E0 270- Machine Learning
Assignment 1
Due date: 17 Feb 2014

1. **Bayes optimal classifier**

   Consider the case of Binary classification with the 0-1 loss, and instance space $\mathcal{X} = \mathbb{R}$. Calculate the Bayes classifier and Bayes error for the following distributions. Let $X \sim \mathcal{N}(0,1)$, and $h \in (0,1)$.

   (a) $P(Y = 1|X = x) = \mathbf{1}(x \geq 0)$

   (b) $P(Y = 1|X = x) = \max(0, \min(\frac{x+h}{2h}, 1))$

   (c) $P(Y = 1|X = x) = 0.4 * \mathbf{1}(x \in [0.1, 0.5]) + 0.4 * \mathbf{1}(x \in [0.3, 0.7])$

   Repeat the above for $X \sim \text{Uniform}([-1, 1])$ and $X \sim \text{Uniform}([-\frac{1}{2}, 1])$.

2. **Link between Fisher linear discriminant and least squares regression**

   Let $X = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ with $\mathbf{x}_i \in \mathbb{R}^d$. Let $C_1 \subseteq X$ and $C_{-1} \subseteq X$ denote the elements in X with positive and negative label respectively. Let $|C_1| = N_1$ and $|C_2| = N_2$. Consider the following line (hyperplane) fitting problem.

   Find $\mathbf{w} \in \mathbb{R}^d$ and $w_0 \in \mathbb{R}$ such that the 'prediction' at $\mathbf{x} \in \mathbb{R}^d$, given by $\mathbf{w}^\top \mathbf{x} + w_0$, is 'close' to $\frac{N}{N_1}$ if $\mathbf{x} \in C_1$ and is 'close' to $\frac{-N}{N_2}$ if $\mathbf{x} \in C_2$. Where 'closeness' refers to a small squared difference. i.e.

   $$(\mathbf{w}, w_0) = \text{argmin}_{\mathbf{a} \in \mathbb{R}^d, b \in \mathbb{R}} \sum_{\mathbf{x} \in C_1} \left(\mathbf{a}^\top \mathbf{x} + b - \frac{N}{N_1} \right)^2 + \sum_{\mathbf{x} \in C_2} \left(\mathbf{a}^\top \mathbf{x} + b + \frac{N}{N_2} \right)^2$$

   What is the relation of the obtained $\mathbf{w}$ with the Fisher linear discriminant on the data?

3. **Perceptron**

   (a) **Programming exercise:** Write a piece of MATLAB code that implements the Perceptron algorithm. Your program should take as input training set $(\mathbf{X}, \mathbf{y})$, and return a separating hyperplane through the origin. (You may assume it exists). Run the Perceptron algorithm on the 2-dimensional synthetic datasets `Synth2a` and `Synth2b`. (By running the algorithm on the dataset, we mean that you should cycle through the dataset repeatedly until no mistake is made on one whole cycle). Report the number of mistakes made by the algorithm. Give a reason by looking at the scatter plot of the dataset. (No need to submit the code.)

   (b) **Effect of dimension:** Assume you have a stream of d-dimensional datapoints, $\mathbf{x}_i$ and its labels $y_i \in \{+1, -1\}$. Let each *component* of $\mathbf{x}_i$ be within $[-1, 1]$. Let there exist a $\mathbf{u} \in \mathbb{R}^d$ such that $||\mathbf{u}|| = 1$ and $y_i \mathbf{u}^\top \mathbf{x}_i \geq \gamma$ for all i. Use the proof of Perceptron for giving a mistake bound in this case.

4. **Programming exercise: Effect of C on SVMs**

   Write a piece of MATLAB code to learn an SVM classifier from a 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

$K(\mathbf{x}, \mathbf{x}') = e^{-||x-x'||^2/2\sigma})$. 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.

Run the SVM program with linear kernels on the two, 2-d datasets `Synth1a` and `Synth1b` with C values ranging in $\{0.01, 0.1, 1, 10\}$. Plot the training points and the separating hyperplane for a total of 8 plots (you can use the provided code `decision_boundary_SVM.m`). Comment on the changes in the separating hyperplane with increase in C. Does increasing C to values greater than 10 affect the separating hyperplane in either of the datasets? Justify. (No need to submit the code.)

5. **Programming exercise: Support Vector Machine**

   Implement an SVM in MATLAB as before.

   Explanation of Cross-Validation: Many machine learning algorithms take several parameters as input. For example the SVM requires a C parameter and kernel parameters. These parameters are set via validation on a hold out set. A part of the training set is set aside during training. Models with various parameter values are constructed using the remaining part of the training. All these models are run on the remaining part and the model with the best performance on this set is returned as the answer. While this is a good strategy for cases with large training set, not using a significant part of the training set for training the model is detrimental with small training sets. One popular way to overcome this is via cross-validation. In k-fold-cross-validation one maintains k-folds of the training set. Where each fold is a partition of the training set (which has n samples) into two parts : one of size $\frac{k-1}{k}n$ and another of size $\frac{n}{k}$. The k-folds are such that the smaller $fracnk$ size part of each fold are disjoint. For every parameter value to be considered, one learns a model on all the folds using the larger $\frac{k-1}{k}n$ part as the training set and tests on the other $\frac{n}{k}$ size part. Thus a model corresponding to a parameter value gives k-accuracy numbers corresponding to each fold. The average of these k-numbers is taken as the accuracy estimate for the corresponding model. The best model is then chosen as the one with the highest accuracy estimate. The parameter values corresponding to the best model is then used for training a new model using all n samples and is returned.

   (a) Run your implementation of the SVM learning algorithm with linear kernel on the 57-dimensional spam classification data set `Spambase`. Selecting C from the range $\{10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3, 10^4\}$ through 5-fold cross-validation on the training set. Report the average cross-validation error for each value of C in this range and the training and test errors achieved by the best value of C.

   Hint: Randomly split the training set into 5 parts, and in each fold use 4 of the parts for training and the other for testing.

   (b) You are provided a 2-dimensional synthetic data set, `Synth3` 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 $\{10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3, 10^4\}$ 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 for each value of C and σ and the training and test errors achieved by the best parameter value(s). Also, draw a 2-d scatter plot of the test 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.

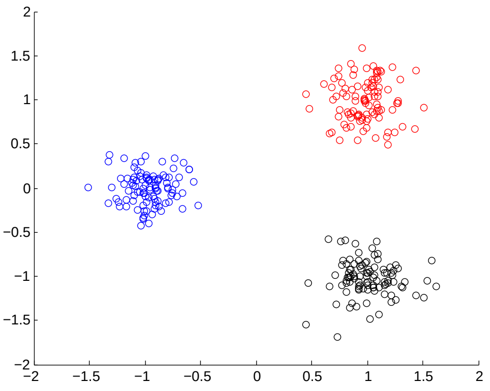
   Note: The train and test data are in separate files.

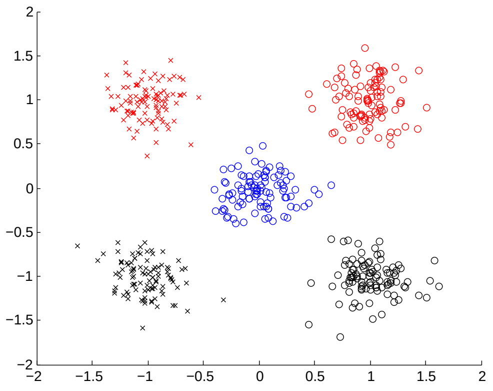
6. **Multiclass classification**

   (a) One of the most popular approaches for constructing multiclass learners is breaking the multiclass problem into several binary problems, and using binary learners for each of them during the training phase, and somehow aggregating the predictions of all the binary learners into a single multiclass prediction during testing. In particular two of these deserve special mention

- **One Vs All**: A k-class problem is broken into k binary problems. In binary problem i, the objective is to build a classifier that separates class i (+ve example), from all other classes (-ve example). In the prediction phase the instance is fed to the k learned binary models, and the class corresponding to the model which predicts the instance to be positive. (Ideally only one model fires for a given instance, in case more than one model fires or no model fires the right prediction is not clear)
- **All pairs**: A k-class problem is broken into $\binom{k}{2}$ binary problems, indexed by (i, j). In binary problem (i, j), the objective is to build a classifier that separates class i (+ve example), from class j (-ve example). In the prediction phase the instance is fed to the $\binom{k}{2}$ learned binary models, giving rise to a $k \times k$ binary matrix which can be viewed as the results of tournament between the k classes, where every class plays every other class exactly once. One standard way to make a multiclass prediction out of this matrix is to return the class which has 'won' the most 'matches'.

Which of the above 2 methods is better (in terms of accuracy) for the two multiclass problems, with 3 and 5 classes respectively, in below figure?. You may assume the base binary learner is a SVM with linear kernel. Justify qualitatively. Each color, and mark type in the figure denotes a different class.





(b) **Multiclass Perceptron:** The Perceptron is an online method for binary learning, the Multiclass Perceptron follows the Perceptron method very closely and is given below. In the Multiclass Perceptron, one maintains k vectors $\mathbf{w}_1, \ldots, \mathbf{w}_k$ in $\mathbb{R}^d$, each vector corresponds to a class. After receiving an instance $\mathbf{x}_t \in \mathbb{R}^d$ in round t, the Multiclass Perceptron predicts $\hat{y}_t = \text{argmax}_j \mathbf{w}_j^\top \mathbf{x}_t$. After receiving the truth $y_t \in \{1, \ldots, k\}$, it updates the vectors $\mathbf{w}_{y_t}$ and $\mathbf{w}_{\hat{y}_t}$.

Algorithm 1 Multiclass Perceptron

```
$\mathbf{w}_1 = \mathbf{w}_2 = \ldots = \mathbf{w}_k = \mathbf{0}$
for $t = 1$ to $T$
    receive $\mathbf{x}_t$
    predict $\hat{y}_t = \text{argmax}_j \mathbf{w}_j^\top \mathbf{x}_t$
    receive $y_t$
    if $y_t \neq \hat{y}_t$
        $\mathbf{w}_{y_t} = \mathbf{w}_{y_t} + \mathbf{x}_t$
        $\mathbf{w}_{\hat{y}_t} = \mathbf{w}_{\hat{y}_t} - \mathbf{x}_t$
    end if
end for
```

Let $||\mathbf{x}_t|| \leq 1$ for all t. Let there exist vectors $\mathbf{w}_1^*, \ldots, \mathbf{w}_k^*$ such that $(\mathbf{w}_{y_t}^*)^\top \mathbf{x}_t \geq 1 + (\mathbf{w}_j^*)^\top \mathbf{x}_t$ for all $j \neq y_t$. Then show that the Multiclass Perceptron has the following mistake bound

$$M = \sum_t \mathbf{1}(y_t \neq \hat{y}_t) \leq 2 \sum_{j=1}^{k} \sum_{i=1}^{d} (w_{j,i}^*)^2$$

where $w^*_{j,i}$ refers to the i^{th} element of $\mathbf{w}^*_j$.

Note: $\sim$`/Assignments/Assignment1-Files/Files_data_codes.zip` contains the required files. The datasets for the problems are in the correspondingly named folders. The helper code for SVMs is in a separate folder.

Note: The format of all dataset files is as below. Each row represents one example with a comma separated list of attributes. The last attribute is the label of the datapoint (i.e. $+1$ or -1).