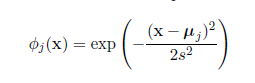
# Machine Learning Linear Regression Project

# AIM

This 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:

y(x,w) = wT *Φ* (x)

where, w = (w0,w1,...wM-1) is a weight vector to be learnt from training samples and *Φ* =(*Φ* 0,….. *Φ* M-1)T is a vector of M basis functions. Each basis function *Φ* j(x), j = 0,…M – 1 converts input vector x into a scalar value. An example is the Gaussian basis function



where μ j is a vector in feature space and s is an isotropic spatial scale.

The training dataset consists of N exemplar vectors X = (x1,… xN) together with the corresponding target values t = (t1,…tN).

# MODEL

Part A: Closed form solution method

Model:

The parameters that need to me tuned are the complexity M and regularization parameter λ. The hyper parameters are mean μ and variance s.

To begin with by looking at the data we set S=0.7 initially as the data given is in the normalized form and randomly calculating we find the variance to be quite small. The value of μ is chosen by randomly selecting rows from the training data and computing the mean for every feature. This is done for each value of j as given in the basis function.

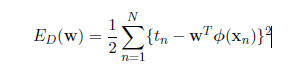
Next we compute the design matrix given as:



We then proceed to calculate the weight for each given value of complexity M using the given formula:



The next step involves the calculation of error using the least squared error method and using the subsequent formulae:



The above gives us the maximum likelihood solution using a closed form solution.



Here λ is the regularization parameter. And hence to avoid over fitting we use:



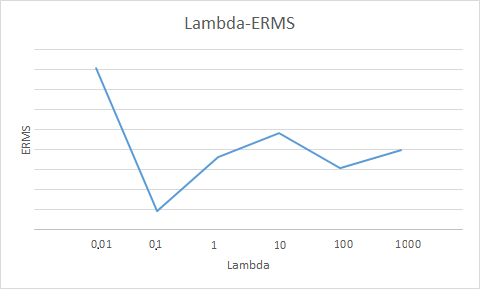
Using the above process we select M such that it best fits the solution without over fitting.

Model Complexity:

The model complexity is the number of basis functions that we choose to fit over the data. If we choose higher model complexity, the fitting of data might be better for the chosen model but over fitting arises by tuning the parameters perfectly over all the values in the training data .This will lead to huge error while testing the data.

Validation:

In the validation we tune the regularization function λ. We take the values of λ in logarithmic increments and compute the fit using the least squared error approach for the selected model M.



Testing:

After computing the parameters M and λ and tuning hyper parameters μ and s, where μ is normalized by selecting random values at each iteration we apply the model to the test dataset. And here we observe the least squared errors are comparable to the ones obtained after training and hence we conclude that we have avoided over fitting and have selected a model that best fits the data.

# RESULT

The final model, parameters and hyper parameters obtained using this approach are documented below:

Complexity (M) = 14

Lambda (λ) = 0.1

ERMS = 0.5619

ERMSTEST = 0.6238

Variance (S) = 0.7

# PART B: Stochastic Gradient Descent

In this approach we calculate the optimization by using the sum minimization approach. The change in error is evaluated at each iteration of computing the weights. When the change in error becomes a negligible value we stop the iteration and hence prevent from iterating over all the values in the data set.

The gradient descent is formulated using the given formula:



The rest of the process remains the same as closed form solution approach. The cycle of validation and testing are carried out in a similar fashion.

# MODEL

Model:

The parameters that need to me tuned are the complexity M and regularization parameter λ. The hyper parameters are mean μ and variance s.

To begin with by looking at the data we set S=0.7 initially as the data given is in the normalized form and randomly calculating we find the variance to be quite small. The value of μ is chosen by randomly selecting rows from the training data and computing the mean for every feature. This is done for each value of j as given in the basis function.

For each M, we first randomly initialize W, the weight matrix of order M X 1.

Then we compute error change for the next iteration of the weight matrix using:



If the change in error is grows then we decrease the value of the coefficientη by half.

Otherwise if the change in error decreases we compute the next iteration. We continue the iteration on W until the change in error becomes significantly small.

Validation:

In the validation we tune the regularization function λ. We take the values of λ in logarithmic increments and compute the fit using the least squared error approach for the selected model M.

Testing:

After computing the parameters M and λ and tuning hyper parameters μ and s, where μ is normalized by selecting random values at each iteration we apply the model to the test dataset. And here we observe the least squared errors are comparable to the ones obtained after training and hence we conclude that we have avoided over fitting and have selected a model that best fits the data.

# RESULT

The final model, parameters and hyper parameters obtained using stochastic descent approach are documented below:

Complexity (M) = 13

Lambda (λ) = 0.1

ERMS = 0.6040

ERMSTEST = 0.6172

Variance (S) = 0.7

# PERFORMANCE COMPARISION

On comparing the performance of the two methods we observe that for the closed form solution approach, we need to calculate NxM design matrix for each values of M. So that becomes approximately NxM2. However for small number of features M is a constant and run time will be in the order of N.

In comparison, for stochastic gradient descent approach we need to calculate nxM order design matrix. (n<N, where n is the value when the error change becomes minimal). However if we have a design matrix already computed the runtime is of the order nxM. Computing the whole design matrix, or saving the already computed values iteratively results in a much faster implementation of the gradient descent approach and will be much faster than closed form solution approach as n<N, for n being the value when we stop the weight matrix iterations when the error change become significantly small.