CMSE 820: Homework #8

Due on Nov 10, 2019 at 11:59pm

Professor Yuying Xie

Boyao Zhu

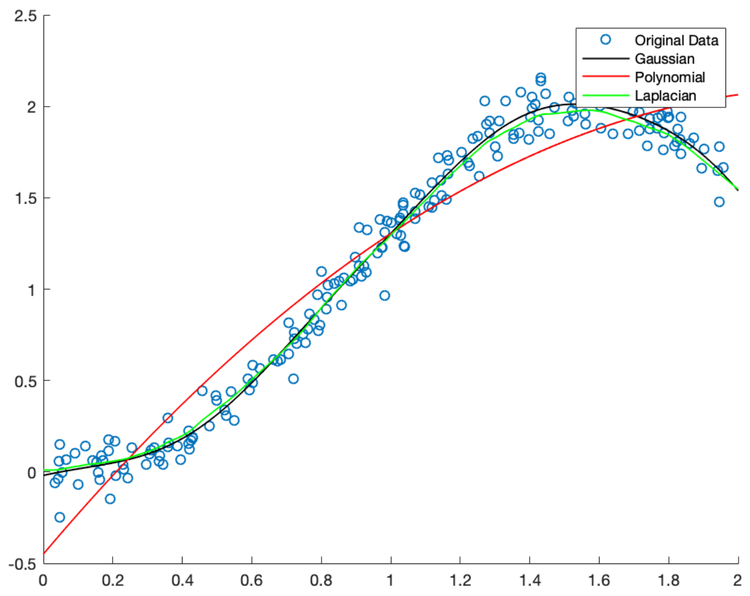

Figure 1: Plot of kernel regression functions $f(x)$ vs x and original data

Problem 1

Solution

The plot is shown above.

The regression kernels and tuning parameter λ used are

(1) Gaussian kernel: $k(x, x') = \exp(-\|x - x'\|^2/0.25)$ with $\lambda = 0.01$

(2) Polynomial kernel: $k(x, x') = (\langle x, x'\rangle + 1)^2$ with $\lambda = 0.1$

(3)Laplacian kernel: $k(x, x') = \exp(-\|x - x'\|)$ with $\lambda = 1$

By comparing the RSS for the regression fit, we find that the quality of Gaussian kernel regression and Laplacian kernel regression is much better than that of Polynomial kernel regression. We can explain this by noting that the feature maps of Gaussian kernel and Laplacian kernel map $\mathcal{X}$ to an infinite dimensional space, while the feature map of the polynomial kernel is of a finite dimension.

Residual(Gaussian) = 0.0079

Residual(Polynomial) = 0.0391

Residual(Laplacian) = 0.0081

Problem 2

Solution

We perform binary classifications for pairs of 1, 2, 3, 4, i.e., {1, 2}, {1, 3}, $\cdots$, {3, 4}. The data set for each pair is taken from the MNIST data set and split into two parts: training data set (60%) and test data set (40%). RKHS ridge regression with the quadratic polynomial $k(x, x') = (\langle x, x'\rangle + 1)^2$ is performed. The tuning parameter λ is fixed at $\lambda = 10$ for simplicity.

	2	3	4
1	99.70%	99.68%	99.81%
2		99.19%	99.73%
3			99.95%

Problem 3

Solution
Let $\mathcal{B} = \{f_\alpha \in \mathcal{H}_k : f_\alpha = \sum_{i=1}^n \alpha_i k(x_i, \cdot), \alpha \in \mathbb{R}$ be a subspace of $\mathcal{H}_k$. Since $\mathcal{B}$ is of finite dimension, $\mathcal{B}$ is closed, and therefore $\mathcal{H}_k = \mathcal{B} \oplus \mathcal{B}^\perp$. That is, $\forall f \in \mathcal{H}_k$, there exists uniquely $f_\alpha \in \mathcal{B}$ and $f' \in \mathcal{B}^\perp$ such that $f = f_\alpha + f'$. Moreover, for any $i = 1, \cdots, n$,

$$f(x_i) = \langle f_\alpha + f', k(x_i, \cdot)\rangle = \langle f_\alpha, k(x_i, \cdot)\rangle = f_\alpha(x_i)$$

Note also that, for any $f \in \mathcal{H}_k, c[x_i, y_i, f(x_i)] = c[x_i, y_i, f_\alpha(x_i)]$, and

$$\Omega(\|f\|_{\mathcal{H}_k}) = \Omega(\|f_\alpha + f'\|_{\mathcal{H}_k}) = \Omega(\sqrt{\|f_\alpha\|^2_{\mathcal{H}_k} + \|f'\|^2_{\mathcal{H}_k}}) \geq \Omega(\|f_\alpha\|_{\mathcal{H}_k})$$

So the minimizer $\tilde{f}^* \in \mathcal{B}\oplus$ span$\{\phi_j\}_{j=1}^m \subseteq \mathcal{F}$. That is, there exists $\alpha_i \in \mathbb{R}$ for all integers $1 \leq i \leq n$ and $\beta_j \in \mathbb{R}$ for all integers $1 \leq j \leq n$ such that

$$\tilde{f}^* = \sum_{i=1}^n \alpha_i k(x_i, \cdot) + \sum_{j=1}^m \beta_j \psi_j$$

Once α is determined, the coefficient β can be found from

$$\Psi\beta = \hat{Y} - \mathbf{K}\alpha$$

where $\hat{Y} = (y_1, y_2, \cdots, y_n) \in \mathbb{R}^n, K_{ij} = k(x_i, x_j)$ and $\Psi_{ij} = \psi_j(x_i)$. Since Ψ is of full rank, it is invertible and β can be uniquely determined by

$$\beta = \Psi^{-1}P_\Psi(\hat{Y} - textbfK\alpha) = (\Psi^T\Psi)^{-1}\Psi^T(\hat{Y} - \mathbf{K}\alpha)$$

Problem 4

Solution
Is it obvious that $\tilde{k}$ is symmetric by construction. ($tildek$ is well-defined when $k(x, x) > 0$ for all $x \in \mathcal{X}$.) For any positive integer $N \in \mathbb{N}^+$, for any real number $c_i \in \mathbb{R}$, for any $x_i \in \mathbb{X}$ where $i = 1, \cdots, N$, we have

$$\sum_{ij} c_i c_j \frac{k(x_i, x_j}{\sqrt{k(x_i, x_i)k(x_j, x_j}} = \sum_{ij} \Big(\frac{c_i}{\sqrt{k(x_i, x_i}}\Big)\Big(\frac{c_j}{\sqrt{k(x_j, x_j}}\Big)k(x_i, x_j) \geq 0$$

The last inequality follows from the positive semi-definiteness of k. by the statement above, $\tilde{k}$ is also p.s.d.

Problem 5

Solution 1.
Solve

$$\hat{f} = \arg\min_{f\in\mathcal{H}} \sum_{i=1}^n (y_i - f(x_i))^2 + \lambda\|\omega\|^2, \quad (*)$$

Problem 5 continued on next page. . .

Let Ψ be an $n \times N$ matrix such that $\Psi_{ij} = \sqrt{\lambda_j}\psi_j(x_i)$ for $i = 1, \cdots, n$ and $j = 1, \cdots, N$ Define $y = (y_1, y_2, \cdots, y_n)^T \in \mathbb{R}^n, \hat{y} = (f(x_1), f(x_2), \cdots, f(x_n))^T = \Psi\omega \in \mathbb{R}^n$. We can rewrite the original problem as

$$\hat{\omega} = \arg\min_{\omega \in \mathbb{R}^N} \|Y - \Psi\omega\|^2 + \lambda\|\omega\|^2$$

The corresponding Lagrangian is $\mathcal{L}(\omega) = \|Y - \Psi\omega\|^2 + \lambda\|\omega\|^2$. Setting its partial derivative to 0 yields

$$\frac{\partial \mathcal{L}}{\partial \omega} = -2\Psi^T(Y - \Psi\omega) + 2\lambda\omega = 0 \implies \hat{\omega} = (\Psi^T\Psi + \lambda\mathbf{I})^{-1}\Psi^T Y$$

so $\hat{f} = \sum_{j=1}^N \hat{\omega}_j \sqrt{\lambda_j}\psi_j$.

Solution 2.
Recall the RKHS Ridge regression problem

$$\hat{f} = \arg\min_{f \in \mathcal{H}} \sum_{i=1}^n (y_i - f(x_i))^2 + \lambda\|f\|_{\mathcal{H}}^2 \iff \hat{\alpha} = \arg\min_{\alpha \in \mathbb{R}^n} \|Y - \mathbf{K}\alpha\|^2 + \lambda\alpha^T\mathbf{K}\alpha$$

where $\mathbf{K}$ is an $n \times n$ matrix such that $\mathbf{K}_{ij} = k(x_i, x_j)$. The solution to this problem is given by

$$\hat{\alpha} = (\mathbf{K} + \lambda(I))^{-1}Y$$

Note also that by the Mercer's theorem

$$\mathbf{K} = \Psi\Psi^T$$

We will show that the RKHS Ridge regression solution is also a solution to the problem (*). Firstly, we need to show that the residual sums of squares are the same

$$\begin{aligned}
&\|Y - \hat{Y}_{\hat{\alpha}}\|^2 = \|Y - \mathbf{K}\hat{\alpha}\|^2 = \|Y - \Psi\hat{\omega}\|^2 = \|Y - \hat{Y}_{\hat{\omega}}\| \\
\iff& \mathbf{K}\hat{\alpha} = \Psi\hat{\omega} \\
\iff& \mathbf{K}(\mathbf{K} + \lambda\mathbf{I})^{-1}Y = \Psi(\Psi^T\Psi + \lambda\mathbf{I})^{-1}\Psi^T Y \\
\iff& \mathbf{K}(\mathbf{K} + \lambda\mathbf{I})^{-1} - \Psi(\Psi^T\Psi + \lambda\mathbf{I})^{-1}\Psi^T = 0 \\
\iff& \mathbf{K} - \Psi(\Psi^T\Psi + \lambda\mathbf{I})^{-1}\Psi^T(\mathbf{K} + \lambda\mathbf{I}) = 0 \\
\iff& \mathbf{K} - \Psi(\Psi^T\Psi + \lambda\mathbf{I})^{-1}\Psi^T(\Psi\Psi^T + \lambda\mathbf{I}) = 0 \\
\iff& \mathbf{K} - \Psi(\Psi^T\Psi + \lambda\mathbf{I})^{-1}(\Psi^T\Psi + \lambda\mathbf{I})\Psi^T = 0 \\
\iff& \mathbf{K} - \Psi\Psi^T = 0
\end{aligned}$$

Secondly, we need to show that the penalty terms are equal as well. Let $\omega_{\hat{\alpha}}$ denote the corresponding ω of the model $\hat{\alpha}$ that satisfies $f_{\hat{\alpha}} = \sum_j \omega_{\hat{\alpha}}\sqrt{\lambda_j}\psi_j(\cdot) = \sum_i \alpha_i k(\cdot, x_i)$. In particular, considering the data points x_i only, we have $\Psi\omega_{\hat{\alpha}} = \mathbf{K}\alpha$. Since $\mathbf{K}\hat{\alpha} = \Psi\hat{\omega}$

$$\Psi\omega_{\hat{\alpha}} = \Psi\hat{\omega} \Longrightarrow \omega_{\hat{\alpha}} = \hat{\omega} \Longrightarrow \|\omega_{\hat{\alpha}}\| = \|\hat{\omega}\|$$

Thus, the model $f_{\hat{\alpha}}$ given by $\hat{\alpha}$ also solves the problem $(*)$. Since both the problem $(*)$ and the RKHS Ridge regression problem are strictly convex optimization problems, so the solution is unique, implying that $f_{\hat{\alpha}} = f_{\hat{\omega}}$ solves both of the problems. Hence, the two problems are equivalent.

HW8

November 10, 2019

[4]:
```
M = csvread('HW8_dat.csv',1);
M = transpose(M);
X = M(2,:);
Y = M(1,:);
Y_T = transpose(Y);
XX = linspace(0,2,200);

% Gaussian
K_G = KappaMatrix(X,'Gaussian');
alpha_G = RKHSRegression(K_G,Y_T,0.01);
YY_G = zeros(1,200);
for i = 1:200
    YY_G(i) = RegressionFunction(XX(i),X,alpha_G,'Gaussian');
end
% figure();
% scatter(X,Y);
% hold on;
% plot(XX,YY_G,'LineWidth',1,'Color','black');
% hold off;
RSS(X,Y,alpha_G,'Gaussian')

% Polynomial
K_P = KappaMatrix(X,'Polynomial');
alpha_P = RKHSRegression(K_P,Y_T,0.1);
YY_P = zeros(1,200);
for i = 1:200
    YY_P(i) = RegressionFunction(XX(i),X,alpha_P,'Polynomial');
end
% figure();
% scatter(X,Y);
% hold on;
% plot(XX,YY_P,'LineWidth',1,'Color','black');
% hold off;
RSS(X,Y,alpha_P,'Polynomial')

% Laplacian
K_L = KappaMatrix(X,'Laplacian');
```

```
alpha_L = RKHSRegression(K_L,Y_T,1);
YY_L = zeros(1,200);
for i = 1:200
    YY_L(i) = RegressionFunction(XX(i),X,alpha_L,'Laplacian');
end
% figure();
% scatter(X,Y);
% hold on;
% plot(XX,YY_L,'LineWidth',1,'Color','black');
% hold off;
RSS(X,Y,alpha_L,'Laplacian')

figure();
scatter(X,Y);
hold on;
plot(XX,YY_G,'Linewidth',1,'Color','black');
hold on;
plot(XX,YY_P,'Linewidth',1,'Color','red');
hold on;
plot(XX,YY_L,'Linewidth',1,'Color','green');
legend('Original Data','Gaussian','Polynomial','Laplacian')
hold off;

function G = GaussianKernel(x,y)
    G = exp(-(norm(x-y)^2/0.25));
end

function P = PolynomialKernel(x,y,c)
    P = (dot(x,y)+c)^2;
end

function L = LaplacianKernel(x,y)
    L = exp(-norm(x-y));
end

function K = KappaMatrix(X,type)
  p = size(X,1);
  n = size(X,2);
  K = zeros(n,n);
  for i=1:n
      xx = X(:,i);
      for j=i:n
         yy = X(:,j);
         if type == "Gaussian"
           K(i,j)=GaussianKernel(xx,yy);
         elseif type == "Polynomial"
           K(i,j)=PolynomialKernel(xx,yy,1);
         elseif type == "Laplacian"
           K(i,j)=LaplacianKernel(xx,yy);
         end
         K(j,i)=K(i,j);
      end
   end
end

function alpha = RKHSRegression(K,Y,lambda)
     n = size(K,1);
     alpha = (K+lambda*eye(n,n))\Y;
end

function y = RegressionFunction(x,X,alpha,type)
    n = size(X,2);
    y = 0;
    for i = 1:n
         xx = X(:,i);
         if type == "Gaussian"
           y = y + alpha(i)*GaussianKernel(xx,x);
         elseif type == "Polynomial"
           y = y + alpha(i)*PolynomialKernel(xx,x,1);
         elseif type == "Laplacian"
           y = y + alpha(i)*LaplacianKernel(xx,x);
         end
    end
end

function residual = RSS(X,Y,alpha,type)
    residual = 0;
    n = size(X,2);
    for i = 1:n
        diff = Y(i) - RegressionFunction(X(i),X,alpha,type);
        residual = residual + diff*diff;
    end
    residual = residual/n;
end
```

```
  File "<tokenize>", line 90
    end
    ^
IndentationError: unindent does not match any outer indentation level
```

[6]:
```
L = load('MNIST_20x20.mat');
labels = L.labels;
imgs = L.imgs;
n = size(labels,1);
counter = zeros(4);
for i = 1:n
    if labels(i) == 1
        counter(1) = counter(1) + 1;
        IMG(counter(1),1) = i;
    elseif labels(i) == 2
        counter(2) = counter(2) + 1;
        IMG(counter(2),2) = i;
    elseif labels(i) == 3
        counter(3) = counter(3) + 1;
        IMG(counter(3),3) = i;
    elseif labels(i) == 4
        counter(4) = counter(4) + 1;
        IMG(counter(4),4) = i;
    end
end

% pair: k & l
for k = 1:3
    for l = k+1:4
        total = counter(k)+counter(l);
        X = zeros(401,total);
        for i = 1:counter(k)
            X(1:400,i) = reshape(imgs(:,:,IMG(i,k)),[400,1]);
            X(401,i) = 1;
        end
        for i = 1:counter(l)
            j = counter(k)+i;
            X(1:400,j) = reshape(imgs(:,:,IMG(i,l)),[400,1]);
            X(401,j) = -1;
        end

        ntraining = round(total*0.6);
        temp = zeros(401,1);
        for i = 1:ntraining
            r = randi([i,total]);
            temp = X(:,i);
            X(:,i) = X(:,r);
            X(:,r) = temp;
        end
        K = KappaMatrix(X(1:400,1:ntraining),'Polynomial');
        alpha = RKHSRegression(K,transpose(X(401,1:ntraining)),10);
        correct = 0;
        for i = ntraining+1:total
            xx = X(1:400,i);
            yy = RegressionFunction(xx,X(1:400,1:ntraining),alpha,'Polynomial');
            if yy*X(401,i)>0
                correct = correct + 1;
            end
        end
        accuracy(k,l) = correct/(total-ntraining);
    end
end





function P = PolynomialKernel(x,y,c)
    P = (dot(x,y)+c)^2;
end

function K = KappaMatrix(X,type)
  p = size(X,1);
  n = size(X,2);
  K = zeros(n,n);
  for i=1:n
      xx = X(:,i);
      for j=i:n
         yy = X(:,j);
         if type == "Gaussian"
           K(i,j)=GaussianKernel(xx,yy);
         elseif type == "Polynomial"
           K(i,j)=PolynomialKernel(xx,yy,1);
         elseif type == "Laplacian"
           K(i,j)=LaplacianKernel(xx,yy);
         end
         K(j,i)=K(i,j);
      end
   end
end

function alpha = RKHSRegression(K,Y,lambda)
     n = size(K,1);
     alpha = (K+lambda*eye(n,n))\Y;
end

function y = RegressionFunction(x,X,alpha,type)
    n = size(X,2);
    y = 0;
    for i = 1:n
         xx = X(:,i);
         if type == "Gaussian"
           y = y + alpha(i)*GaussianKernel(xx,x);
         elseif type == "Polynomial"
           y = y + alpha(i)*PolynomialKernel(xx,x,1);
         elseif type == "Laplacian"
           y = y + alpha(i)*LaplacianKernel(xx,x);
         end
    end
end
```

```
      File "<tokenize>", line 85
    end
    ^
IndentationError: unindent does not match any outer indentation level
```

[ ]:

```

```