E0 270 Machine Learning *Due:* Feb 7, 2012

Assignment 2

TAs: Goutham Tholpadi (gtholpadi@csa), Arun Rajkumar (arun_r@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 electronic submission can be accessed here. 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 $(\mathbf{x}, y) \in \mathcal{X} \times \mathbb{R}$ 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 $g : \mathcal{X} \to \mathbb{R}$ is applied to $\mathbf{x}$ and $g(\mathbf{x})$ is computed, where
   $$g(\mathbf{x}) \quad = \quad \sin(x_1) - x_2\,;$$
   - the label y for instance $\mathbf{x}$ is set to $y = g(\mathbf{x}) + \eta$, where $\eta \in \mathbb{R}$ is drawn randomly from the standard normal distribution $\mathcal{N}(0, 1)$ (independently of $\mathbf{x}$).

   This defines a joint probability distribution D on $\mathcal{X} \times \mathbb{R}$. Show that among all possible real-valued functions $f : \mathcal{X} \to \mathbb{R}$ that you could use for predicting labels of new instances in $\mathcal{X}$, g has the lowest expected squared error, i.e. show that
   $$\mathbf{E}_{(\mathbf{x},y)\sim D}\left[(g(\mathbf{x}) - y)^2\right] \quad \leq \quad \mathbf{E}_{(\mathbf{x},y)\sim D}\left[(f(\mathbf{x}) - y)^2\right] \quad \text{for all } f : \mathcal{X} \to \mathbb{R}\,.$$
   What is the value of this lowest possible error, i.e. what is the value of $\mathbf{E}_{(\mathbf{x},y)\sim D}\left[(g(\mathbf{x}) - y)^2\right]$? Explain your answer.

2. **Least squares regression and ridge regression.**

   (a) Write a small piece of MATLAB code `linear_least_squares_learner.m` to implement the linear least squares regression 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 $y_i \in \mathbb{R}$ being a real-valued label associated with the i-th instance in $\mathbf{X}$): your program should take as input the training set $(\mathbf{X}, \mathbf{y})$, and should return as output a weight vector $\mathbf{w} \in \mathbb{R}^n$ representing a linear regression model; the weight vector $\mathbf{w}$ should be saved to a file.

   (b) Write a small piece of MATLAB code `linear_predictor.m` that reads in a learned linear regression model (weight vector) from a file and uses it to predict the (real-valued) label of a test instance $\mathbf{x}$: your program should take as input a new instance $\mathbf{x}$ (an n-dimensional vector) and the name of the file from which the model is to be read, and should return as output a predicted real-valued label for $\mathbf{x}$.

   (c) Run your code on the provided data set `regression_dataset.mat`, and report the average squared error on both the training set and the test set (you can use the provided code `squared_error.m` to help you compute the errors).

   (d) Adapt your code from part (a) to implement the linear ridge regression algorithm, which adds a term $\frac{\lambda}{2}\|\mathbf{w}\|_2^2$ to the objective of the basic linear least squares regression algorithm; name the new code `linear_ridge_learner.m`. Your code should take as input a training data set $(\mathbf{X}, \mathbf{y})$ as in part (a) above, and the parameter $\lambda > 0$, and should return as output a linear regression

model as above, which again can be saved to a file; the same code you wrote in part (b) can then be used to read in this model and apply it to make predictions for new instances. Perform 5-fold cross-validation using the provided folds (`regression_folds.mat`) for 5 different values of λ: $0.01, 0.1, 1, 10, 100$; for each value of λ, tabulate the squared error for each fold and compute the average error over all the folds. Which value of λ gives the lowest cross-validation (average) error? Use this value of λ to re-train the model on the complete training set, and report the training and test errors. Compare this to the train/test errors obtained with the other values of λ. Did the cross-validation procedure select the best value of λ in terms of test set error? How does the performance compare with the linear least squares regression model?

(e) Adapt your code from part (d) to implement the kernel ridge regression algorithm, allowing for linear, polynomial, and RBF kernels as was done in the support vector machines problem in Assignment 1; name the new code `kernel_ridge_learner.m`. Similarly adapt your code from part (b) to read in a model using any of these kernels; name this `kernel_predictor.m`. Repeat the experiment in part (d) above using a cubic (degree 3 polynomial) kernel. Which value of λ gives the lowest cross-validation error in this case? What are the corresponding training and test errors – does the cross-validation procedure select the best value of λ? How does the performance compare with the linear ridge regression model?

3. **Support vector regression.**

(a) Write a piece of MATLAB code `SVR_learner.m` to learn a support vector regression (SVR) model from a data set of the form described in Problem 2(a): your program should take as input the training set $(\mathbf{X}, \mathbf{y})$, parameters C, ϵ, kernel type (which can be 'linear', 'poly' or 'rbf'; definitions for these were provided in the files associated with the SVM problem in Assignment 1), and kernel parameter (which as in Assignment 1, you can take 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 SVR model (consisting of the kernel type and parameters, the support vectors, and the coefficients associated with the support vectors) and save the output to a file. You should be able to use the code `kernel_predictor.m` you wrote in Problem 2(e) above to read in the model from the file and apply it to predict the label of a new instance $\mathbf{x}$.

(b) Fix $\epsilon = 0.1$, and apply your code to learn a linear SVR model for the data set used in Problem 2, using 5-fold cross-validation as above to select a value of C from $0.01, 1, 100$. For each of these values of C, tabulate the squared error for each fold and compute the average error over all the 5 folds. Which value of C gives the lowest cross-validation (average) error? As above, use this value of C to re-train the model on the complete training set, and report the training and test errors. Compare this to the train/test errors obtained with the other values of C. Did the cross-validation procedure select the best value of C in terms of test error? How does the performance compare with the linear least squares regression and linear ridge regression models from Problem 2?

(c) Repeat part (b) above with a cubic (degree 3 polynomial) kernel. How does the performance compare with the linear SVR model from part (b), and with the cubic kernel ridge regression model from Problem 2(e)?

4. **Multiclass nearest neighbor classification.** Write a small piece of MATLAB code `kNN_multiclass.m` to implement the k-NN multiclass classification 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 $y_i \in \{0, \ldots, r-1\}$ being a multiclass label associated with the i-th instance in $\mathbf{X}$): your program should take as input the training set $(\mathbf{X}, \mathbf{y})$, parameters k, r, and a new instance $\mathbf{x}$ (an n-dimensional vector), and should return as output a predicted label for $\mathbf{x}$ (in $\{0, \ldots, r-1\}$). You can use the `kNN.m` code you wrote in Assignment 1 as a starting point. Apply your code to the provided 10-class digit classification data set `data_digit_10class.mat` for $k = 1, 9, 99$ and report the corresponding training and test errors (you can use the provided code `multiclass_error.m` to help you compute the errors). Which of the three models gives the lowest test error? For this model, give 10 examples of images that are misclassified, indicating their true class labels in the data set as well as the wrong class labels predicted by the model.

5. **VC-dimension and generalization error.**

   (a) Let $\mathcal{X} = \mathbb{R}^2$, and consider let $\mathcal{H}$ be the class of all binary-valued functions on $\mathcal{X}$ that label all points within some rectangle (not necessarily axis-parallel) as $+1$ and all points outside the rectangle as -1. What is the VC-dimension of $\mathcal{H}$? Justify your answer.

   (b) Let $\mathcal{X} = \mathbb{R}^2$, and consider the class $\mathcal{H}$ of all mappings $h : \mathcal{X} \to \{\pm 1\}$ that can be realized using binary decision trees in which each node partitions based on a decision of the form 'is $x_j \leq \theta$?' for some $j \in \{1, 2\}, \theta \in \mathbb{R}$ (i.e. decision trees that create axis-parallel rectangular partitions of the form described in class). What is the VC-dimension of $\mathcal{H}$? Justify your answer.

   (c) Recall the binary classification data set `data_simulated.mat` provided in Assignment 1. Use the SVM code you wrote in Assignment 1 to learn a linear SVM classifier using 10%, 20%, and so on up to 100% of the training data in this data set; in each case, use 5-fold cross-validation to choose a value of C from $0.01, 1, 100$. In each case, compute the training error and test error of the learned classifier, and the 95% confidence interval you get using the VC-dimension based bound on the generalization error. Plot the training error, test error, the upper bound of the confidence interval, and the lower bound, as a function of the number of training examples m; also show the Bayes error (which you computed in Problem 1 of Assignment 1) in this plot. What are your observations?

6. **Extra credit.** Read the following paper on using error-correcting codes for reducing a multiclass classification problem to a series of binary classification problems:

   - Thomas G. Dietterich and Ghulum Bakiri. Solving Multiclass Learning Problems via Error-Correcting Output Codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995. Available from `http://www.jair.org/media/105/live-105-1426-jair.pdf`

   Implement the approach for the 10-class digit classification problem described in Problem 4 above using your favorite binary classification algorithm and an appropriate method for generating codewords (see the paper above). Include a detailed description of the choices made in your implementation. How does the performance compare with the multiclass k-NN approach implemented in Problem 4? **Optional:** How does the performance compare to the multiclass SVM approach of Crammer and Singer (2001) discussed in class (using different kernels)?

Practice problems (not to be turned in). The following problems are based on material to be covered in Lectures 7-8 and are for your practice. Solutions to these problems are not to be turned in, but you are strongly encouraged to attempt the problems for your own learning.

7. **PAC learning.** Let $\mathcal{X} = \mathbb{R}^2$, and let $\mathcal{H}$ be the class of all binary-valued functions on $\mathcal{X}$ that label all points within some axis-parallel rectangle as $+1$ and all points outside the rectangle as -1:

$$\mathcal{H} = \left\{ h : \mathcal{X} \to \{-1, 1\} \;\middle|\; h(x) = 1 \text{ if } x \in [a, b] \times [c, d] \text{ and } -1 \text{ otherwise, for some } a \leq b, c \leq d \right\}$$

   Is the class $\mathcal{H}$ learnable in the target function setting? If so, give an algorithm demonstrating this. Is $\mathcal{H}$ learnable in the general (agnostic) setting? Justify your answer.

8. **Boosting.**

   (a) Implement a simple 'decision stump' learning algorithm which, given a weighted training sample $S = ((\mathbf{x}_1, y_1, w_1), \ldots, (\mathbf{x}_m, y_m, w_m)) \in (\mathbb{R}^n \times \{\pm 1\} \times [0, 1])^m$ (where $\sum_{i=1}^m w_i = 1$, with w_i representing the 'weight' of the i-th training example), returns a 'decision stump' classifier $h : \mathbb{R}^n \to \{\pm 1\}$ of the form $h(\mathbf{x}) = \mathbf{1}(x_j \leq \theta)$ or $h(\mathbf{x}) = \mathbf{1}(x_j > \theta)$ for some $j \in \{1, \ldots, n\}, \theta \in \mathbb{R}$; use a simple weighted extension of the entropy criterion used in constructing decision trees for this purpose. Apply the algorithm to the simulated binary classification data set provided in Assignment 1. Now implement AdaBoost using this decision stump learner as your weak/base learning algorithm. Run the boosting algorithm for $T = 100$ rounds and plot the training and test errors (of the final classifier at each round t) as a function of the number of rounds t. How does

the performance compare to that of a single decision stump (learned from the original, unweighted sample)?

(b) Recall the 2-digit classification task in Problem 3 of Assignment 1. Consider crafting a small number (say 8-10) of hand-coded 'rules of thumb' for classifying each image as '7' or '9' (e.g. one such rule could be to test the average pixel intensity in the central region of the image, and classify as '9' if this is high and '7' otherwise). You can look at the training set in constructing these rules, but do not look at the test set. Tabulate the training and test errors of each of these simple rules. Now apply AdaBoost on top of a weak learner which at each round picks a single rule that gives the best performance according to the current distribution on the training sample, and evaluate the resulting performance (both training and test errors) at $T = 1, 2, 5, 10, 20, 50, 100$ rounds. How does the performance compare to that of the individual rules?

(c) Read the following paper as an example of a real-world application of boosting to a face detection problem:

- Paul Viola and Michael J. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (CVPR), 2001.