Project 2- Regression on page relevancy

**Learning to Rank using Linear Regression**

**Name: Sneha Balakrishnan Thazhathethil**

**UBID: 50169238**

**Objective**

To implement linear regression to solve regression problem and evaluate its performance. The objective is to learn how to map an input vector x into a scalar target t. In this project, you are required to finish four tasks:

1. Train a linear regression model on real dataset using batch method.
2. Train a linear regression model on real dataset using stochastic gradient descent.
3. Train a linear regression model on synthetic dataset using batch method and evaluate its performance.
4. Train a linear regression model on synthetic dataset using stochastic gradient descent and evaluate its performance.

**Regression model:**

[2]Regression analysis is a statistical tool for the investigation of relationships between variables. Usually, the investigator seeks to ascertain the causal effect of one variable upon another—the effect of a price increase upon demand, for example, or the effect of changes in the money supply upon the inflation rate.

[2]www.law.uchicago.edu/files/files/20.Sykes\_.**Regression**.pdf

**Linear regression model:**

A linear regression model is formed by using the basic concept of linear fitting of a function which matches as closely as possible to the pattern of datasets.



The image above represents scatter plot of x versus y.(figure courtesy mathworks)

The linear regression model is defined by :

y(x,w) = wT \*phi(x)

w= (w0, w1, w2, …..,wm-1)-> weight vector

phi= (phi0,phi1, ….phim-1)-> vector of M basis function

We use the Gaussian radial basis function to convert the input vector into a scalar value.

φj(x)=exp (−1/2(x−μj) TΣ-1j (x−μj))

where μj is the center of basis function and Σj (sigma) decides how broadly the basis function spreads.

**Maximum likelihood solution:**

Maximum likelihood estimation is used to the parameters of a statistical model when dataset is provided which makes the likelihood estimation maximum.

We can maximize the likelihood solution by reducing the sum of squares error to a minimum value.

 ED(w)= 1/2 Summation(j=0 to M-1) {tn −w⊤φ(xn)}2



**Fig: X(1)-> column 1 of feature vector Vs relevancy.**

Example of one feature vector plotted against the relevance values.



Plot using cftool of feature vector column 1 and 2 with relevance. Applying polynomial fit to the two vectors, we obtain the equation shown in the figure.

The major issue with curve fitting model is the problem of over fitting. To avoid that we add a regularization term in the error function.

E(w) = ED(w) + λEW (w)

Where,

EW(w)= ½ summation( j=0 to M-1) |wj2|

Here lambda governs the relative importance of the regularization term.

In this project, we make use of closed form solution and stochastic gradient descent to solve the issue of linear regression and evaluate optimal wML by setting the likelihood’s gradient w.r.t. w equal to zero.

**Closed form likelihood solution**

The closed form likelihood solution to linear regression is as follows: (Assuming Gaussian noise)

wML = (Φ⊤Φ)−1Φ⊤t

where Φ is the design matrix, t(t1,t2,t3..tN) is the outputs in training data.

The solution to the likelihood function is given by:

wML = (λI + Φ⊤Φ)−1Φ⊤t

**Stochastic gradient technique:**

[1]As our dataset is relatively very large, the computation complexity of the learning algorithm is extremely important. This contribution advocates stochastic gradient algorithms for large scale machine learning problems.

[1][Large-Scale Machine Learning with Stochastic Gradient Descent – Leon Bottou]

The stochastic gradient descent algorithm first takes a random initial value w(0). Then it

updates the value of w using

w(τ+1) = w(τ) + ∆w(τ)

∆w(τ) = −η(τ)∇E 🡪weight updates

∇E = ∇ED + λ∇EW

∇ED =−(tn−w(τ)⊤φ(xn))φ(xn)

∇EW = w(τ)

η is the learning rate.

To evaluate the solution, we find the RMS error :

Erms= squareroot(2\*E(w\*)/Nv)

**Dataset interpretation:**

**For real world data set: (Microsoft Letor 4.0 MQ2008)**

The dataset includes 69623 query document pairs with 46 features.

The dataset is broken down as follows:

Column 1: relevance (y)

Column 3 to 48: Features (46 dimensional input vector)

The other columns are not used for this project.

**For synthetic data: (Provided)**

Dataset contains a 2000x10 data point vector.

Data set t – relevance(y)

Data set x- features(input vector)

**Tasks and approach:**

1. Extract feature values and labels from data

Use fopen with file identifies as fid to open the text file with the dataset.

Read the dataset and convert it into a 2x2 vector.

Use textscan to read the file contents.

**%% code snippet**

%fid = fopen('Querylevelnorm.txt','rt');

fid = fopen('Querylevelnorm.txt','rt');

%Read data row-wise

wholerowread = textscan(fid,'%s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s',69623);

% Read relevance

Y = wholerowread{1,1};

%Read features

parameters = wholerowread(:,3:48);

%Convert features to vector

X = [parameters{:}];

\_new = zeros(69623,46);

[rowSize,columnSize] = size(X);

for k=1:rowSize

for j=1:columnSize

%For X\_new enter matrix values from position 3 to 48(features)

strings = strsplit(X{k,j},':');

X\_new(k,j) = str2double(strings(2));

end

end

Y\_new = str2double(Y);

1. Data partition

As mentioned in the project requirement document, I have divided the dataset into training, validation and test data.

Training – 80% of dataset

Validation - 10% of dataset

Test – 10% of dataset

noTrainDocs = floor(0.8 \* length(X\_new));

noValidationDocs = floor(0.9 \* length(X\_new));

X\_rand = randperm(length(X\_new));

X\_rand = (X\_rand).';

trainingIndexes = X\_rand(1:noTrainDocs);

X\_training = X\_new(trainingIndexes,:);

Y\_training = Y\_new(trainingIndexes,:);

Similarly for validation and test set.

1. Train model parameter

Given a set of parameters calculate weights for the model.

The parameters on which weights are based are M, mu, sigma, lambda and eta.

% populate mu2 matrix

for i= 1 : M2

mu2(:,i) = X\_training\_syn(Xforclosed\_syn(i),:);

end

%% sigma

for i=1:10

if sigma\_syn(1,i) < 0.0001

sigma\_syn(1,i) = 0.01;

end

end

%%phi

for j= 2 : M2

for i = 1 : noTrainSyn

a1= inv(Sigma2(:,:,j));

b1= (X\_training\_syn(i,:).'-mu2(:,j)).';

c1= (X\_training\_syn(i,:).'-mu2(:,j));

d1= -0.5 \* b1 \* a1 \* c1;

phi\_syn(i,j) = exp(d1);

end

end

w2 = inv( lambda2\*eye(M2,M2)+ phi\_syn.'\*phi\_syn)\*phi\_syn.'\*Y\_training\_syn;

Err2= 0.5 \* ((Y\_training\_syn-(phi\_syn\*w2)).')\*(Y\_training\_syn-(phi\_syn\*w2));

trainPer2 = sqrt((2\*Err2)/noTrainSyn);

1. Tune hyper parameters

The regression performance for the dataset is validated on validation data. Find out optimized model using varied values of parameters.

**Hyper tuning:**

For hyper tuning, we first choose a range of M and λ

M-> Number of Basis function

λ -> (lambda) regularization term

logic:

Loop M (range)

Calculate mu, sigma, phi, Error

Loop lambda(range)

Calculate Training and validation set.

Check if the error is minimum. If yes, store the minimum values in a min\_variables for all the parameters. Use these set of parameters to model the system.

Test the testing set vector on this optimized solution of the regression model.

End loop

End loop

1. Test machine learning scheme on testing set

Test the model using a different test set and check if its optimal.