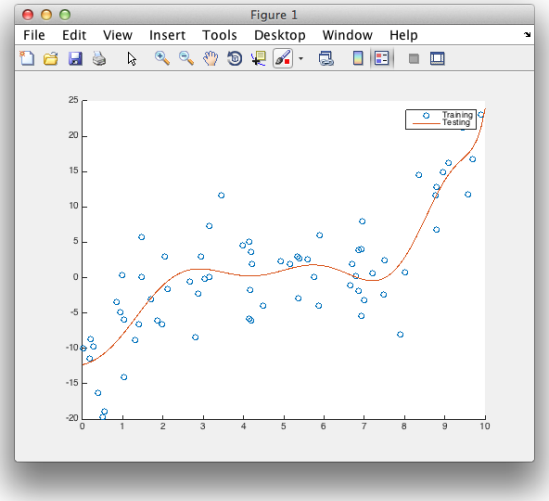
Seyed Mahdi Basiri Azad (Erfan): CS74/174 Homework #2

Machine Learning and Statistical Data Analysis: Winter 2016

Problem 1: Linear Regression

(a) After taking the derivative of $J(\theta)$ and setting it to zero in order to find the minimum, we get:
$argmin J(\theta) = (Y - \theta X^T).X - \alpha W\theta = 0$
$Y^T X - \theta X^T X - \alpha W\theta = 0$
$Y^T X = \theta(X^T X + \alpha W)$
$\theta = Y^T X(X^T X + \alpha W)^T$

(b) Fitting a polynomial



(c) Cross Validation
The code snippet below shows the "else" part of the cross validation function when Nfold is not equal to 2:

```
else
   n = size(Xtrain,1); %size of the training set
   K = ceil(n/Nfold);%max length of each fold
   startIndex = 1;%start of a sub-dataset
   endIndex = K;%end of a sub-dataset
   for i = 1:Nfold
      set{i} = startIndex:endIndex;
      if(i+1 ~= Nfold) %if not the second to last set
        startIndex = endIndex+1;
        endIndex = endIndex+K;
      else
          startIndex = endIndex+1;
          endIndex = n;
```

```
      end
   end

   errIndex = 1;
   for j = 1:Nfold %how many times to train
       theta=train_func(Xtrain(set{j},:), Ytrain(set{j},:)); % train on set(j)
       for k = 1:Nfold
          if(k ~= j)%don't test on the training subset, that's cheating!
              err_xVar_Vec(errIndex)=evaluate_func(Xtrain(set{k},:),
                        Ytrain(set{k},:), theta);  % test on set(k)
              errIndex = errIndex+1;
          end
       end
   end
   err_xVar = mean(err_xVar_Vec); % average the error rates
```

(d) Finding the best alpha:
Given the array of alphas as:

```
alphaVec = [0,0.1,1,10,100];
```

The best alpha value after running the crossValidation with Nfold of 5 was:

```
alphabest =
   100

Testing Error is 2.347429e+01
```

(e) Finding the best degree of polynomial:
Given an array of polynomial degrees as:

```
poly_order_Vec = [1,2,3,4,5,10,20];
```

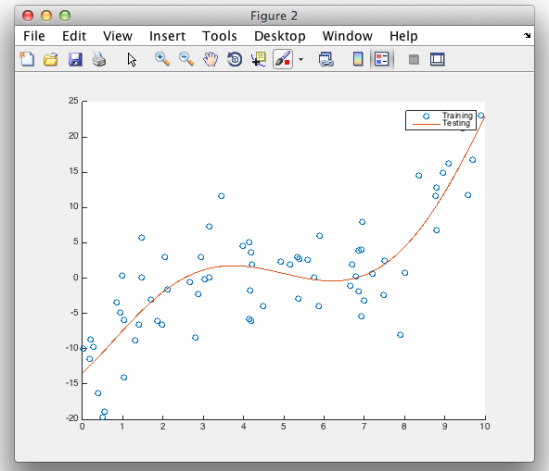
The best polynomial degree along with its error with 5 fold crossValidation was:

```
poly_order_best =
     3

Testing Error is 2.585687e+01
```

(f) Best combinations of alpha and polynomial degrees:
After changing the values for several times the best value of alpha and degree appeared to be:

```
alphabest =
   0

poly_order_best =
     5

Testing Error is 2.459435e+01
```

resulting in the following graph:



However given that especially the value of alpha is very different from the optimal value calculated through he cross validation method, I would suspect that the mentioned value of alpha and degree of the polynomial are over-fitting the testing dataset. In this case it would be valuable to break the data into train, validate, test portions and find the optimal alpha and degree of the polynomial using the "validation set" instead of the "testing set" and use the testing set to report the error.

Problem 2: Logistic Regression

(a) log-sum-exp function

I changed the code for the log-sum-exp function to the following in order to eliminate the over/under flow issue with the naive implementation:

```
tMax = max(x, [], 2);%n X 1 vector containing the maximum of each row of x
y = tMax + log(sum(exp(x-repmat(tMax,1,size(x,2)))));
dydx = exp(x-repmat(tMax,1,size(x,2))) ./
        repmat(sum(exp(x-repmat(tMax,1,size(x,2)))),1,size(x,2));
```

After testing the new implementation i got the following result:

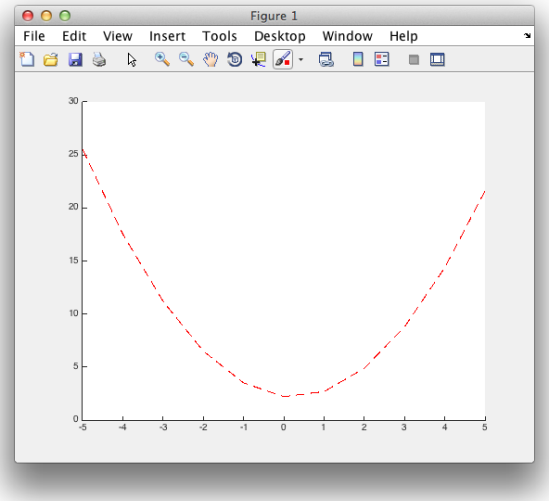
```
%running:
[y,dy]=logsumexp([1000,1001.2, 999.9])
[y,dy]=logsumexp([-1000,-1001])

%result:
y =
   1.0017e+03
dy =
    0.1914    0.6354    0.1732

y =
 -999.6867
dy =
    0.7311    0.2689
```

(b) Negative log-likelihood function

I spent a good amount of time on this function. It was difficult to convert the math to MATLAB code in an optimized way so I implemented the loop version first to calculate the negative log-likelihood function. I then realized that it would be much easier to have the hypothesis as a variable in order to use it for calculating the derivative. While trying to learn how to use the vector form I came across the links below and used it as a guideline:
http://www.aiqus.com/forum/questions/39087/softmax-regression-has-anyone-successfully-impl
http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/
below is the vector (optimized version) the loop version is commented out but is in the file:

```
t=Xtrain*theta';
t = bsxfun(@minus, t, max(t, [], 1));% to deal with the over/under flow (basically the same as logs
hyp = exp(t);%hypothesis
hyp = bsxfun(@rdivide, hyp, sum(hyp));
neglogp = -(sum(Ytrain'*log(hyp)))/n + sum(sum(theta.^2));
%========derivative=========
negDlogp = (bsxfun(@minus, hyp,Ytrain)'*Xtrain)/n + alpha*theta;
```

This resulted in the following graph:



Problem 3: SVM

(a) Linearly Separable data

The given data set is linearly separable because there exist a straight line that pass in between $x^{(2)}$ and $x^{(3)}$ such that it separates the dataset into two parts of $[x^{(1)}, x^{(2)}]$ with label y=+1 and $[x^{(3)}]$ with label y=-1.

(b) Primal Form

Given the data set any wx+b=0 that passes in between 0 and 1 will satisfy the problem. We can start by writing the constraints:
1) $w(-1) + b < 1 \implies b < w + 1$
2) $w(0) + b < 1 \implies b < 1$
3) $w(1) + b > -1 \implies b > -w - 1$

so we are looking for $min_{w,b}\frac{1}{2}w^2$ such that it meets the said constraints.

(c) Dual Form
Let's rewrite the constraints as:
1) $w(-1)+b<1 ==> -1-w+b<0$
2) $w(0)+b<1 ==> -1+b<0$
3) $w(1)+b>-1 ==> 1+w+b<0$

The Lagrangian form:
$L(w,b,\theta)=\frac{1}{2}w^2+\alpha_1(-1-w+b)+\alpha_2(-1+b)+\alpha_3(1+w+b)$
$L(w,b,\theta)=\frac{1}{2}w^2+(-\alpha_1+\alpha_3)w+(\alpha_1+\alpha_2+\alpha_3)b+(-\alpha_1-\alpha_2-\alpha_3)$

we are looking to do the following optimization:
$max_{\alpha,\alpha_j>0}\Sigma_j\alpha_j-\frac{1}{2}\Sigma_i\Sigma_j\alpha_i\alpha_j y^i y^j(x^i.x^j)$ such that $\Sigma_j\alpha_j y^j=0$

(d) Support Vectors
points of x=0 and x=1 in the given data set are the support vectors because they are on the margin lines. Also they are the only points which their α_j in the dual form in non-zero.