Machine Learning Assignment1

Xiaoxu Gao 4504348

Question a

The closed form solution of regular ridge regression (p=2) is

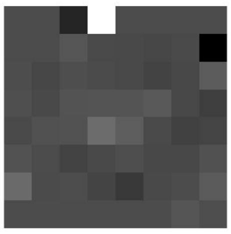
$$w = (X^T X + \lambda I)^{-1} X^T Y$$

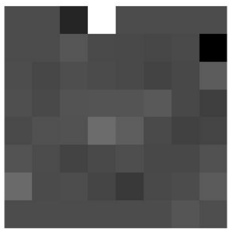
```
Matlab code:
data = load('optdigitsubset.txt');
dim = 64;
num = 1125;
X = data;
X = [ones(num,1) X];
Y = [ones(554,1).*(-1);ones(571,1)];
lambda = 1;
w = (X'*X+lambda*[zeros(1,dim+1);[zeros(dim,1) eye(dim)]];)^(-1)*X'*Y;
w = w(2:end);
```

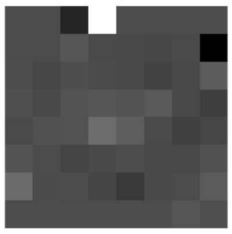
Question b

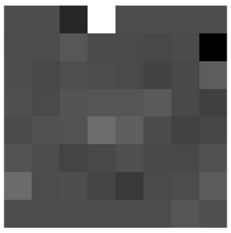
Based on Question a, the following is the Matlab code realted to plot the images.

```
img = reshape(w,[8,8]); %resize the images
img = mat2gray(img);
imshow(img,'InitialMagnification','fit'); %fit the screen
```

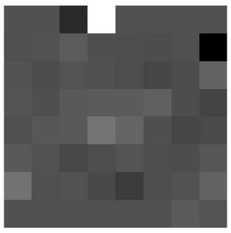

$\lambda = 10^{-2}$


$\lambda = 10^{-1}$

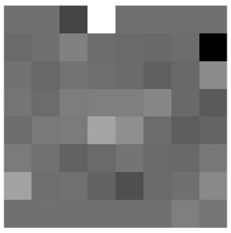

$\lambda = 1$



$\lambda = 10^1$



$\lambda = 10^2$

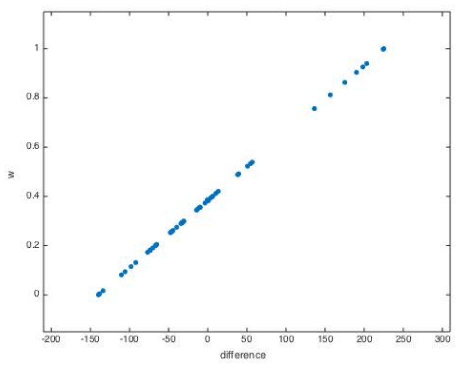


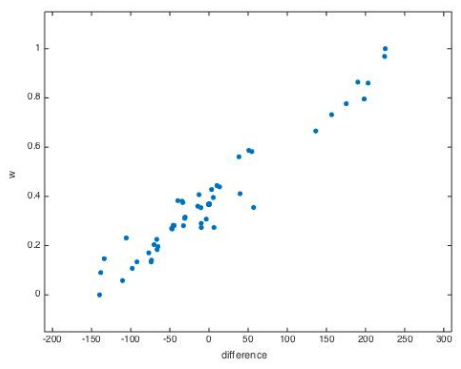
$\lambda = 10^3$

Question c

It's assumed that there are two classes with equal size. The first 554 rows are class0, the rest 544 rows are class1. The following is Matlab code. To better show the tendency, extremely large λ are used here. We can see that for large λ, the solution w is bascially propotional to the difference between the two class means.

```
Matlab code:
data = load('data.txt');
dim = 64;
num = 1108; %every class has 554 samples
X = data((1:num),:);
X = [ones(num/2,1).*(-1);ones(num/2,1)];
lambda = 10^(10);
w = (X'*X+lambda*eye(dim+1))^(-1)*X'*Y;
w = w(2:end);
w = mat2gray(w); % the result of w
class0 = X((1:num/2),:);
class1 = X((num/2+1:end),:);
result0 = []; % the means of class0
for i = 1:65
    temp0 = sum(class0(:,i))/(num/2);
    result0 = [result0;temp0];
end
result1 = []; % the means of class1
for j = 1:65
    temp1 = sum(class1(:,j))/(num/2);
    result1 = [result1;temp1];
end
difference = result1 - result0;
difference = differenct(2:end) % difference between the two class means
plotmatrix(difference,w); %scatter plot to show the tendency
```





$\lambda = 10^{10}$ $\lambda = 10^{7}$

Question d

$$\min_{w \in \Re^d} \sum_{i=1}^{N} (w^T x_i - y_i)^2 + \lambda \sum_{i=1}^{d} |w_i|^2$$

The derivateive of this formula to w_k equals:

$$2 \sum_{i=1}^{N} \left[x_i k (w^T x_i - y_i) \right] + 2\lambda w_k$$

$$2 \sum_{i=1}^{N} (w^T x_{ik} - x_{ik} y_i) + 2\lambda w_k$$

Since$\sum_{j=1}^{d} x_{ij} w_j$ equals to $w^T x_i$, we can get the following formula:

$$2 \sum_{i=1}^{N} \left[\left(\sum_{j=1}^{d} x_{ij} w_j \right) - x_{ik} y_i \right] + 2\lambda w_k$$

Question e

Gradient Descent

step1: start with some $\theta_0, \theta_1 ... \theta_{64} (\theta_0 = \theta_1 = ... = \theta_{64} = 0)$
step2: keep changing $\theta_0, \theta_1 ... \theta_{64}$ until get minimum value of $J(\theta_0, \theta_1, ... \theta_{64})$

Update process:
1. $temp_0 := \theta_0 - \alpha \frac{\mathrm{d}}{\mathrm{d}\theta_0} J(\theta_0, \theta_1, ... \theta_{64})$
2. $temp_j := \theta_j - \alpha \frac{\mathrm{d}}{\mathrm{d}\theta_\mathrm{j}} J(\theta_0, \theta_1, ... \theta_{64})$
3. $\theta_0 = temp_0$
4. $\theta_j = temp_j$

Optimization Toolbox

Matlab toolbox can be used to accomplish this task.Here, I use $fminunc$ function to find the minimum cost.

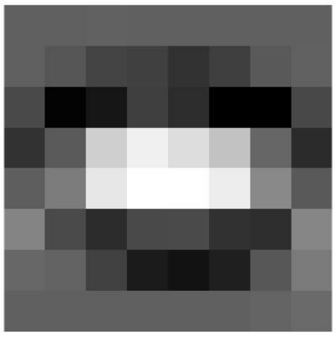
```
Matlab code:
theta0 = zeros(size(X,2),1);
lambda = 1;
options = optimset('GradObj','on','MaxIter',400);
[theta,J]=fminunc(@(t)(costFunction(t,X,Y,lambda)),theta0,options);

function[cost,grad] = costFunction(theta,X,Y,lamdba)
    num = length(Y);
    h = X*theta;
    cost = (1/(2*num))*(sum((h-Y).^2)+lamdba*sum(theta(2:end).^2));
    w = zeros(size(theta));
    for i = 1:num
        w = w + (h(i)-Y(i))*X(i,:)';
    end
    r = lamdba*[0;theta(2:end)];
    grad = (1/num)*(w+r);
end
```

Question f

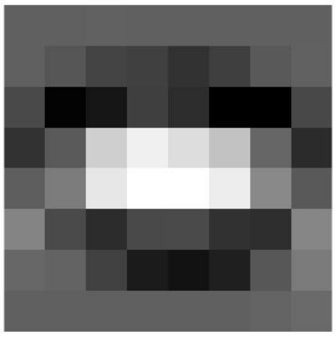
Comparison between Normal Equation and Gradient Descent:

	Result	**Computation time**
Normal Equation	see Questionb	0.157219
Gradient Descent	see Questionf	1.972519

Basically, the difference between these two results are really small. And with the increase of λ, the difference get smaller.



$NormalEquation(\lambda = 10^{10})$



$GradientDescent(\lambda = 10^{10})$

Question g

$$\lambda||w||_p^p$$

equals:

$$\lambda \sum_{i=1}^{d} |w_i|^p$$

The derivative of this formula is

$$\lambda psign(w_k)|w_k|^{p-1}$$

which equals to $\lambda pw_k|w_k|^{p-2}$

Question h

Extend gradient descent to deal with the general L_p norm regularizer.
Based on the formula in Question g, we can get the following code.Since the main program is the same, I only show the code for costFunction.

```
Matlab code:
function[cost,grad] = costFunction(theta,X,Y,lamdba)
    num = length(Y);
    h = X*theta;
    cost = (1/(2*num))*(sum((h-Y).^2)+lamdba*sum(theta(2:end).^2));
    w = zeros(size(theta));
    for i = 1:num
        w = w + (h(i)-Y(i))*X(i,:)';
    end
    p = 3;
    r = lamdba*(p/2)*[0;theta(2:end)].*[0;theta(2:end)].^(p-2); %new
regularization
    grad = (1/num)*(w+r);
end
```

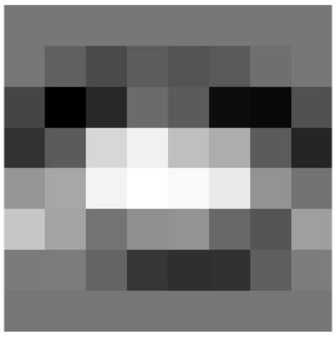
Question i

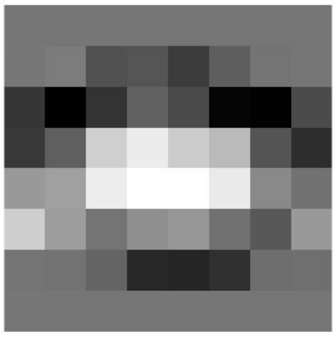
In this question, I choose $\lambda = 10^6$,$p = 2, p = 1$.There are two classes, each class has ten samples. Since we are talking about $p = 2$ and $p = 1$ here, we need to use the following formular:

$$\lambda psign(w_k)|w_k|^{p-1}$$

```
Matlab code:
data = load('data.txt');
dim = 64;
num = 20;
X1 = data(1:20,:);
X2 = data(571:590,:);
X  = [X1;X2];
X  = [ones(40,1) X];
Y  = [ones(20,1).*(-1);ones(20,1)];
lambda = 10^6;
theta0 = zeros(size(X,2),1);
options = optimset('GradObj','on','MaxIter',400);
[theta, J] = fminunc(@(t)(costFunction(t,X,Y,lambda)),theta0,options);
theta = theta(2:end);
plot(theta);

function[cost,grad] = costFunction(theta,X,Y,lamdba)
    num = length(Y);
    h = X*theta;
    cost = (1/(2*num))*(sum((h-Y).^2)+lamdba*sum(theta(2:end).^2));
    w = zeros(size(theta));
    for i = 1:num
        w = w + (h(i)-Y(i))*X(i,:)';
    end
    p = 2; %change the value of p
    r = lamdba*(p/2)*sign([0;theta(2:end)]).*[0;theta(2:end)].^(p-1);
    grad = (1/num)*(w+r);
end
```

Images of solution w($\lambda = 10^6$):





$p = 1$ $p = 2$

When p gets closer to 1:

When p gets closer to 1, the value of cost function gets bigger.

The reason why some pixel values of w get closer to zero:

Here we use regularization to avoid overfitting, so making some ws small is one of the solutions.

Question j

When there are many features but only a few samples, it's highly possible that we will face the problem of overfitting.