LINEAR REGRESSION WITH BASIS FUNCTIONS

By

Aravind Kumar Ramesh
CSE574-Introduction to Machine Learning
Dr. S N Srihari
STATE UNIVERSITY OF NEW YORK-
UNIVERSITY AT BUFFALO

OVERVIEW

"The project is to implement and evaluate several supervised machine learning approaches to the task of linear regression. The objective is to learn how to map an input vector x into a target value t using the model

$$\mathbf{y(x, w)= w^T \Phi(x)}$$

Where w = (w_0, w_1, w_{M-1}) is a weight vector to be learnt from training samples and **Φ** is a vector of M basis functions. Each basis function converts input vector into a scalar value."[1]

We can apply any basis functions as desired and calculate the respective feature space and isotropic spatial scale to obtain the scalar value.

DATASET

For this particular task of regression we will be using the Learning to Rank (LeTOR 4.0 MQ2007-Supervised ranking) dataset provided by Microsoft (http://research.microsoft.com/en-us/um/beijing/projects/letor/letor4dataset.aspx)

The dataset consists of a total of 69623 query-documents pairs and 47 dimensions of feature vectors. Hence the dataset can be imagined as a matrix of dimension (69623 x 47).

The first column contains a relevance label with values 0, 1 or 2. The relevance label is a quantitative measure of how relevant a corresponding document-query pair is. The 46 dimensional feature vectors are real numbers that are normalized to be in the range of [0, 1]. The dataset also contains fields for the query id, document id and other comments about the pair of the document.

A sample field of the input dataset is as shown.

```
0 qid:10 1:0.000000 2:0.000000 3:0.000000 4:0.000000 5:0.000000 6:0.000000 7:0.000000 8:0.000000 9:0.000000 10:0.000000
11:0.000000 12:0.000000 13:0.000000 14:0.000000 15:0.000000 16:0.001348 17:0.000000 18:0.222222 19:0.000000 20:0.001282
21:0.000000 22:0.000000 23:0.000000 24:0.000000 25:0.000000 26:0.000000 27:0.000000 28:0.000000 29:0.000000 30:0.000000
31:0.000000 32:0.000000 33:0.000000 34:0.000000 35:0.000000 36:0.000000 37:0.000000 38:0.000000 39:0.000000 40:0.000000
41:0.000000 42:0.000000 43:0.017241 44:0.000000 45:0.000000 46:0.000000 #docid = GX000-00-0000000 inc = 1 prob = 0.0246906
1 qid:10 1:0.031310 2:0.666667 3:0.500000 4:0.166667 5:0.033206 6:0.000000 7:0.000000 8:0.000000 9:0.000000 10:0.000000
11:0.023327 12:0.641157 13:0.498951 14:0.323153 15:0.026674 16:0.029246 17:0.500000 18:0.222222 19:0.111111 20:0.029398
21:0.689128 22:0.636228 23:0.869764 24:0.716400 25:0.725186 26:0.554961 27:0.695985 28:0.504060 29:0.602946 30:0.679534
31:0.730286 32:0.687414 33:0.529688 34:0.436996 35:0.643739 36:0.372337 37:0.646890 38:0.686107 39:0.823908 40:0.750092
41:0.385426 42:0.923077 43:0.086207 44:0.333333 45:0.448276 46:0.000000 #docid = GX000-24-12369390 inc = 0.600318836372593
prob = 0.416367
```

The dataset.txt file is first parsed and the irrelavant comments and labels removed. After which the dataset is loaded into the matlab as a .mat file. To start with computing the closed form solution using basis function we first need to divide the input dataset into training, validation and testing data. We divide the dataset as 80-10-10 where the first 80% of the dataset is the

Training data, and the next 10 percents being the validation and testing data respectively. This is a rough heuristics that we consider before carrying out our computation.

CLOSED FORM SOLUTION USING BASIS FUNCTION

The closed form maximum likelihood solution to linear regression is carried out by computing the basis function (Gaussian, Sigmoidal etc) and finding out the respective hyper parameter mean and variance. We use the linear basis function model which is given by the formula

$$y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + \sum_{j=1}^{M-1} \sum_{i=1}^{D} w_{(i,j)} \phi_j(x_i)$$

Where M is the complexity and D is the dimensions (D=46 in this case). We have considered the Gaussian basis function and hence as a result the Φ_j can be determined by the following formula.

$$\phi_j(\boldsymbol{x}) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$$

We consider $\Phi_0(\mathbf{x}) = 1$ to introduce a bias into the system. This bias parameter compensates for the difference in the average values of target vector in the training data with that of averages of the basis function values.

Now to compute the values of the hyper parameter we use a simple yet effective technique. We calculate the hyper parameter mean in a randomized fashion. We make use of the randi function in matlab to pick out random columns from the training dataset and we iteratively compute the mean of all the values. Now we take the means obtained in the 46 iterations and compute the mean of the means to give a more accurate hyper parameter mean value. We do the same for variance. We store the mean and the variance in a Mx2 matrix where M is the Complexity. The code snippet shown below demonstrates the process.

```
for i=1:complexity
      for  j=1:46   %randomly compute the columns and store
      the mean and variance of the column
      mu(j)=mean(train_data(:,randi(45,1)));
      sigma(j)=var(train_data(:,randi(45,1)));
```

```
    end
   parameter(i,1)=mean(mu); % compute the mean(mu(j))
   parameter(i,2)=mean(sigma); % Variance
end
```

With this value of hyper parameter mean and variance we now begin to compute our design matrix **Φ**. The design matrix is of dimension M x N applied over D dimensions (D=46 in this case) with M being the complexity of the model. The design matrix can be mathematically represented as

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}.$$

Using the design matrix **Φ** we can now calculate the closed-form maximum likelihood solution to the linear regression using Gaussian basis function in the following form

$$\mathbf{w}_{\mathrm{ML}} = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}$$

Where t represents the target vector.

In order to smoothen out the curve and to prevent it from over fitting we add a penalty term to the error function. This is done in order to discourage the weights from reaching a large values and hence restrict the increase in M. Thus this regularization term can be added to the weight function w_{ML} Hence the w* represents the regularized weights given by

$$\mathbf{w}^* = (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda\mathbf{I})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}$$

where I is an identity matrix.

With the Weights obtained from the above step we calculate the sum of squared errors by subtracting the computed values with the values given in the dataset. Thus E_{RMS} is given by

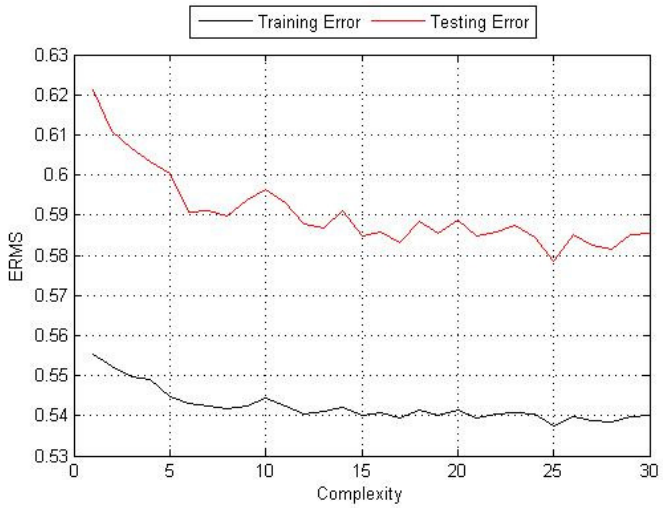
$$E_{RMS}(w) = [(\Phi^{T}w - t)(\Phi w - t)/N]^{1/2}$$

We repeat the same process for the validation and testing dataset by tuning the value of **λ** and M.

RESULT

To obtain the lowest squared error we take five values for λ, which are λ=0.001, 0.01, 0.1, 1 and 10. We also compute the E_{RMS} by considering complexity M from 1 to 30. The lowest Values is obtained for M=26, and λ=0.001.

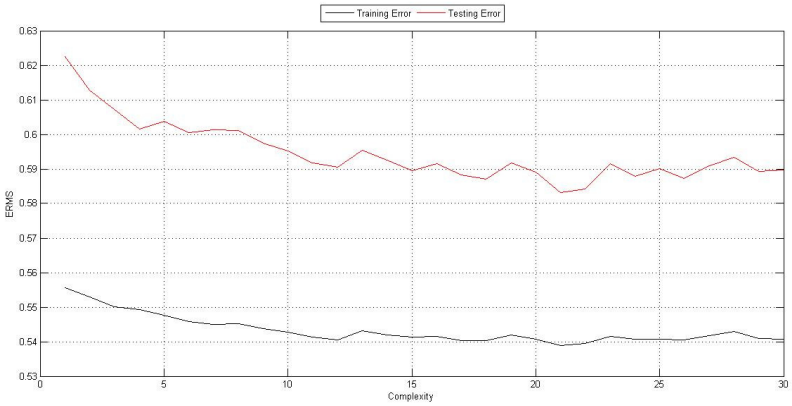
a) λ =0.001 , $E_{RMS(Training)}$=0.537290 $E_{RMS(Testing)}$=0.578535, M=26



	Training Error	Validation Error	Testing Error
M= 2	0.555450	0.543648	0.621390
M= 3	0.552303	0.534428	0.610866
M= 4	0.549854	0.529029	0.606709
M= 5	0.548918	0.526590	0.603351
M= 6	0.544679	0.521648	0.600355
M= 7	0.542994	0.514324	0.590897
M= 8	0.542514	0.514842	0.591082
M= 9	0.541734	0.513330	0.589721
M= 10	0.542476	0.516899	0.593711
M= 11	0.544452	0.518536	0.596377
M= 12	0.542410	0.516200	0.592943
M= 13	0.540520	0.512037	0.587670
M= 14	0.541093	0.510759	0.586933
M= 15	0.542242	0.514718	0.591129
M= 16	0.540068	0.509196	0.584919
M= 17	0.540781	0.509707	0.585637
M= 18	0.539596	0.509393	0.583177
M= 19	0.541353	0.513104	0.588604
M= 20	0.539951	0.510300	0.585469
M= 21	0.541303	0.513287	0.588828

M= 22	0.539568	0.509264	0.584792
M= 23	0.540350	0.510506	0.585811
M= 24	0.540759	0.512177	0.587405
M= 25	0.540354	0.509174	0.584568
M= 26	0.537290	0.503687	0.578535
M= 27	0.539931	0.510304	0.585260
M= 28	0.538872	0.508171	0.582575
M= 29	0.538371	0.506597	0.581458
M= 30	0.539917	0.509524	0.584979

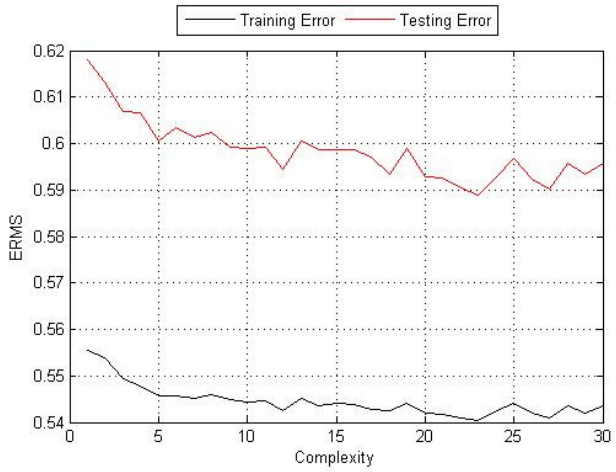
b) $\lambda = 0.01$, $E_{RMS(Training)} = 0.538844$ $E_{RMS(Testing)} = 0.583165$, M=22



	Training Error	Validation Error	Testing Error
M= 2	0.555510	0.543728	0.622545
M= 3	0.552871	0.535456	0.612664
M= 4	0.550062	0.529161	0.607214
M= 5	0.549189	0.525327	0.601488
M= 6	0.547594	0.524605	0.603732
M= 7	0.545810	0.521925	0.600472
M= 8	0.545043	0.522226	0.601329
M= 9	0.545204	0.522102	0.601114
M= 10	0.543700	0.519230	0.597540
M= 11	0.542712	0.518413	0.595247
M= 12	0.541246	0.515406	0.591727
M= 13	0.540476	0.514481	0.590609
M= 14	0.543144	0.518581	0.595481
M= 15	0.541992	0.515964	0.592492
M= 16	0.541420	0.513604	0.589475
M= 17	0.541508	0.515782	0.591626
M= 18	0.540403	0.512883	0.588288
M= 19	0.540247	0.512159	0.587085
M= 20	0.541985	0.515094	0.591725
M= 21	0.540719	0.513760	0.588996

M= 22	0.538844	0.508308	0.583165
M= 23	0.539527	0.509958	0.584224
M= 24	0.541573	0.515260	0.591478
M= 25	0.540651	0.512480	0.587902
M= 26	0.540769	0.514069	0.590152
M= 27	0.540513	0.512376	0.587231
M= 28	0.541791	0.514652	0.590939
M= 29	0.542967	0.515924	0.593278
M= 30	0.540909	0.513697	0.589339
M= 31	0.540704	0.514192	0.589665

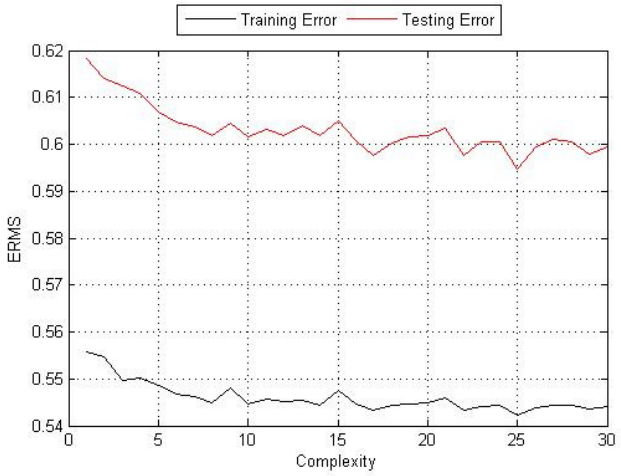
c) $\lambda = 0.1$, $E_{RMS(Training)} = 0.540406$ $E_{RMS(Testing)} = 0.588986$, M=24



	Training Error	Validation Error	Testing Error
M= 2	0.555443	0.543018	0.617894
M= 3	0.553969	0.535443	0.613223
M= 4	0.549411	0.529073	0.606980
M= 5	0.547693	0.528574	0.606672
M= 6	0.545576	0.522191	0.600586
M= 7	0.545819	0.524433	0.603552
M= 8	0.545162	0.523040	0.601437
M= 9	0.545937	0.524175	0.602322
M= 10	0.544961	0.520315	0.599221
M= 11	0.544393	0.520916	0.599012
M= 12	0.544542	0.520240	0.599312
M= 13	0.542649	0.518041	0.594443
M= 14	0.545142	0.523020	0.600650
M= 15	0.543700	0.519896	0.598705
M= 16	0.544179	0.520367	0.598644
M= 17	0.543924	0.520326	0.598695
M= 18	0.542752	0.518656	0.597156

M= 19	0.542574	0.516868	0.593434
M= 20	0.544035	0.520663	0.598911
M= 21	0.541897	0.516494	0.592876
M= 22	0.541622	0.516475	0.592654
M= 23	0.540968	0.515169	0.590543
M= 24	0.540406	0.514017	0.588986
M= 25	0.542515	0.516908	0.592883
M= 26	0.544066	0.519064	0.596872
M= 27	0.542116	0.516010	0.592426
M= 28	0.540939	0.514234	0.590107
M= 29	0.543468	0.518253	0.595730
M= 30	0.541984	0.516715	0.593470

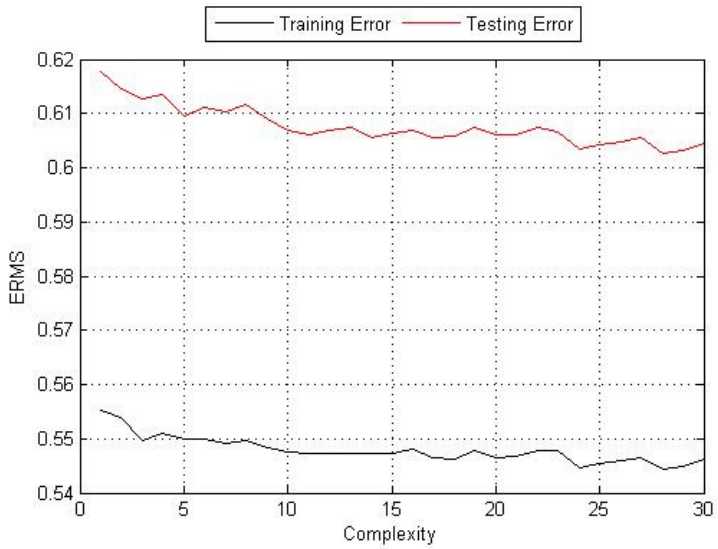
d) $\lambda = 1$, $E_{RMS(Training)} = 0.542357$ $E_{RMS(Testing)} = 0.594776$, M=26



	Training Error	Validation Error	Testing Error
M= 2	0.555750	0.543438	0.618375
M= 3	0.554822	0.536433	0.614156
M= 4	0.549727	0.534393	0.612382
M= 5	0.550237	0.531940	0.610777
M= 6	0.548704	0.528711	0.606932
M= 7	0.546734	0.526490	0.604788
M= 8	0.546133	0.525434	0.603813
M= 9	0.544844	0.522585	0.601966
M= 10	0.547984	0.526565	0.604536
M= 11	0.544605	0.522367	0.601694
M= 12	0.545575	0.524395	0.603299
M= 13	0.545096	0.523583	0.601832
M= 14	0.545503	0.524841	0.604077
M= 15	0.544431	0.522385	0.601842
M= 16	0.547547	0.526821	0.604968

M= 17	0.544540	0.521596	0.600924
M= 18	0.543403	0.519376	0.597624
M= 19	0.544242	0.521607	0.600372
M= 20	0.544525	0.522402	0.601653
M= 21	0.544810	0.522709	0.601811
M= 22	0.546045	0.525046	0.603497
M= 23	0.543188	0.519825	0.597650
M= 24	0.544198	0.521732	0.600598
M= 25	0.544312	0.521390	0.600580
M= 26	0.542357	0.518447	0.594776
M= 27	0.543733	0.521411	0.599598
M= 28	0.544384	0.521803	0.601112
M= 29	0.544259	0.521573	0.600428
M= 30	0.543647	0.520065	0.597890

e) $\lambda = 10$, $E_{RMS(Training)} = 0.544401$ $E_{RMS(Testing)} = 0.602746$, M=29



	Training Error	Validation Error	Testing Error
M= 2	0.555251	0.542055	0.617738
M= 3	0.553898	0.536855	0.614648
M= 4	0.549625	0.534298	0.612700
M= 5	0.551092	0.536293	0.613627
M= 6	0.549988	0.531455	0.609502
M= 7	0.549832	0.532310	0.611059
M= 8	0.549210	0.531588	0.610382
M= 9	0.549722	0.532990	0.611607
M= 10	0.548225	0.530334	0.609026
M= 11	0.547487	0.527793	0.606785
M= 12	0.547270	0.527511	0.605980
M= 13	0.547410	0.527915	0.606952

M= 14	0.547266	0.528964	0.607505
M= 15	0.547392	0.527968	0.605528
M= 16	0.547241	0.527763	0.606301
M= 17	0.548191	0.528744	0.607014
M= 18	0.546430	0.526815	0.605500
M= 19	0.546291	0.526574	0.605716
M= 20	0.547827	0.528960	0.607380
M= 21	0.546426	0.527318	0.606119
M= 22	0.546773	0.527577	0.606073
M= 23	0.547761	0.528860	0.607299
M= 24	0.547758	0.528090	0.606504
M= 25	0.544701	0.523836	0.603358
M= 26	0.545416	0.525213	0.604164
M= 27	0.545955	0.525607	0.604782
M= 28	0.546597	0.527100	0.605526
M= 29	0.544401	0.523126	0.602746
M= 30	0.545014	0.524165	0.603145

GRADIENT DESCENT APPROACH

In this approach we use the stochastic gradient method to compute the weight matrix which in turn helps us find the lowest mean error. We have kept the learning rate as a constant value but ensured a very small step size thereby ensuring that the value does not overshoot the local minima. As an optimization measure, a ball-parked value of Regularization constant has been added along with the learning rate to ensure that there is no over fitting. The Computation is carried out for values of M ranging from M=1 to M=25 and the optimum values of M is obtained by choosing the lowest corresponding error for the Testing data.

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \eta(t_n - \mathbf{w}^{(\tau)\mathrm{T}}\phi_n)\phi_n$$

With the weights obtained via the gradient descent approach we use the design matrix obtained in the previous technique to the following formula

$$E_{RMS}(w) = [(\Phi^{T}w - t)(\Phi w - t)/N]^{1/2}$$

The optimum value obtained is λ =0.001, $E_{RMS(Training)}$= 0.6055
$E_{RMS(Testing)}$= 0.684606, M=11