Artificial Neural Network Coursework

Karan
230293944

April 2024

Abstract

We are constructing and training a dataset $\mathcal{D} = \{(x_n, t_n)\}_{n=1}^{N}$ with Gaussian noise embedded into its target values(t_n). The dataset is applied on neural network consisting input-output layer with 1 hidden layer in between. This neural network is trained and compared using Error back propagation and Radial basis function interpolation. The approach compare on basis of their mean square error value. The first training method is Error back-propagation(EBP) method which have some cost function E, and cost function had been minimised. The neural network is trained for 3 different cases in EBP where we modify the number of units in the hidden layer such as $k =\ 3, 5, 11$ with most optimally trained when hidden layer unit is 5. The loss function is the usual sum of square function. To minimize the over-fitting in the network, regularization is inferred and hessian matrix calculation have been performed and eigenvalues are calculated in order to state that it is not a definite positive hessian matrix. The second method used to trained the neural network is radial basis function(RBF) interpolation method consisting 5 centers on regular grids. The trained model have been compared keeping mean square error as our comparing parameter. The neural network is trained slightly better with optimal value of EBP approach.

1 Introduction

Regression problems fall under the domain of supervised learning, a cornerstone of machine learning. These problems revolve around predicting continuous outcomes by discerning patterns and relationships between input variables and the dependent variable. In essence, regression tasks entail expressing the anticipated behavior of the dependent variable as a function of the input features.

The artificial data set $\mathcal{D}$ that contains N numbers of data points which which can visualize in 2 dimensional scattered plot form. $\mathcal{D} = \{(x_n, t_n)\}_{n=1}^{N}$ where $x_n \in\ X,\ X \subset \mathbb{R}^d$ represent the input values and $t_n \subset \mathbb{R}$ represent the actual target values. The target data point are given by

$$t_n = x_n + \frac{1}{4} * cos(\pi * x_n) + \epsilon_n$$

where ϵ_n is represent noise which is drawn from Gaussian distribution with $\mu = 0$ and $variance = 0.1$. The Gaussian noise can be define as $\mathcal{N}(\epsilon_n|0,\sigma^2)$ is:

$$\mathcal{N}(\epsilon_n|0,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\{-\frac{(t_n - \phi(\vec{x_n}))^2}{2\sigma^2}\}$$

The domain ranges is $X = \pm\{1\}^d$.

The neural network consisting 3 layers such that 1 input layer, 1 hidden layer and 1 output layer. And Instantaneous value of error energy or cost function is in form of sum of square that expressed as

$$E = \frac{1}{2}\sum_{n=0}^{N}(t_{(n)} - z^{(2)}(x_n))^2$$

This cost function should be minimised to make neural network optimal and efficient. The distance between actual output and target output should be lesser so the prediction for unseen data would have lesser amount of error.
Initially, Applying Error Back Propagation (EBP) for minimising the instantaneous value of error energy by adjustment of weight in every epoch. EBP use gradient descent approach for the minimisation of cost function. The cost function (E) require to be partially derivate with respect to weight of the inputs ($w_{k,j}$). But there is no direct connection between our error cost function and weight's input. Mathematically, it can be solved by using chain rule of calculus. And using this method, we will achieve the minimised cost function of the neural network.
To assess the impact of varying the number of hidden layer units (k), EBP is executed for 3 cases:$k = [3, 5, 11]$. While increasing the number of hidden layer units may enhance the network's capacity to capture complex patterns, it can also lead to slower learning rates and potential overfitting. To address the latter, regularization terms are incorporated into the cost function, imposing constraints on model parameters and mitigating the risk of overfitting. In mathematical term, regularized function defined as

$$\tilde{E}(\vec{w}) \;=\; E(\vec{w}) + \frac{\upsilon}{2}\langle w|w\rangle \quad \upsilon \geq 0 \tag{1}$$

where $\tilde{E}(\vec{w})$ is regularised loss function ,$E(\vec{w})$ is normal loss function and $\frac{\upsilon}{2}\langle w|w\rangle$ is length of weights. In order to regularise the cost function, Hessian matrix has been introduced and defined as squared symmetric matrix and used in second-order optimization algorithms, such as Newton's method and its variants, to determine the step direction and size for updating the parameters during optimization.

$$E(\vec{w}) \approx E_0 + \langle b|w\rangle + \frac{1}{2}\langle w|H|w\rangle \tag{2}$$

E_0 is minimum cost error that can be obtained,b is a suitable vector of order with w and H contains the information about the second derivative of loss function with respect to weight w. By taking into account the curvature of the cost

function, these algorithms can converge more efficiently towards the minimum. The Hessian matrix helps determine the appropriate balance between fitting the data and penalizing model complexity.
Radial basis interpolation have ability to handle irregular space data and approximate complex function or datasets(including nonlinear relationship) even in the region with sparse or noisy data. This application involves approximating a function or dataset by constructing a linear combination of radial basis functions centered at data points. It directly computes the weights of the radial basis functions based on interpolation conditions.

Comparison of the curve obtained from Radial basis interpolation with Error back propagation as both approximate functions but with different approach.

2 Methods

2.1 Data Generation

Dataset $\mathcal{D}$ consists of N data points in the form of (x_n, t_n), where x_n represents the input data points and t_n represents the target output that are being calculated as $t_n = x_n + \frac{1}{4} \cdot \cos(\pi x_n) + \epsilon_n$. Here, x_n has been randomly selected within the domain $X = \pm\{1\}^d$. The Gaussian noise ϵ with $\mu = 0$ and $\sigma = 0.1$ is produced in the process to achieve the target value. Thus, the dataset contains target points with some noise embedded into it. To represent this graphically, a scatter plot is best suited method.

2.2 Error Back Propagation

Definition : The process of propagating the network error from the output layer to the input layer is called backward propagation[1], or simple back propagation. The back propagation algorithm is the set of steps used to update network weights to reduce the network error.

Let the data space $\mathcal{D} = \{\vec{x_m}, \vec{t_m}\}_{m=1}^{L}$ such that there is function h that belongs to $\mathbb{R}^{\mathbb{M}}$. The function h is defined as basic idea of regression with input X_m and will produce output t_m and some noise ϵ_m.

$$h(\vec{X_m}) = t_m + \epsilon_m$$

The function h is approach by using Error back propagation. The difference between t_m and output $z^{(2)}(\vec{x_m})$ should as close as possible to 0.
The only thing which change/modify in our network is weights. So, the loss function is a function of weights from input layer to hidden layer and weights from hidden layer to output. It can be expressed as

$$\mathcal{L}(W^{(1)}, W^{(2)}) = \sum_{m=1}^{M} \mathcal{L}_m \tag{3}$$

$$\mathcal{L}_m = \frac{1}{2}\langle \vec{t}_m - \mathbf{z}^{(2)}(\vec{X}_m) \,|\, \vec{t}_m - \mathbf{z}^{(2)}(\vec{X}_m)\rangle \tag{4}$$

And if there is only one feature that is M=1, then

$$\mathcal{L}_m = \frac{1}{2}\{\vec{t_m} - \vec{z^{(2)}}(\vec{X_m})\}^2$$

$$\nabla_{w_n^{(2)}}\mathcal{L}_m = \nabla_{w_n^{(2)}} * \mathcal{L}_m = \frac{1}{2}\langle \vec{t}_m - \mathbf{z}^{(2)}(\vec{X}_m) \,|\, \vec{t}_m - \mathbf{z}^{(2)}(\vec{X}_m)\rangle$$

Here, multiplying both side with gradient of $w_n^{(2)}$, with n is number of weight inputs from hidden layer. Here, we find the derivative of loss function with respect to activation function of layer 2.

$$\nabla_{w_n^{(2)}}\mathcal{L}_m = \frac{\partial \mathcal{L}_m}{\partial a_n^{(2)}}.\nabla_{w_n^{(2)}}.a_n^{(2)}$$

such that $a_n^{(2)}$ is an activation function for the output layer. The component can be define in such a way

$$\nabla_{w_n^{(2)}} = a_n^{(2)} = \begin{bmatrix} \frac{\partial}{\partial w_1^{(2)}}.a_n^{(2)} \\ \frac{\partial}{\partial w_2^{(2)}}.a_n^{(2)} \\ \vdots \\ \vdots \end{bmatrix}$$

Also, we can define derivative of our loss function with activation function in such a way

$$\delta_n^{(2)} = \frac{\partial \mathcal{L}_m}{\partial a_n^{(2)}}$$

The activation function for output layer is inner product of weights from output layer with the output of the hidden layer. It can expressed by

$$\frac{\partial}{\partial w_{n=k}^{(2)}}a_n^m = \frac{\partial}{\partial w_{n=k}^{(2)}}\sum_{s=0}^{k} w_{ns}^{(2)}.z^{(1)}$$

Here, s is used for bias as there is bias coming from hidden layer. So, if $s = 0$ then it is bias and $s = 1$ then we have weight such as w_n.

$$= \sum_{s=0}^{k} z_s^{(1)}\delta_{k,s} = z_k^{(1)}$$

Here, the $w_{n=k}$ will only going to affect the w_{ns}. The output here, $z_k^{(1)}$ is k_{th} output of the hidden layer. Let use the equation,

$$\nabla_{w_n^{(2)}} \mathcal{L}_m = \frac{\partial \mathcal{L}_m}{\partial a_n^{(2)}} . \nabla_{w_n^{(2)}} . a_n^{(2)}$$

Let define the local gradient such that it comes out to be

$$\delta_n^{(2)} = (a_s^{(2)} - t_{mn})$$

Here, we have written $a_s^{(2)} = z_n^{(2)}$, so $\nabla_{w^{(1)}} \mathcal{L}_m = \delta_n^{(2)} |\vec{z^{(1)}}\rangle$ where $\delta_n^{(2)} = z_n^{(2)} - t_{mn}$ Now, similarly we will solve for previous layer and partial derive with respect to weights corresponded to first layer. Therefore,

$$\nabla_{w_k^{(1)}} . a_k^{(1)} = |z^{(0)}\rangle$$

Where $a_k^{(1)}$ is activation function for layer 1.

$$\frac{\partial}{\partial a_k^{(1)}} \mathcal{L}_m = \sum_{s=1}^{m} \frac{\partial}{\partial a_k^{(1)}} a_s^{(2)} . \frac{\partial}{\partial a_s^{(1)}} \mathcal{L}_m \tag{5}$$

Where $\mathbf{a}^{(2)} = \langle w^{(2)} | z_k^{(1)} \rangle$ and $z_k^{(1)} = \sigma(a_k^{(1)})$

$$\frac{\partial}{\partial a_k^{(1)}} a_s^{(2)} = \sum_{r=0}^{k} \frac{\partial}{\partial a_k^{(1)}} . w_{sk}^{(2)} z_r^{(1)}$$

Where $z_r^{(1)}$ is an activation of function sigmoidal.

$$\frac{\partial}{\partial a_k^{(1)}} a_s^{(2)} = \sum_{s=1}^{m} w_{s,k}^{(2)} \frac{\partial}{\partial a_k^{(1)}} . \sigma(a_k^{(1)})$$

$\frac{1}{2}[1 - (z_k^{(1)})^2]$ is an activation function. Therefore, the equation can be written as

$$\frac{\partial}{\partial a_k^{(1)}} \mathcal{L}_m = \frac{1}{2}[1 - (z_k^{(1)})^2] . \sum_{s=1}^{m} w_{s,k}^{(2)} . \delta_s^{(2)} \tag{6}$$

As we can write the $\frac{\partial}{\partial a_k^{(1)}} \mathcal{L}_m = \delta_k^{(1)}$,

$$\delta_k^{(1)} => \nabla_{w_k^{(1)}} . \mathcal{L}_m = \delta_k^{(1)} |z^{(k)}\rangle$$

All gradient of loss function have been calculated and next stage is to update weights on layer(1) and layer(2), and it can be expressed as

$$|W_k^{(1)}\rangle_{m+1} = |W_k^{(1)}\rangle_m - \eta . \nabla_{w_k^{(1)}} . \mathcal{L}_m$$

$$|W_k^{(2)}\rangle_{m+1} = |W_k^{(2)}\rangle_m - \eta . \nabla_{w_k^{(2)}} . \mathcal{L}_m$$

The update of weight will run till some stopping criteria which can number of iteration or due to level of discrepancy.

2.3 Regularization

Definition : It is set of technique that can use to minimise the chances of over-fitting of the data.

Note : Over-fitting is occur when model learn the details and noise to the extend that it show negative results on the performance of the model. Capturing noise or random fluctuation as meaningful patterns that leads to training of data on unseen test dataset performance very poor.

Regularized loss function is define as sum of loss function with proportional of length of weights with υ.

$$\tilde{E}(\vec{w}) \;=\; E(\vec{w}) + \frac{\upsilon}{2}\langle w|w\rangle \quad \upsilon \geq 0 \tag{7}$$

where $\tilde{E}(\vec{w})$ is regularised loss function ,$E(\vec{w})$ is normal loss function and $\frac{\upsilon}{2}\langle w|w\rangle$ is length of weights.

The behaviour of loss function can be plot in form of parabola with more than 1 dimension. And if we make Taylor approximation up to the order 2 then we will get:

$$E(\vec{w}) \approx E_0 + \langle b|w\rangle + \frac{1}{2}\langle w|H|w\rangle \tag{8}$$

where E_0 is minimum possible error that can be obtained, b is a suitable vector of order with w and H contains the information about the second derivative of loss function with respect to weight w. H is defined as Hessian matrix. Also Hessian is symmetric matrix.
Use equation (10) in equation (9), we get

$$\tilde{E}(\vec{w}) \approx E_0 + \langle b|w\rangle + \frac{1}{2}\langle w|H + \upsilon\mathbb{I}|w\rangle \tag{9}$$

Where Hessian matrix has been move by a bit and introduce with identity matrix. Inner product of weight matrix can be written in identity matrix form as $\langle w|w\rangle = \langle w|\mathbb{I}|w\rangle$.

Now, In order to minimise, gradient will be introduced with respect to w and will be equate to vector zero. That is,

$$\begin{aligned} \vec{\nabla}_w E &= |0\rangle \\ |0\rangle &= |b\rangle + H|w^*\rangle \end{aligned} \tag{10}$$

We can write equation as

$$|b\rangle = -H|w^*\rangle \tag{11}$$

Similarly, we apply gradient with respect to weight for equation and equate it to vector zero to get minimise term:

$$\begin{aligned} \vec{\nabla}_w \tilde{E} &= |0\rangle \\ |0\rangle &= |b\rangle + (H + \upsilon\mathbb{I}|\tilde{w}\rangle \end{aligned} \tag{12}$$

To see the difference between $\tilde{w}$ and w^*, we are going to use the basis of Hessian matrix such that

$$\lambda_k = eigenvector\ of\ H$$

where each eigenvector corresponded to the eigenvalues are perpendicular. To find eigenvalue of Hessian matrix,

$$\begin{aligned} H|\lambda_k\rangle &= \lambda_k|\lambda_k\rangle \\ 0 &= \langle\lambda_q|\lambda_k\rangle \quad where \lambda_k \neq \lambda_q \end{aligned}$$

we can write the w^* as the linear combination of Hessian

$$|w^*\rangle = \sum_k |\lambda_k\rangle\langle\lambda_k|w^*\rangle \tag{13}$$

Similarly, the equation for for regularised weight

$$|\tilde{w}\rangle = \sum_k |\lambda_k\rangle\langle\lambda_k|\tilde{w}\rangle \tag{14}$$

Modify the equation (6) by substituting equation(5), the new equation can be written as

$$|0\rangle = (H + \upsilon\mathbb{I})|\tilde{w}\rangle - H|w^*\rangle \tag{15}$$

Now substitute values of equation(7) and equation (8) in equation (9),we get

$$|0\rangle = (H + \upsilon\mathbb{I})\sum_k\{|\lambda_k\rangle\langle\lambda_k|\tilde{w}\rangle\} - H\sum_k\{|\lambda_k\rangle\langle\lambda_k|w^*\rangle\} \tag{16}$$

The right hand side of equation, $(H + \upsilon\mathbb{I})$ is a matrix which is multiplied by sum of vectors, it output will be linear. So, we can modify by putting matrix within sum,

$$|0\rangle = \sum_k(H + \upsilon\mathbb{I}).|\lambda_k\rangle\langle\lambda_k|\tilde{w}\rangle - \sum_k H.|\lambda_k\rangle\langle\lambda_k|w^*\rangle \tag{17}$$

As we already stated that $H|\lambda_k\rangle = \lambda_k|\lambda_k\rangle$ and $|\upsilon\rangle\mathbb{I} = \upsilon$ we can substitute while solving the above equation, we will get

$$|0 >= \sum_k(\lambda_k + \upsilon)|\lambda_k\rangle\langle\lambda_k|\tilde{w}\rangle - \sum_k \lambda_k|\lambda_k\rangle\langle\lambda_k|w^*\rangle \tag{18}$$

In equation (12), we have $|\lambda_k\rangle$ as common vector and we also know that the $\langle\lambda_k|\tilde{w}\rangle$ will produce a number as output(inner product). So, we can take summation over the whole equation and can re-write as

$$|0 >= \sum_k\{(\lambda_k + \upsilon)\langle\lambda_k|\tilde{w}\rangle - \lambda_k\langle\lambda_k|w^*\rangle\}\lambda_k \tag{19}$$

Here on right hand side, terms before subtraction are term with regularised weight and after subtraction is terms without regularization and $|0\rangle$ is one vector equation which can transmitted to many scalar equation. Therefore, we can write this as

$$\langle \lambda_k | \tilde{w} \rangle = \frac{\lambda_k}{(\lambda_k + \upsilon)} \langle \lambda_k | w^* \rangle$$

We knew, λ_k is positive and υ is non-negative (mentioned in equation (1)) value so therefore it will always show positive value and within the range of 0 and 1.

If $\lambda_k >> \upsilon$ then the ratio

$$\frac{\lambda_k}{(\lambda_k + \upsilon)} \approx 1$$

Therefore, we can say that the regularised model produced the similar results with normal model. While in the vice versa case, where $\lambda_k << \upsilon$, we will get

$$\langle \lambda_k | \tilde{w} \rangle = \frac{\lambda_k}{\upsilon} \langle \lambda_k | w^* \rangle$$

2.4 Radial Basis Function

Radial basis function methods are the mean to approximate the multivariate functions.RBFs depends on the distance between the input vector and a set of predefined centers. Let suppose we have data set in such form

$$\mathcal{D} = \{(x_n, t_n)\}_{n=1}^{L}$$

And there is function y which will provide us interpolated values and defined as $y(x_n) = t_n$. The network consist multiple inputs inputs with weights.

$$t_m = y(x) = \sum_{n=1}^{L} w_n \phi(|x_n - x_m|)$$

The ϕ is function which measure distance between point to our feature x. And for each m we have different equations and structure.

$$\Phi : \Phi_{nm} = \phi(|x_n - x_m|) = \Phi_{mn}$$

If we have 2 data points then it would be in form of $2 * 2$ matrix. Also, it would be a symmetric matrix. $\Phi \in \mathbb{R}^{L*L}$ and there is inverse exist for this. This formulation demonstrates how RBF methods use the distances between input points to construct an interpolation function that accurately represents the underlying multivariate function.

3 Results

3.1 Data Generation

The target dataset has been generated such that there 50 data points scattered in the domain $\chi = [-1, 1]$. The presence of the Gaussian noise in the network affecting the range of the points and therefore the data points are not purely in the domain range.

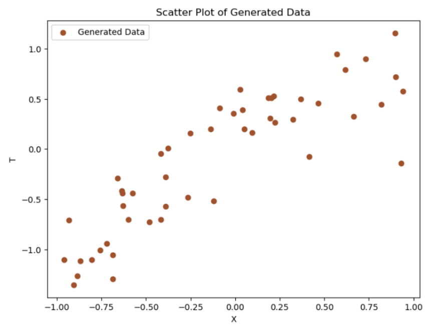

Figure 1: Target point visualization using scattered plot

3.2 Error back propagation

3.2.1 Cost function minimization

3 layers have been used as such the first unit is

$$z_i^{(0)} = x^i \ such \ that \ i = 0, 1, 2, 3$$

and the first hidden layer is set to 1, $z_0^{(1)} = 1$ where (1) denote the layer number and 0 represent the first unit of hidden layer. The other unit of hidden layer are computed by using sigmoidal function

$$z_k^{(1)} = \frac{1}{1 + exp(-\sum_{j=0}^{3} w_{k,j}^{(1)} x^j)} \ such \ that \ k = 1, 2..., K$$

and output of the network is given mathematically

$$z^{(2)} = \sum_{k=0}^{K} w_k^{(2)} z_k^{(1)}$$

Here, we are required to minimise the instantaneous value of error energy or cost function E that we already mentioned in the previous section. To perform minimisation, we required to partially differentiate with respect to the input weights $w_{k,j}$.

In order to differentiate, we required to use partial differentiation with the help of chain rule as there is no direct connection between the cost function and input weight.

$$\begin{aligned}\frac{\partial E}{\partial e_j(n)} &= \frac{\partial \frac{1}{2}\sum e_j^2(n)}{\partial e_j}\\ &= e_j(n)\\ \frac{\partial E}{\partial e_j(n)} &= \sum_{n=0}^{N}(t_{(n)} - z^{(2)}(x_n)) \quad \text{eq}^n(1)\end{aligned}$$

$$\begin{aligned}\frac{\partial e_j}{\partial z^{(2)}} &= \frac{\sum_{n=0}^{N}(t_{(n)} - z^{(2)}(x_n))}{\partial z^{(2)}}\\ \frac{\partial e_j}{\partial z^{(2)}} &= -x_n \quad \text{eq}^n(2)\end{aligned}$$

$$\begin{aligned}\frac{\partial z^{(2)}}{\partial z^{(1)}} &= \frac{\partial \sum_{k=0}^{K} w_k^{(2)} z_k^{(1)}}{\partial z_k^{(1)}}\\ \frac{\partial z^{(2)}}{\partial z^{(1)}} &= w_k^{(2)} \quad \text{eq}^n(3)\end{aligned}$$

$$\frac{\partial z^{(1)}}{\partial w_{k,j}} = \frac{\partial[\frac{1}{1+\exp(-\sum_{j=0}^{3} w_{k,j}^{(1)} x^j)}]}{\partial w_{k,j}}$$

Using quotient rule, we can write this as

$$\begin{aligned}\frac{\partial z^{(1)}}{\partial w_{k,j}^{(1)}} &= \frac{[1+\exp(-\sum_{j=0}^{3} w_{k,j}^{(1)} x^j)] \cdot \frac{\partial}{\partial w_{k,j}^{1}} 1 - 1 \cdot \frac{\partial}{\partial w_{k,j}^{1}}[1+\exp(-\sum_{j=0}^{3} w_{k,j}^{(1)} x^j)]}{[1+\exp(-\sum_{j=0}^{3} w_{k,j}^{(1)} x^j)]^2}\\ &= \frac{\exp[\sum_{j=0}^{3} w_{k,j}^{(1)} x^j]}{[1+\exp(-\sum_{j=0}^{3} w_{k,j}^{(1)} x^j)]^2}\end{aligned}$$

We can write this as

$$\begin{aligned}&= \frac{1}{[1+\exp(-\sum_{j=0}^{3} w_{k,j}^{(1)} x^j)]} \cdot \frac{[\exp \sum_{j=0}^{3} w_{k,j}^{(1)} x^j]}{[1+\exp(-\sum_{j=0}^{3} w_{k,j}^{(1)} x^j)]}\\ &= z_k^{(1)} \cdot \left[1 - \frac{1}{1+\exp\left(-\sum_{j=0}^{3} w_{k,j}^{(1)} x^j\right)}\right]\\ &= z_k^{(1)}[1 - z_k^{(1)}] \quad \text{eq}^n(4)\end{aligned}$$

Using all equations and multiplying in order to achieve

$$\frac{\partial E}{\partial w_{k,j}} = \sum_{n=0}^{N} (t_{(n)} - z^{(2)}(x_n)) \cdot (-x_n) \cdot z_k^{(1)} [1 - z_k^{(1)}]$$

or,

$$\frac{\partial E}{\partial w_{k,j}} = -[z_k^{(1)} [1 - z_k^{(1)}] \sum_{n=0}^{N} [t_{(n)} - z^{(2)}(x_n)] \cdot x_n$$

The above equation will be the minimised cost function for our error back propagation neural network.

3.2.2 EBP Implementation

The error back propagation is implemented for data point that is generated in such a way that each target point carries some embedded Gaussian noise while generating. The EBP is implemented for 3 different number of unit in hidden layer k of the network. The 3 instances are $k = [3, 5, 11]$. The following graph in figure represent the behaviour of output point for corresponding input data point.

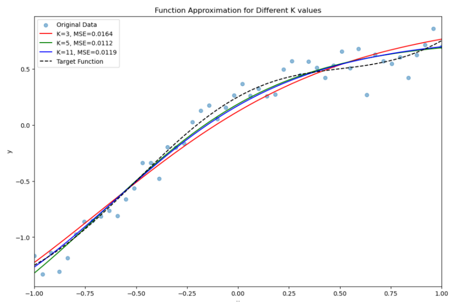

Figure 2: Visualization of outputs with different hidden unit values

The following parameter are kept constant for the implementation of EPB on 3 cases:

$$learning\ rate(\eta)\ :\ 0.01$$
$$Number\ of\ epochs:\ 600$$

This observation aligns with the typical behavior of neural networks, where increasing model complexity by adjusting hidden layer units initially improves performance by allowing the model to capture more complex patterns in the data. However, there is point of diminishing returns, beyond which adding more complexity does not yield significant performance gains and may even lead to overfitting. In this case, it seems that 5 hidden units shows a good balance between model complexity and performance.

Hidden Layer unit	Mean square error
3	0.0164
5	0.0112
11	0.0119

Table 1: Mean square error for hidden layer units

3.2.3 Sum of square of cost function

Lets take a regression equation such that

$$t = h(x) + \epsilon$$

where ϵ is noise produced when we apply a function on x to achieve our target value t.And we know that $h(x)$ is function of input and weights. So, further we can define this as

$$t = \phi(\vec{x}, \{\vec{w}\}) + \epsilon$$
$$\epsilon = t - \phi(\vec{x}, \{\vec{w}\})$$

Suppose noise is drawn out from Gaussian distribution with 0 mean and variance of σ^2, then we can expressed it as

$$\begin{aligned}\mathcal{N}(\epsilon|0, \sigma^2) &= \frac{1}{\sqrt{2\pi\sigma^2}} exp\{-\frac{\epsilon^2}{2\sigma^2}\} \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} exp\{-\frac{(t - \phi(\vec{x}, \{\vec{w}\}))^2}{2\sigma^2}\} = \hat{P}(t|\vec{x})\end{aligned}$$

When $\hat{P}(t|\vec{x})$ is an estimator of the conditional probability distribution such that given input. Let see from where we get this conditional probability.
Let define Dataset such

$$\mathcal{D} = \{(\vec{x_n} t_n)\}_{n=1}^{L}$$

such that $x_n \in \mathcal{X} \in \mathbb{R}^{d}$. Now suppose, the joint probability distribution $\mathcal{P}_{X,T}(\vec{x}, t)$, so we can be define the conditional probability as well. We can expressed this as

$$\mathcal{P}_{T|X}(t|\vec{x}) = \frac{\mathcal{P}_{X,T}(t, \vec{x})}{\mathcal{P}_X(\vec{x})}$$
$$\mathcal{P}_{X,T}(t, \vec{x}) = \mathcal{P}_{T|X}(t|\vec{x}).\mathcal{P}_X(\vec{x})$$

We can express the expectation and variance of this as well such that

$$Var = \mathbb{E}_{T|\vec{X}}[t^2|\vec{x}] - \mathbb{E}_{T|\vec{X}}[t|\vec{x}]^2$$

Now, we can define the likelihood as well which will tell us that how likely the dataset would be look like. It can be express as product of all probabilities such that

$$L = \Pi_{n=1}^{L} \mathcal{P}(\vec{x}_n, t_n)$$

And to get the maximum likelihood, multiply log on both side and to obtain minimum likelihood, use negative of log such that

$$\begin{aligned}-\log L &= -\sum_{n=1}^{L} \log \mathcal{P}(\vec{x}_n, t_n) \\ &= -\sum_{n=1}^{L} \{(\log \mathcal{P}_{T|X}(t|\vec{x})) + \log \mathcal{P}_X(\vec{x})\}\end{aligned}$$

Substituting value of conditional probability in log likelihood equation, we get

$$-\log L = -\sum_{n=1}^{L} \log\{\frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{(t-\phi(\vec{x_n}, \{\vec{w}\}))^2}{2\sigma^2})\} - \sum_{n=1}^{L} \log \mathcal{P}_X(\vec{x_n}) \quad (20)$$

The only parameter we can change to minimize the noise in network is weights, so the marginal term in above equation is remain constant and does not affect, therefore it is redundant and new equation can be simplified as:

$$-L^{'}(\phi) = \frac{1}{2\sigma^2} \sum_{n=1}^{L} ((t_n - \phi(\vec{x_n}))^2 - L \log \frac{1}{\sqrt{2\pi\sigma^2}} \quad (21)$$

In equation, the second term is constant and will be same for all the value. It is just a product of all term with this. And $\frac{1}{2\sigma^2}$ is independent of term x therefore, it can be taken out from summation.
If we look particularly the first term here, it is summation over square term. The sum of square come from supposing that the deviation from expectation behaviour comes from Gaussian distribution. So, we can say that Gaussian noise is embedded in form of sum of square error. So, whenever we use sum of square, it implies that Gaussian distribution adding noise to our measurement.

3.3 Regularization - Hessian matrix

The network is formed by the input $z^{(0)} = x$ with a single hidden unit $z^{(1)} = vz^{(0)}$ and output is given by $z^{(2)} = wz^{(1)}$. The cost function E is similar as define before. The hessian matrix hereby define in such a way

$$H = \begin{bmatrix} \frac{\partial^2 E}{\partial v^2} & \frac{\partial^2 E}{\partial w \partial v} \\ \frac{\partial^2 E}{\partial v \partial w} & \frac{\partial^2 E}{\partial w^2} \end{bmatrix}$$

Here, the values of network can redefine as follow

$$\begin{aligned} z^{(0)} &= x \\ z^{(1)} &= vz^{(0)} = vx \\ z^{(2)} &= wz^{(1)} = wvx \end{aligned}$$

So we can define the given cost function in the form of w, v *and* x and can be expressed as

$$E = \frac{1}{2}\sum_{n=0}^{N}(t_{(n)} - wvx(x_n))^2$$

Use partial derivative and calculate each term of the hessian matrix.

$$\frac{\partial E}{\partial v} = \frac{\frac{1}{2}\sum_{n=0}^{N}(t_{(n)} - wvx(x_n))^2}{\partial v}$$

$$\frac{\partial E}{\partial v} = \sum_{n=0}^{N}(t_n - wvx(x_n)).(wx(x_n)) \tag{22}$$

$$\frac{\partial^2 E}{\partial v^2} = \sum_{n=0}^{N}(wxx_n)^2 \tag{23}$$

$$\frac{\partial E}{\partial w} = \frac{\frac{1}{2}\sum_{n=0}^{N}(t_{(n)} - wvx(x_n))^2}{\partial w}$$

$$\frac{\partial E}{\partial w} = -\sum_{n=0}^{N}(t_n - wvx(x_n)).(vxx_n) \tag{24}$$

$$\frac{\partial^2 E}{\partial w^2} = \sum_{n=0}^{N}(vxx_n)^2 \tag{25}$$

Using above equation, and partial differentiate with respect to w.

$$\frac{\partial^2 E}{\partial v \partial w} = -\sum_{n=0}^{N}[t_n xx_n - 2wv(xx_n)^2] \tag{26}$$

Using above equation and partial differentiate with respect to v.

$$\frac{\partial^2 E}{\partial w \partial v} = -\sum_{n=0}^{N}[t_n xx_n - 2wv(xx_n)^2] \tag{27}$$

We have calculated all parameter of hessian matrix, the matrix will be expressed as

$$H = \begin{bmatrix} \sum_{n=0}^{N}(wxx_n)^2 & -\sum_{n=0}^{N}[t_n xx_n - 2wv(xx_n)^2] \\ -\sum_{n=0}^{N}[t_n xx_n - 2wv(xx_n)^2] & \sum_{n=0}^{N}(vxx_n)^2 \end{bmatrix} \tag{28}$$

Now, if the minimum weights are define as $v = v_0$ and $w = w_0$, the new hessian matrix that produces the minimum over-fitting to the network will be define as

$$H = \begin{bmatrix} \sum_{n=0}^{N}(w_0xx_n)^2 & -\sum_{n=0}^{N}[t_n xx_n - 2w_0v_0(xx_n)^2] \\ -\sum_{n=0}^{N}[t_n xx_n - 2w_0v_0(xx_n)^2] & \sum_{n=0}^{N}(v_0xx_n)^2 \end{bmatrix} \tag{29}$$

$$H^T = \begin{bmatrix} \sum_{n=0}^{N}(w_0xx_n)^2 & -\sum_{n=0}^{N}[t_nxx_n - 2w_0v_0(xx_n)^2] \\ -\sum_{n=0}^{N}[t_nxx_n - 2w_0v_0(xx_n)^2] & \sum_{n=0}^{N}(v_0xx_n)^2 \end{bmatrix} \tag{30}$$

As $H_{12} = H_{21}$, we can say that the it is symmetric hessian matrix.

3.3.1 Hessian matrix

In order to find its eigenvalues of the Hessian matrix, The values are substituted in calculated hessian matrix when gradient at $v0 : 0.3151$ and gradient at $w0 : 0.3348$ and matrix comes out to be

$$H = \begin{bmatrix} 0.06927792 & 4.01501055 \\ 4.01501055 & 0.0781597 \end{bmatrix} \tag{31}$$

And eigenvalues comes out to be

$$Eigenvalues : [-3.94129419\ ,\ 4.08873181]$$

From results value, we can observed that it is not definite positive matrix as one eigenvalue is positive and other one is negative. The negative eigenvalue suggest that the function is locally concave along at least one direction and positive eigenvalue indicates that the function is locally convex along another direction.

3.4 Radial Basis Function

The implementation of radial basis interpolation considering 5 centers with regular grid is performed with similar constructed data-points. Here we are using 5 centers, meaning we will have 5 Gaussian radial basis functions, each centered at a different point and the centers are distributed evenly on a regular grid within the range of the dataset. This ensures that the centers are uniformly spaced, which can help in capturing the features of the data distribution effectively.

Centers	Mean square error
5	0.0114

Table 2: Mean square error value for RBF interpolation with 5 centers

3.5 Curves of RBF and EBP

Comparing the Mean Square Error (MSE) of Error Back-Propagation (EBP) with 5 hidden units with 5 centers on regular grid, we observed EBP method with 5 hidden units has a slightly lower MSE 0.0112 compared to the RBF

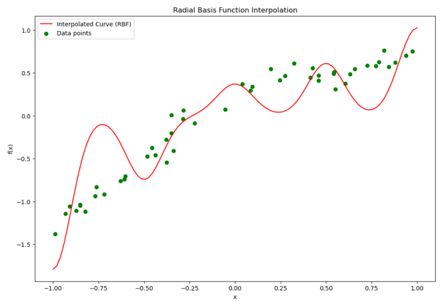

Figure 3: RBF interpolation with 5 centers

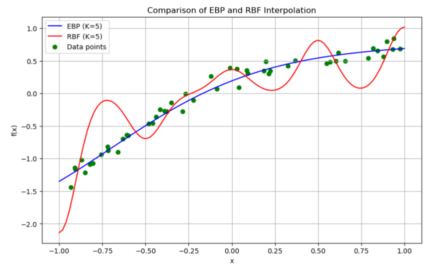

Figure 4: EBP with 5 hidden layer units vs RBF with 5 centers on regular grid

method with 5 centers on a regular grid MSE = 0.0114. The difference in MSE between the two methods is minuscule, indicating that both methods are fairly effective in modeling the dataset.

4 Conclusions

In this study, we conducted an analysis of two training methods, Error Back-Propagation (EBP) and Radial Basis Function (RBF) interpolation, applied to a neural network with one input, one output, and one hidden layer. The data points were generated from randomly chosen input points within the domain of $X = \pm\{1\}^d$, with target values featuring Gaussian noise (mean = 0, variance = 0.1). Visualization using scattered plot technique aided in understanding the data distribution.

Our analysis revealed that the EBP method, particularly when implemented with 5 hidden layer units, performed optimally among various configurations tested (k = 3, 5, 11). Notably, the mean square error (MSE) showed a measurable difference between 3 and 5 hidden units, but increased notably with 11 units, suggesting a point of diminishing returns where additional complexity did not significantly improve performance.

To mitigate potential overfitting in the EBP model, we implemented regularization techniques. Additionally, we calculated the Hessian matrix and observed that it was symmetric and contained one positive and one negative eigenvalue. This indicated that the function represented by the matrix was concave along at least one direction and convex along another.

Furthermore, we trained the neural network using radial basis interpolation with 5 centers on a regular grid. Upon comparison, we found that the EBP model with 5 hidden layer units slightly outperformed the RBF method by a very small margin, as evidenced by mean square error evaluation.

In conclusion, our study highlights the effectiveness of the EBP method, particularly with a moderate number of hidden layer units, in modeling complex relationships within the dataset. Further investigation into regularization techniques and optimization strategies could enhance the performance of neural networks in similar contexts.

References

[1] Hollan Houle, Medium, [online], 11th April 2024. Available at: `https://towardsdatascience.com/error-backpropagation-5394d33ff49b`

[2] Christopher M Bishop(1995).Neural networks for pattern recognition. Oxford; New York. Oxford University Press. pp 161-162

[3] A. Engel; C. van den Broeck. Statistical mechanics of learning(2001). Cambridge, U.K.; New York. Cambridge University Press.pp 170-180.