9.520 Problem Set 1

Due March 15, 2010

Note: there are six problems total in this set.

Problem 1 A common way to think of a RKHS is by looking at the kernel as an inner product in some feature space where the data are mapped (see for example [1, 2]). More precisely, a feature map is a (vector valued) map

$$\Phi : X \to \mathcal{F}$$

that maps the data X into some possibly infinite dimensional feature space $\mathcal{F}$ endowed with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{F}}$. A feature map can be related to a RKHS $\mathcal{H}$ by the condition

$$\langle \Phi(x), \Phi(s) \rangle_{\mathcal{F}} = K(x, s)$$

and the function in the RKHS can be seen as a hyperplane in $\mathcal{F}$. That is to say,

$$f(x) = \langle w, \Phi(x) \rangle_{\mathcal{F}}$$

for some $w \in \mathcal{F}$. In this exercise you are asked to derive a canonical example of feature map using Mercer Theorem that we describe below.

A well known result due to Mercer says that if a (symmetric, positive definite) kernel is continuous and the domain X is compact, then the integral operator

$$L_K f(x) = \int_X K(x, x') f(x') p(x') dx'$$

acting on the space $L^2(X, p(x)dx)$ of square integrable functions, is compact, self-adjoint and positive. We indicate the (ordered) eigenvalues with $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_D$, and the eigenfunctions with $(\psi_j)_{j=1}^D$, where D can be infinite. In this case the functions ψ define an orthonormal basis in $L^2(X, p(x)dx)$ and the kernel can be expanded in the following way

$$K(x, x') = \sum_{j=1}^{D} \sigma_j \psi_j(x) \psi_j(x').$$

(a) Give an example of a feature map using the above results.

(b) What is the associated feature space if the operator has finite range (finite number of eigenfunctions)?

(c) Given the map in (a), show that a function in $\mathcal{H}$ can be written as a hyperplane in $\mathcal{F}$.

(d) Show that the square of w for the corresponding hyperplane is simply the square of the norm of the function in the RKHS, and write it explicitly.

Problem 2 The distance between two elements $\Phi(x), \Phi(s)$ of a feature space induced by some kernel K can be seen as a new distance $d(x, x')$ in the input space. Show that such a distance can always be calculated without knowing the explicit form of the feature map itself.

Problem 3 You are given a dataset of x, y pairs $\{(x_i, y_i)\}_{i=1}^N$, with $x_i \in X$ and $y_i \in \{-1, 1\}$. Assume that n_+, n_- of the x_i have label $+1, -1$, respectively (so $n_+ + n_- = N$), and let's also assume that we are given a kernel K and an associated feature map $\Phi : X \to \mathcal{F}$ satisfying

$$K(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{F}}.$$

Derive a classification rule, involving only kernel products (and the sign function), that assigns to a new test point the label of the class whose mean is closest *in the feature space.*

Problem 4 In (binary) classification problems one aims at finding a classification rule (also called the "decision rule") which is a binary valued function on the input space $c : X \to \{1, -1\}$. The quality of a classification rule can be naturally measured by means of the so called misclassification error defined by

$$R(c) = \mathbb{P}\{c(x) \neq y\}.$$

If we introduce the misclassification loss $\ell(c(x), y) = \theta(-yc(x))$, where $\theta(s) = 1$ if $s > 0$ and $\theta(s) = 0$ otherwise, the misclassification error can be rewritten as

$$R(c) = \int_{X \times Y} \theta(-yc(x)) p(x) p(y|x) dx dy.$$

Direct minimization of the misclassification error is not computationally feasible mostly because the misclassification loss is not convex. In practice, one usually looks for real valued (rather than binary valued) functions $f : X \to \mathbf{R}$ and replaces $\theta(-yc(x))$ with some convex loss $\ell(-yf(x))$. A classification rule is then obtained by taking the sign, that is $c(x) = \text{sign}(f(x))$. Commonly chosen loss functions are the hinge loss and square loss (see class). Note that in this case the error is measured by the expected error

$$\mathcal{E}(f) = \int_{X \times Y} \ell(-yf(x)) p(x) p(y|x) dx dy.$$

However, there is still the problem of relating the convex approximation to the original classification problem.

With the above discussion in mind, answer the following questions:

(a) The minimizer of $R(c)$ over all possible decision rules is the so called Bayes decision rule $b : X \to \{1, -1\}$. Find the explicit form of b.
(hint: Consider $R(c|x) = \int_Y \theta(-yc(x)) p(y|x) dy$ and calculate $b(x)$ point-wise.)

(b) Calculate the explicit form of the minimizer of $\mathcal{E}(f)$ if ℓ is the hinge loss. (hint: use a trick similar to the previous hint) How is it related to Bayes decision rule?

(c) Check that the square loss can be written as $\ell(-yf(x))$. Calculate the explicit form of the minimizer of $\mathcal{E}(f)$ if ℓ is the square loss. How is it related to Bayes decision rule?

Problem 5 In class, we discussed RLS and the SVM, Tikhonov regularization algorithms derived by choosing the square-loss, $V(f(x), y) = (y - f(x))^2$, and the hinge-loss, $V(f(x), y) = (1 - yf(x))_+$, respectively. For classification, does one of these losses make more sense than the other? Why? Try to draw (or plot) some simple data sets (in 2 or even 1 dimensions) in which one loss function or the other "does the wrong thing." In general, do you think the optimal setting for λ will be the same for each algorithm? Why or why not?

Note: This problem is admittedly a bit open-ended. There is not necessarily a right or wrong answer. We are just interested in seeing what ideas you might have. Please limit your work on this problem to one page total.

Problem 6 *Matlab exercise.*

In this excercise you will implement and use regularized least squares on an artificial classification problem. You will do the following:

- Implement RLS using the linear and polynomial kernels. You should write three functions:
  - `rlsTrain(Ytrain,Xtrain,whichKernel,lambda)` takes four inputs
    - `Ytrain` the training labels;
    - `Xtrain` the training inputs;
    - `whichKernel` the kernel to use, e.g. `'linear'`;
    - `lambda` the regularization parameter

    and returns one output
    - `coeffs` the optimal RLS coefficients
  - `rlsPredict(Xtest,Xtrain,coeffs,whichKernel)` takes four inputs
    - `Xtest` the test inputs;
    - `Xtrain` the training inputs;
    - `coeffs` the coefficients to use;
    - `whichKernel` the kernel to use, e.g. `'linear'`

    and returns one output
    - `Ytest` the predicted values at the test inputs
  - `rlsLOO(Ytrain,Ktrain,lambdas)` implements the efficient leave-one-out method we derived in class. It takes three inputs
    - `Ytrain` the training labels;
    - `Ktrain` the kernel matrix on the training set;
    - `lambdas` a vector of values for the regularization parameter λ

    and returns one output
    - `looe` a vector of leave-one-out errors on the training set – errors for the RLS solutions, that is – one for each value of λ in `lambdas`
- (You might want to write a helper function to construct the kernel matrix, too.)
- Download the *two moons* data set from the course page. It contains a training set `x,y` and a test set `xt, yt` each one containing 200 samples. The inputs in `x` and `xt` should have two-dimensions. Plot them and verify that the dataset lives up to its name.
- Use RLS to train a linear classifier on the training set, choosing the regularization parameter λ to minimize the leave-one-out error. (A tip: a reasonable value for λ might range up to the maximum eigenvalue of the kernel matrix.)
- Do the same thing with a polynomial kernel, using at least 3 different polynomial degrees.

- Compare the obtained classifiers by testing them on the test set and plotting the obtained decision boundaries.

Please include the following in your writeup.

- All of the code you wrote.
- Figures showing:
  - The training set error and leave-one-out error vs. λ for each of the kernels you tried. Plot both kinds of error on the same figure (one figure for each kernel).
  - The decision boundaries overlaid on the training set points (plotted in two dimensions), for each of the kernels you tried.
- A table giving the training and test error and best λ for each kernel you tried. Report the error in terms of the true positive rate: percentage of test points correctly classified.

References

[1] T. Evgeniou and M. Pontil and T. Poggio. Regularization Networks and Support Vector Machines. Advances in Computational Mathematics, 2000.

[2] V. N. Vapnik. Statistical Learning Theory. Wiley, 1998.