuracy. Plot a curve of the test accuracy vs the number of training examples per class. What trend do you see in the plot? Please give a brief explanation. Submit your code and the plot, along with the writeup for the other problems, in a single zip file.

Note: Your implementation should ideally be in MATLAB but since we are not providing you any skeleton MATLAB code, if you want to use any other language, e.g., Python/R, that should be fine too.