Statistical Machine Learning (W4400)
Spring 2013
http://stat.columbia.edu/∼porbanz/teaching/W4400/

Peter Orbanz
porbanz@stat.columbia.edu
Benjamin Reddy
bmr2136@columbia.edu

Homework 3

Due: 29 October 2013

Homework submission: We will collect your homework **at the beginning of class** on the due date. If you cannot attend class that day, you can leave your solution in my postbox in the Department of Statistics, 10th floor SSW, at any time before then.

Problem 1 (Convex hulls and classifiers)

Suppose we are given two training sets

$$\mathcal{C}_1 = \{\mathbf{x}^1_1, \dots, \mathbf{x}^1_m\} \quad \text{and} \quad \mathcal{C}_2 = \{\mathbf{x}^2_1, \dots, \mathbf{x}^2_n\} \; .$$

in $\mathbb{R}^d$. In class, we have derived the support vector machine based on the idea that separating the two classes means separating their convex hulls. In this problem, you should check that this idea is valid.

Homework problems:

1. Show that if the convex hulls of $\mathcal{C}_1$ and $\mathcal{C}_2$ intersect, the two sets cannot be linearly separable.
2. Show that the converse is also true: If the convex hulls do not intersect, the classes are linearly separable.

To do so, recall the definition of the convex hull: If $\mathcal{C} = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$ is a set of points in $\mathbb{R}^d$, its convex hull conv($\mathcal{C}$) is the set of all points $\mathbf{x} \in \mathbb{R}^d$ which can be represented as "convex combinations" of points in $\mathcal{C}$, that is, points which can be written as

$$\mathbf{x} = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i$$

where $\alpha_i \geq 0$ and $\sum_{i=1}^m \alpha_i = 1$. The convex hulls of the two sets are

$$\text{conv}(\mathcal{C}_1) = \left\{ \mathbf{x} = \sum_{i=1}^{m} \alpha_i \mathbf{x}^1_i \,\middle|\, \alpha_i \geq 0, \sum_{i=1}^{m} \alpha_i = 1 \right\} \quad \text{and} \quad \text{conv}(\mathcal{C}_2) = \left\{ \mathbf{x} = \sum_{j=1}^{n} \beta_j \mathbf{x}^2_j \,\middle|\, \beta_j \geq 0, \sum_{j=1}^{n} \beta_j = 1 \right\} \; .$$

Recall also that $\mathcal{C}_1$ and $\mathcal{C}_2$ are linearly separable if and only if there is a vector $v_{\text{H}} \in \mathbb{R}^d$ and a scalar $c \in \mathbb{R}$ such that the affine hyperplane defined by (v_{H}, c) classifies the sets correctly:

$$\langle \mathbf{x}, v_{\text{H}} \rangle - c = \begin{cases} > 0 & \mathbf{x} \in \mathcal{C}_1 \\ < 0 & \mathbf{x} \in \mathcal{C}_2 \end{cases} \; .$$

Problem 2 (Combining kernels)

It was already mentioned in class that kernel functions can be combined and modified in various ways to obtain new kernel functions. In this problem, we will convince ourselves that this is indeed true in two simple cases. Recall that a function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a kernel if there is *some* function $\phi : \mathbb{R}^d \to \mathcal{F}$, into *some* space $\mathcal{F}$ with scalar product $\langle \, . \, , \, . \, \rangle_{\mathcal{F}}$, such that

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{F}}$$

for all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$.

Homework problems:
Let $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$ be kernels on $\mathbb{R}^d$.

1. Show that, for any positive real number a, $k(\mathbf{x}, \mathbf{x}') := ak_1(\mathbf{x}, \mathbf{x}')$ is a kernel.
2. Show that $k(\mathbf{x}, \mathbf{x}') := k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}')$ is a kernel.
3. Show that, for any positive integer p, $k(\mathbf{x}, \mathbf{x}') := k_1(\mathbf{x}, \mathbf{x}')^p$ is a kernel.

Problem 3 (Boosting)

The objective of this problem is to implement the AdaBoost algorithm. We will test the algorithm on handwritten digits from the USPS data set.

AdaBoost: Assume we are given a training sample $(\mathbf{x}^{(i)}, y_i), i = 1, ..., n$, where $\mathbf{x}^{(i)}$ are data values in $\mathbb{R}^d$ and $y_i \in \{-1, +1\}$ are class labels. Along with the training data, we provide the algorithm with a training routine for some classifier c (the "weak learner"). Here is the AdaBoost algorithm for the two-class problem:

1. Initialize weights: $w_i = \frac{1}{n}$
2. for $b = 1, ..., B$
   (a) Train a weak learner c_b on the weighted training data.
   (b) Compute error: $\epsilon_b := \frac{\sum_{i=1}^n w_i \mathbb{I}\{y_i \neq c_b(\mathbf{x}^{(i)})\}}{\sum_{i=1}^n w_i}$
   (c) Compute voting weights: $\alpha_b = \log\left(\frac{1-\epsilon_b}{\epsilon_b}\right)$
   (d) Recompute weights: $w_i = w_i \exp\left(\alpha_b \mathbb{I}\{y_i \neq c_b(\mathbf{x}^{(i)})\}\right)$
3. Return classifier $\hat{c}_B(\mathbf{x}^{(i)}) = \operatorname{sgn}\left(\sum_{b=1}^B \alpha_b c_b(\mathbf{x}^{(i)})\right)$

Decision stumps: Recall that a stump classifier c is defined by

$$c(\mathbf{x}|j, \theta, m) := \begin{cases} +m & x_j > \theta \\ -m & \text{otherwise.} \end{cases} \tag{1}$$

Since the stump ignores all entries of $\mathbf{x}$ except x_j, it is equivalent to a linear classifier defined by an affine hyperplane. The plane is orthogonal to the jth axis, with which it intersects at $x_j = \theta$. The orientation of the hyperplane is determined by $m \in \{-1, +1\}$. We will employ stumps as weak learners in our boosting algorithm. To train stumps on weighted data, use the learning rule

$$(j^*, \theta^*) := \arg\min_{j,\theta} \frac{\sum_{i=1}^n w_i \mathbb{I}\{y_i \neq c(\mathbf{x}^{(i)}|j, \theta, m)\}}{\sum_{i=1}^n w_i} . \tag{2}$$

In the implementation of your training routine, first determine an optimal parameter θ_j^* for each dimension $j = 1, ..., d$, and then select the j^* for which the cost term in (2) is minimal.

Homework problems:

1. Implement the AdaBoost algorithm in R. The algorithm requires two auxiliary functions, to train and evaluate the weak learner. We also need a third function which implements the resulting boosting classifier. We will use decision stumps as weak learners, but a good implementation of the boosting algorithm should permit you to easily plug in arbitrary weak learners. To make sure that is possible, please use function calls of the following form:

- `pars <- train(X, w, y)` for the weak learner training routine, where `X` is a matrix the columns of which are the training vectors $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$, and `w` and `y` are vectors containing the weights and class labels. The output `pars` is a list which contains the parameters specifying the resulting classifier. (In our case, `pars` will be the triplet (j, θ, m) which specifies the decision stump).
- `label <- classify(X, pars)` for the classification routine, which evaluates the weak learner on `X` using the parametrization `pars`.
- A function `c_hat <- agg_class(X, alpha, allPars)` which evaluates the boosting classifier ("aggregated classifier") on `X`. The argument `alpha` denotes the vector of voting weights and `allPars` contains the parameters of all weak learners.

2. Implement the functions `train` and `classify` for decision stumps.
3. Run your algorithm on the USPS data (the digit data we used in Homework 2) and evaluate your results using cross validation.

   More precisely, your AdaBoost algorithm returns a classifier that is a combination of B weak learners. Since it is an incremental learning, we can evaluate the AdaBoost at every iteration b by considering the sum up to the b-th weak learner. At each iteration, perform 5-fold cross validation to estimate the training and test error of the current classifier (that is, the errors measured on the cross validation training and test sets, respectively).
4. Plot the training error and the test error as a function of b.

Submission. Please make sure your solution contains the following:

- Your implementation for `train`, `classify` and `agg_class`.
- Your implementation of `AdaBoost`.
- Plots of your results (training error and cross-validated test error).