1. Logistic regression Hessian (10 points)

Determine a formula for the gradient and the Hessian of the regularized logistic regression objective function. Argue that the objective function

$$J(\boldsymbol{\theta}) = -\ell(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|^2$$

is convex, and that for regularization parameter $\lambda > 0$, the objective function is strictly convex.

1.

$$J(\theta) = \sum_{i=1}^{n} \log\left(1 + e^{y_i \theta^T \tilde{x}_i}\right) + \lambda \|\theta\|^2$$

$$= \sum_{i=1}^{n} \log\left(1 + e^{y_i \theta^T \tilde{x}_i}\right) + \lambda \theta^T \theta$$

$$\frac{\partial J}{\partial \theta} = \sum_{i=1}^{n} y_i \tilde{x}_i e^{y_i \theta^T \tilde{x}_i} \frac{1}{1 + e^{y_i \theta^T \tilde{x}_i}} + 2\lambda\theta$$

$$\frac{\partial}{\partial \theta^T}\left(\frac{\partial J}{\partial \theta}\right) = \sum_{i=1}^{n} y_i^2 \tilde{x}_i \tilde{x}_i^T e^{y_i \theta^T \tilde{x}_i} \frac{1}{(1 + e^{y_i \theta^T \tilde{x}_i})^2} + 2\lambda I$$

$$x^T \frac{\partial}{\partial \theta^T}\left(\frac{\partial J}{\partial \theta}\right) x = \sum_{i=1}^{n} y_i^2 \frac{e^{y_i \theta^T \tilde{x}_i}}{(1 + e^{y_i \theta^T \tilde{x}_i})^2} \underbrace{x^T \tilde{x}_i \tilde{x}_i^T x}_{\|\tilde{x}_i^T x\|^2} + \underbrace{2\lambda x^T I x}_{2\lambda \|x\|^2}$$

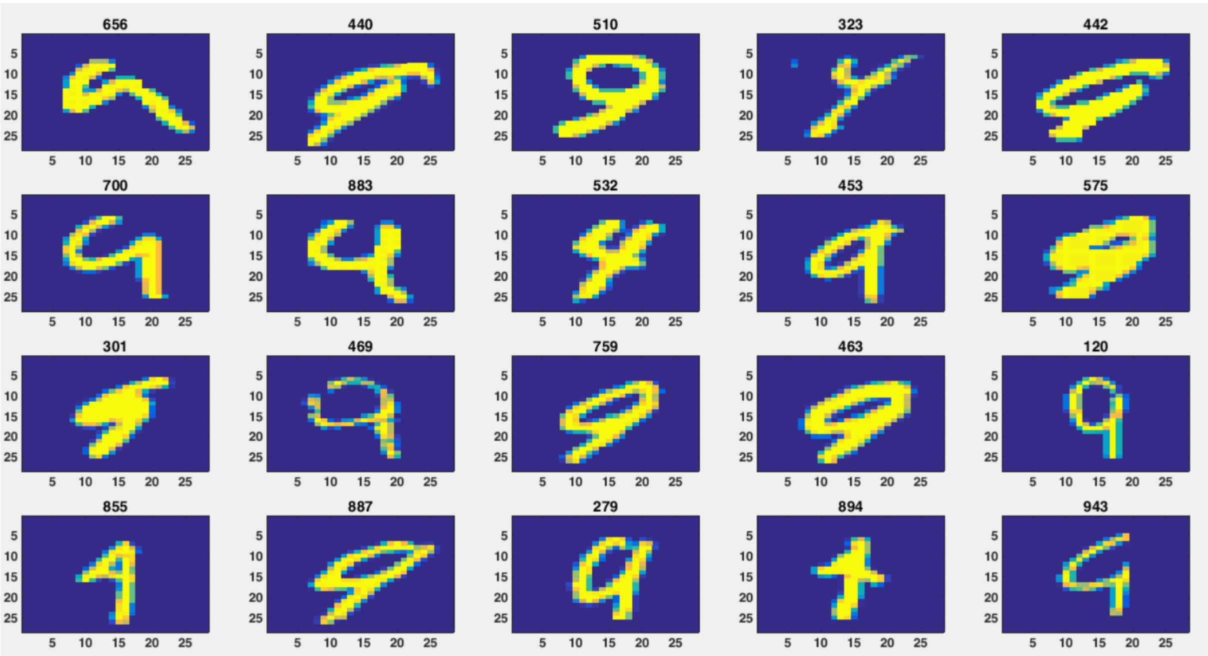
This expression is ≥ 0 in general. Therefore, $J(\theta)$ is convex

If $\lambda > 0$, second term is > 0 for $x \neq 0$. Therefor, expression is > 0. So, $J(\theta)$ is strictly convex

2. Handwritten digit classification with logistic regression (5 points each)

Download the file mnist_49_3000.mat from Canvas under Files → Data and Helper Functions. This is a subset of the MNIST handwritten digit database, which is a well-known benchmark database for classification algorithms. This subset contains examples of the digits 4 and 9.

Test shows 48 errors (i.e. 4.8% misclassification rate). Termination criterion was 10% error of optimizer ($\hat{\theta}$). Value of objective function at optimum is 307.84.



Code for calculating thetahat

```
close all; clear all; clc
load mnist_49_3000

global x_train y_train lambda thetahat

[d,n]=size(x);

x_train=x(:,1:2000);
y_train=y(1:2000);
x_test=x(:,2001:3000);
y_test=y(2001:3000);
lambda=10;

k=1;
theta(:,1)=zeros(size(x_train,1)+1,1);
theta_old=ones(size(x_train,1)+1,1);
epsilon=1e-1;
while norm(theta_old-theta(:,k))>epsilon
    grad(theta(:,k));
```

```
    hes(theta(:,k));
    theta_old=theta(:,k);
    theta(:,k+1)=theta(:,k)-hes(theta(:,k))\grad(theta(:,k));
    k=k+1;
end
thetahat=theta(:,k);
J(thetahat)
save('thetahat')
```

Code for calculating gradient

```
function grad=grad(theta)
global x_train y_train lambda

xTilde_train=[ones(1,size(x_train,2)); x_train];
grad=2*lambda*theta;
for i=1:numel(y_train)

grad=grad+y_train(i)*xTilde_train(:,i)*exp(y_train(i)*theta'*xTilde_train(:,
i))/(1+exp(y_train(i)*theta'*xTilde_train(:,i)));
end
end
```

Code for calculating Hessian

```
function hes=hes(theta)
global x_train y_train lambda

xTilde_train=[ones(1,size(x_train,2)); x_train];
hes=2*lambda*eye(size(theta,1));
for i=1:numel(y_train)

hes=hes+y_train(i)^2*xTilde_train(:,i)*xTilde_train(:,i)'*exp(y_train(i)*the
ta'*xTilde_train(:,i))/(1+exp(y_train(i)*theta'*xTilde_train(:,i)))^2;
end
end
```

Code for calculating LR objective function

```
function J=J(theta)
global x_train y_train lambda

xTilde_train=[ones(1,size(x_train,2)); x_train];
J=lambda*(norm(theta))^2;
for i=1:numel(y_train)
    J=J+log(1+exp(y_train(i)*theta'*xTilde_train(:,i)));
end
end
```

Code for calculating etahat

```
function pr=etahat(x_test)
global thetahat

pr=1/(1+exp(-thetahat'*[1;x_test]));

end
```

Code for classifier

```
function [y_pred,conf]=LRclsf(x_test)

if etahat(x_test)>=.5
    y_pred=-1;
else
    y_pred=1;
end
conf=abs(etahat(x_test)-.5);
end
```

Code for obtaining results

```
close all; clear all; clc
load mnist_49_3000
load thetahat

[d,n]=size(x);

x_test=x(:,2001:3000);
y_test=y(2001:3000);

for i=1:size(x_test,2)
    [y_pred(1,i),conf(i)]=LRclsf(x_test(:,i));
end
error=abs(y_pred-y_test);
ind=find(error)
misclsConf=conf(ind);
errorSum=0;
for i=1:numel(error)
    errorSum=errorSum+error(i)/2;
end

errorSum
misclas_rate=errorSum/numel(y_test)

[misclsConfSrt,IndVec]=sort(misclsConf);
ind1=fliplr(ind(IndVec));

k=1;
plotInd=ind1(1:20);
for i=plotInd
```

```
    subplot(4,5,k)
    imagesc(reshape(x_test(:,i),[sqrt(d),sqrt(d)])');
%     title(num2str(conf(i)))
    title(num2str(i))
    k=k+1;
end
```