ENGN 4520: Introduction to Machine Learning

Alex Smola, RSISE ANU
Problem Sheet — Week 4
The due date for these problems is Monday, May 28

Teaching Period
April 30 to June 8, 2001

A Theory

Problem 14 (Convolutions and Random Variables, 4 Points)
Show that for two random variables ξ_1, ξ_2 with densities $p_1(\xi_1)$ and $p_2(\xi_2)$ the density of the random variable $\xi := \xi_1 + \xi_2$ is given by $p(\xi) = p_1 \circ p_2(\xi)$.

Problem 15 (Kernels, B_n-Splines, and Mercer's Condition, 10 Points)
In this problem we will introduce a new class of kernels. For this purpose denote by B_0 the indicator function on the interval $[-\frac{1}{2}, \frac{1}{2}]$, i.e.

$$B_0(x) = \begin{cases} 1 & \text{if } |x| \leq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

Furthermore we introduce the splines B_n on $\mathbb{R}$ via $B_{n+1} := B_n \circ B_0$.

1. *Compute the splines B_1 and B_2 analytically.*
2. *Show that B_n is a spline of order n, i.e. it is piecewise polynomial up to order n.* **Hint:** *use induction, i.e. assume that it is true for B_n and show that it then holds for B_{n+1}.*
3. *Compute the Fourier transform of B_0. Why does it follow from this that $k(x, x') := B_0(x - x')$ is not a vaild kernel?*
4. *Show that the Fourier transform of B_n is given by $\tilde{B}_n = (2\pi)^{\frac{n}{2}} \left(\tilde{B}_0\right)^n$. Which $k(x, x') := B_n(x - x')$ is therefore a valid kernel?*
5. **Bonus question (difficult):** *Show that $p_n(x) := \frac{n+1}{12} B_n\left(\frac{n+1}{12} x\right)$ converges to a normal distribution with zero mean and unit variance.*

   Hint: *Use the result of Problem 14. Next show that B_n is the density corresponding to a sum of $n+1$ random variables uniformly distributed on $[-\frac{1}{2}, \frac{1}{2}]$. Finally, show that p_n has zero mean and unit variance and apply the central limit theorem (from second week) to prove the claim.*

Problem 16 (Radial Basis Function Kernels, 6 Points)
Denote by $k(\mathbf{x}, \mathbf{x}') := \kappa(\|\mathbf{x} - \mathbf{x}'\|)$ a radial basis function kernel.

1. *Show that for a strictly monotonically decreasing $\kappa : [0, \infty) \to \mathbb{R}$ the mapping into a feature space is neighbourhood preserving, i.e. that*

   $$d(\Phi(\mathbf{x}), \Phi(\mathbf{x}')) \leq d(\Phi(\mathbf{x}), \Phi(\mathbf{x}'')) \text{ is equivalent to } d(\mathbf{x}, \mathbf{x}') \leq d(\mathbf{x}, \mathbf{x}'')$$

2. *Show that for the kernels given below the feature map Φ maps all $\mathbf{x}$ onto the surface of a sphere, and more precisely, into an orthant of $90°$.*

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{x}'\|^2\right) \quad (1)$$

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{\sigma}\|\mathbf{x} - \mathbf{x}'\|\right) \quad (2)$$

Alex Smola, RSISE ANU
Problem Sheet — Week 4
The due date for these problems is Monday, May 28

Teaching Period
April 30 to June 8, 2001

B Programming

Problem 17 (Kernels and Regression, 20 Points) *As in previous week's exercise, we assume squared loss and we also keep the regularization terms. However, this time we are going to use kernels. This means that f is given by*

$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i k(\mathbf{x}_i, \mathbf{x})$$

where the $\mathbf{x}_i$ are the training points.

1. *Implement in MATLAB an algorithm that takes $(\mathbf{x}_1, \ldots, \mathbf{x}_m)$, $(y_1, \ldots, y_m)$ and the kernel object as an input and produces the vector of α_i which minimize the* **empirical risk** *as an output.*

   Note: the matrix K may have very small eigenvalues. *You can use* `pinv` *in MATLAB to deal with the problem.*

2. *Test your program on data generated by*

   $$y = f(x) + \xi \text{ where } f(x) = 1 + 2x + 3\exp(-(3-x)^2) + 2\exp(-(5-x)^2)$$

   More specifically, draw x unformly at random from $[0, 10]$ and let ξ be normally distributed with zero mean and variance σ. Plot the estimate of $f(x)$ on $[0, 10]$ for ($m = 50, \sigma = 0$) and ($m = 50, \sigma = 0.5$)
   with the following kernels

   - *a Gaussian RBF kernel with kernel width $\omega = 1$ and with $\omega = 0.5$.*
   - *a polynomial kernel $k(\mathbf{x}, \mathbf{x}') = (\langle \mathbf{x}, \mathbf{x}' \rangle + 1)^5)$ and with $(\langle \mathbf{x}, \mathbf{x}' \rangle + 1)^3)$.*

3. *Now we introduce a quadratic regularization term via*

   $$\Omega[f] = \frac{1}{2}\|\mathbf{w}\|^2$$

   to minimize

   $$R_{\text{reg}}[f] = R_{\text{emp}}[f] + \lambda\Omega[f]$$

   Modify your MATLAB code from above such that the algorithm takes $(\mathbf{x}_1, \mathbf{x}_m)$, $(y_1, \ldots, y_m)$ and the kernel object as an input and produces the vector of α_i which minimize the **regularized risk** *as an output.*

4. *Test your program in the above settings for $\lambda = 1, 0.1, 0.01$ where ($m = 50, \sigma = 0.5$) and the following kernels*

   - *a Gaussian RBF kernel with kernel width $\omega = 1$ and with $\omega = 0.5$.*
   - *a polynomial kernel $k(\mathbf{x}, \mathbf{x}') = (\langle \mathbf{x}, \mathbf{x}' \rangle + 1)^5)$ and with $(\langle \mathbf{x}, \mathbf{x}' \rangle + 1)^3)$.*

(m: number of observations, σ: variance of additive noise, ω: width of the Gaussian RBF kernel, λ: regularization constant)

ENGN 4520: Introduction to Machine Learning

Alex Smola, RSISE ANU
Problem Sheet — Week 4
The due date for these problems is Monday, May 28

Teaching Period
April 30 to June 8, 2001

Instructions for SVLab Toolbox

Download SVLab `http://axiom.anu.edu.au/~smola/engn4520/svlab.zip` (lite version) and install it on your system. To use it you have to set

```
path('svlab', path);
path('svlab/common', path);
```

It allows you to compute kernel functions efficiently. You need to initialize a kernel object first. This is done via

```
kernel = vanilla_dot;       %the normal dot product as a kernel

kernel = rbf_dot;           %this creates an RBF kernel
kernel.sigma = 10;          %this uses the kernel exp(-sigma ||x-x'||^2)

kernel = poly_dot;          %this creates a polynomial kernel
kernel.degree = 5;          %polynomial kernel of degree 5
kernel.offset = 2;          %and offset 2, i.e. (<x,x'> + 2)^5
```

Among others, the toolbox offers the functions `sv_dot` and `sv_mult`. They work as follows

```
mtrain = 1000;              %generate 1000 training observations
mtest  = 500;               %test set 500
n = 100;                    %in 100 dimensions
xtrain = randn(n,mtrain);
xtrain = randn(n,mtest);

y = randn(mtrain,1);        %labels
alpha = randn(mtrain,1);    %weights

k = sv_dot(kernel, xtrain);
%this computes the kernel matrix K_ij = k(x_i, x_j)
f = sv_mult(kernel, xtrain, alpha);
%compute the function values f(x_j) given by sum_i alpha_i k(x_i, x_j)

ktest = sv_dot(kernel, xtrain, xtest);
%this computes the kernel matrix K_ij = k(xtrain_i, xtest_j)
ftest = sv_mult(kernel, xtest, xtrain, alpha);
%compute the function values f(xtest_j) given by sum_i alpha_i k(xtrain_i, xtest_j)
```