Scalable Machine Learning CS281B, STAT241B
http://alex.smola.org/teaching/berkeley2012
UC Berkeley

March 13, 2012
due on April 3, 2012
Alexander J. Smola

Problem Set 3 — Kernels, SVMs and Gaussian Processes

1 Variable Class Labels (10)

An immediate extension of binary large margin methods can be found in multiclass classification problems. That is, we observe an object x, and subsequently decide that it's an apple, an orange, a car, or part of any other category from a set previously observed categories.

1.1 Multiclass Loss

Your first task is to derive a large margin multiclass loss function and show how the associated subgradient can be computed efficiently using the large margin loss

$$l(x, y, f) = \sup_{y' \in \mathcal{Y}} [f(x, y') - f(x, y) + \Delta(y, y')] \tag{1}$$

In this context assume that $\mathcal{Y} = \{1, \dots, k\}$ and that $\Delta(y, y')$ is an arbitrary nonnegative function with $\Delta(y, y) = 0$ for all $y \in \mathcal{Y}$. Specifically do the following:

1. State the multiclass loss for $f(x, y) = \langle w_y, x \rangle$.
2. Compute the derivative of $l(x, y, f)$ with respect to w. Describe an algorithm for computing it.

Note that you need not compute the full dual of an associated SVM optimization problem. Just derive the subgradient equation and describe how to compute it efficiently.

1.2 Variable Labels

In some cases the number of labels itself can be variable. This occurs, e.g. in image tagging when several objects might be present in an image (pedestrian, car, house, sky, all might be valid tags for the same picture). Consequently we have that $\mathcal{Y} \subseteq 2^{\{1,\dots,k\}}$. That is, $y[i] \in \{0, 1\}$ denotes whether the i-th tag is present. Without loss of generality we will identify y with a binary vector in $\{0, 1\}^k$. Furthermore we assume that the loss function $\Delta(y, y')$ is given by

$$\Delta(y, y') = \sum_{i=1}^{k} |y[i] - y'[i]| \delta_i(y). \tag{2}$$

Here $y[i]$ denotes the i-th coordinate of y and $\delta_i(y) > 0$ is the cost of having the wrong class label for class i, i.e. the cost for having $y[i] \neq y'[i]$. It is our goal to construct a function $f(x, y)$ which is maximized for the correct number and selection of tags. To simplify matters we assume that f is given by

$$f(x, y) = \sum_{i=1}^{k} y[i] \langle w_i, x \rangle + c(\|y\|). \tag{3}$$

That is, we sum over the class label specific scores. Here $c : \{1, \dots k\} \to \mathbb{R}$ is a score dependent only on the *number* of chosen labels. Your task is the following:

1. State the multilabel loss for f given by (3).
2. Compute the derivative of $l(x, y, f)$ with respect to w. Derive an $O(k \log k)$ algorithm to solve this.

Problem Set 3 — Kernels, SVMs and Gaussian Processes

2 Regularization and Fourier Transforms (10)

Radial basis functions are popular choices for kernel functions. They take on the form

$$k(x, x') = \kappa(x - x'). \tag{4}$$

One popular choice of function interpolation is to use B-Splines. They are defined as follows:

$$B_0(x) := \chi_{[-0.5,0.5]}(x) \text{ and } B_{n+1}(x) := B_n \circ B_0. \tag{5}$$

Furthermore denote by $\bar{B}_n(x) := \sqrt{n} B_n(\sqrt{n} x)$ a rescaled version of B_n. Prove the following properties:

1. B_n is a kernel on $\mathbb{R}$ for odd n. Hint — use the Fourier property of kernels.
2. What does this mean in terms of regularization and smoothness properties of any function f given by a kernel expansion (a simple reasoning in terms of the spectral properties of B_n is sufficient)?
3. The limit of $\bar{B}_n$ is a normal distribution with variance $\frac{1}{12}$. Hint — use the fact that B_0 corresponds to a uniform distribution on $[-0.5, 0.5]$ and interpret convolutions in a probabilistic fashion.

To design kernels on $\mathbb{R}^n$ we have a number of options. One possibility is to take coordinate-wise product of kernels. That is, we use

$$k(x, x') = \prod_{i=1}^{d} k_i(x_i, x_i'). \tag{6}$$

4. To see that this is a kernel prove the more general result that for kernels k_1, k_2 on $\mathcal{X} \times \mathcal{X}$ also the product

   $$k(x, x') = k_1(x, x') k_2(x, x') \tag{7}$$

   is a kernel. Hint — use the fact that you can expand k_1 and k_2 in terms of an inner product and rearrange sums to obtain an inner product representation for k.
5. Show that for any function in a Reproducing Kernel Hilbert Space $\mathcal{H}$ the following bound holds

   $$|f(x)| \leq \|f\|_{\mathcal{H}} \sqrt{k(x, x)} \tag{8}$$

Problem Set 3 — Kernels, SVMs and Gaussian Processes

3 Robust Spam Filter (10)

Spammers like to manipulate their e-mails in such a way as to maximize their chance of success at overcoming spam filters. Your job is to ensure that this doesn't happen. You are given two ingredients:

- A set of labeled e-mails (spam, ham) drawn from the *current* distribution of incoming e-mails.
- A linear classifier using the bag-of-words model. That is, the classifier simply converts e-mails into a vector of word counts. E.g. the phrase x

  To be or not to be.

  is converted into the sparse vector $\phi(x)$

  {*(be, 2), (not, 1), (or, 1), (to, 2)*}

  and the classification then occurs by computing

$$f(x) = \sum_i \phi(x)_i w_i + b \tag{9}$$

The spammer is willing to obfuscate his e-mails slightly. That is, he's willing to change up to k words arbitrarily by removing them or by replacing them by any set of k words, i.e. instead of x he may choose any $x' \in B_k(x)$. You counter his move by designing a classifier that is maximally robust to such changes via the loss function:

$$l(x, y, w) = \sup_{x' \in B_k(x)} \max\left(0, 1 - y[\langle w, \phi(x')\rangle + b]\right) \tag{10}$$

1. For a given choice of w find the worst change a spammer can perform such as to increase the misclassification error. It is sufficient that you describe the algorithm. Hint — the size of the coordinates of w matters.
2. Describe a subgradient procedure to minimize the robust loss in (10).
3. We now modify the loss function (10) take a penalty for drastic changes of the text into account. That is, we rescale the margin with the number of modified words via

$$l(x, y, w) = \sup_{x'} \max\left(0, c(\|x - x'\|) - [y\,\langle w, \phi(x')\rangle + b]\right) \tag{11}$$

   Should $c(\cdot)$ be an increasing or decreasing function of its argument? Explain your answer.

Problem Set 3 — Kernels, SVMs and Gaussian Processes

4 Receiver Operating Curve Optimization (10)

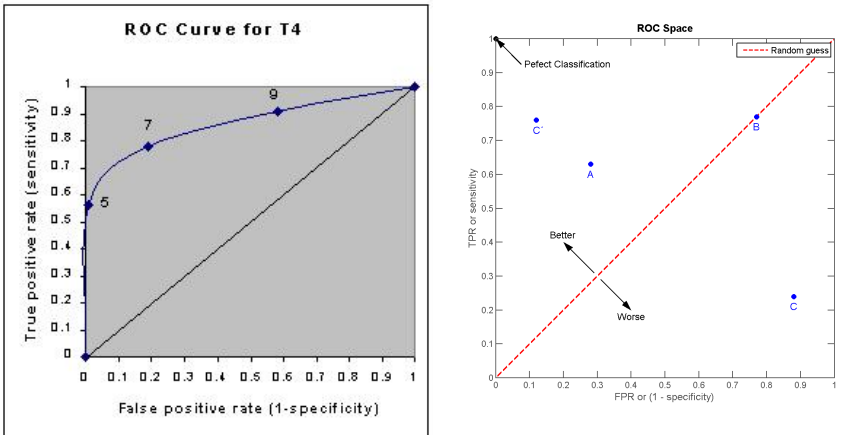
The Receiver Operating Characteristic (ROC) is a useful means of evaluating the overall properties of a *binary* classifier. After all, if we are given a function $f : x \to f(x)$ we are at liberty of selecting any threshold to decide whether an instance is considered positive or negative. In this context the observations can be decomposed into four groups:

True Positives (TP) are instances with class label $y = 1$ which are correctly classified as positive.
False Positives (FP) are instances with class label $y = -1$ which are *incorrectly* classified as positive.
True Negatives (TN) are instances with class label $y = -1$ which are correctly classified as negative.
False Negatives (FN) are instances with class label $y = 1$ which are *incorrectly* classified as negative.

See also http://en.wikipedia.org/wiki/Receiver_operating_characteristic.

1. How do the numbers of TP, FP, TN, FN change if we increase the threshold t for $f(x) > t$ to decide whether an instance is positive or negative.

Denote by $r_{\mathrm{TP}} := \frac{\mathrm{TP}}{m_+}$ and by $r_{\mathrm{FP}} := \frac{\mathrm{FP}}{m_-}$ the normalized ratio of true and false positives respectively. m_+ and m_- denote the number of positive and negative instances. The curve of r_{TP} vs. r_{FP} looks as follows:



The left figure contains an example of such a curve. The right displays several points on this curve. It allows one to compare different algorithms even if their true and false positive rates are quite different.

2. Show that the random classifier, i.e. the classifier which assigns random numbers to $f(x_i)$, has the diagonal line as the dependence between true and false positives.
3. The area under the ROC curve (AROC) is defined as

$$\mathrm{AROC} := \int_0^1 r_{\mathrm{TP}} dr_{\mathrm{FP}}. \tag{12}$$

Show that its empirical average on a dataset is given by

$$\mathrm{AROC} := \frac{1}{m_+ m_-} \sum_{i,j:y_i=1,y_j=-1} \{f(x_i) > f(x_j)\}. \tag{13}$$

4. Given m pairs $(y_i, f(x_i))$ design an $m \log m$ algorithm to compute (13), since a naive implementation would take $O(m_+ m_-)$ time. Hint — you need to sort the list by $f(x_i)$.

A paper that discusses direct optimization of such multivariate performance measures can be found at http://svmlight.joachims.org/svm_perf.html

Problem Set 3 — Kernels, SVMs and Gaussian Processes

5 Gaussian Process Regression (20)

Gaussian Process regression assumes that the observations are jointly normal with a given covariance matrix K whose entries are given by $K_{ij} = k(x_i, x_j)$. We assume an additive Gaussian model

$$t \sim \mathcal{N}(0, K) \text{ and } y_i = t_i + \epsilon_i \text{ where } \epsilon_i \sim \mathcal{N}(0, \sigma^2). \tag{14}$$

It is your goal to implement a GP regression estimator and to apply it to the infamous 'Abalone' dataset measuring the size, age, weight, etc. of Abalone caught in Tasmania. You can get the dataset and its description at http://archive.ics.uci.edu/ml/datasets/Abalone.

This assignment is best done using a programming language that already has a matrix library installed. Otherwise this could be a fair amount of work. In other words, you really should use MATLAB, Octave, or Python (NumPy and SciPy) unless you have considerable issues with any of these three systems. While there are efficient C++ and Fortran libraries for linear algebra available, coding in them and processing the data will take some more time than otherwise.

1. Randomly permute the instances in the datasets to remove any inherent order in the data.
2. Write a script that reads the data, converts it into vectors and partitions out the variable to be predicted (the number of rings). The categorical variable `sex` which can take three values {`male`, `female`, `infant`} is best converted into three dimensional vectors $(1, 0, 0), (0, 1, 0)$, and $(0, 0, 1)$ respectively. Split out the number of rings (age) since this is what we want to predict. We call the number of rings y_i and the vector of the remainder x_i. You should have 4177 such pairs.
   Save this dataset to disk. Call it `raw`.
3. We create a second copy of this dataset by computing the coordinate-wise mean and variance. Subtract the mean and rescale to unit variance (don't rescale y).
   Save this dataset to disk. Call it `cooked`.
4. Compute an approximation of the $0.05, 0.1, 0.5, 0.9, 0.95$ quantiles of the distance between pairs of instances in the datasets. You can achieve this by drawing 1000 pairs of instances (x_i, x_j) from each dataset respectively and by computing the quantiles on the subsets. That is, sort the distances $\|x_i - x_j\|$ and pick the 50th, 100th, 500th, 900th and 950th element in the list.
   These distances will be useful in calibrating the scale used for the Gaussian RBF kernel
   $$k(x, x') = \exp\left(-\lambda^{-2} \|x - x'\|^2\right). \tag{15}$$
   More specifically, we want to ensure that the argument of the exponent is $O(1)$. A good guess is to use one of these 5 quantiles as a good value for λ.
5. Implement a Gaussian Process regression algorithm using the Gaussian RBF kernel. That is, obtain estimates of the mean.
6. We need to determine good values for σ and λ such that the regression error is small. Very large values of λ amount to overly smooth functions. Likewise, very large values of σ amount to underfitting due to the fact that the estimator attributes too much of the signal to noise. Small λ and σ, on the other hand, lead to overfitting. That is, we will get estimates that look too good to be true on the training set but with poor generalization performance. For the experiments use $\sigma \in \{0.01, 0.03, 0.1, 0.3, 1, 3\}$.
   It is your job to evaluate the accuracy of the GP regressor using 5-fold crossvalidation. That is, partition the dataset into 5 blocks. For each block, use the remainder for training the model and then validate the estimates on the held-out block. Average over the 5 partitions. You should compute the following statistics:
   (a) For each pair of values of (σ, λ) compute the 5-fold crossvalidation error on both datasets. Note that this will require $5 \cdot 6 \cdot 5 \cdot 2 = 300$ regression estimates. You need to use the built-in linear algebra libraries (easy if you use python, octave or matlab) for reasonable speed.

Problem Set 3 — Kernels, SVMs and Gaussian Processes

We define it as follows: using the loss function

$$l(y, y') := \frac{1}{2}(y - y')^2 \tag{16}$$

Compute the average error over all instances in all validation sets.

(b) Also compute the training error (least mean squares) for the regression problem for all choices of (σ, λ).

(c) For a given value of λ compare the validation error and training errors by plotting both of them in a graph. Can you identify regions of overfitting and underfitting?

A few hints:

- When using NumPy or Octave, make sure that it has been compiled against the fast linear algebra libraries of your system. E.g. on Linux this is ATLAS and on a Mac this is the Apple Accelerate Framework. Most Linux distributions and MacPorts do this by default.
- When computing the RBF kernel, compute the full kernel before partitioning for different validation sets. Then just pull out submatrices. This is somewhat faster than recomputing. Dont bother if you have a very fast machine.
- Reuse the kernel matrices for different values of σ^2.
- Do not compute the kernel matrix via a double for loop. Instead, use the fact that you can apply functions to matrices. More specically, the dominant cost is to compute $\|x_i - x_j\|^2$. We may write it as
$$\|x_i - x_j\|^2 = \|x_i\|^2 + \|x_j\|^2 - 2x_i^\top x_j$$
This allows us to compute the matrix of all distances via $d1^\top + 1d^\top - 2X\top X$.
- You need a computer with at least 300MB RAM to do this comfortably. At a pinch you could get away with about 35MB but the code would be a lot messier to write.
- The dominant portion of your code will be the solution of a linear system. If possible, *do not compute the matrix inverse explicitly* via $T = M^{-1}$ and Ty but instead have your linear algebra system solve $M^{-1}y$. If you want to do things explicitly use the fact that the matrix youre going to invert is positive semidenite and that you can obtain a Cholesky factorization of it.
- If you choose to code it up in C/C++/Fortran, you want to use BLAS and the key routines to solve the linear system are DPOTRF and DPOTRS. Alternatively use a good linear algebra library such as http://www.mathematik.uni-ulm.de/lehn/FLENS/.
- The actual regression estimator should take less than 10 lines of code in Matlab, Python, or Octave