Intro to Machine Learning (CS771A, Autumn 2018)
Practice Problem Set 1

Problem 1

(Prototype based Classification) 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 prototype based classification, $f(\boldsymbol{x})$ 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

(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 prototype based classifier?

Problem 3

(Getting used to matrix-vector notations) First, verify (or convince yourself) that you can also write down the sum of squared errors loss function (assuming no regularization) for linear regression, $\mathcal{L}(\boldsymbol{w}) = \sum_{n=1}^N (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2$, directly as $\mathcal{L}(\boldsymbol{w}) = (\boldsymbol{y} - \mathbf{X}\boldsymbol{w})^\top (\boldsymbol{y} - \mathbf{X}\boldsymbol{w})$. Here $\mathbf{X}$ is the $N \times D$ feature matrix and $\boldsymbol{y}$ is the $N \times 1$ response vector. Note that this is also the same as $||\boldsymbol{y} - \mathbf{X}\boldsymbol{w}||^2$, i.e., the ℓ_2 squared norm of the error vector.

Using this (equivalent) form of the loss function, find the least squares solution for $\boldsymbol{w}$ and verify that you get the same solution as we saw in the class. The purpose of this simple exercise is to get yourself used to working directly with matrices and vectors, and taking derivatives w.r.t. vectors (the derivatives needed here should be obvious - basically linear and quadratic terms in $\boldsymbol{w}$, but if needed, you may also refer to the Matrix Cookbook for results on derivatives, including derivatives w.r.t. matrices, which we will actually need in the next problem).

Problem 4

(Multi-output Regression) Consider the multi-output regression in which each output $\boldsymbol{y}_n \in \mathbb{R}^M$ in a real-valued vector, rather than a scalar. Assuming a linear model, we can model the outputs as $\mathbf{Y} = \mathbf{X}\mathbf{W}$, where $\mathbf{X}$ is the $N \times D$ feature matrix and $\mathbf{Y}$ is $N \times M$ response *matrix* with row n being $\boldsymbol{y}_n^\top$ (note that each column of $\mathbf{Y}$ denotes one of the M responses), and $\mathbf{W}$ is the $D \times M$ **weight matrix**, with its M columns containing the M weight vectors $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_M$. Let's define a squared error loss function $\sum_{n=1}^N \sum_{m=1}^N (y_{nm} - \boldsymbol{w}_m^\top \boldsymbol{x}_n)^2$, which is just the usual squared error but summed over all the M outputs. Firstly, verify that this can also be written in a more compact notation as $\text{TRACE}[(\mathbf{Y} - \mathbf{X}\mathbf{W})^\top (\mathbf{Y} - \mathbf{X}\mathbf{W})]$. Using this form of the loss function, derive the solution for $\mathbf{W}$. It should look very similar to the solution of linear regression with single output.

Note: Due to the form of the problem, you can also break the estimation problem for $\mathbf{W}$ into M independent linear regression problems for $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_M$. But writing the objective function in the trace form makes it more amenable to directly solving the problem by taking derivatives w.r.t. the *matrix* $\mathbf{W}$.

Problem 5

(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. You can try with and without ℓ_2 regularization on the weight vector.

Problem 6

(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 replace each input $\boldsymbol{x}_n$ by a "noisy" input $\boldsymbol{x}_n + \epsilon_n$, where ϵ_n is a vector having the same size as $\boldsymbol{x}_n$, and each entry of ϵ_n is assumed drawn i.i.d. from a Gaussian with zero mean and variance σ^2 (so you can write $\epsilon_n \sim \mathcal{N}(0, \sigma^2\mathbf{I})$. Let's use $\tilde{L}(\boldsymbol{w})$ to denote the new loss function defined on the noisy inputs. 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 $\mathbb{E}[\epsilon_m \epsilon_n^\top] = \delta_{mn}\sigma^2\mathbf{I}$ (where $\delta_{mn} = 1$ if $m = n$, and 0 otherwise).