E0 270 Machine Learning *Due:* 11.59 pm, March 6, 2017; Total: 100 Mark

Assignment 2

Submission Instructions

1. Please submit the assignment on submission page linked from the course webpage.
2. Upload the source code files (Matlab/Python) for the programming assignment as a single compressed (zip) file. Follow the code structure provided in code-data.zip.
3. You should report the entire assignment as a soft copy (single pdf) generated using LaTeX. All graphs should be generated programmatically and should be included in the report with proper numbering.
4. Use the pdf submission link in the assignment submission page to submit your pdf.
5. Upon multiple submissions till the deadline, the most recent submission will overwrite the previous one.
6. Please download and use the code blocks provided in the assignment submission page for doing the programming assignments. Please follow the folder structure/template provided. Follow the same folder structure as given in the Matlab template, for Python implementations too.
7. Please make sure that the file formats are correct, i.e. code is in zip format and report is in pdf format. TA's will not evaluate defaulting assignment submissions.
8. Datasets are included in the data folder of respective problems in the hw2_code_data.zip file. Please submit only ZIP files (no other formats like rar, tar etc)
9. Questions about the assignment can be posted on the group email or directed to the TA @ `sharmistha@grad.cds.iisc.ac.in`

Some useful concepts and definitions

1. Training Set and Test Set: A typical machine learning model is trained using some data. The subset of data used for learning model parameters is called Training Set. Test set of the subset of data (it's intersection with train data is a null set), used to evaluate your model on unseen data. Good performance on the test set indicates generalisation capability of the model.
2. Cross Validation: Typically, the data is divided into two groups, train and test. The evaluations obtained in this case tend to reflect the particular way the data are divided up. The solution is to divide up data in such a way that, we test the model on each data point atleast once. This method is called cross-validation. Following is a sketch of the procedure.
   (a) Divide data randomly into k folds (subsets) of equal size.
   (b) Train the model on k-1 folds, use remaining one fold for testing.
   (c) Repeat this process k times so that all folds are used atleast once for testing.
   (d) Compute the average test performance on these k test sets.
3. Precision, Recall, Accuracy: Important metrics to evaluate the performance of learned algorithms. Read more here: `https://www.cs.cornell.edu/courses/cs578/2003fa/performance_measures.pdf`

Questions

1. (20 points) **Support Vector Machine.** Write a piece of code to learn an SVM classifier from the spam classification data set. Your program should take as input the training set $(\mathbf{X}, \mathbf{y})$, parameter C, kernel type (which can be 'linear', 'poly' or 'rbf'; use the provided code `compute_kernel.m` for computing these kernels), and kernel parameter (which you can take here to be a single real number r; this is used as the degree parameter q in the polynomial kernel $K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + 1)^q$, and as the width parameter σ in the RBF kernel $K(\mathbf{x}, \mathbf{x}') = e^{-\|\mathbf{x}-\mathbf{x}'\|_2^2/2\sigma^2}$); the program should return as output an SVM model (consisting of the kernel type and parameters, the support vectors, the coefficients corresponding to the support vectors and the bias term). You can use the provided template `SVM_learner.m` as a starting point. Use MATLAB's quadratic program solver `quadprog` to solve the SVM dual problem.

   (a) Run your implementation of the SVM learning algorithm with linear kernel on the spam classification data set (see folder `Problem-2\Spambase`), selecting C from the range $\{1, 10, 10^2, 10^3, 10^4\}$ through 5-fold cross-validation on the training set. Report the average cross-validation error on train and test set for each value of C in this range, using a plot with C value on the x-axis and the error on y-axis.

   (b) You are provided a 2-dimensional synthetic data set for this problem. Run your implementation of SVM on the training set provided using a linear, degree-2 polynomial, degree-3 polynomial and RBF kernel, selecting C from the range $\{1, 10, 10^2, 10^3, 10^4\}$ via 5-fold cross-validation on the training set; in the case of the RBF kernel, in addition to C, you will also have to choose the width paramter σ from the range $\{1/32, 1/4, 1, 4, 32\}$ using cross-validation. Report the average cross-validation error on test set (cv-test) for each value of C (and σ). Also, report the training and test errors (full training data: train.txt and full testing data test.txt) achieved by the best parameter value(s). Draw a 2-d scatter plot of the test (test.txt) instances and visualize how well the decision boundaries learnt using the different kernels separate the positive and negative test instances; you can use the provided code `decision_boundary_SVM.m` to visualize the decision boundaries.

   (c) Compare your results with previous assignment's Logistic Regression results with SVM results in part (a). State which algorithm performs better on the test set [best accuracy on test.txt by the algorithm]. Also, state why the better algorithm gives good results.

2. **Least squares regression and ridge regression.** [40 points]

   (a) Write 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 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` (use the full training set), 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) For ridge regression problem:
1.) At optimality show that there exists β^* such that $w = \sum_{i=1}^{N} \beta_i^* x_i$
2.) Let $w = \sum_{i=1}^{N} \beta_i x_i$ and restate the Ridge regression problem in β. Compute and state the optimal β^*.

(f) Adapt your code from part (d) to implement the kernel ridge regression algorithm (hint: proof written in part (e) of the problem should help you see the gram matrix term), allowing for linear, polynomial, and RBF kernels as was done in the support vector machines in problem 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.** [25 points]

(a) Write 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 in problem 1), and kernel parameter (which as in problem 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(f) 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. **K-means and Kernelized K-means [15 points]**

- Consider the synthetic dataset *data/synthetic/syndata2.txt* . Each row is a 2D feature-vector, and there are 500 of them. In each vector, the two components are the X and Y coordinates. Run the K-means algorithms on this dataset, using $K = 5$. Plot all the 500 datapoints, using a separate color for each cluster formed by K-means.
- Are the results satisfactory? If not, what do you think is the reason that K-means fails on this data?
- Now implement the Kernelized K-means algorithm, using the Kernel matrix (data/synthetic/syndata2-kernel.txt), using 5 clusters. Once again plot the results.
- One quantitative measure for clustering performance is the RAND index. This is can be calculated by the script *code/RAND.m*. Use the provided label files *data/synthetic/syndata2 − lab.txt*, to compute RAND indices of the 2 clusterings (K-means and Kernelized K-means) on syndata2.txt.