CSE 574: Introduction to Machine Learning (Fall 2018)
Instructor: Sargur N. Srihari

Project 1.2: Learning to Rank using Linear Regression

October 10, 2018

Report By:
Siddheswar Chandrasekhar

Objective

The goal of this project is to use machine learning to solve a problem that arises in Information Retrieval, one known as the Learning to Rank (LeToR) problem. We formulate this as a problem of linear regression where we map an input vector x to a real-valued scalar target y(x, w).

There are two tasks:
1. Train a linear regression model on LeToR dataset using a closed-form solution.
2. Train a linear regression model on the LeToR dataset using stochastic gradient descent (SGD).

The LeToR training data consists of pairs of input values x and target values t. The input values are real-valued vectors (features derived from a query-document pair). The target values are scalars (relevance labels) that take one of three values 0, 1, 2: the larger the relevance label, the better is the match between query and document. Although the training target values are discrete we use linear regression to obtain real values which is more useful for ranking (avoids collision into only three possible values).

Some of the concepts used and questions addressed by this project are:

What is Linear Regression?

In statistics, linear regression is a linear approach to modelling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The case of one explanatory variable is called simple linear regression. For more than one explanatory variable, the process is called multiple linear regression. This term is distinct from multivariate linear regression, where multiple correlated dependent variables are predicted, rather than a single scalar variable.

In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models. Most commonly, the conditional mean of the response given the values of the explanatory variables (or predictors) is assumed to be an affine function of those values; less commonly, the conditional median or some other quantile is used. Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of the response given the values of the predictors, rather than on the joint probability distribution of all of these variables, which is the domain of multivariate analysis.[1]

In regression problem, the goal of the algorithm is to predict real-valued output.
Linear regression is a very simple approach for supervised learning. It is used to predict a quantitative response Y from the predictor variable X.
Linear Regression is made with an assumption that there's a linear relationship between X and Y.

Linear Regression vs Classification

In classification problems we are trying to predict a discrete number of values.
The labels(y) generally comes in categorical form and represents a finite number of classes.

In regression problems we try to predict continuous valued output.

Our linear regression function $y(x, w)$ has the form:

$$y(x, w) = w^T \phi(x)$$

where,

$w = (w_0, w_1, w_2, \ldots\ldots, w_{M-1})$ is a weight vector to be learnt from training samples
$\phi = (\phi_0, \phi_1, \phi_2, \ldots\ldots, \phi_{M-1})^T$ is a vector of M basis functions.

Each basis function $\phi_j(x)$ converts the input vector x into a scalar value.

In this project, we use the Gaussian Radial Basis Fumction which has the form:

$$\phi_j(x) = exp\left[-\tfrac{1}{2} (x - \mu_j)^T \Sigma_j^{-1} (x - \mu_j) \right]$$

where,

μ_j is the center of the basis function, and
Σ_j decides how broadly the basis function spreads

What is Basis Function?

The basic idea of basis function is that in addition to the original inputs, we add inputs that are calculated as deterministic functions of the existing inputs and treat them as additional inputs.

The challenge is to find problem specific basis functions which are able to effectively model the true mapping.
If we include too few basis functions or unsuitable basis functions, we might not be able to model the true dependency.
If we include too many basis functions, we need many datapoints to fit all the unknown parameters.

The number of "sensible" basis functions increases exponentially with the number of inputs. [4]

Gaussian Radial Basis Function

RBF doesn't necessarily have a probabilistic interpretation. In this method, the normalization term is unimportant as the bias function is multiplied by the weight.

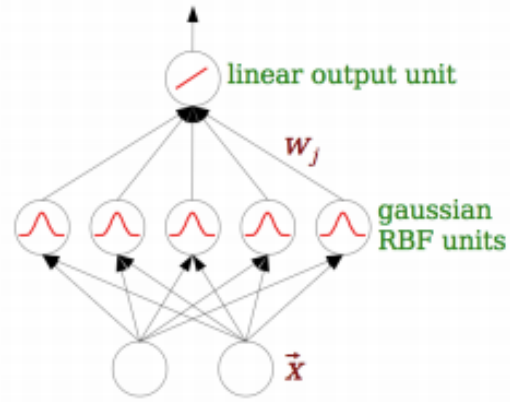

Fig RBF Network

Clustering

At times it is better to first cluster data in input space and then use the cluster centers as positions for the Gaussian basis functions.

The widths of the Gaussian basis functions might be derived from the variances of the data in the cluster.
An alternative is to use one RBF per data point. The centers of the RBFs are simply the data points themselves and the widths are determined via validation.

What happens in the training process? [2]

Model Coefficients / Parameters:

When training a linear regression model, we try to find out coefficients for the linear function that best describe the input variables.

Loss Function:

When building a linear model, we try to minimize the error by making predictions, and we do that by choosing a function to help us measure the error, called the loss function.

How do we estimate the coefficients?

For this task, we use Stochastic Gradient Descent.

What is Closed-Form Solution?

A closed-form solution is any formula that can be evaluated in a finite number of standard operations. Closed form solutions and numerical solutions are similar in that they both can be evaluated with a finite number of standard operations. They differ in that a closed-form solution is exact whereas a numerical solution is only approximate.

What is LeToR?

LETOR is a package of benchmark data sets for research on Learning To Rank, which contains standard features, relevance judgments, data partitioning, evaluation tools, and several baselines. [3]

Evaluation

We evaluate our solution using Root Mean Square (RMS) Error.

RMSE is a quadratic scoring rule that measures the average magnitude of error. It is the square root of the average of squared differences between prediction and actual observation.

$$E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^*)/N_{\mathrm{V}}}$$

Observations

How do we reduce the size of our dataset?

We have a dataset with forty-six features. However, five of these are always zero and hence do not contribute towards our prediction. We therefore discard these five features and consider only forty-one features.

How do we restrict Σ ?

Σ has the form:

$$\Sigma = \begin{pmatrix} \sigma_1^2 & & & \\ & \sigma_2^2 & & \\ & & \ddots & \\ & & & \sigma_D^2 \end{pmatrix}$$

This is because we are only concerned with the covariance of σ_i with itself. We hence disregard all the other values.

Why do we use Moore – Penrose inverse ?

Moore – Penrose inverse is the most widely known type of matrix pseudo-inverse.

Since we have forty-one features and ten basis functions, our design matrix ϕ is not a square matrix. Therefore, we cannot simply find the inverse of the matrix. We hence have to use some other mean to inverse the matrix. We use Moore – Penrose pseudo inverse for achieving this, which has the form:

$$w_{ML} = (\phi^T \phi)^{-1} \phi^T t$$

where,

$t = \{ t_1 , t_2 , \ldots\ldots , t_N \}$ is the vector of outputs in the training data, and

ϕ = design matrix having the form:

$$\mathbf{\Phi} = \begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \phi_2(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{bmatrix}$$

We also observe that unlike Closed-Form Solution, Gradient Descent doesn't update the weights of the model.

Observations based on different values of hyperparameters

Number of Basis Functions (M)	Regularization Term (λ) [For Closed form and GD]	Basis Function	Learning Rate
10	0.03	Radial	0.01

Closed-form Solution
E_{RMS} Training = 0.5494
E_{RMS} Validation = 0.4938
E_{RMS} Testing = 0.5906

Gradient Descent
E_{RMS} Training = 0.5496
E_{RMS} Validation = 0.4906
E_{RMS} Testing = 0.5789

Number of Basis Functions (M)	Regularization Term (λ) [For Closed form and GD]	Basis Function	Learning Rate
10	0.2	Radial	0.1

Closed-form Solution
E_{RMS} Training = 0.5495
E_{RMS} Validation = 0.4939
E_{RMS} Testing = 0.5906

Gradient Descent
E_{RMS} Training = 0.5841
E_{RMS} Validation = 0.5090
E_{RMS} Testing = 0.5986

Number of Basis Functions (M)	Regularization Term (λ) [For Closed form and GD]	Basis Function	Learning Rate
10	3	Radial	0.2

Closed-form Solution
E_{RMS} Training = 0.5503
E_{RMS} Validation = 0.4965
E_{RMS} Testing = 0.58867

Gradient Descent
E_{RMS} Training = 25.409
E_{RMS} Validation = 25.35
E_{RMS} Testing = 25.13

Number of Basis Functions (M)	Regularization Term (λ) [For Closed form and GD]	Basis Function	Learning Rate
10	0.5	Radial	0.2

Closed-form Solution
E_{RMS} Training = 0.5643
E_{RMS} Validation = 0.5079
E_{RMS} Testing = 0.5933

Gradient Descent
E_{RMS} Training = 74.7318
E_{RMS} Validation = 74.6626
E_{RMS} Testing = 74.8181

Number of Basis Functions (M)	Regularization Term (λ) [For Closed form and GD]	Basis Function	Learning Rate
15	0.15	Radial	0.15

Closed-form Solution
E_{RMS} Training = 0.5643
E_{RMS} Validation = 0.5079
E_{RMS} Testing = 0.5933

Gradient Descent
E_{RMS} Training = 86.4056
E_{RMS} Validation = 86.336
E_{RMS} Testing = 86.492

Number of Basis Functions (M)	Regularization Term (λ) [For Closed form and GD]	Basis Function	Learning Rate
10	0.15	Radial	0.15

Closed-form Solution
E_{RMS} Training = 0.5495
E_{RMS} Validation = 0.4940
E_{RMS} Testing = 0.5905

Gradient Descent
E_{RMS} Training = 0.5528
E_{RMS} Validation = 0.4896
E_{RMS} Testing = 0.5871

From the above observations, we can infer that changing the hyperparameters doesn't make any significant difference in the error rate.

We however notice that increasing the number of basis functions, or highly increasing the regularization term increases the root mean squared error of our gradient descent model.
Apart from that, increasing the learning rate, regularization term, changing the number of basis functions, doesn't have much of an impact on the final E_{RMS} of our model.

We hence infer that the data in the dataset is insufficient to have a model with these parameters and get a low error rate.

References

1. Linear Regression
   https://en.wikipedia.org/wiki/Linear_regression

2. David Fumo
   "Learning Regression – Intro to Machine Learning #6"
   https://medium.com/simple-ai/linear-regression-intro-to-machine-learning-6-6e320dbdaf06

3. Tao Qin, Tie-Yan Liu
   "Introducint LETOR 4.0 Datasets"

4. Volker Tresp
   "Basis Functions"
   http://www.dbs.ifi.lmu.de/Lehre/MaschLernen/SS2016/Skript/BasisFunctions2016.pdf