Machine Learning Module

Week 3

Laboratory Exercise, Week 3

Probabilistic & Bayesian Methods

Mark Girolami
girolami@dcs.gla.ac.uk
Department of Computing Science
University of Glasgow

January 18, 2006

1 Laboratory Exercise

In this session we will now use our knowledge of probabilistic linear modeling to look at a richer class of function approximators than the simple polynomial-based methods.

1.1 Radial Basis Expansions

So far we have used linear models of the form

$$t = \sum_{m=1}^{M} w_m x^m$$

which provide a polynomial basis expansion of the functions we are attempting to model. There are however many forms of basis functions which we could decide to use (casting your mind back to first year maths you may remember the Fourier basis composed of multiples of sine and cosine functions). One basis which is particularly useful is the Radial Basis Function which is defined as

$$\phi(x) = \exp\left(-\beta|x-\mu|^2\right)$$

using this basis we can define flexible models of the form

$$t = \sum_{m=1}^{M} w_m \phi_m(x) = \sum_{m=1}^{M} w_m \exp\left(-\beta|x-\mu_m|^2\right)$$

Now of course we have introduced additional parameters into our model in the form of each location parameter μ_m and the common spread or influence parameter β which will have to be estimated in some way. On the other hand given that we have N data points in our sample then we can define our model as

$$t = \sum_{n=1}^{N} w_n \phi_n(x) = \sum_{n=1}^{N} w_n \exp\left(-\beta|x-x_n|^2\right)$$

So now we are taken a little radial-basis bump and placing one onto each of our data-points weighting each of the responses with a w_n and summing to obtain our model response. We will see that this provides a highly flexible method for modeling and we have only introduced one additional parameter (commonly called a hyper-parameter). Such a radial-basis model can be viewed as a **Kernel Machine** which we will meet later in the course.

1.2 Modeling the Sinc Function

The sinc function is defined as

$$f(x) = \frac{sin(x)}{x}$$

and is a relatively simple function but is of interest as the scale of the response if location dependent and as such modeling this function, with a finite amount of data, using polynomial basis expansions is doomed to failure. However using the radial-basis function introduced above it is fairly straightforward to provide a good predictive model for this function.

1.3 Exploring the Influence of Radial-Basis Width in Solutions

The following simple Matlab script (the file wk3_lab_1_sol.m is available in the week 3 laboratory folder) will generate *training* and *testing* data sets where noisy observations of the sinc function are simulated and these are used to obtain the posterior mean and associated posterior variance predictions at each of the *test points*. Use this code to see the effect that the Radial-Basis width parameter has on the predictive quality achieved. Print out some illustrative plots.

```
Range = 20;
N=30;
sigma = 0.25;
xt = [-Range/2:0.1:Range/2]';
xt(find(xt == 0))=[];
ft = sin(xt)./xt;

x = Range.*rand(N,1) - Range/2;
f = sin(x)./x;
[i,j] = sort(x);

e=sigma*randn(N,1);
t = f + e;

alpha = 100;
```

```
k=0.15;
X=kernel_func(x,x,'gauss',k,k)';
Xt=kernel_func(x,xt,'gauss',k,k)';

pos_cov = sigma*inv(X'*X + (sigma/alpha)*eye(N));
mu=pos_cov*X'*t./sigma;
pred_mean = Xt*mu;
pred_cov=diag(Xt*pos_cov*Xt');

hold on
plot(i,t(j),'k.');
plot(xt,ft)
plot(xt,pred_mean,'r')
plot(xt,pred_mean+sqrt(pred_cov),'r');
plot(xt,pred_mean-sqrt(pred_cov),'r');
```

1.4 Exploring the Influence of the Prior on the Solution

From your experience in the previous section you should now be aware that the radial-basis width has an enormous influence on the predictive capability of the model. How does the prior variance affect the model performance for a given Radial-Basis width parameter? Discuss you qualitative findings.

1.5 Using Cross Validation to Select the Optimal Prior and Radial-Basis Hyper-Parameters

In a real application there is no possibility to generate test-sets on which to assess how good your model is and we introduced LOOCV as a general model assessment and selection method. In our model we now have a pair of hyper-parameters which will index the possible model set, so we will have quite a large number of α & β combinations to consider.

Write a Matlab script which will do the following

1. Generate a *train* and *test* set for the sinc function for a pre-set and known noise level.

2. For a range of α & β values compute

(a) the MSE on the *training* data

(b) the *MSE* on the *test* data

(c) the LOOCV *MSE*

3. Identify the α, β pair that yields the minimum *train, test* and LOOCV errors

4. How well does LOOCV do in terms of locating optimal α, β pairings?

Write up your findings in the form of a report.

2 Radial-Basis Function in Matlab

You will notice that you require an auxilliary Matlab function which creates the $N \times N$ matrix of basis-function responses. A naive way of implementing this is to loop over all N data points and populate the $\mathbf{X}$ matrix as follows

```
for i = 1:N
    for j=1:N
        X(i,j) = exp(-h*(x(i) - x(j))^2);
    end
end
```

Of course as you will have no doubt noticed the radial-basis function is symmetric in its arguments (hence the name radial-basis) and so we can reduce the computational cost of populating the matrix from $\mathcal{O}(N^2)$ to $\mathcal{O}(0.5N(N-1))$ so we could use

```
for i = 1:N
    for j=i:N
        X(i,j) = exp(-h*(x(i) - x(j))^2);
        X(j,i) = X(i,j);
    end
end
```

Now Matlab performance degrades when loops are used and if at all possible Matlab code should be vectorised as much as possible so converting for loops to vector operations will yield a significant increase in performance. The function `kernel-func.m` is in vectorised form.