CSE 4/574
Introduction to Machine Learning
Homework 2
Total Marks 50 (You may also earn +20 Bonus!!)

Your Name:
Your email ID:
Your UB Person ID:

1.[10] Define Linear Basis Functions to support building Linear models for regression. Describe with examples.

While solving a linear regression problem we aim to find the optimal weights(w) for the

equation(1) :y(x,w) = W0 + W1*x1 +..Wn*xn = W^T * x

using the given data points. However, by doing slight modification to the above formula can help us to perform linear regression on more complex fixed non-linear functions. Thus, in place of simple linear 'x', we assume 'w' to be in combination of non-linear function Q_j(x). This Q_j(x) (phi of x) is now called 'linear basis' to the problem of regression. Hence new equation(1) is similar to

new equation1(2): y(x,w) = W0 + W1*x1 +..Wn*Q_n(x) = W^T * Q(x)

Here Q_j(x) extrinsically acts similar to static data points in x_i in Eq(1), although intrinsically they are totally different. Hence, We can use the same to perform linear regression & learn optimal weights for basis function Q(x) with given data points.

Apart for inducing non-linearity to regression problem via polynomial basis, we can use variety of linear basis functions that induce various effect into our model. For example:

a) Gaussians function: exp(-(x - mu_j)^2 / 2*s^2)
b) Sigmoidal function: sigma((x - mu_j) / s)
c) Tanh function: 2*sigma(a) - 1

Both of the above function are of wavelet form(periodical & finite), types of which are generally preferred for learning application unless the case demands an exception like 'Fourier series'.

2. [6] What is a design matrix and how does it contribute in getting the Maximum Likelihood solution to the regression problem.

As we know that MLE is about maximising the likelihood function. In terms of regression problem, we can formulate the equation as

Eq(1): p(t|x,w, β) = N(t | y(x,w), β^–1). OR (for 'N' data points): p(t|X,w, β) = Product(N(tn | wTϕ(xn), β^–1))

Now to maximise this likelihood Eq(1), we use concept of log-likelihood to simplify the objective to get Eq(2) : ln p(t|w, β). Now we use the concept of gradient descent to find the optimal weights (w) for above regression problem.

Solving this gradient descent gives us Eq(3) : ∇ ln p(t|w, β) = Sum({ tn – w^T*ϕ(xn)} * ϕ(xn)T

Eq(3) gives us the Eq(4) optimal weights (W_ml): (Φ^T*Φ)^–1 * Φ^T * t

Where 'Φ' is called design matrix whose elements are output of basis function for given 'x' (i.e. Φ(x_n)). The inverse in Eq(4) is called Moore-Penrose pseudo inverse.

The whole idea of this whole activity is to get us a 'Closed form solution' to the regression problem using concepts of MLE. And design matrix forms an important part of it.

3. [6] What is sequential learning and how does it address some of the drawbacks of Maximum Likelihood solution.

Sequential learning is also called online learning. In MLE we try to maximise the likelihood in one go, via finding a closed form solution using Moore-Penrose pseudo inverse. In real life scenario, this is not the best option as it consumes a lot of computing power & it also gobbles up a huge mass of data, which is inefficient from a learner's prospective.

Thus, Sequential learning proposes a better alternative in learning iteratively in chunks in comparison to finding a closed form solution. Here we uses the concept of Stochastic gradient descent to break the learning into small chunks of data & learn iteratively form it. We update the weights in each iteration using Eq(1): w(τ+1) = w(τ) – η∇En
Where weights are improved(incremented) based upon calculated error for just that set of data & a hyper parameter of 'learning rate'(η).

Sequential learning is most useful when the data at the moment is scarce but keeps accumulating(incoming) at a regular pace. A very common real life life problem. Hence we can start learning the hypothetical model and keep on updating the weights once new set of data is ready.

4. [8] What is the advantage of introducing regularization and how does it impact variance of the predicted curves?

As we know that the complex models tend to overfit, especially when data available is low. Though we can control the overfitting using regularization, but only to some extent. By introducing regularization we extend an extra penalty in loss function on the norm of the weights in order to reduce the complexity of the model. This puts extra constraints on the distribution of weights. This results in suppressing the variance of the predicted curve as the complexity of model (weights) have been reduced due to regularization. Before regularization, High complexity of the model was the reason of the hight variance. The amount of regularization to be done depends upon specific case.

5. [6] What is the difference between Lasso and Ridge Regression. Describe using equations.

We can use various types of regularization methods to achieve the goal. One of the main method is to add an extra penalty for weights in loss function for the regression problem. The kind of the penalty(regularization term) defines the type of regularization (regression). If the added term is L1-norm of weights(w), it's called Lasso regression. If the added term is L2-norm of weights(w), it's called Ridge regression. The final loss function is of the following form, where |wj|^q is the q-norm of weights(w):

Eq(1): 1/2*Sum{tn – w^T*ϕ(xn)}^2 + (λ/2)*Sum[| w_j |^q] —> Last term is Regularization Term

Ridge regression(L2 norm of weights) tries to suppress the weights of the model equitably, so that model is not dependent upon just few weight of the model. This leads to better a generalisation of the ground truth and less sparse solution in comparison to Lasso regularization.

On other hand, Lasso(L1 norm of weights), tries to suppress some weights more on the cost of others. This makes model rely more upon few important weight. This might give you better results but some times lead to not so good generalisation of the ground truth. This also leads to a more sparse solution, where less important weights might be suppressed to 0.

Ridge Reg. results in a circular contours & Lasso Reg. results in a square contour, when minimising the loss function using concepts of Lagrangian multiplier for constrained optimization.

6. [6] What is Bias and Variance and how does it help in gaining insights on model complexity?

There are 2 components to the expected loss.
Eq(1): E[L] = Integral ({y(x) – h(x)}^2*p(x)dx + Integral {h(x) – t}^2*p(x, t)dxdt.

The second component (h(x) – t) is due to noise(not dependent upon ground truth) in generating target variable. This is represents minimum achievable loss for the equation.

The First term(y(x) – h(x)) can be decomposed into (Bias)^2 + Variance.

Where Bias is the average difference between the predicted model function & ground truth function. This gives us an idea how well model has been able to generalise & learn the data.

The variance is the loss due to uncertainty prevailing in the training data. This uncertainty creeps into learned model and contributes in expected error.

7. [8] Briefly describe the Bayesian Gaussian Regression and how does it help in addressing some of the potential drawbacks of Maximum Likelihood solution for the linear regression model.

Maximum likelihood is about maximising:
Eq(1): p(tlx,w, β) = N(t l y(x,w), β^–1). OR (for 'N' data points): p(tlX,w, β) = Product(N(tn l wTϕ(xn), β^–1))

Thus, MLE depends upon the complexity of model & its basis function. This put extra constrained on the learning of the model.

In comparison to this, Bayes Gaussian Regression, uses the concept of Bayesian probability & makes assumption about the underlying distribution of input data to be a 'Gaussian'. This causes a cascading effect on the distributions of further in-situ calculated terms to become 'Gaussian'. This phenomenon is also called conjugate distribution. Thus for a conjugate Gaussian prior, posterior will be also a Gaussian. This assumptions leads us to nullify various in-situ terms that we couldn't do so in MLE. Also optimal weights vector for gaussian will be its mean only.

As, the The posterior Gaussian is defined by the mean(m_N) & co-variance(S_N) given by:
Eq(2): m_N = S_N *(S_0^–1 * m_0 + β*^ΦT*t)

Eq(3): S_N^–1 = S_0^–1 + β*Φ^T*Φ.

Where m_0 & S_0 are prior mean & covariance.

As one can see our posterior is not dependent upon weight (w), our model is not constrained by the limitations of MLE.

8. [**Bonus**][10] Consider a data set in which each data point tn is associated with a weighting factor $r_n > 0$, so that the sum-of-squares error function becomes

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(t_n - W^T\phi(\mathbf{x}_n))^2 \tag{1}$$

Find an expression for the solution $\mathbf{w}^*$ that minimizes this error function.

For given information, equation(1) will become Eq(2) for considering weighing factor(r_n):
Eq(2): ED(w)= 1/2 * Sum[r_n * (tn –WTϕ(xn))]^2

Now we just need to take the gradient of Eq(2) w.r.t ‘w’ and find the minima in closed for solution.

Doing this will give us optimal ‘w’ that minimises the error:

w = [sum (r_n * ϕ(xn)*ϕ(xn)^T)]^-1 * [sum(r_n * t_n * ϕ(xn))]

9.[**Bonus**][10] Consider a linear basis function regression model for a multivariate target variable t having a Gaussian distribution of the form $p(\mathbf{t}|\mathbf{W}, \mathbf{\Sigma}) = \mathcal{N}(\mathbf{t}|\mathbf{y}(\mathbf{x}, W), \Sigma)$, where $\mathbf{y}(\mathbf{x}, W) = \mathbf{W}^T\phi(\mathbf{x})$ together with a training data set comprising input basis vectors $\phi(x_n)$ and corresponding target vectors $\mathbf{t}_n$, with $n = 1, ..., N$. Show that the maximum likelihood solution W_{ML} for the parameter matrix W has the property that each column $\mathbf{w}$ of W_{ML} is given by an expression of $\mathbf{w} = (\phi^T\phi)^{-1}\phi^T\mathbf{t}$. Note that this is independent of the covariance matrix $\mathbf{\Sigma}$. Show that the maximum likelihood solution for Σ is given by

$$\Sigma = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{t}_n - W_{ML}^T\phi(\mathbf{x}_n))(\mathbf{t}_n - W_{ML}^T\phi(\mathbf{x}_n))^T \tag{2}$$

We can use log-likelihood of the
the given Gaussian distribution (p(t | W, Σ)).

This will give us:
Eq(1): ln p(t | W, Σ) = ln p(T | X,W, β) = Sum(ln N(tn | W^Tϕ(xn), β^–1*I))

Where β is precision, i.e. inverse of Σ

Thus we differentiate the Eq(1) w.r.t W & set it equal to 0.
Thus we get:
Eq(2): W_ML = (Φ^T*Φ^)–1 * Φ^T*T
Eq(2) is similar to one derived for a single variable 't'.
Hence we can decompose Eq(2) into a number of Eq(3) for each target variable (t_k).
Where Eq(3) is one mentioned in question:
Eq(3): w = (ϕ^T * ϕ)^–1ϕ^T * t

Now, to get Σ= 1/N * Sum [(t –W_ML^T * ϕ(x))(t –W_ML^T * ϕ(x))^T]

We need to compare Eq(1) & Eq(2) with general MLE estimation equation for covariance (single variable case):
Eq(4): Σ_ML = 1/N *Sum(xn – μ_ML)(xn – μ_ML)^T

Given the fact that weights given by MLE (i.e. W_ML) tend to accumulate around the mean of the distribution & we can link this with Eq(1) & Eq(2) for each target variable t_k