E0 270 Machine Learning *Due:* Feb 18, 2016

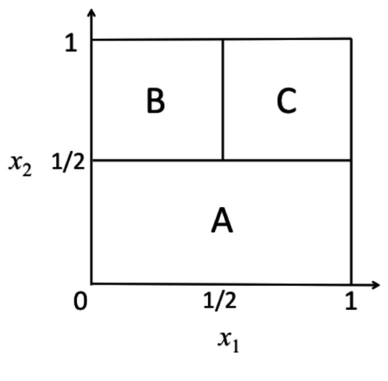
Assignment 1

Instructions

- Please submit your assignment before class on Feb 18, 2016.
- The data and helper code for the programming problems have been provided in the zip file 'hw1-data-code.zip' which can be downloaded from the course website.
- For the programming problems attach the plots with the assignment. Also, submit your code as a zip file before class on Feb 18, 2016. The submission link and instructions can be found on the course website.
- Read the plagiarism policy on the course website and make sure that you do not indulge in it.

1. **Refresher:** Let $\mu \in \mathbb{R}^d$ and $\Sigma, \Sigma' \in \mathbb{R}^{d\times d}$. Let X, Y be d-dimensional random vectors with $X \sim \mathcal{N}(\mu, \Sigma)$, and $Y|X \sim \mathcal{N}(X, \Sigma')$. Calculate the distribution each of the following:
   (a) The unconditional distribution of Y.
   (b) The joint distribution of the random variable (X, Y).
   (c) The conditional distribution of X given Y.

2. **Bayes Error:** Consider a binary classification task on the unit square: $\mathcal{X} = [0,1] \times [0,1]$ and $\mathcal{Y} = \widehat{\mathcal{Y}} = \{\pm 1\}$. Let $A = [0,1] \times [0, \frac{1}{2}]$, $B = [0, \frac{1}{2}] \times (\frac{1}{2}, 1]$, and $C = (\frac{1}{2}, 1] \times (\frac{1}{2}, 1]$ as shown in the figure; clearly, A, B, C form a partition of $\mathcal{X}$.



Consider a joint probability distribution D on $\mathcal{X} \times \{\pm 1\}$ under which labeled examples $(\mathbf{x}, y)$ are generated as follows:

- First $\mathbf{x}$ is drawn uniformly at random from $\mathcal{X}$;
- Given $\mathbf{x}$, $y \in \{\pm 1\}$ is drawn randomly as follows:

$$\mathbf{P}(y = 1|\mathbf{x}) = \begin{cases} 0.3 & \text{if } \mathbf{x} \in A \\ 0.6 & \text{if } \mathbf{x} \in B \\ 0.7 & \text{if } \mathbf{x} \in C. \end{cases}$$

(a) Find a Bayes optimal classifier $h^* : \mathcal{X} \to \{\pm 1\}$ for D. What is the Bayes error for D, i.e. $\mathbf{P}_{(X,Y)\sim D}(h^*(X) \neq Y)$?

(b) Consider the classifier $h : \mathcal{X} \to \{\pm 1\}$ given by

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in C \\ -1 & \text{if } \mathbf{x} \in A \cup B. \end{cases}$$

Find the 0-1 error of h w.r.t. D, i.e. $\mathbf{P}_{(X,Y)\sim D}(h(X) \neq Y)$.

3. **Bayes Classifier:** Let $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \widehat{\mathcal{Y}} = \{\pm 1\}$. Assume that given the label $y \in \{\pm 1\}$, the features are conditionally independent, and moreover, are distributed according to Laplace distribution, so that the class-conditional densities $f_y(\mathbf{x})$ can be written as

$$f_y(\mathbf{x}) = \prod_{k=1}^{d} f_y(x_k)$$

with

$$f_y(x_k) = \frac{1}{2} \exp\Big(- |x_k - \mu_{y,k}| \Big),$$

where $\mu_{y,k}$ are the mean parameters of the Laplace distribution. Let $p = \mathbf{P}(Y = 1)$.

(a) Find an expression for the conditional class probability function $\eta(\mathbf{x}) = \mathbf{P}(Y = 1|X = \mathbf{x})$ in this setting.

(b) Use this to construct the Bayes optimal classifier.

(c) Suppose that the parameters $\mu_{y,k}$, p are unknown, but you are given a training sample $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)) \in (\mathcal{X} \times \{\pm 1\})^m$ where the examples $(\mathbf{x}_i, y_i)$ are drawn i.i.d. from the above joint distribution (which can be viewed as generating a labeled example $(\mathbf{x}, y)$ by first drawing a label y according to p and then drawing an instance $\mathbf{x}$ according to $f_y(\mathbf{x})$). Describe an algorithm for constructing the Bayes optimal classifier in this case.

4. **Cost Sensitive Classification:** Consider a cost-sensitive binary classification task in which misclassifying a positive instance as negative incurs a cost of $c \in (0, 1)$, and misclassifying a negative instance as positive incurs a cost of $1-c$. This can be formulated as a learning task with instance space $\mathcal{X}$, label and prediction spaces $\mathcal{Y} = \widehat{\mathcal{Y}} = \{\pm 1\}$, and cost-sensitive loss $\ell_c : \{\pm 1\} \times \{\pm 1\} \to \{0, c, 1-c\}$ defined as

$$\ell_c(y, \widehat{y}) = \begin{cases} 1-c & \text{if } y = -1 \text{ and } \widehat{y} = 1 \\ c & \text{if } y = 1 \text{ and } \widehat{y} = -1 \\ 0 & \text{if } \widehat{y} = y. \end{cases}$$

Let D be a joint probability distribution on $\mathcal{X} \times \{\pm 1\}$, with marginal distribution μ on $\mathcal{X}$ and conditional label probabilities given by $\eta(x) = \mathbf{P}(y = 1|x)$, and for any classifier $h : \mathcal{X} \to \{\pm 1\}$, define

$$\mathrm{er}^c_D[h] = \mathbf{E}_{(x,y)\sim D}\big[\ell_c(y, h(x))\big].$$

Derive a Bayes optimal classifier in this setting, i.e. a classifier $h^* : \mathcal{X} \to \{\pm 1\}$ with

$$\mathrm{er}^c_D[h^*] = \inf_{h:\mathcal{X}\to\{\pm 1\}} \mathrm{er}^c_D[h].$$

How does your derivation change if the loss incurred on misclassifying a positive instance as negative is a and that for misclassifying a negative instance as positive is b for some arbitrary $a, b > 0$?

5. **Programming Exercise: Perceptron and Winnow.**

You are given three data streams, `data_stream_1.mat`, `data_stream_2.mat` and `data_stream_3.mat`. The first two contain 1000 instances $\mathbf{x}_t \in \mathbb{R}^2$ and corresponding labels $y_t \in \{\pm 1\}$, $t \in \{1, \ldots, 1000\}$, while the last contains 8124 instances $\mathbf{x}_t \in \mathbb{R}^{112}$ and corresponding labels $y_t \in \{\pm 1\}$. You are also given two MATLAB routines `get_instance(`s`,`t`)` and `get_label(`s`,`t`)`, which return the t-th instance in data stream s and the t-th label in data stream s, respectively ($s \in \{1, 2, 3\}$). The instances in `data_stream_1.mat` happen to be drawn iid from a uniform distribution over the surface (perimeter) of the unit ball (circle) in $\mathbb{R}^2$, i.e. from a uniform distribution over the set $S = \{\mathbf{x} \in \mathbb{R}^2 \mid \|\mathbf{x}\|_2 = 1\}$. The instances in `data_stream_2.mat` are generated as follows: $\mathbf{x}_1$ is set to be $(1, 0)$; then for each $t \geq 2$, $\mathbf{z}_t$ is drawn uniformly at random from $[-1, 1]^2$, and $\mathbf{x}_t$ is set to $(\mathbf{x}_{t-1} + \mathbf{z}_t)/\|\mathbf{x}_{t-1} + \mathbf{z}_t\|_2$. Thus the instances in `data_stream_2.mat` are not independent. In both cases, the labels y_t are given by a fixed linear separator, $y_t = \text{sign}(\mathbf{u} \cdot \mathbf{x}_t)$, where $\mathbf{u} = \left(\frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}\right)$. The last data stream is taken from a real world dataset and there do not exist a fixed linear separator.

(a) Write a piece of MATLAB code `perceptron.m` which implements the perceptron algorithm on a given data stream, where instances and labels are obtained via the `get_instance(`s`,`t`)` and `get_label(`s`,`t`)` routines. Apply the algorithm to each of the three data streams above, and in each case, plot the ratio of the number of mistakes made relative to the total number of examples seen as a function of the number of examples seen. For the first two data streams, also plot an upper bound on this ratio obtained from the mistake bound discussed in class.

(b) In the case of `data_stream_1.mat`, where the instances are drawn iid from a fixed distribution, it can also be of interest to evaluate the performance of learned models in terms of expected error on new instances drawn from the same distribution. Write a small MATLAB function `error_uniform.m` which takes as input two vectors $\mathbf{u}, \mathbf{w} \in \mathbb{R}^2$, and implements your solution to Problem 1 to compute the expected misclassification error on a new example $\mathbf{x}$ drawn uniformly from S, when the true label is $\text{sign}(\mathbf{u} \cdot \mathbf{x})$ and the predicted label $\text{sign}(\mathbf{w} \cdot \mathbf{x})$. Use this MATLAB function to compute the expected error of the linear predictor learned by the perceptron algorithm on each iteration, and plot this error as a function of the number of examples seen.

(c) Write a piece of MATLAB code `winnow.m` which implements the winnow algorithm on a given data stream, where instances and labels are obtained via the `get_instance(`s`,`t`)` and `get_label(`s`,`t`)` routines. Repeat part (a) for the winnow algorithm. Compare the total number of mistakes for the perceptron and the winnow for each of the data streams.

6. **Programming Exercise: Support Vector Machine.**

Write a piece of MATLAB code to implement SVM algorithm on a given 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 $\mathbf{X}$): 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. 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.

(a) Run your implementation of the SVM learning algorithm with linear kernel on the spam classification training set, for each value of C in the range $\{1, 10, 10^2, 10^3, 10^4\}$. Plot the training and test error achieved by each value of C (C on the x-axis and the classification error on the y-axis) and identify the value of C that achieves the lowest test error (the train and test data are provided in the folder `Problem-6\Spambase`). Now, select C 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-6\Spambase\CrossValidation`); report the average cross-validation error (across the five folds) for each value of C and identify the value of C that achieves the lowest average cross-validation error. Does the cross-validation procedure select the right value of C (you can resolve ties in favor of the smallest value of C)?

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

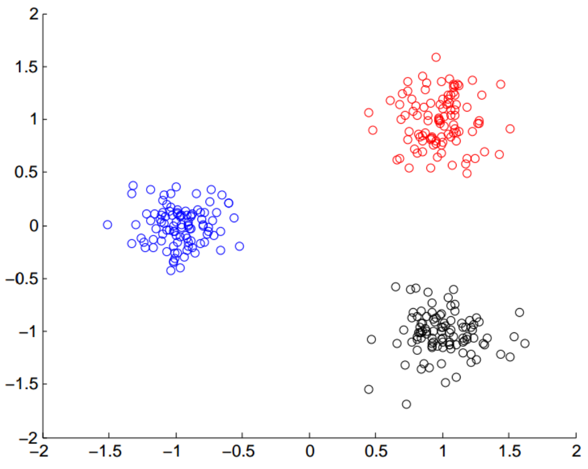
(b) You are provided a 2-dimensional synthetic data set for this problem. Run your implementation of SVM on the training set provided using a linear, degree-2 polynomial, degree-3 polynomial and RBF kernel, selecting C from the range $\{1, 10, 10^2, 10^3, 10^4\}$ via 5-fold cross-validation on the training 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 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. (See folder `Problem-6\Synthetic` for the relevant files.)

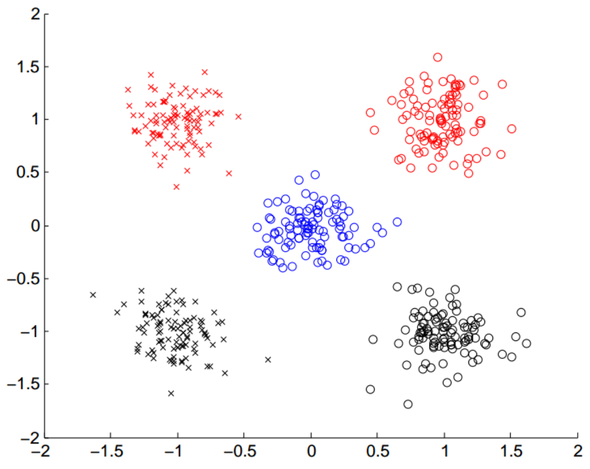
7. Multiclass classification

(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) Consider an online multiclass classification problem with $r > 2$ classes, where on each trial t, the learner receives an instance $\mathbf{x}^t \in \mathbb{R}^d$, must make a prediction $\widehat{y}^t \in [r]$, and then receives the true label $y^t \in [r]$ and incurs zero-one loss $\ell_{0\text{-}1}(y^t, \widehat{y}^t) = \mathbf{1}(y^t \neq \widehat{y}^t)$. Consider the following extension of the perceptron algorithm for this problem, which maintains a weight vector $\mathbf{w}_y^t$ for each class

$y \in [r]$, predicts according to the class y whose weight vector $\mathbf{w}_y^t$ has the highest dot product $\mathbf{w}_y^t \cdot \mathbf{x}^t$ with the current instance $\mathbf{x}^t$, and if a mistake is made, updates only weight vectors $\mathbf{w}_y^t$ corresponding to the true class $y = y^t$ and to classes that have a higher dot product than the true class, $\mathbf{w}_y^t \cdot \mathbf{x}^t > \mathbf{w}_{y^t}^t \cdot \mathbf{x}^t$ (such algorithms are said to be *ultra-conservative*):

```
Algorithm Multiclass Perceptron (MP)
Initial weight vectors $\mathbf{w}_y^1 = \mathbf{0} \in \mathbb{R}^d \;\forall y \in [r]$
For $t = 1, \ldots, T$:
  – Receive instance $\mathbf{x}^t \in \mathbb{R}^d$
  – Predict $\widehat{y}^t = \arg\max_{y \in [r]} \mathbf{w}_y^t \cdot \mathbf{x}^t$
  – Receive true label $y^t \in [r]$
  – Incur loss $\ell_{0\text{-}1}(y^t, \widehat{y}^t)$
  – Update: If $\widehat{y}^t \neq y^t$ then
        $E^t \leftarrow \big\{ y \in [r] \;\big|\; \mathbf{w}_y^t \cdot \mathbf{x}^t > \mathbf{w}_{y^t} \cdot \mathbf{x}^t \big\}$
        $\mathbf{w}_y^{t+1} \leftarrow \begin{cases} \mathbf{w}_y^t + \mathbf{x}^t & \text{if } y = y^t \\ \mathbf{w}_y^t - \mathbf{x}^t/|E^t| & \text{if } y \in E^t \\ \mathbf{w}_y^t & \text{otherwise} \end{cases}$
    else
        $\mathbf{w}_y^{t+1} \leftarrow \mathbf{w}_y^t \;\forall y \in [r]$
```

Let $S = ((\mathbf{x}^1, y^1), \ldots, (\mathbf{x}^T, y^T)) \in (\mathbb{R}^d \times [r])^T$, and let $R_2 = \max \big\{ \|\mathbf{x}^t\|_2 \;\big|\; t \in [T] \big\}$. Suppose there exist some weight vectors $\mathbf{u}_1, \ldots, \mathbf{u}_r \in \mathbb{R}^d$ and $\gamma > 0$ such that $\mathbf{u}_{y^t} \cdot \mathbf{x}^t - \max_{y \neq y^t} \mathbf{u}_y \cdot \mathbf{x}^t \geq \gamma \;\forall t$. Then show that the number of mistakes made by the above algorithm on S satisfies

$$L_S^{0\text{-}1}[\text{MP}] \;\leq\; \frac{2R_2^2 \left(\sum_{y=1}^r \|\mathbf{u}_y\|_2^2 \right)}{\gamma^2} \,.$$

You may find it helpful to note that, by Cauchy-Shwarz inequality, for any two sets of vectors $\mathbf{p}_y, \mathbf{q}_y \in \mathbb{R}^d$ for $y \in [r]$, $\sum_{y=1}^r \mathbf{p}_y \cdot \mathbf{q}_y \leq \sqrt{\left(\sum_{y=1}^r \|\mathbf{p}_y\|_2^2 \right) \left(\sum_{y=1}^r \|\mathbf{q}_y\|_2^2 \right)}$.