Exercises
Machine Learning
AS 2012

Machine Learning Laboratory
Dept. of Computer Science, ETH Zürich
Prof. Dr. Joachim M. Buhmann
Web http://ml2.inf.ethz.ch/courses/ml/

Email questions to: Dmitry Laptev
dlaptev@inf.ethz.ch

Series 5, Nov 21th, 2012 (Adaboost)

Problem 1 (Combining Kernels):

Recall the definition of positive semidefiniteness.

eigenvalues Show that positive semidefinite matrices have non-negative eigenvalues.

In this exercise, we explore several common ways to combine known kernels so that we obtain new kernels. For some input data $x, x' \in \mathcal{X}$, let $k_1(x, x')$ and $k_2(x, x')$ be kernels over $\mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Most of the following proofs can be done by giving the corresponding mapping Φ to feature space.

scaling For a positive real number a, show that $k(x, x') := ak_1(x, x')$ is a kernel.

sum Show that $k(x, x') := k_1(x, x') + k_2(x, x')$ is a kernel.

linear combination For real positive numbers a and b, show using the results before that $k(x, x') := ak_1(x, x') + bk_2(x, x')$ is a kernel.

product Show that $k(x, x') := k_1(x, x')k_2(x, x')$ is a kernel.

exponentiated Show that for a positive integer p, $k(x, x') := k_1(x, x')^p$ is a kernel

metric For $x, x' \in \mathbb{R}^d$, and for a positive definite matrix M, show that $k(x, x') := x^\top M x'$ is a kernel.

Problem 2 (Deriving the SVM Dual):

Recall the soft margin C-SVM:

$$\text{minimize}_{w,b,\xi} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \tag{1}$$

$$\text{subject to} \quad y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i \text{ for all } i = 1, \ldots, n. \tag{2}$$

$$\xi_i \geq 0 \text{ for all } i = 1, \ldots, n \tag{3}$$

1. Write down the Lagrangian, using α_i as the Lagrange multiplier corresponding to constraint (2) and γ_i as the Lagrange multiplier corresponding to constraint (3).
2. Compute the derivative of the Lagrangian with respect to w, b and ξ.
3. Solve for the dual, and show that this is given by

$$\begin{aligned} \text{maximize}_\alpha \quad & \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle \\ \text{subject to} \quad & \sum_{i=1}^{n}\alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C \text{ for } i = 1, 2, \ldots, n. \end{aligned}$$

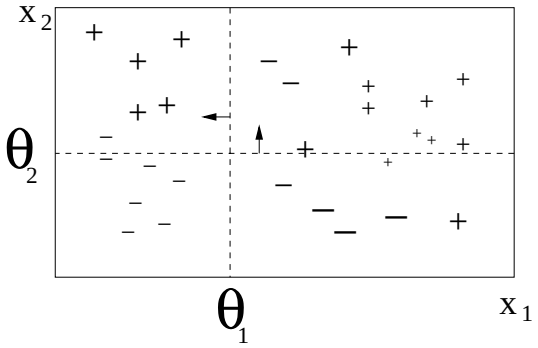
Problem 3 (Adaboost):

The objective of this problem is to implement the AdaBoost algorithm. We will use a simple type of decision tree as weak learners and run 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", also called the *base classifier*). 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 base classifier 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: In the lecture, we discussed decision tree classifiers. The simplemost non-trivial type of decision tree (a root node with two leaves) is called a *decision stump*:



A stump classifier c is defined by

$$c(\mathbf{x}|j, \theta, m) := \begin{cases} +m & x_j > \theta \\ -m & \text{otherwise.} \end{cases} \tag{4}$$

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 base learners in our boosting algorithm. To train stumps on weighted data, use the learning rule

$$(j^*, \theta^*) := \arg\min_{j,\theta} \frac{\sum_{i=1}^n w_i \mathbb{I}\{y_i \neq c(\mathbf{x}^{(i)}|j, \theta, m)\}}{\sum_{i=1}^n w_i} \,. \tag{5}$$

Implement this in your training routine by first finding an optimal parameter θ_j^* for each dimension $j = 1, ..., d$, and then select the j^* for which the cost term in (5) is minimal.

1. Implement the AdaBoost algorithm in Matlab. The algorithm requires two auxiliary functions, to train and evaluate the base classifier. We also need a third function which implements the resulting boosting classifier. To ensure that an arbitrary base learner can easily be plugged into your boosting algorithm, please use function calls of the following form:
   - `pars = train(X, w, y)` for the base classifier training routine, where `X` is a matrix the columns of which are the training vectors $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$, `w` and `y` are vectors containing the weights and class labels, and `pars` is a *structure* of parameters specifying the resulting classifier.
     In our case, the structure contains the triplet (j, θ, m) which specifies the decision stump. See the respective Matlab help page (`Programming > Data Types > Structures`) if you are unfamiliar with the concept.

- `label = classify(X, pars)` for the classification routine, which evaluates the base classifier on `X` using the parametrization `pars`.
- A function `c_hat = agg_class(X, alpha, allPars)` which evaluates the boosting classifier ("aggregated classifier") on `X`. `alpha` denotes the vector of voting weights and `allPars` is a structure array containing the parameters of all base classifiers.

2. Implement the functions `train` and `classify` for decision stumps.

3. Your AdaBoost algorithm returns a classifier that is a combination of B base 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 base learner. Use your functions for Cross-Validation and Bootstrap to estimate (test) error. Furthermore, we require you to compute the training error for each CV-fold / Bootstrap sample.

   Run your algorithm on the USPS data from the previous Exercise series. Evaluate your results using 5 fold CV and 200 draws of Bootstrap. Boost the decision stumps and compute training and test error estimates for each iteration b.

4. Run your algorithm on one other task listed under the classification data in the "Machine Learning Repository" at UC Irvine: `http://www.ics.uci.edu/~mlearn/MLSummary.html`. Please do this using 5 fold cross validation.

Submission: Solving this exercise takes some effort, so we award you two points for a full solution. Here is what we ask from you in detail:

1. For the first point:
   - Your implementation for `train`, `classify` and `agg_class`.
   - Your implementation of `AdaBoost`
2. For the second point:
   - Plots your results (Training error, Test error on CV, and Training error, Test error on Bootstrap), on the USPS data, and your comments on the results. You should have a total of 4 plots. Each plot shows the mean error and the corresponding error bars.
   - Your results on at least one other dataset from the UCI machine learning repository