**E0 270 Machine Learning** *Due:* April 5, 2016

# Assignment 2

**Instructions**

- For this assignment **we will only accept one single soft copy (pdf)** as almost all the problems are programming. You can write the theory answers in latex/word or can scan (or take photo of) handwritten answers, and include it in the soft copy.
- You have to include **all the plots in one single soft copy**. We will not consider any plots that are given in the code folders and are not a part of the soft copy.
- Submit your soft copy using the submission link on the website before 11:59 pm on 5 Apr. Code also needs to be submitted as a single zip file.
- The data and helper code for the programming problems have been provided in the zip file 'hw1-data-code.zip' .

1. **Least Squares Regression.** Write a small piece of code to implement (linear) least-squares regression (with and without regularization). The input to your code is a training data set $(\mathbf{X}, \mathbf{y})$, where $\mathbf{X}$ is an $m \times d$ matrix of training examples and $\mathbf{y}$ is now an $m$-dimensional vector of real-valued labels associated with the instances in $\mathbf{X}$. You are provided with two synthetic data sets for this problem, each split into training and test sets.

   (a) Data Set 1 (1-dimension). Run (unregularized) least-squares regression on the training set provided and measure the mean squared error of the learnt linear function on both the training and test examples (you can use the provided code `mean_squared_error.m` to compute the errors). In a single figure, draw a plot of the learnt linear function (input instance on the $x$-axis and the predicted value on the $y$-axis), along with a scatter plot depicting the true label associated with each test instance. You can use the provided script to plot this. (See folder `Problem-1\Data-set-1` for the relevant files.)

   (b) Data Set 2 (250-dimension). Run L-2 regularized least-squares regression on this training set. Repeat step (a) of Problem 6 in Assignment 1 on this data set, i.e. plot the training error and test error as a function of lambda. Also, perform cross validation and report the average cross-validation errors for each lambda. Use the mean squared error as the performance measure. For regularization use the range $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100\}$ for selecting the regularization parameter $\lambda$. (See folder `Problem-1\Data-set-2` for the relevant files.)

   *Hints:* Do not forget to include a bias (threshold) term $b$ in the model. This can be done as follows: add an extra column of 1's to (the right of) the training data matrix $\mathbf{X}$ and create an augmented matrix $\tilde{\mathbf{X}}$ of dimension $m \times (n+1)$; then the solution to the unregularized least squares regression (with the bias term included) is given by $[\widehat{w}; \widehat{b}] = (\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^\top \mathbf{y}$; in the case of the $L_2$-regularized least squares regression (with the bias term included), the solution takes the form $[\widehat{w}; \widehat{b}] = (\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}} + \lambda m \mathbf{R})^{-1} \tilde{\mathbf{X}}^\top \mathbf{y}$, where $\mathbf{R}$ is the $n \times n$ identity matrix augmented with an extra column of 0's to the right and an extra row of 0's to the bottom (i.e., $\mathbf{R}$ is a $(n+1) \times (n+1)$ matrix with $\mathbf{R}_{ij} = 1$ if $i = j < n+1$ and 0 otherwise). (If the matrix $\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}$ is not invertible, you can take the pseudoinverse of the matrix instead of the usual matrix inverse; the MATLAB function `pinv` can be used for pseudoinverse computation.)

2. **Gaussian Process Regression.**

   Write a piece of code that implements Gaussian process regression. The input to your code is the same as for least squares regression. You are provided with one dataset for this problem in the folder `Problem-2\Data`. The instances are one dimensional and the labels are of the form

$$y = f(x) + \epsilon$$

where $f(x) = x * \sin(x)$ and $\epsilon$ is drawn from a normal distribution with 0 mean and unknown variance. Use the rbf kerbel, i.e.

$$K(x, x') = \exp(-\frac{\|x - x'\|^2}{2\sigma^2}) .$$

You can use $\sigma = 1$ for solving this problem or you chose to learn this parameter using maximum-likelihood estimation as described in Section 6.4.3 of Bishop' book. Use the script provided in folder `Problem-2\Code` to plot the true x*sin(x) function, given labels for the train instances, predicted labels for the test instances and standard deviation for each test prediction obtained using gaussian process regression, in a single figure.

*Hint:* For learning the precision parameter $\beta$ of the Gausssian process either you can use maximum-likelihood estimation or you can use a fixed value like $\beta = 10$.

3. **Generative models for binary biclassification.** In generative models for classification we model the conditional label distribution $\mathbf{P}(y|\mathbf{x})$ directly. One common practice is to use a sigmoid function to model this distribution. More specifically, we assume

$$\mathbf{P}(y|\mathbf{x}) = \frac{1}{1 + e^{-y(\mathbf{w}^\top \mathbf{x} + b)}}$$

for some parametesr $\mathbf{w}$ and $b$. If one does not know the parameter $\mathbf{w}$ and $b$, one generally uses maximum likelihood estimation to learn the parameter from data.

   (a) Write the log-likelihood expression for the parameters $\mathbf{w}$ and $b$ given a binary classification data set $(\mathbf{X}, \mathbf{y})$, where $\mathbf{X}$ is an $m \times d$ matrix ($m$ instances, each of dimension $d$) and $y$ is an $m$-dimensional vector (with $y_i \in \{\pm 1\}$ being a binary label associated with the $i$-th instance in $\mathbf{X}$).

   (b) Suppose that you maximized the log-likelihood and found $\widehat{\mathbf{w}}$ and $\widehat{b}$ as the optimal parameters. Write an expression for the best classifier $h(\mathbf{x}) : \mathcal{X} \rightarrow \{\pm 1\}$ in terms of $\widehat{\mathbf{w}}$ and $\widehat{b}$.

   (c) Write a piece of code that maximizes the log-likelihood for a given dataset: your program should take the training data $(\mathbf{X}, \mathbf{y})$ as input and output an optimal $d$-dimensional vector $\widehat{\mathbf{w}}$ and bias term $\widehat{b}$.

   For this problem, you are provided a spam classification data set[1], where each instance is an email message represented by 57 features and is associated with a binary label indicating whether the email is spam (+1) or non-spam (−1). The data set is divided into training and test sets. The goal is to learn from the training set a classifier that can classify new email messages in the test set.

   (d) Use above implementation to learn a classifier from 10% of the training data, then 20% of the training data, then 30% and so on upto 100% (separate files containing $r$% of the training data are provided in the folder `Problem-3\Spambase\TrainSubsets`). In each case, measure the classification error on the training examples used, as well as the error on the given test set (you can use the provided code `classification_error.m` to help you compute the errors). Plot a curve showing both the training error and the test error (on the $y$-axis) as a function of the number of training examples used (on the $x$-axis). Such a plot is often called a **learning curve**. (Note: the training error should be calculated only on the subset of examples used for training, not on all the training examples available in the given data set.)

   (e) Incorporate a $L_2$-regularizer in the above log-likelihood expression by subtracting the term $\lambda \|\mathbf{w}\|_2^2$ (taking the regularization parameter $\lambda$ as an additional input). Maximize the regularized objective on the entire training set for different values of $\lambda$ in the range $\{10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$. Plot the training and test error achieved by each value of $\lambda$ ($\lambda$ on the $x$-axis and the classification error on the $y$-axis) and identify the value of $\lambda$ that achieves the lowest test

[1] UCI Machine Learning Repository: `http://archive.ics.uci.edu/ml/datasets/Spambase`.

error (resolving ties in favor of the smallest value of $\lambda$). Now, select $\lambda$ from the same range through 5-fold cross-validation on the training set (the train and test data for each fold are provided in a separate folder `Problem-3\Spambase\CrossValidation`); report the average cross-validation error (across the five folds) for each value of $\lambda$ and identify the value of $\lambda$ that achieves the lowest average cross-validation error (again resolving ties in favor of the smallest value of $\lambda$).

*Hints:* For maximizing the log-likelihood the `fminunc` MATLAB function for unconstrained optimization will be useful. Also, do not forget to include the bias (threshold) terml.

4. **Ranking using multi-class perceptron**

Consider a ranking problem in which a set of items are to be ranked in an order (for example, a set of movies are to be ranked for a movie database like IMDB, a set of research papers are to be ranked for deciding the best paper awards). In this problem we follow what is called a point-wise approach in ranking. In this approach we rate each item on a scale from 1 to $k$, (for example, a movie is rated on a scale of 1 to 10 by users in IMDB, or a research paper is rated on a scale of 1 to 5 by reviewers). Once we have learnt the ratings for each item, we rank the items based on some aggregated form of these ratings.

Let each item belong to $\mathbb{R}^d$ and the set of ratings be $\mathcal{Y} = \{1, \ldots, k\}$. Given a training set of ratings $S = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$, the goal is to learn a rating model $h : \mathbb{R}^d \times \mathcal{Y}$.

In this problem we learn a function $h$ by learning a vector $\mathbf{w} \in \mathbb{R}^d$ and a set of $k$ thresholds $b_1 \leq b_2 \cdots b_{k-1} \leq b_k$. We assume $b_k = \infty$. The rating of an item $x \in \mathbb{R}^d$ is predicted to be $r$ if $b_{r-1} \leq \mathbf{w}^\top \mathbf{x} < b_r$. More formally, $h(x) = \min_{r \in \mathcal{Y}} \{r : \mathbf{w}^\top \mathbf{x} - b_r < 0\}$.

In this problem you are asked to implement a modified version of the multi-class perceptron algorithm given below, to learn the weights and the thresholds in an online manner.

---

Algorithm **Multiclass Perceptron (MP) for Ranking**

---

**Initialize:** $\mathbf{w}^1 = \mathbf{0} \in \mathbb{R}^d$ and $b_1, \cdots b_{k-1} = 0$, $b_k = \infty$

For $t = 1, \ldots, T$:

- Receive item $\mathbf{x}^t \in \mathbb{R}^d$
- Predict $\widehat{y}^t = \min_{r \in \mathcal{Y}} \{r : \mathbf{x}^\top \mathbf{w}^t - b_r^t < 0\}$
- Receive true label $y^t \in \mathcal{Y}$
- Update: If $\widehat{y}^t \neq y^t$ then
  1. For $r = 1, ..., k-1$ : If $y^t \leq r$, then $y_r^t = -1$ else $y_r^t = 1$.
  2. For $r = 1, ..., k-1$ : If $(\mathbf{x}^\top \mathbf{w}^t - b_r^t) y_r^t \leq 0$, then $\tau_r^t = y_r^t$ else $\tau_r^t = 0$.
  3. Update $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + (\sum_r \tau_r^t) \mathbf{x}^t$
     For $r = 1, ..., k-1$ : update $b_r^{t+1} \leftarrow b_r^t - \tau_r^t$

  else

  $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t$ and $b_r^{t+1} \leftarrow b_r^t \;\; \forall r \in \mathcal{Y}$

---

You are required to apply the above algorithm on the dataset provided in `Problem-4\Data` where the items are two dimensional and $k = 5$. The instances and labels are obtained via the `get_instance(`$t$`)` and `get_label(`$t$`)` routines. Run the algorithm for 1 epoch. Plot the ratio of the absolute error divided by the total number of examples seen as a function of the number of examples seen. The `absolute_error(`$y$`,`$\widehat{y}$`)` routine can be used to compute the absolute error ( See folder `Problem-4\Code`).