E0 270 Machine Learning *Due:* Jan 24, 2012

Assignment 1

TAs: Arun Rajkumar (arun_r@csa), Raman Sankaran (ramans@csa)

Submission instructions: Assignment solutions must be prepared in LaTeX and the PDF file together with any code must be submitted **both electronically *and* as a hard copy in class, before the start of class on the date above**. The link for submitting your assignments electronically can be accessed by clicking this. Any plots/tables/figures must be included in your PDF file/hard copy. Questions about the assignment can be directed to the TAs above. Extra credit problems are optional.

---

1. **Warm-up.** Let $\mathcal{X} = \mathbb{R}^2$, and suppose examples $(\mathbf{x}, y) \in \mathcal{X} \times \{\pm 1\}$ are generated randomly and independently as follows:

   - instance $\mathbf{x} = (x_1, x_2) \in \mathcal{X}$ is drawn randomly from a uniform distribution over $[-10, 10] \times [-3, 3]$;
   - a fixed function $g : \mathcal{X} \to \{\pm 1\}$ is applied to $\mathbf{x}$ and $g(\mathbf{x})$ is computed, where

   $$\begin{aligned} g(\mathbf{x}) &= \max(g_0(\mathbf{x}), g_1(\mathbf{x}), g_2(\mathbf{x}), g_3(\mathbf{x})) \\ g_0(\mathbf{x}) &= \mathrm{sign}(\sin(x_1) - x_2) \\ g_1(\mathbf{x}) &= \mathrm{sign}(1 - ((x_1 - 8)^2 + 2(x_2 - 2.25)^2)) \\ g_2(\mathbf{x}) &= \mathrm{sign}(1 - ((x_1 - 1)^2 + 2(x_2 - 2)^2)) \\ g_3(\mathbf{x}) &= \mathrm{sign}(1 - ((x_1 + 8)^2 + 2(x_2 - 1.75)^2)) \end{aligned}$$

   with

   $$\mathrm{sign}(z) = \begin{cases} 1 & \text{if } z > 0 \\ -1 & \text{otherwise;} \end{cases}$$

   - with probability 0.9, the label y for instance $\mathbf{x}$ is set to be equal to $g(\mathbf{x})$, and with probability 0.1, y is set to $-g(\mathbf{x})$.

   This defines a joint probability distribution on $\mathcal{X} \times \{\pm 1\}$. Now say that, given some examples drawn from this distribution, you learn a (deterministic) classifier $h : \mathcal{X} \to \{\pm 1\}$ that predicts labels of new instances in $\mathcal{X}$. What is the best possible error (in expectation) that the classifier can achieve on a randomly drawn example from the same distribution? Explain your answer.

2. **Nearest neighbor – simulated data.**

   (a) Write a small piece of MATLAB code to implement the k-NN algorithm on a given data set $(\mathbf{X}, \mathbf{y})$, where $\mathbf{X}$ is an $m \times n$ matrix (m instances, each of dimension n) and $\mathbf{y}$ is an m-dimensional vector (with $y_i \in \{\pm 1\}$ being a binary label associated with the i-th instance in $\mathbf{X}$): your program should take as input the training set $(\mathbf{X}, \mathbf{y})$, parameter k, and a new instance $\mathbf{x}$ (an n-dimensional vector), and should return as output a predicted label for $\mathbf{x}$ ($+1$ or -1). You can use the provided code template `kNN.m` as a starting point.

   (b) Fix $k = 5$, and run the k-NN algorithm on the data set `data_simulated.mat`, which is drawn according to the distribution described in Problem 1 above, using 10% of the training data, then 20%, then 30%, and so on up to 100% (in each case, you can use the first r% of the training examples in the given training set, or you can use a random subset of r% training examples). 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 actually used for training, not on all the training examples available in the given data set.)

(c) Repeat the above experiment with $k = 1$. What do you observe? How do the training and test errors differ from those obtained in part (b) above?

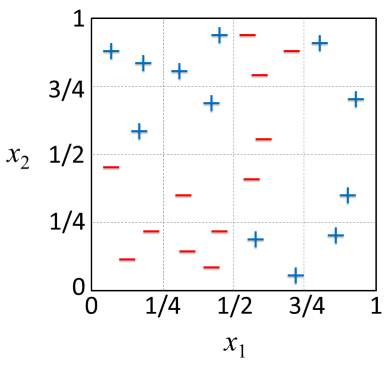
(d) In this experiment, you will investigate the effect of increasing k. Fix the number of training examples to be $m = 5000$ (i.e. use the full training set), and evaluate the train and test errors obtained with k-NN for $k = 1, 9, 49, 99, 499$. Plot the training error and test error (on the y-axis) as a function of k. What do you observe? Is this what you expected? How can you explain your findings?

(e) Use the provided code `decision_boundary_NN.m` to visualize the decision boundaries obtained with $k = 1, 49, 499$ (include the resulting figures in your assignment). Intuitively, which of these models has the most 'complex' decision boundary, and which has the most 'simple' (smooth) decision boundary? When learning, given that your goal is to perform well on new test data, how would you select a good value of k based on the training data?

3. **Nearest neighbor – handwritten digit classification data.** In this problem, you are given a real world handwritten digit recognition data set in `data_digit.mat`. The data set consists of 16×16 images, represented as 256-dimensional vectors, each containing a handwritten digit 7 or 9, together with corresponding labels ($+1$ for digit 7 and -1 for digit 9). The data set is divided into a training set and a test set. The goal is to learn from the training examples a classifier that can classify new images of digits 7 and 9. Use the code you wrote in Problem 2(a) above to learn a 5-NN classifier for this problem. Again, learn classifiers using 10%, 20%, and so on up to 100% of the training data, and plot a learning curve as before (including both training and test errors). Give 4 examples of images from the test set that are misclassified by the final classifier that uses all 100% of the training examples (include the images in your assignment).

4. **Decision trees.**

(a) Let $\mathcal{X} = [0, 1]^2$, and consider a training set consisting of 24 labeled examples in $\mathcal{X} \times \{\pm 1\}$ as follows:



Draw two decision trees of depth ≤ 3 that classify all the above training examples perfectly.

(b) If you used the entropy measure to learn a decision tree from the data in part (a), what would the root node be? Show all your calculations (in general, for each variable, you would consider all transition points in the training data as possible split points; in this case, restrict yourself to considering split points at only 1/4, 1/2, and 3/4 for each variable).

(c) Repeat part (b) above using the Gini index (again, restrict yourself to considering the split points indicated in part (b)).

(d) **(Optional – no need to turn in solution to this part)** Write a piece MATLAB code, say `DT_learner.m`, to learn a decision tree classifier of depth at most k from data of the form described in Problem 2(a). In this case, given training data $(\mathbf{X}, \mathbf{y})$ and parameter k, your code must output a decision tree model (which can be stored in a file); a separate piece of code, say `DT_classifier.m`, can then take as input a new test instance $\mathbf{x}$ and apply the learned model (which can be read from the file) to the instance to produce a prediction of $+1$ or -1. Use your code to learn decision tree classifiers for various values of k from the simulated data given in Problem 2. Plot the training and test errors as a function of k, for $k = 3, 4, 5, 6, 10, 20$. Modify the code `decision_boundary_NN.m` (or the code `decision_boundary_SVM.m` given in Problem 5) to visualize the decision boundaries produced by the decision tree classifiers for different values of k.

5. **Support vector machines.**

   (a) Write a piece of MATLAB code to learn an SVM classifier from a data set of the form described in Problem 2(a): your program should take as input the training set $(\mathbf{X}, \mathbf{y})$, parameter C, kernel type (which can be 'linear', 'poly' or 'rbf'; definitions for these are provided in the associated files), and kernel parameter (which you can take here to be a single real number r; this is used as the degree parameter d in the polynomial kernel $K(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}' + 1)^d$, and as the width parameter σ in the RBF kernel $K(\mathbf{x}, \mathbf{x}') = e^{-\|\mathbf{x}-\mathbf{x}\|^2/2\sigma^2}$); the program should return as output an SVM model (consisting of the kernel type and parameters, the support vectors, and the coefficients α corresponding to the support vectors). You can use the provided template `SVM_learner.m` as a starting point; this automatically saves the learned model to a mat file `SVM_model.mat`. Use MATLAB's quadratic program solver `quadprog` in your code.

   (b) Write a piece of MATLAB code that reads in a learned SVM model from the saved mat file (`SVM_model.mat`) and uses it to classify a test instance $\mathbf{x}$. You can use the provided template `SVM_classifier.m` as a starting point.

   (c) Use the code you wrote in parts (a-b) above to learn SVM classifiers from the simulated data given in Problem 2, using 5 different kernels: linear, polynomial of degree $d = 2$, polynomial of degree $d = 3$, RBF with $\sigma = 1$, and RBF with $\sigma = 4$; for each kernel, use 3 different values of C: $0.01, 1, 100$. Tabulate the training and test errors for the 15 resulting classifiers. Use the provided code `decision_boundary_SVM.m` to visualize the decision boundary for the best classifier learned for each kernel (include the figures in your assignment). What are your observations? In general, what would be a good way to automatically select the kernel, kernel parameters, and parameter C using only the training data?

   (d) Use your code to learn SVM classifiers for the handwritten digit classification task in Problem 3, using an RBF kernel with $\sigma = 1$, and 5 different values of C: $0.01, 0.1, 1, 10, 100$. Plot the resulting training and test errors as a function of C. How does this compare with the results of the 5-NN classifier learned in Problem 3?

6. **Extra credit.** Consider a binary classification problem in which the error or cost associated with misclassifying a positive example (such as a diseased patient) as negative (normal) is c times higher than the cost associated with misclassifying a negative example (normal individual) as positive (disease). How would you modify the support vector machine algorithm to learn a classifier in this setting?