MASTER 2 MICAS

Machine Learning Communications and Security

Course : MICAS931 - Introduction to Optimization.

School : Institut Polytechnique de Paris

Teacher : HADI GHAUCH

Subject : Final Project - Submit before 01/12/2020

Etudiant : Panongbene Jean Mouhamed Sawadogo.
Email : panongbene.sawadogo@telecom-paris.fr

SOMMAIRE :

Build your own DNN

Part I : Deep linear network

We will start with a 2-layer deep linear network, which is a special case case obtained by setting all activation functions to an identity. Thus, the corresponding training problem is given by the following optimization problem :

$$\min_{\mathbf{W_1},\mathbf{W_2}} \frac{1}{N} \sum_{i\in[N]} \|\mathbf{W_2}\mathbf{W_1}x_i - y_i\|_2^2 + \lambda_1\|\mathbf{W_1}\|_F^2 + \lambda_2\|\mathbf{W_2}\|_F^2$$

We will use the BodyFat data set. The goal of this part is to solve (1) by implementing and comparing several methods : backpropagration (BP), minibatch SGD with an appropriate minibatch size, ADAM, and block-coordinate descent (BCD). Ensure that you select a 'good choice' the model complexity (dimensions of $\mathbf{W_1}, \mathbf{W_2}$) then fix them for all the above methods. Recall that, it is better to start with complex model (large dimensions for $\mathbf{W_1}, \mathbf{W_2}$) and mitigate overfitting with regularization.

a) **BP with constant step-size** : derive the equations of BP for this problem (use the matrix form for conciseness), and implement them. Find the best hyper-parameters, λ_1, λ_2, and the stepsize, using cross validation on a help-out test set.

Answer :

Define the **f** function by :

$$f(W_1, W_2) = \frac{1}{N} \sum_{i\in[N]} \|W_2W_1x_i - y_i\|_2^2 + \lambda_1\|\mathbf{W_1}\|_F^2 + \lambda_2\|\mathbf{W_2}\|_F^2$$

$$\Longrightarrow \boxed{f(W_1, W_2) = \frac{1}{N}\|W_2W_1X^T - Y\|_2^2 + \lambda_1\|\mathbf{W_1}\|_F^2 + \lambda_2\|\mathbf{W_2}\|_F^2}$$

The update weight in BP is given in this case by :

$$W_j^{(k+1)} = W_j^{(k)} - \alpha\nabla_{W_j} f_{|W_j=W_j^{(k)}} \qquad j \in \{1, 2\}$$

For j = 2 , we have :

$$\nabla_{W_2} f(W_1, W_2) = \nabla_{W_2} \left(\frac{1}{N} \|W_2 W_1 X^T - Y\|_2^2 + \lambda_1 \|\mathbf{W_1}\|_F^2 + \lambda_2 \|\mathbf{W_2}\|_F^2 \right)$$

$$\Longrightarrow \nabla_{W_2} f(W_1, W_2) = \nabla_{W_2} \left(\frac{1}{N} \|W_2 W_1 X^T - Y\|_2^2 \right) + 2\lambda_2 W_2$$

Let $Z_2 = W_2 W_1 X^T$

$$\nabla_{W_2} f(W_1, W_2) = \nabla_{W_2} \left(\frac{1}{N} \|Z_2 - y_i\|_2^2 \right) + 2\lambda_2 W_2$$

Using chaine rule :

$$\Longrightarrow \nabla_{W_2} f(W_1, W_2) = \nabla_{Z_2} \left(\frac{1}{N} \|Z_2 - Y\|_2^2 \right) \nabla_{W_2} Z_2 + 2\lambda_2 W_2$$

$$\Longrightarrow \nabla_{W_2} f(W_1, W_2) = \nabla_{Z_2} \left(\frac{1}{N} \|Z_2 - Y\|_2^2 \right) \nabla_{W_2} W_2 W_1 X^T + 2\lambda_2 W_2$$

$$\Longrightarrow \boxed{\nabla_{W_2} f(W_1, W_2) = \frac{2}{N} \left(W_2 W_1 X^T - Y \right) \left(W_1 X^T \right)^T + 2\lambda_2 W_2}$$

For j = 1 we have :

$$\nabla_{W_1} f(W_1, W_2) = \nabla_{W_1} \left(\frac{1}{N} \|W_2 W_1 X^T - Y\|_2^2 + \lambda_1 \|\mathbf{W_1}\|_F^2 + \lambda_2 \|\mathbf{W_2}\|_F^2 \right)$$

$$\Longrightarrow \nabla_{W_1} F(W_1, W_2) = \nabla_{W_1} \left(\frac{1}{N} \|W_2 W_1 X^T - Y\|_2^2 \right) + 2\lambda_1 W_1$$

Let $Z_1 = W_1 X^T$ and $Z_2 = W_2 W_1 X^T = W_2 Z_1$ then ,

Using chain rule have :

$$\nabla_{W_1} f(W_1, W_2) = \nabla_{W_1} \left(\frac{1}{N} \|Z_2 - Y\|_2^2 \right) + 2\lambda_1 W_1$$

$$\Longrightarrow \nabla_{W_1} f(W_1, W_2) = \nabla_{Z_2} \left(\frac{1}{N} \|Z_2 - Y\|_2^2 \right) \nabla_{Z_1}(Z_2) \nabla_{W_1} Z_1 + 2\lambda_1 W_1$$

$$\Longrightarrow \nabla_{W_1} f(W_1, W_2) = \nabla_{Z_2} \left(\frac{1}{N} \|Z_2 - Y\|_2^2 \right) \nabla_{Z_1}(W_2 Z_1) \nabla_{W_1}(W_1 X^T) + 2\lambda_1 W_1$$

$$\Longrightarrow \nabla_{W_1} f(W_1, W_2) = \frac{2}{N} W_2^T \left((Z_2 - Y) X \right) + 2\lambda_1 W_1$$

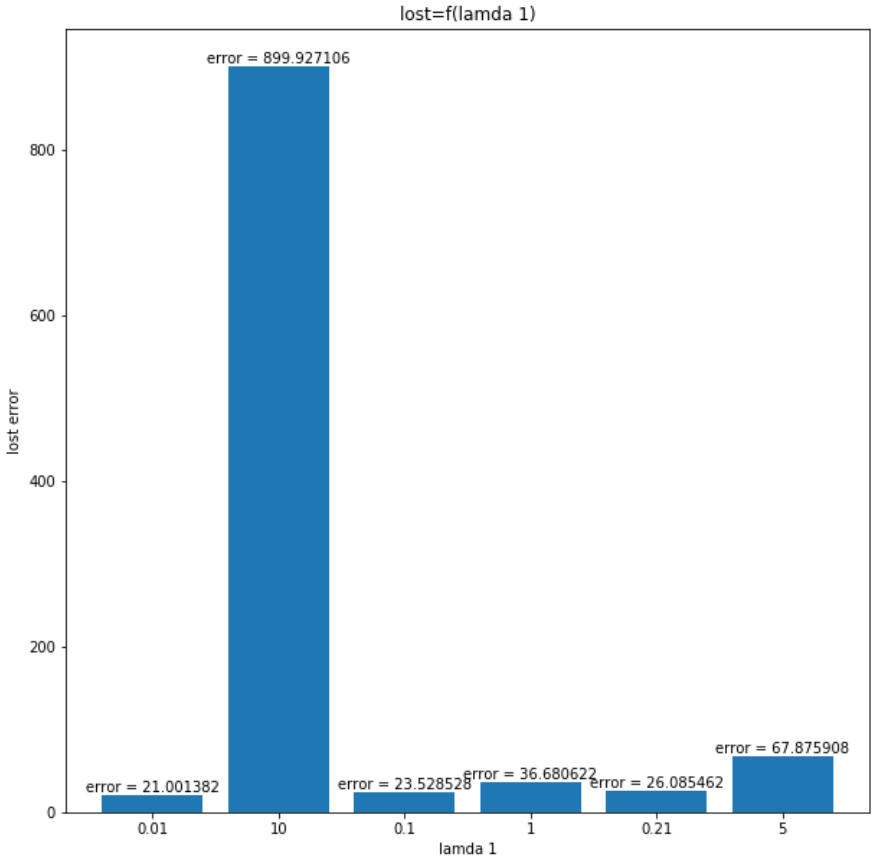
$$\Longrightarrow \boxed{\nabla_{W_1} f(W_1, W_2) = \frac{2}{N} W_2^T \left(\left(W_2 W_1 X^T - Y \right) X \right) + 2\lambda_1 W_1}$$

So we have for BP :

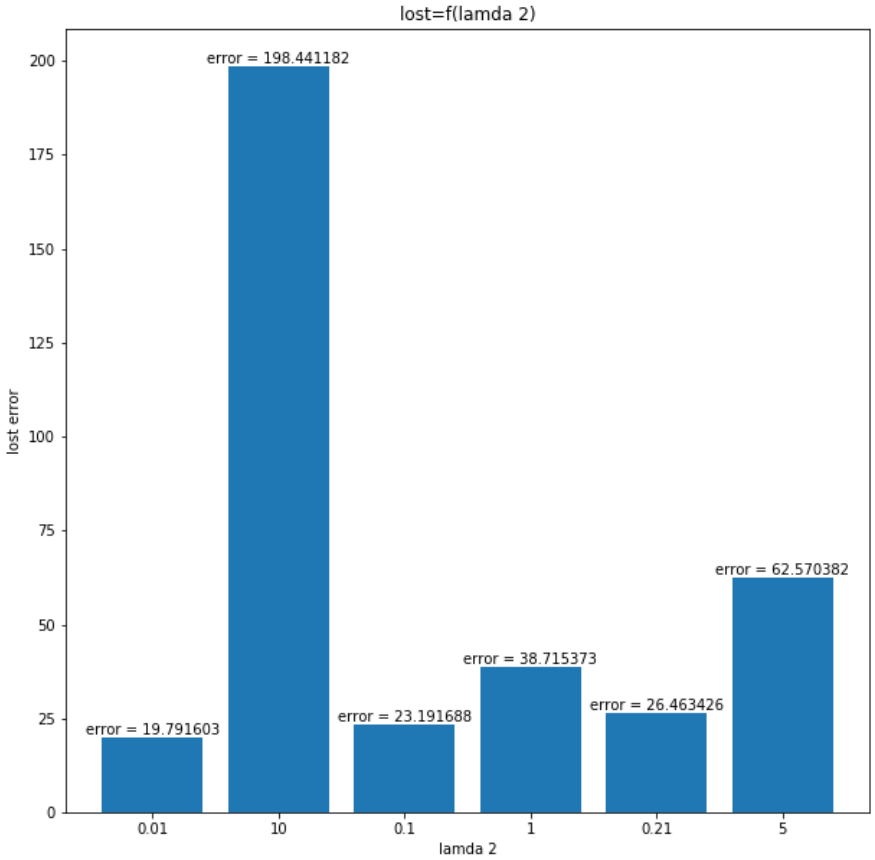
$$\begin{cases} W_1^{(k+1)} = W_1^{(k)} - \alpha \nabla_{W_1} f(W_1^{(k)}, W_2^{(k)}) \\ W_2^{(k+1)} = W_2^{(k)} - \alpha \nabla_{W_2} f(W_1^{(k)}, W_2^{(k)}) \end{cases}$$

$$\boxed{\begin{cases} W_1^{(k+1)} = W_1^{(k)} - \alpha \dfrac{2}{N} W_2^{(k)T} \left(\left(W_2^{(k)} W_1^{(k)} X^T - Y \right) X \right) + 2\alpha\lambda_1 W_1^{(k)} \\ \\ W_2^{(k+1)} = W_2^{(k)} - \alpha \dfrac{2}{N} \left(W_2^{(k)} W_1^{(k)} X^T - Y \right) \left(W_1^{(k)} X^T \right)^T + 2\lambda_2 W_2^{(k)} \end{cases}}$$

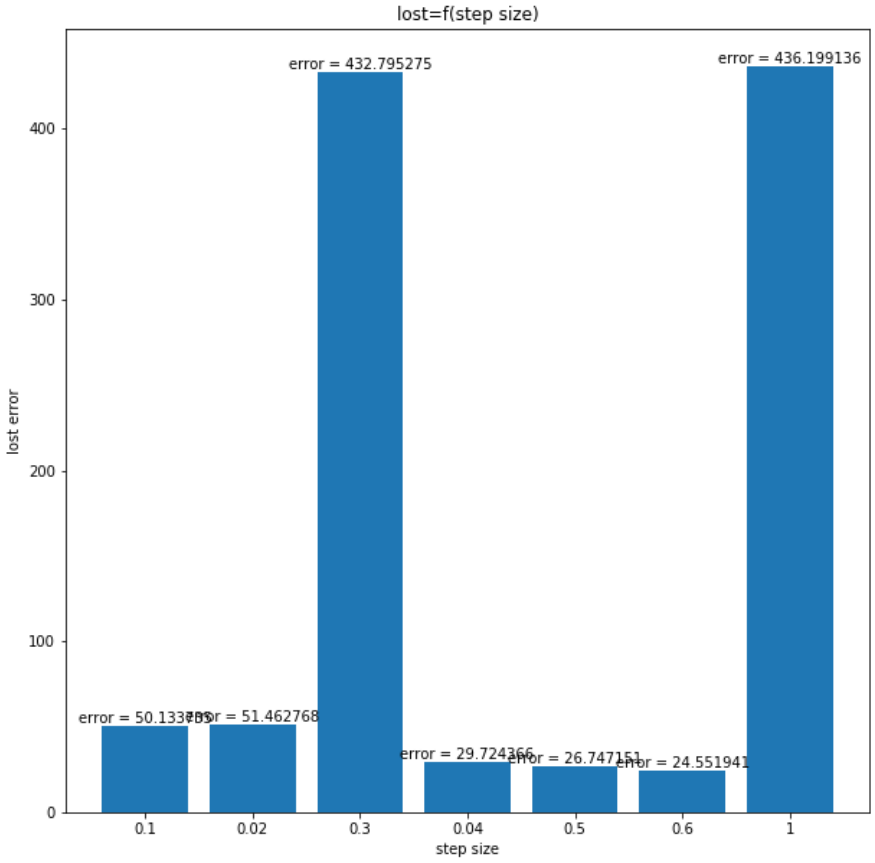
We set the number of neurons in the hidden layer to 10 and we set λ_2 to 0.1 and the step size to 0.01. So by varying λ_1 and using cross validation we get the below histogram. So we can say that the best λ_1 is 0.01.



We set the number of neurons in the hidden layer to 10 and we set λ_1 to 0.1 and the step size to 0.01. So by varying λ_2 and using cross validation we get the below histogram. So we can say that the best λ_2 is 0.01.



We set the number of neurons in the hidden layer to 10 and we set λ_1 to 0.01 and the λ_2 to 0.01. So by varying the step size and using cross validation we get the below histogram. So we can say that the best step size is 0.6.



b) **mini-bath SGD** : For the SGD method, you may use the mini-batch version. Write the update eqts for each layer as a function of the mini-batch and implement these updates.
Find the best hyper-parameters, λ_1, λ_2 , the stepsize, using cross validation on a held-out test set.
Propose your method to select the batch size. Explain your reasoning
Sanity check : set the batch size equal to the training set, and verify (numerically) that you recover a similar behavior as BP (from part a)).

Answer :
The update weight in mini-batch SGD is given in this case by :

$$W_j^{(k+1)} = W_j^{(k)} - \alpha_k \hat{g}_{\beta,j}^{(k)}$$

- where α_k is the stepsize(in here we suppose, for all k, α_k=constant)

- β is the batch of sample it is uniformly at random from[N]

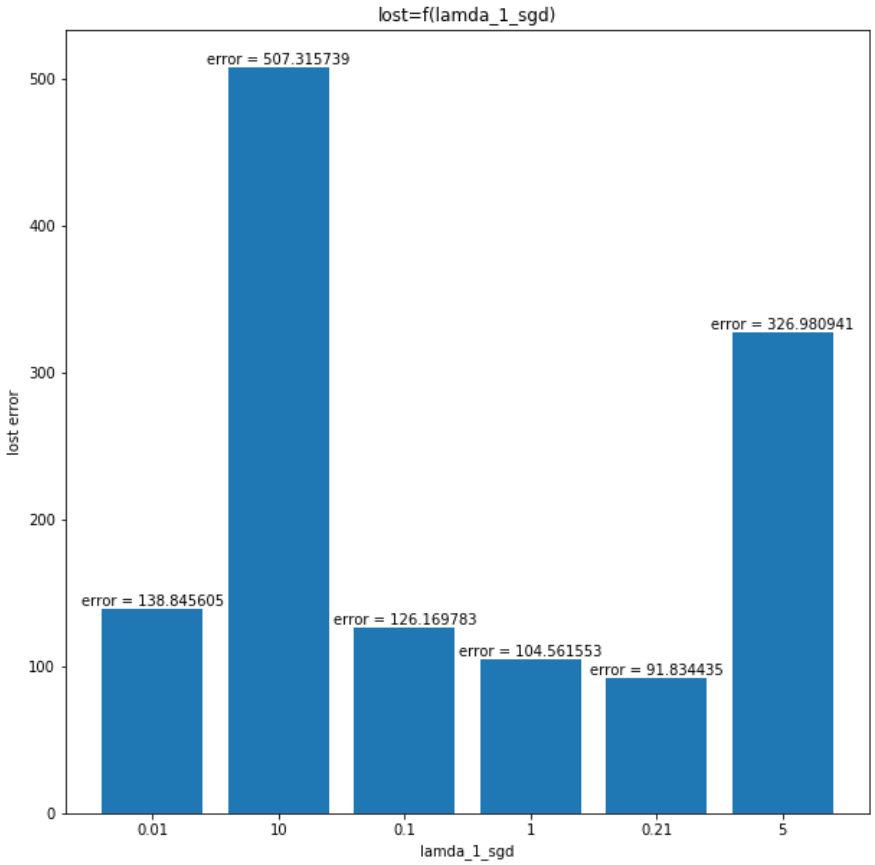
-$\hat{g}_{\beta,j}^{(k)} = \sum_{i\in\beta} \nabla_{W_j} g_i\left(W_1, W_2\right)_{|W_j=W_j^k}$ where $g_i\left(W_1, W_2\right) = \|W_2 W_1 x_i - y_i\|_2^2 + \lambda_1\|W_1\|_F^2 + \lambda_2\|W_2\|_F^2$

So, we have :

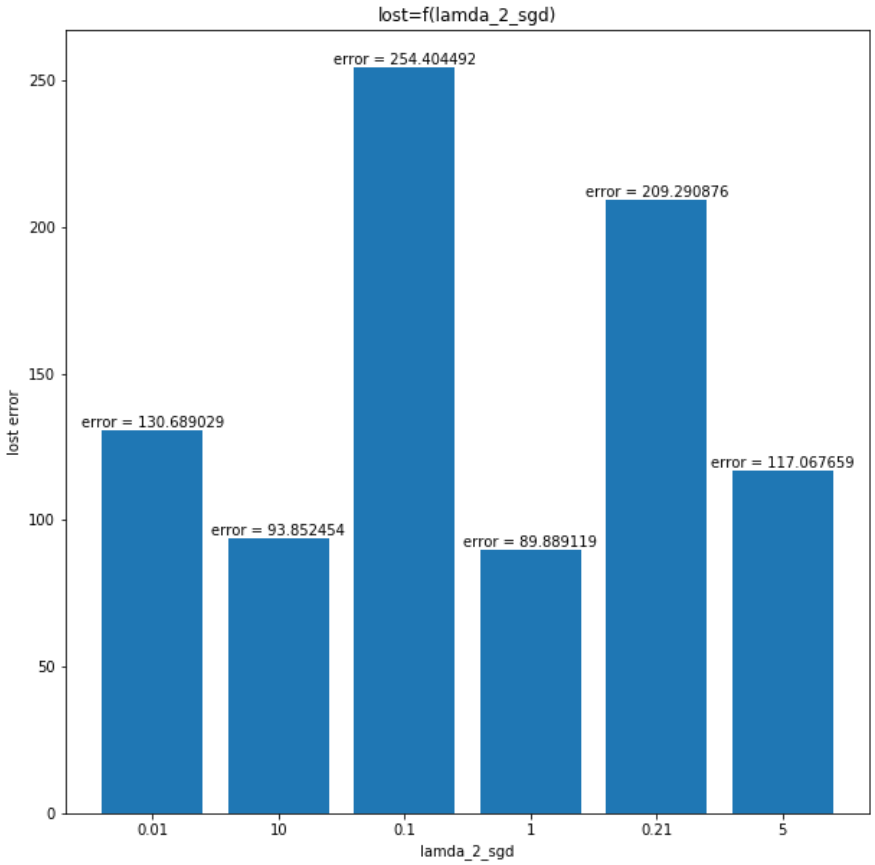
$$\boxed{\hat{g}_{\beta,1}^{(k)} = \sum_{i\in\beta}\left(2W_2^T\left(W_2W_1x_i - y_i\right)x_i^T + 2\lambda_1 W_1\right)}$$

$$\boxed{\hat{g}_{\beta,2}^{(k)} = \sum_{i\in\beta}\left(2\left(W_2W_1x_i - y_i\right)\left(W_1x_i\right)^T + 2\lambda_2 W_2\right)}$$

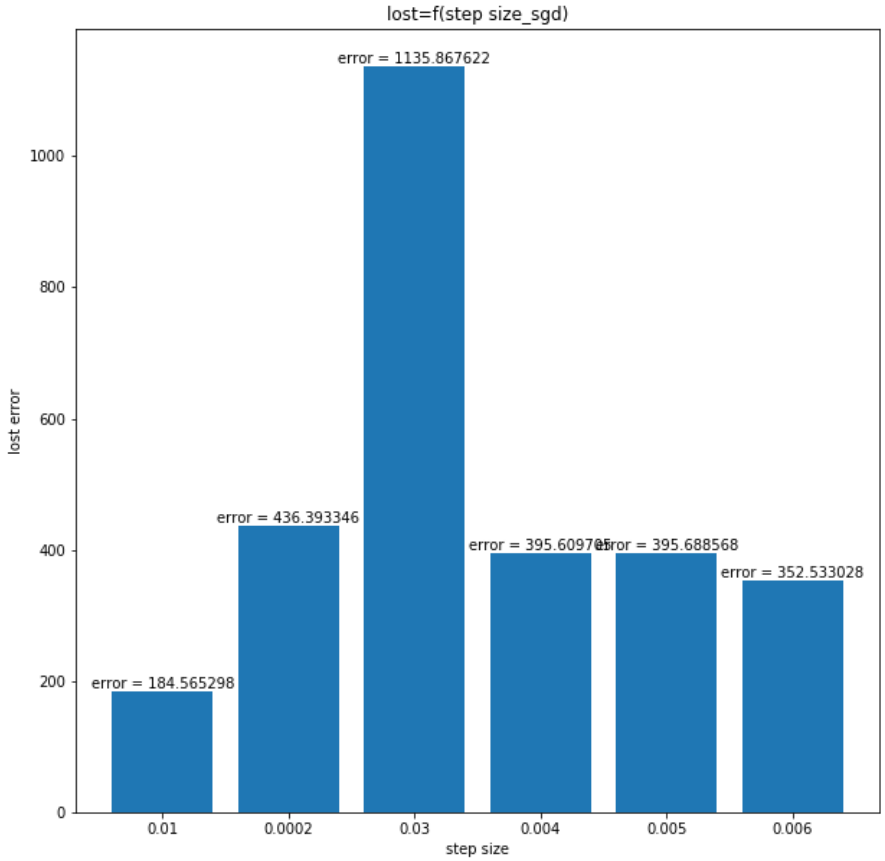
We set the number of neurons in the hidden layer to 10 and we set λ_2 to 0.01,the batch to 50 and the step size to 0.01. So by varying λ_1 and using cross validation we get the below histogram. So we can say that the best λ_1 is 0.21.



We set the number of neurons in the hidden layer to 10 and we set λ_1 to 0.01, the batch size to 50 and the step size to 0.01. So by varying λ_2 and using cross validation we get the below histogram. So we can say that the best λ_2 is 1.



We set the number of neurons in the hidden layer to 10 and we set λ_1 to 0.01 and the λ_2 to 0.01. So by varying the step size and using cross validation we get the below histogram. So we can say that the best step size is is 0.01.



–To get the best batch size, we recommend doing tests by setting all the other parameters (step size, lamda1, lamda2). And to plot the curve of the error according to the size of the batch. Then take the size of the batch which gives the smallest error.

–When the size of the lot is equal to the set of data, the sum of the functions

g i will be equal to N multiplied by the function f, so we get to a constant the same behavior as with the BP.

c) **ADAM** : Refer to algorithm presented in the lecture. Write the update eqts of ADAM for each layer as a function of the exponential decay factors for the first and second order moments, β_1, β_2
. Find the best hyper-parameters, λ_1, λ_2 , the stepsize, and β_1, β_2 using cross validation on a held-out test set..

Answer :
The update weight in ADAM algorithm is given in this case by

$$W_j^{k+1} = W_j^k - \left(\left(\hat{V}_i^{k+1} \right)^{\frac{1}{2}} + \delta 1 \right)^{-1} o \hat{U}_i^{k+1} \qquad j \in \{1, 2\}$$

Where :

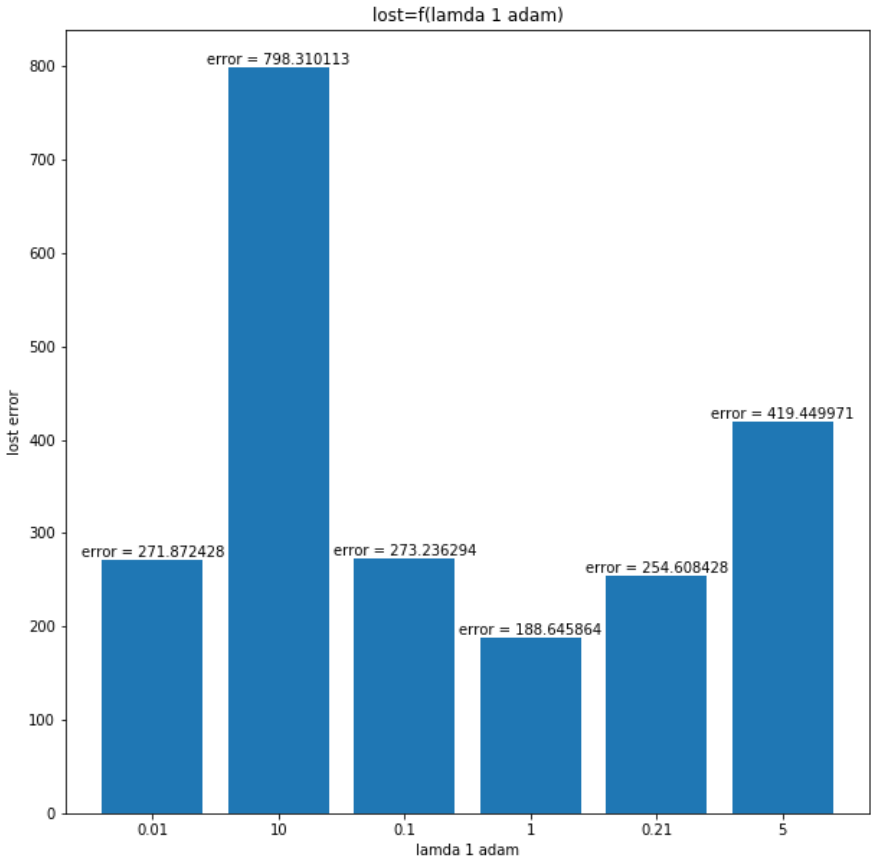
- i is pick uniformely at random from [N]

- $\hat{U}_i^{k+1} = \dfrac{U_i^{k+1}}{1 - \beta_1^k}$ where $U_i^{k+1} = \beta_1 U_i^k + (1 - \beta_1)\hat{g}_i^k$

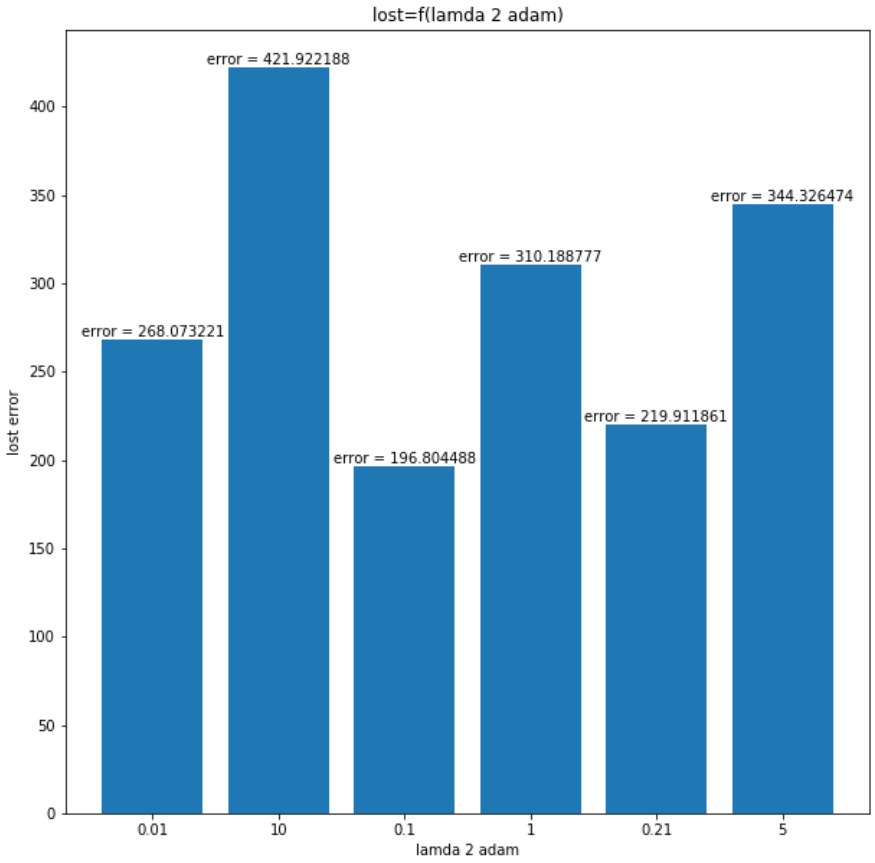
- $\hat{V}_i^{(k+1)} = \dfrac{V_i^{(k+1)}}{1 - \beta_2^{(k)}}$ where $V_i^{(k+1)} = \beta_2 V_i^{(k)} + (1 - \beta_2)\hat{g}_i^{(k)} o \hat{g}_i^{(k)}$

- $\hat{g}_i^{(k)} = \nabla f_i \left(W_j^{(k)} \right)$

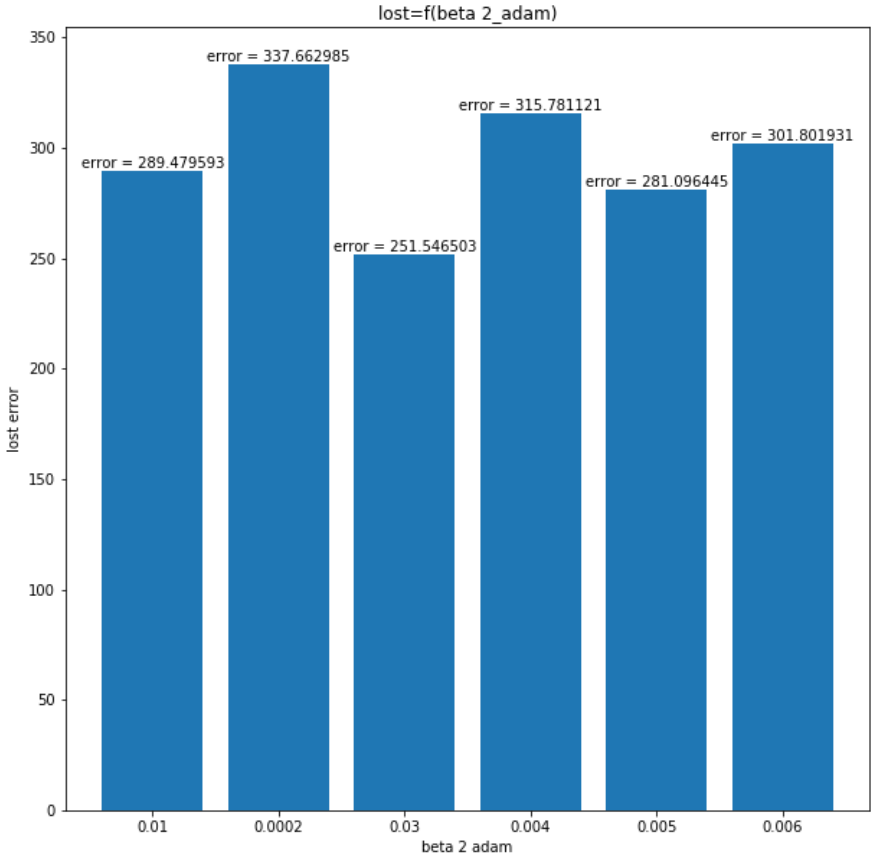
We set the number of neurons in the hidden layer to 10 and we set λ_2 to 0.01, β_1 to 0.01, β_2 to 0.01 and the step size to 0.01. So by varying λ_1 and using cross validation we get the below histogram. So we can say that the best λ_1 is 1.



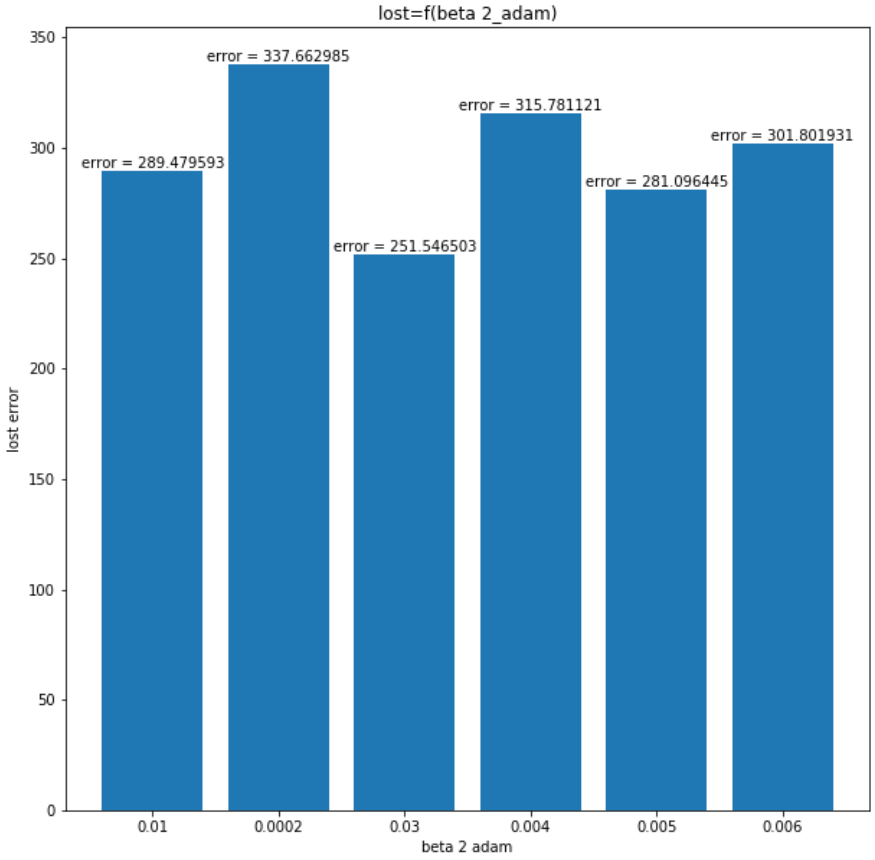
We set the number of neurons in the hidden layer to 10 and we set λ_1 to 0.01, β_1 to 0.01, β_2 to 0.01 and the step size to 0.01. So by varying λ_2 and using cross validation we get the below histogram. So we can say that the best λ_2 is 0.1.



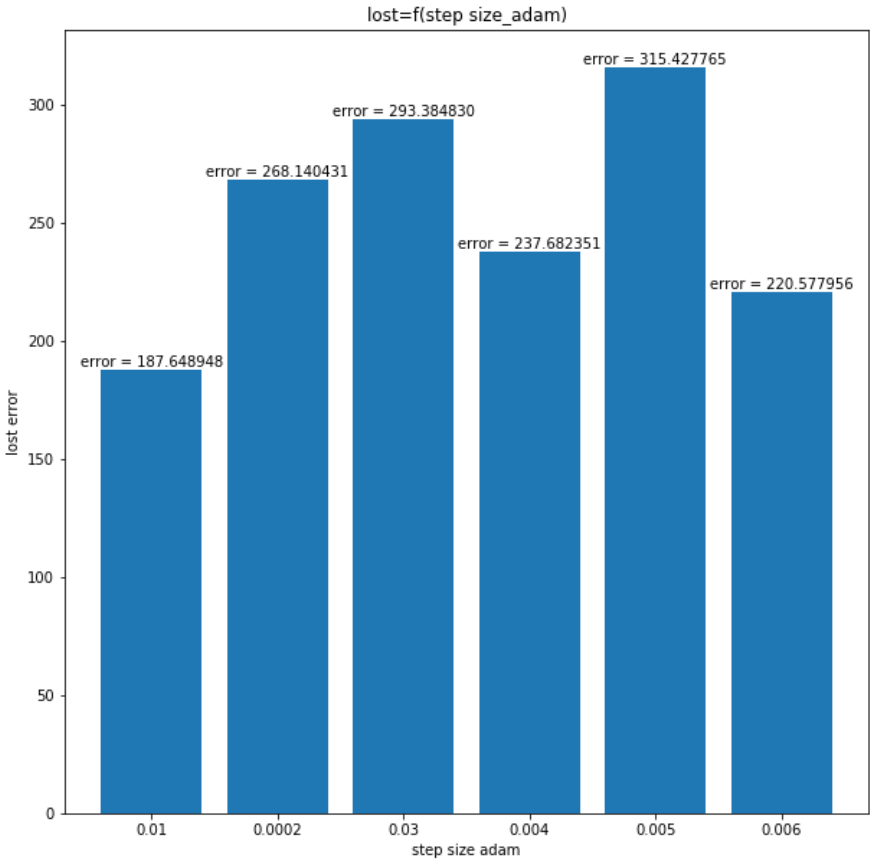
We set the number of neurons in the hidden layer to 10 and we set λ_1 to 0.01, λ_2 to 0.01, β_2 to 0.01 and the step size to 0.01. So by varying β_1 and using cross validation we get the below histogram. So we can say that the best β_1 is 0.0002.



We set the number of neurons in the hidden layer to 10 and we set λ_1 to 0.01, λ_2 to 0.01, β_1 to 0.01 and the step size to 0.01. So by varying β_2 and using cross validation we get the below histogram. So we can say that the best β_2 is 0.03.



We set the number of neurons in the hidden layer to 10 and we set λ_1 to 0.01, λ_2 to 0.01, β_1 to 0.01 and the β_1 to 0.01. So by varying the step size and using cross validation we get the below histogram. So we can say that the best step size is 0.01.



d) **BCD** : Derive each BCD subproblem, and the resulting update eqts for each layer, in closed (matrix) form. You may assume for this part that $\mathbf{W_1}$ is a tall matrix, and $\mathbf{W_2}$ is fat. Implement the resulting algorithm. Find the best hyper-parameters, λ_1, λ_2 , using cross validation on a held-out test set.

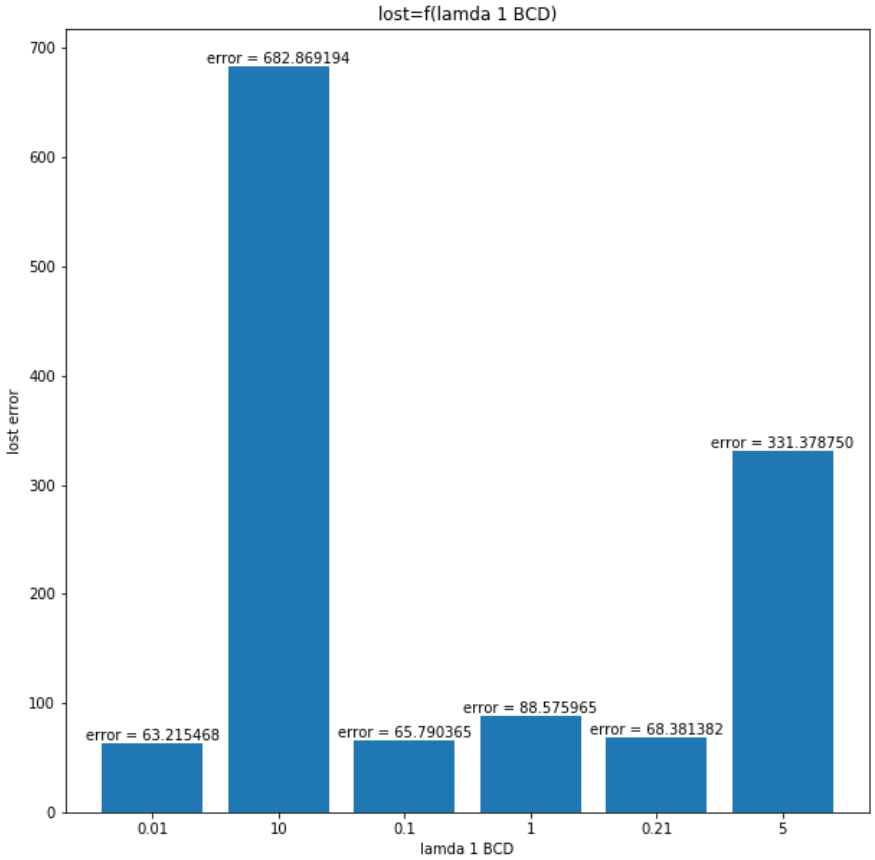
Answer :

The update weight in BCD algorithm is given in this case by

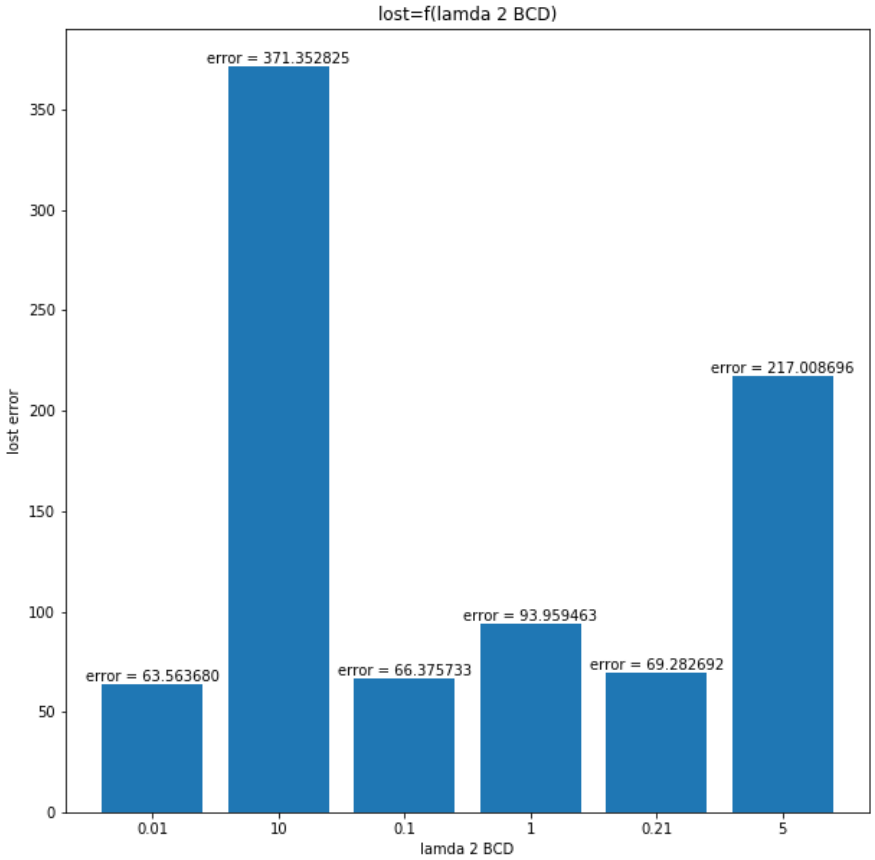
$$W_1^{(k+1)} = W_1^{(k)} - \alpha \nabla_{W_1} f_{|_{W_1 = W_1^{(k)}}} \left(W_1^{(k)}, W_2^{(k)} \right)$$

$$W_2^{(k+1)} = W_2^{(k)} - \alpha \nabla_{W_2} f_{|_{W_2 = W_2^{(k)}}} \left(W_1^{(k+1)}, W_2^{(k)} \right)$$

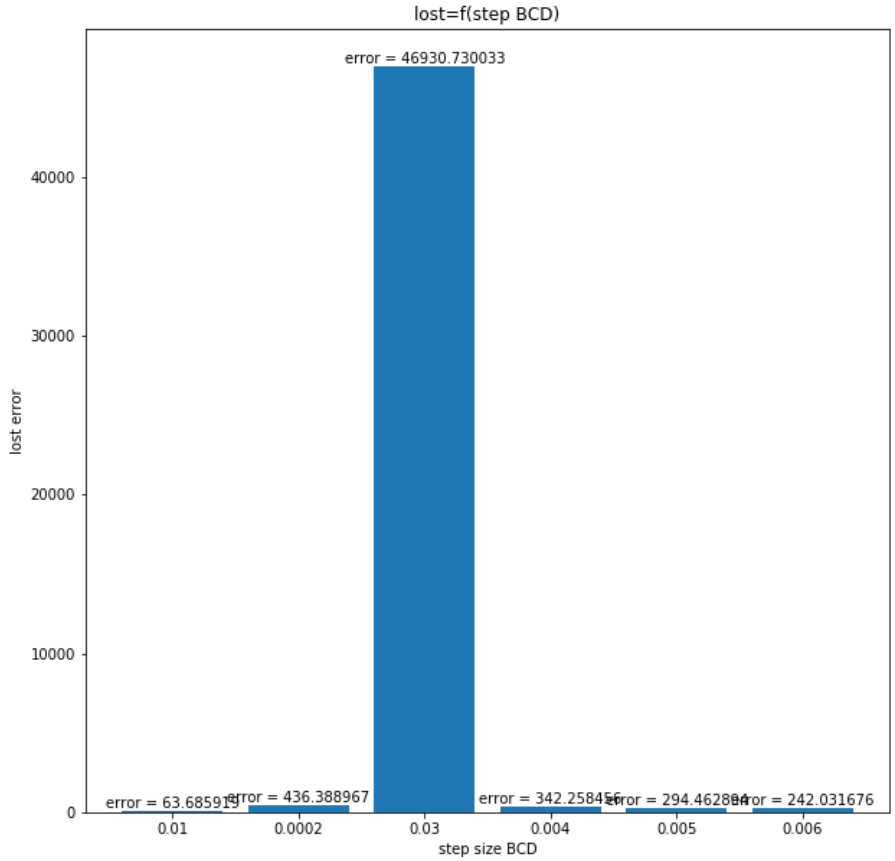
We set the number of neurons in the hidden layer to 10 and we set λ_2 and the step size to 0.01. So by varying λ_1 and using cross validation we get the below histogram. So we can say that the best λ_1 is 0.01.



We set the number of neurons in the hidden layer to 10 and we set λ_1 to 0.01, and the step size to 0.01. So by varying λ_2 and using cross validation we get the below histogram. So we can say that the best λ_2 is 0.01.



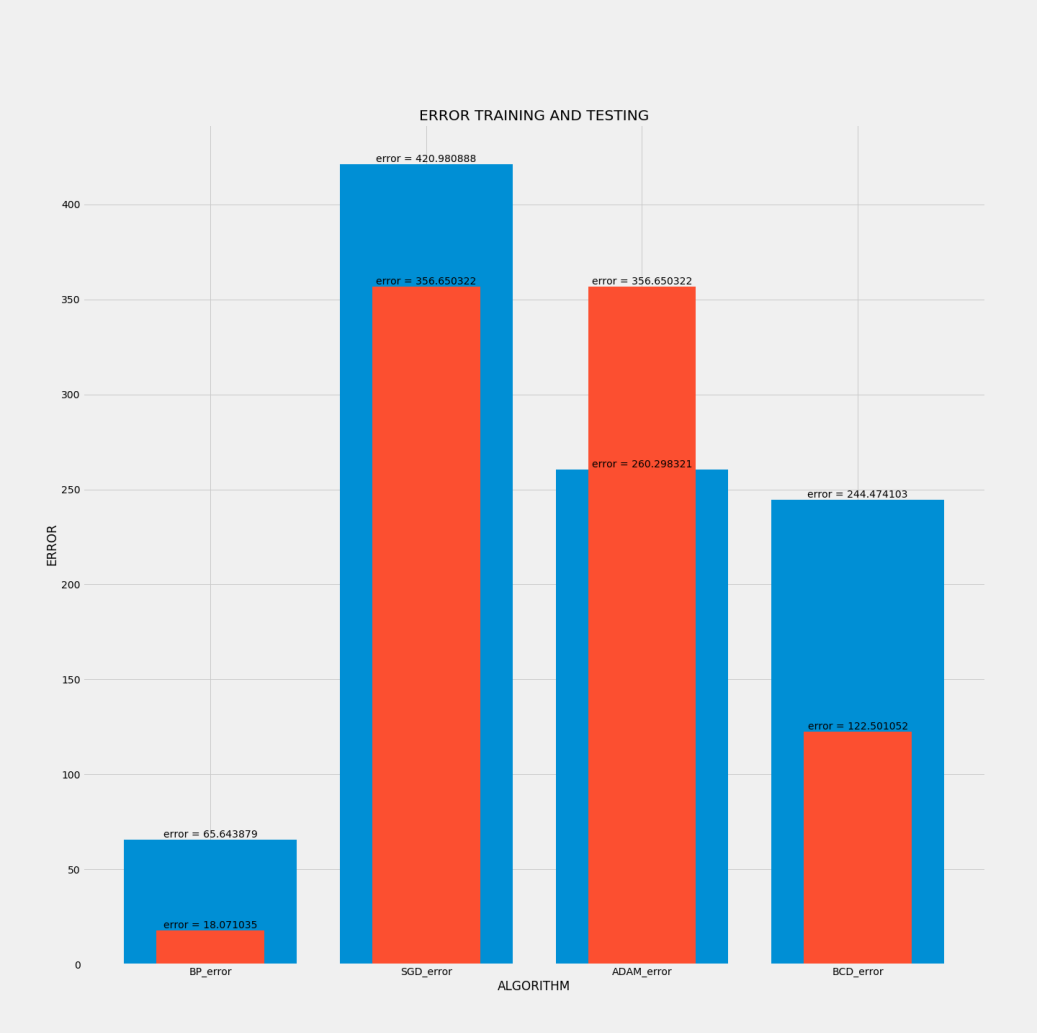
We set the number of neurons in the hidden layer to 10 and we set λ_1 to 0.01, and the λ_2 to 0.01. So by varying the step size and using cross validation we get the below histogram. So we can say that the best step size is 0.01.



e) Compare the training error and test error for all these methods, after convergence is reached (with an appropriate choice of convergence criteria for each). Then computing their test MSE on a held-out set for cross validation.

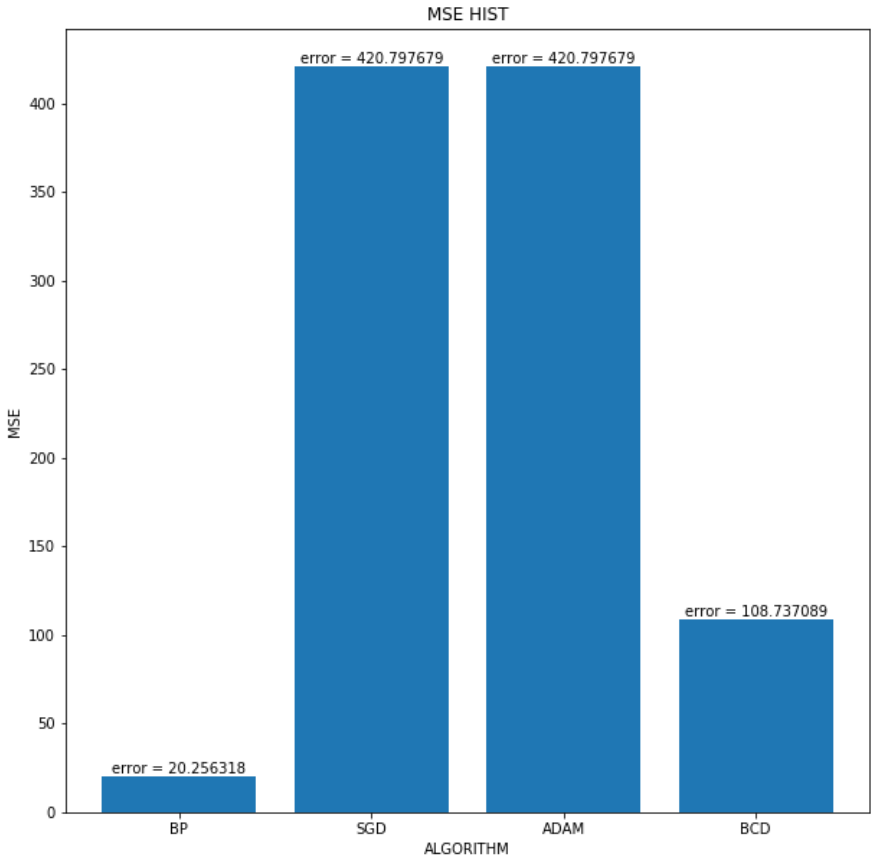
Answer :

In this figure, we can see in red the average error obtained on the test data



and in blue the average error obtained on the data. It is thus noted that the error on the training data is greater than the error on the test data on all

the algorithms except for the ADAM algorithm. This may be justified by the fact that the adam algorithm is not optimal for our data. We can see in



this figure that the mse obtained with the BP algorithm is the smallest so we deduce that this algorithm is more optimal for our data. Theoretically, this can also be justified by the fact that the other algorithms are only an approximation of the BP. It is therefore normal that this algorithm is the best.

f) Compare these methods in terms of computational complexity, complexity of hyper- parameter tuning, and the # of outer-loop iterations.

Answer :

→ computational complexity :

All the methods implemented in this part require calculating the gradients of the elementary loss functions g_i. Once the gradient of the functions g_i has been calculated, we have :

* For the BP, we then calculate the gradient of the loss function f which is equal to the sum of the gradients of the functions g_i applied to the data set. Once done, we move on to updating the weight..

* The mini-batch sgd is similar to the BP except that this time the gradient used to update the weights is equal to the sum of the gradients of the functions g_i evaluated on a part (batch) of the data.

* For the ADAM algorithm, it is necessary to calculate in addition to the gradients of the functions g_i, the elementary functions U_i and V_i at each iteration on a single data.

* For the BCD algorithm, we need to evaluate two the gradient of the loss function f over the set of datasets at each iteration.

Thus, we can say that in terms of computational complexity, the best algorithm is the SGD minibatch, followed by the BP. Then we have the BCD and finally we have the ADAM algorithm.

→ Number of parameters :

* In terms of setting the number of parameters, BP and BCD algorithms require the same number of parameters, i.e. 3.

* For the SGD mini-batch, in addition to the 3 parameters mentioned above, the batch size must also be defined, which brings the number of parameters to 4.

* For the ADAM algorithm, in addition to the 3 parameters, the parameters beta1 and beta2 must be added which brings the number of parameters to 5.

Thus, we can say that the best algorithms in terms of parameterization are the BP and BCD algorithms followed by the SGD mini-batch algorithm and finally by the ADAM algorithm.

→ Number of iterations :

For the number of iterations, the BP algorithm required in our implementation the least iteration with 100 iterations then we have the SGD algorithm with 115 iterations, then the BCD algorithm with 125 iterations and finally the ADAM algorithm with 200 iterations.

g) How do the derivations of BP change, when another loss function is used, e.g. cross-entropy ? Find an efficient way to write/derive equations of BP, when using the cross entropy loss function. This observation is a significant advantage of BP.

Answer :

Hints : Notice that you will to compute derivative w.r.t a matrix. You may consult this guide on matrix differentiation : https://www.math.uwaterloo.ca/hwolkowi/matrixcookbook.pdf.
You will need to use the invariance of the trace with respect to circular permutation. For matrices of appropriate dimensions, $\mathbf{A}, \mathbf{B}, \mathbf{C}$ the following holds :

$$tr(\mathbf{ABC}) = tr(\mathbf{CAB}) = tr(\mathbf{BCA})$$

Part II : deep neural network

Let us now add the following logistic activation, where $\sigma(x) = 1/(1+exp(-x))$ at each layer. Recall that, the function s is applied in an element-by-element manner on its input argument. Find an appropriate normalization of the training set, to take into account the fact that the logistic activation 'compresses' its output between [0, 1]. Thus, the optimization problem considered here is the following :

$$\min_{\mathbf{W_1},\mathbf{W_2}} \frac{1}{N} \sum_{i\in[N]} \|\sigma(\mathbf{W_2}\sigma(\mathbf{W_1}x_i)) - y_i\|_2^2 + \lambda_1\|\mathbf{W_1}\|_F^2 + \lambda_2\|\mathbf{W_2}\|_F^2$$

General :

Define the f function by :

$$f(W_1, W_2) = \frac{1}{N} \sum_{i \in [N]} \|\sigma(W_2 \sigma(W_1 x_i)) - y_i\|_2^2 + \lambda_1 \|W_1\|_F^2 + \lambda_2 \|W_2\|_F^2$$

so we have :

$$\Longrightarrow \boxed{f(W_1, W_2) = \frac{1}{N} \|\sigma(W_2 \sigma(W_1 X^T)) - Y\|_2^2 + \lambda_1 \|W_1\|_F^2 + \lambda_2 \|W_2\|_F^2}$$

In this part, we will normalize the target values. Such that the target values are between 0 and 1.
For this, we divide all the values by the maximum values of the target

a) Redo the questions from Part I-a) to Part I-f)
What is the limitation in applying the BCD method (in Part I-d)) to optimization problem (2) ?.

Answer :

- **BP with constant step-size** :

  The update weight in BP is given by :

  $$W_j^{(k+1)} = W_j^{(k)} - \alpha \nabla_{W_j} f_{|W_j = W_j^{(k)}} \qquad j \in \{1, 2\}$$

  For j = 2 , we have :

  $$\nabla_{W_2} f(W_1, W_2) = \nabla_{W_2} \left(\frac{1}{N} \|\sigma(W_2 \sigma(W_1 X^T)) - Y\|_2^2 + \lambda_1 \|\mathbf{W_1}\|_F^2 + \lambda_2 \|\mathbf{W_2}\|_F^2 \right)$$

  $$\Longrightarrow \nabla_{W_2} f(W_1, W_2) = \nabla_{W_2} \left(\frac{1}{N} \|\sigma(W_2 \sigma(W_1 X^T)) - Y\|_2^2 \right) + 2\lambda_2 W_2$$

  Let $Z_2 = \sigma(W_2 \sigma(W_1 X^T))$

  $$\nabla_{W_2} f(W_1, W_2) = \nabla_{W_2} \left(\frac{1}{N} \|Z_2 - y_i\|_2^2 \right) + 2\lambda_2 W_2$$

Using chaine rule :

$$\Longrightarrow \nabla_{W_2} f(W_1, W_2) = \nabla_{Z_2} \left(\frac{1}{N} \|Z_2 - Y\|_2^2 \right) o \nabla_{W_2} Z_2 + 2\lambda_2 W_2$$

$$\Longrightarrow \nabla_{W_2} f(W_1, W_2) = \nabla_{Z_2} \left(\frac{1}{N} \|Z_2 - Y\|_2^2 \right) o \nabla_{W_2} \sigma(W_2 \sigma(W_1 X^T)) + 2\lambda_2 W_2$$

so we have :

$$\boxed{\nabla_{W_2} f(W_1, W_2) = \frac{2}{N} \left(\sigma(W_2 \sigma(W_1 X^T)) - Y \right) o \nabla \sigma(W_2 \sigma(W_1 X^T)) \left(\sigma(W_1 X^T) \right)^T + 2\lambda_2 W_2}$$

For j = 1 we have :

$$\nabla_{W_1} f(W_1, W_2) = \nabla_{W_1} \left(\frac{1}{N} \|\sigma(W_2 \sigma(W_1 X^T)) - Y\|_2^2 + \lambda_1 \|\mathbf{W_1}\|_F^2 + \lambda_2 \|\mathbf{W_2}\|_F^2 \right)$$

$$\Longrightarrow \nabla_{W_1} F(W_1, W_2) = \nabla_{W_1} \left(\frac{1}{N} \|\sigma(W_2 \sigma(W_1 X^T)) - Y\|_2^2 \right) + 2\lambda_1 W_1$$

Let $Z_1 = \sigma(W_1 X^T)$ and $Z_2 = \sigma(W_2 \sigma(W_1 X^T)) = \sigma(W_2 Z_1)$ then ,

Using chain rule have :

$$\nabla_{W_1} f(W_1, W_2) = \nabla_{W_1} \left(\frac{1}{N} \|Z_2 - Y\|_2^2 \right) + 2\lambda_1 W_1$$

$$\Longrightarrow \nabla_{W_1} f(W_1, W_2) = \nabla_{Z_2} \left(\frac{1}{N} \|Z_2 - Y\|_2^2 \right) \nabla_{Z_1} Z_2 \nabla_{W_1} Z_1 + 2\lambda_1 W_1$$

$$\Longrightarrow \nabla_{W_1} f(W_1, W_2) = \nabla_{Z_2} \left(\frac{1}{N} \|Z_2 - Y\|_2^2 \right) \nabla_{Z_1} (\sigma(W_2 Z_1)) \nabla_{W_1} (\sigma(W_1 X^T)) +$$
$2\lambda_1 W_1$

$$\Longrightarrow \nabla_{W_1} f(W_1, W_2) = \frac{2}{N} W_2^T \left((Z_2 - Y) \, o \nabla \sigma(W_2 Z_1) \nabla \sigma(W_1 X^T) X^T \right) +$$
$2\lambda_1 W_1$

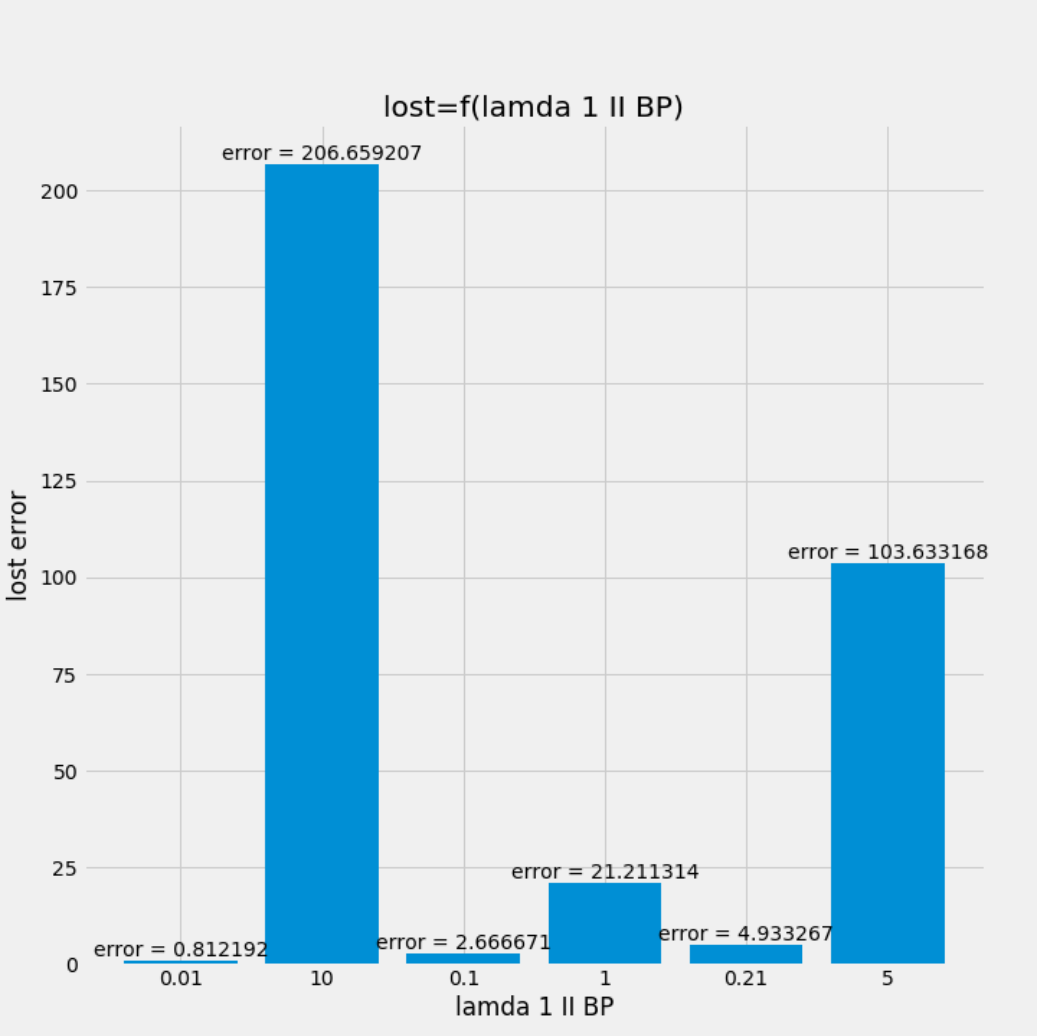
so we have :

$$\boxed{\nabla_{W_1} f = \frac{2}{N} W_2^T \left(\left(\sigma(W_2 \sigma(W_1 X^T)) - Y \right) o \nabla \sigma(W_2 \sigma(W_1 X^T)) o \nabla \sigma(W_1 X^T) X \right) + 2\lambda_1 W_1}$$

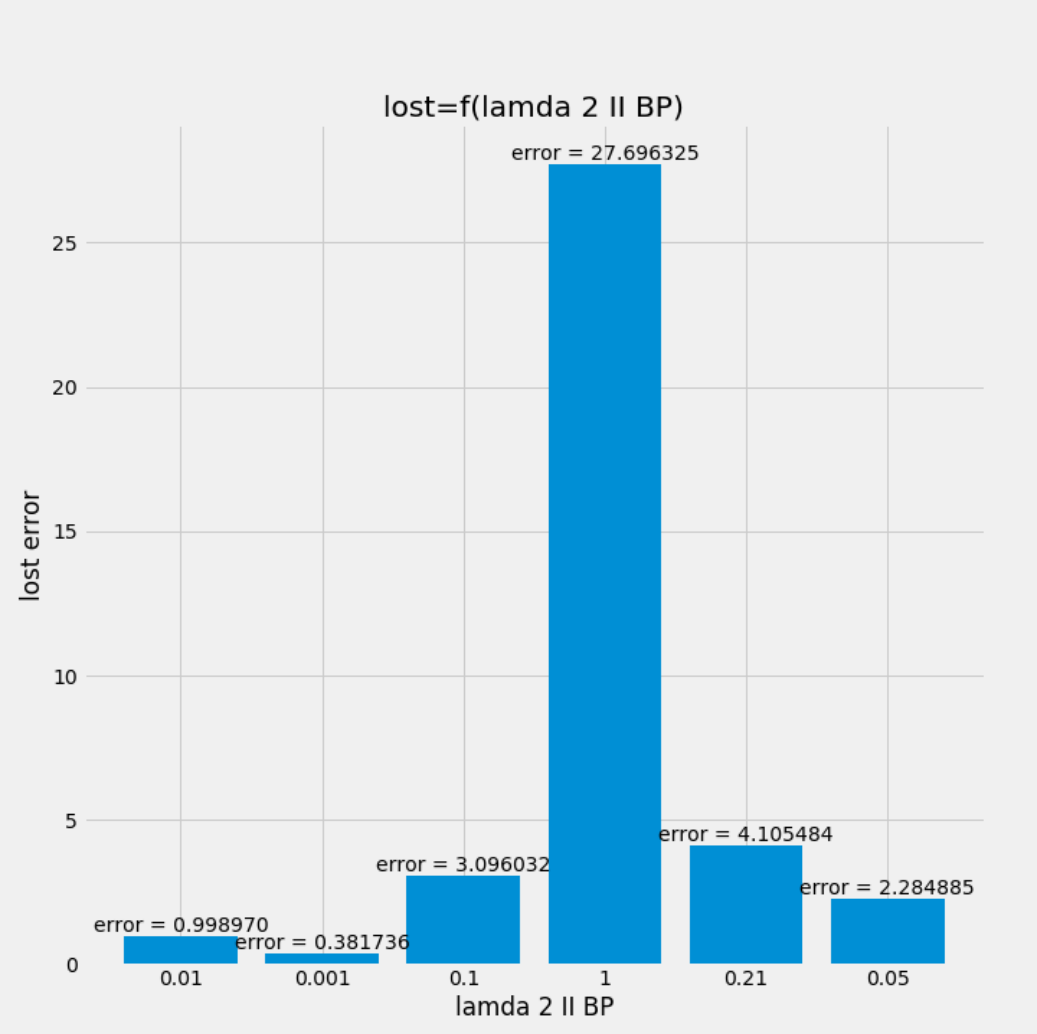
So we have for BP :

$$\left\{ \begin{array}{l} W_1^{(k+1)} = W_1^{(k)} - \alpha \nabla_{W_1} f(W_1^{(k)}, W_2^{(k)}) \\ W_2^{(k+1)} = W_2^{(k)} - \alpha \nabla_{W_2} f(W_1^{(k)}, W_2^{(k)}) \end{array} \right\}$$

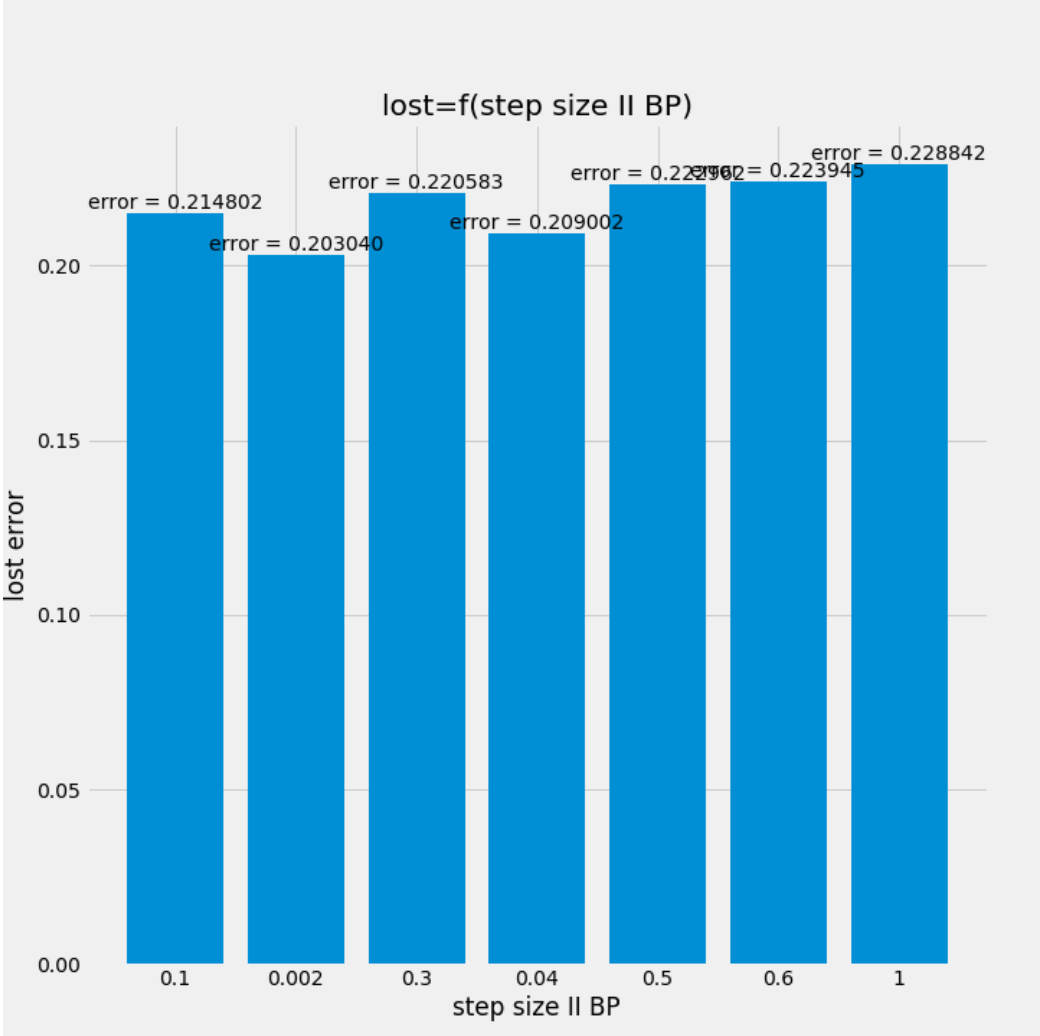
We set the number of neurons in the hidden layer to 5 and we set λ_2 to 0.1, and the step size to 1. So by varying λ_1 and using cross validation we get the below histogram. So we can say that the best λ_1 is 0.01.



We set the number of neurons in the hidden layer to 5 and we set λ_1 to 0.1, and the step size to 1. So by varying λ_2 and using cross validation we get the below histogram. So we can say that the best λ_1 is 0.001.



We set the number of neurons in the hidden layer to 5 and we set λ_1 to 0.1, and λ_2 to 0.001. So by varying step size and using cross validation we get the below histogram. So we can say that the best step size is 0.002.



- **mini-bath SGD** :

The update weight in mini-batch SGD is given in this case by :

$$W_j^{k+1} = W_j^k - \alpha_k \hat{g}_{\beta,j}^{(k)}$$

- where α_k is the stepsize(in here we suppose, for all k, α_k=constant)

- β is the batch of sample it is uniformly at random in [N]

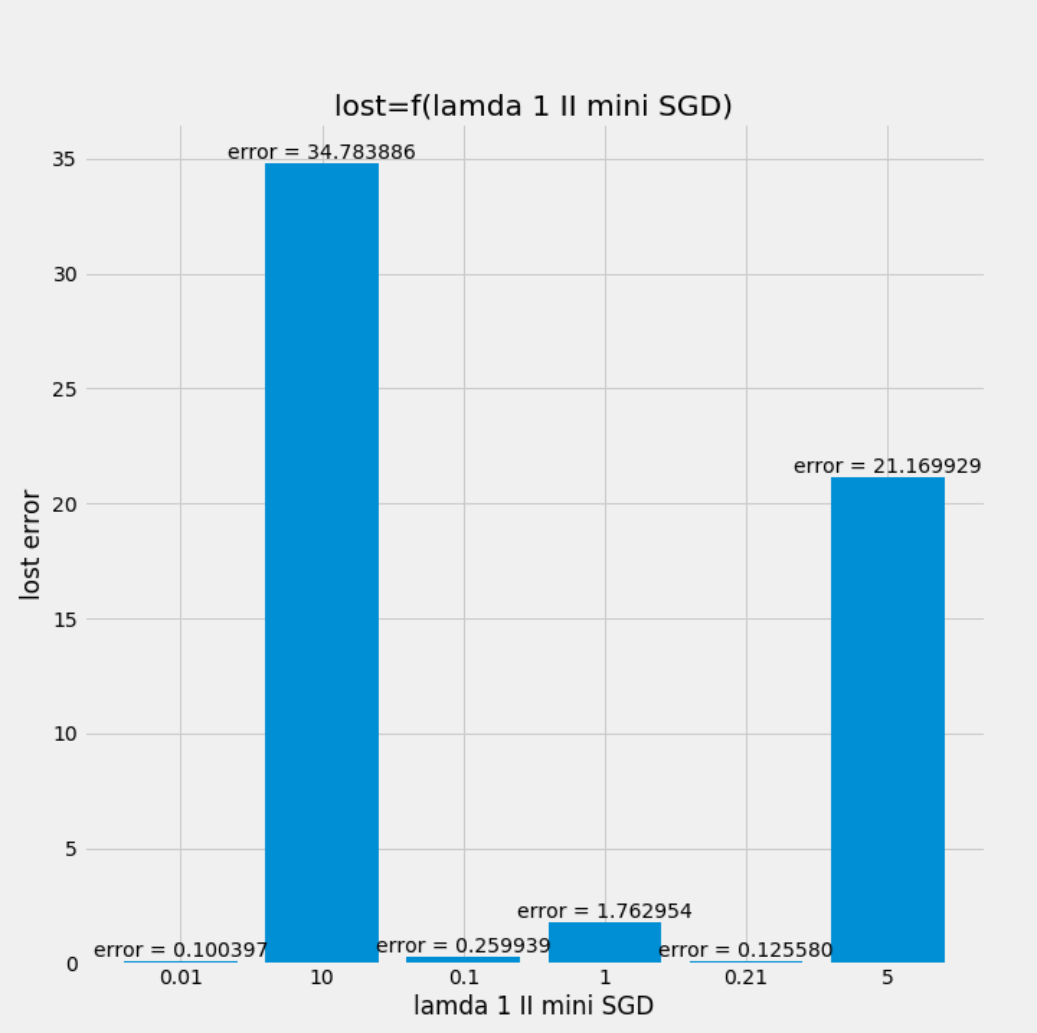
-$\hat{g}_{\beta,j}^{(k)} = \sum_{i\in\beta} \nabla_{W_j} g_i\left(W_1, W_2\right)_{|W_j=W_j^k}$ where $g_i\left(W_1, W_2\right) = \|\sigma(W_2\sigma(W_1 x_i)) - y_i\|_2^2 + \lambda_1\|W_1\|_F^2 + \lambda_2\|W_2\|_F^2$

So, we have :

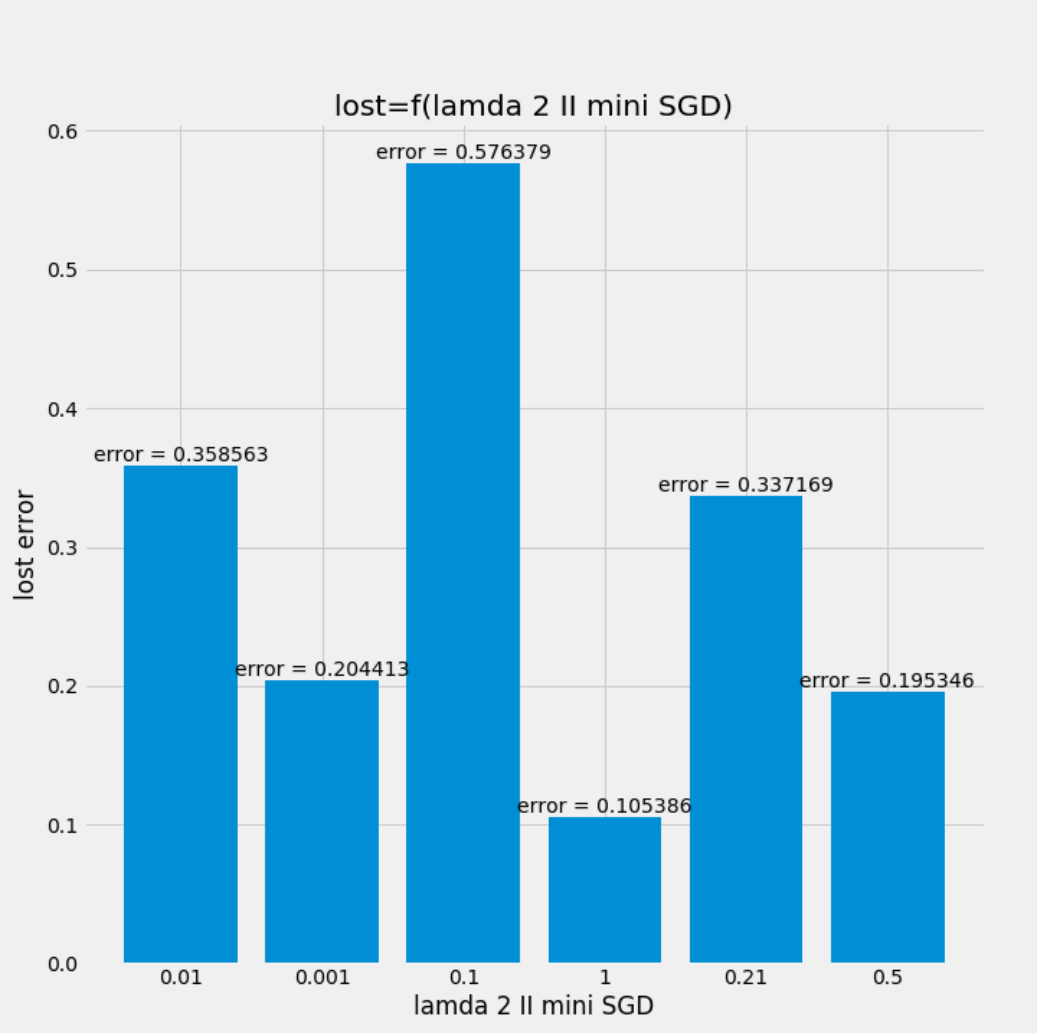
$$\boxed{\hat{g}_{\beta,2}^{(k)} = \sum_{i\in\beta}\left(2\left(\sigma(W_2\sigma(W_1 x_i)) - y_i\right) o \nabla\sigma(W_2\sigma(W_1 x_i))(\sigma(W_1 x_i))^T + 2\lambda W_2\right)}$$

$$\boxed{\hat{g}_{\beta,1}^{(k)} = \sum_{i\in\beta}(2W_2^T\left(\left(\sigma(W_2\sigma(W_1 x_i)) - y_i\right) o \left(\nabla\sigma(W_2\sigma(W_1 x_i))\right) o \nabla\sigma(W_1 x_i) x_i^T\right) + 2\lambda_1 W_1)}$$

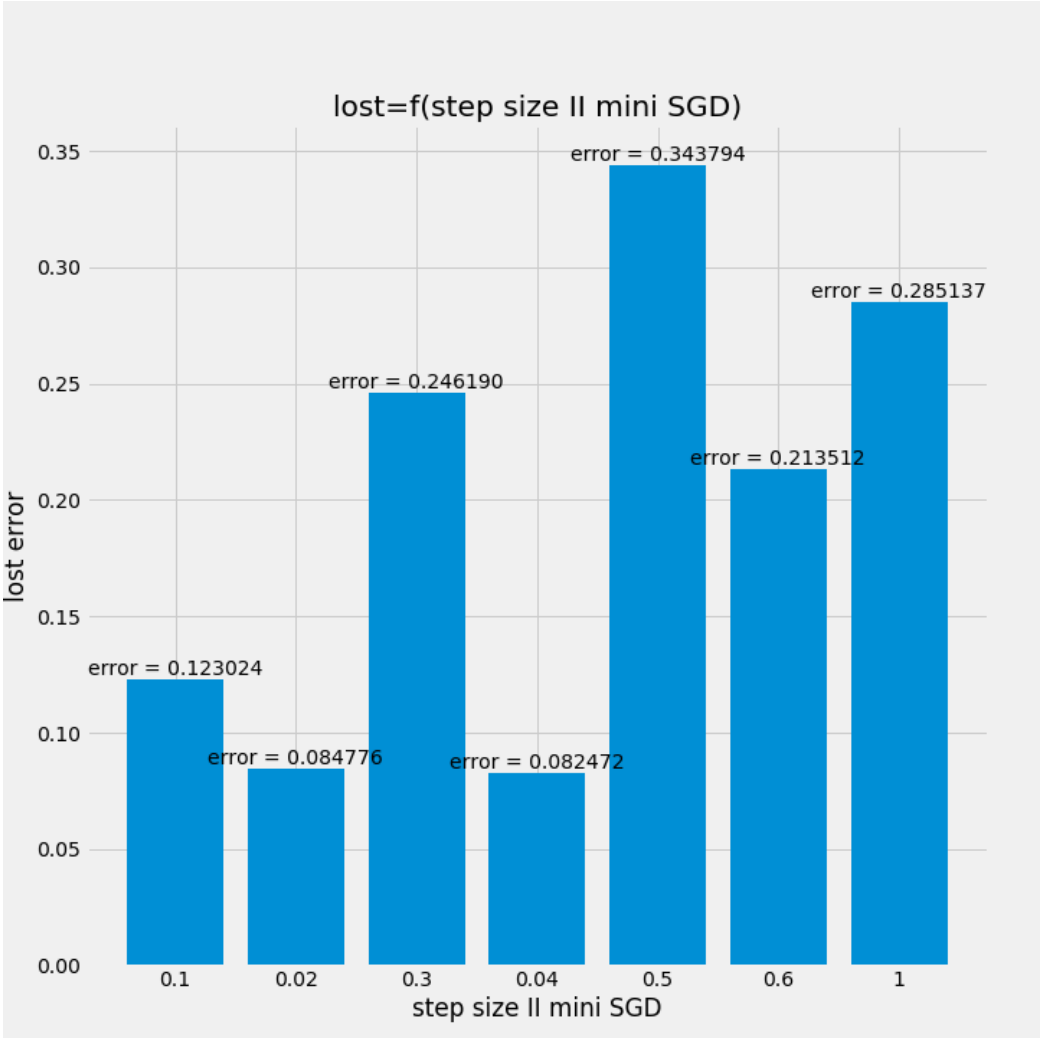
We set the number of neurons in the hidden layer to 5 and we set λ_2 to 0.1, the batch size to 30 and the step size to 1. So by varying λ_1 and using cross validation we get the below histogram. So we can say that the best λ_1 is 0.01.



We set the number of neurons in the hidden layer to 5 and we set λ_1 to 0.1, the batch size to 30 and the step size to 1. So by varying λ_2 and using cross validation we get the below histogram. So we can say that the best λ_2 is 1.



We set the number of neurons in the hidden layer to 5 and we set λ_1 to 0.1, the batch size to 30 and the λ_2 to 0.01. So by varying the step size and using cross validation we get the below histogram. So we can say that the best step size is 0.02.



- **ADAM** :

The update weight in ADAM algorithm is given in this case by

$$W_j^{k+1} = W_j^k - \left(\left(\hat{V}_i^{k+1} \right)^{\frac{1}{2}} + \delta 1 \right)^{-1} o \hat{U}_i^{k+1} \qquad j \in \{1, 2\}$$

Where :

- i is pick uniformely at random from [N]

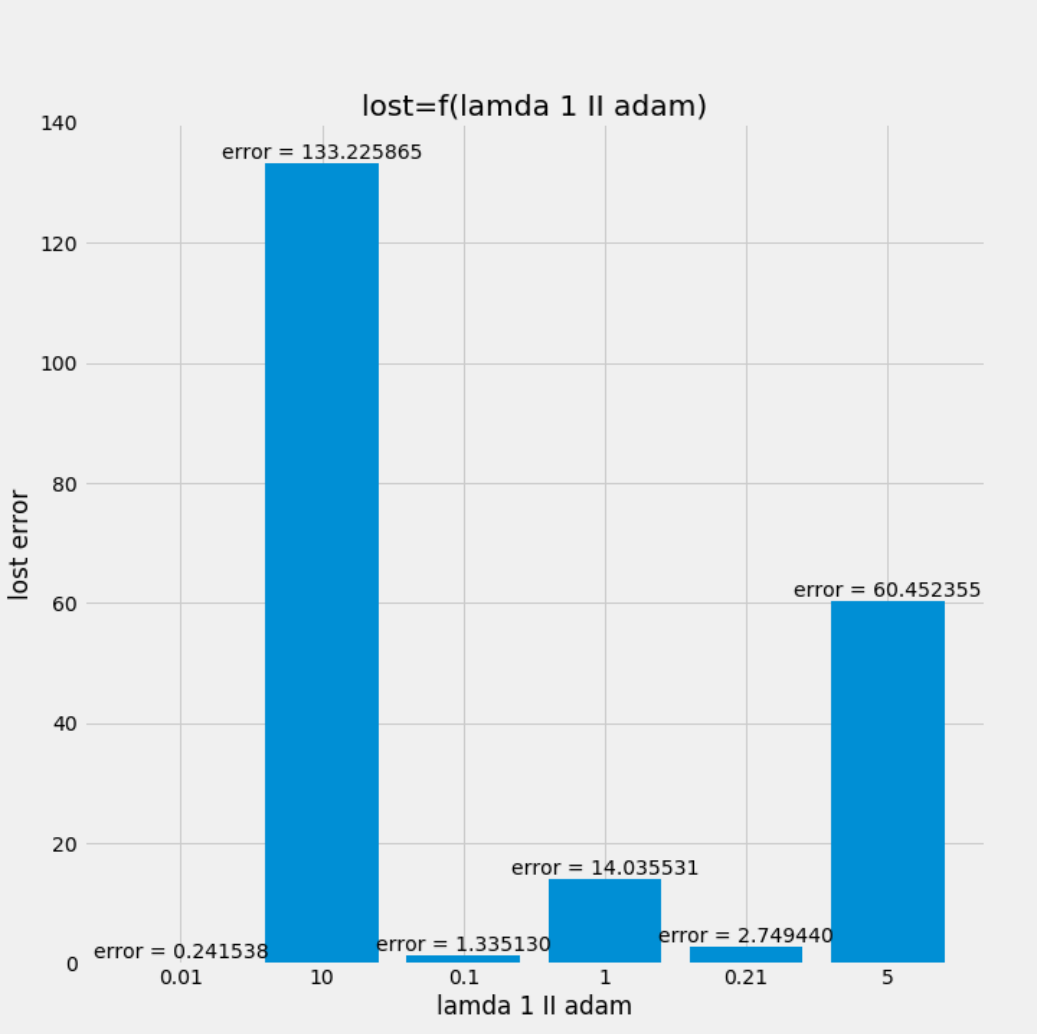
- $\hat{U}_i^{k+1} = \dfrac{U_i^{k+1}}{1 - \beta_1^k}$ where $U_i^{k+1} = \beta_1 U_i^k + (1 - \beta_1)\hat{g}_i^k$

- $\hat{V}_i^{(k+1)} = \dfrac{V_i^{(k+1)}}{1 - \beta_2^{(k)}}$ where $V_i^{(k+1)} = \beta_2 V_i^{(k)} + (1 - \beta_2)\hat{g}_i^{(k)} o \hat{g}_i^{(k)}$

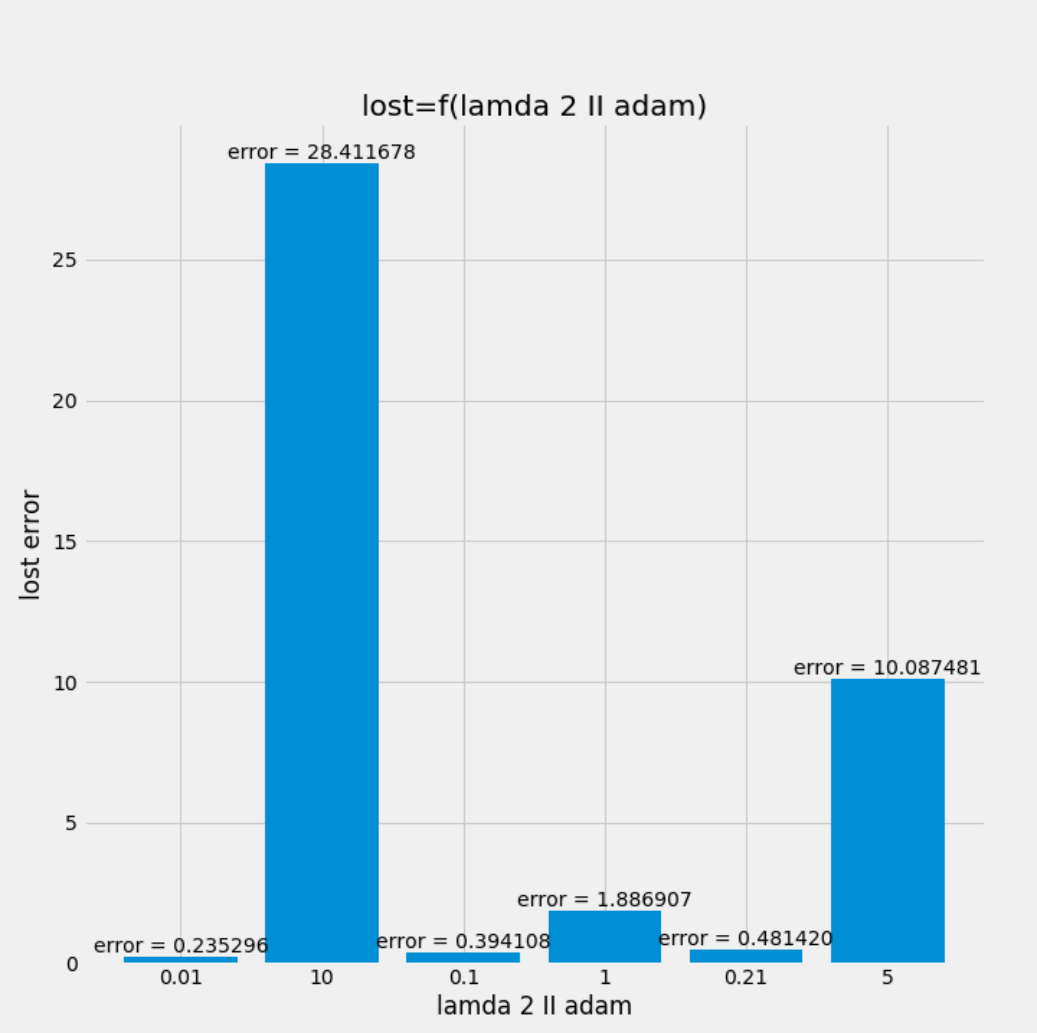
- $\hat{g}_i^{(k)} = \nabla f_i \left(W_j^{(k)} \right)$

- β_1, β_2 is given

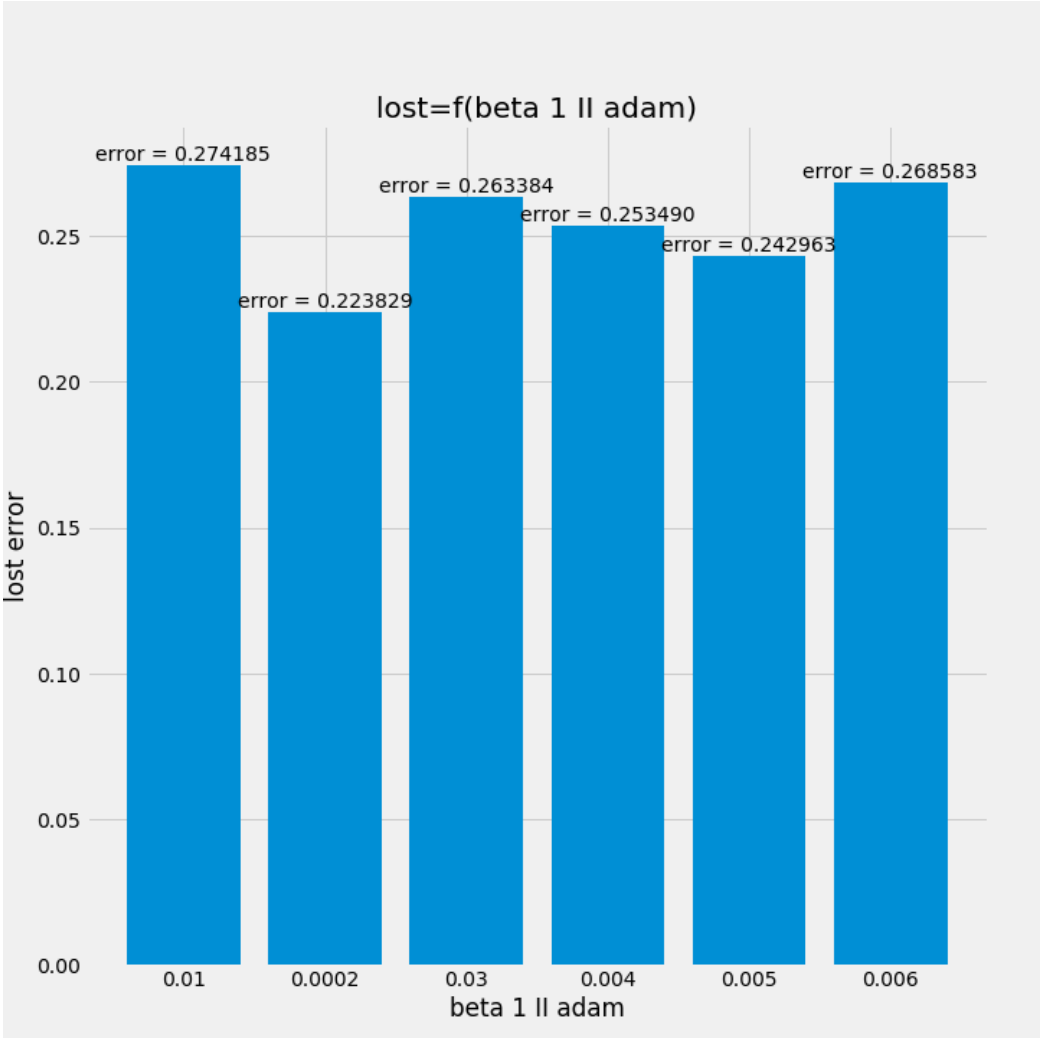
We set the number of neurons in the hidden layer to 5 and we set λ_2 to 0.01, β_1 to 0.01, β_2 to 0.01 and the step size to 0.01. So by varying λ_1 and using cross validation we get the below histogram. So we can say that the best λ_1 is 0.01.



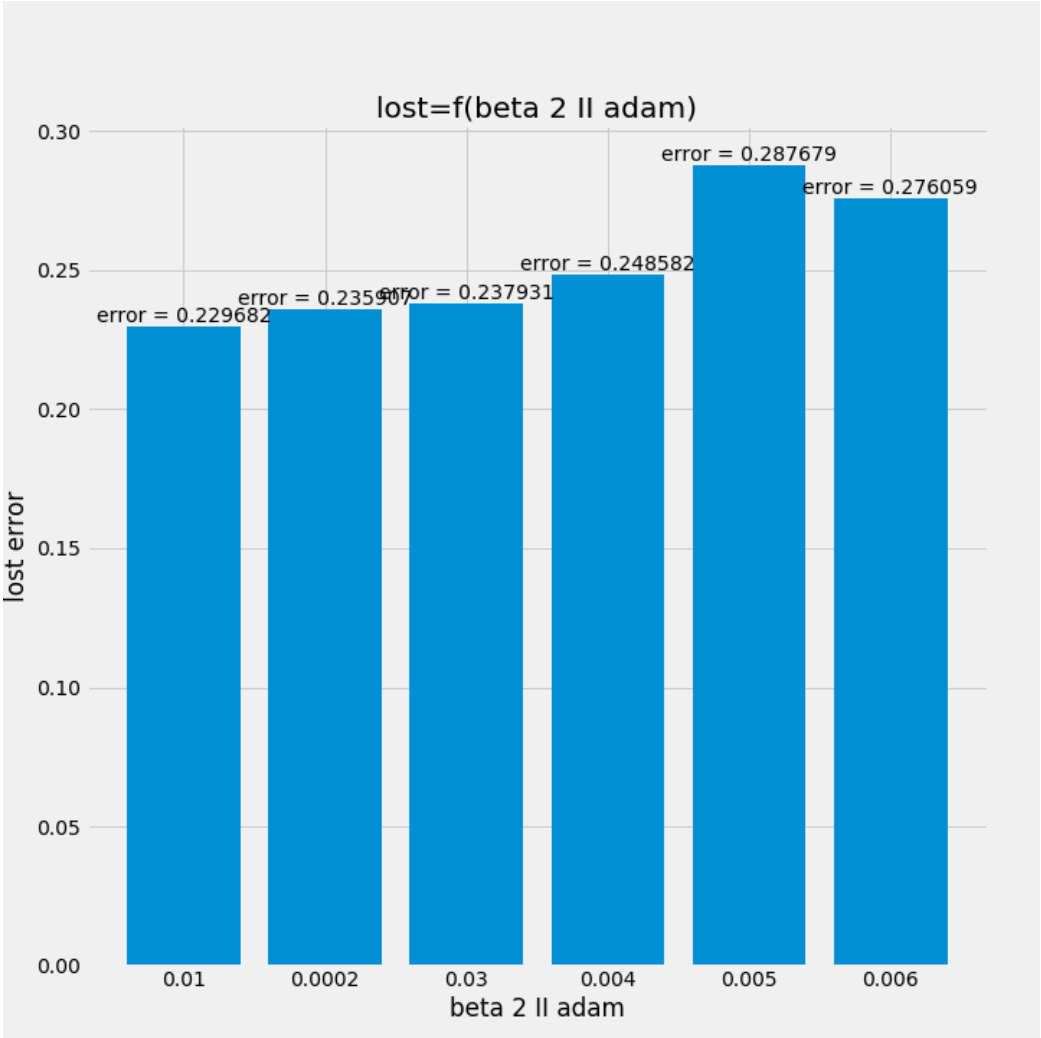
We set the number of neurons in the hidden layer to 5 and we set λ_1 to 0.01, β_1 to 0.01, β_2 to 0.01 and the step size to 1. So by varying λ_2 and using cross validation we get the below histogram. So we can say that the best λ_2 is 0.01.



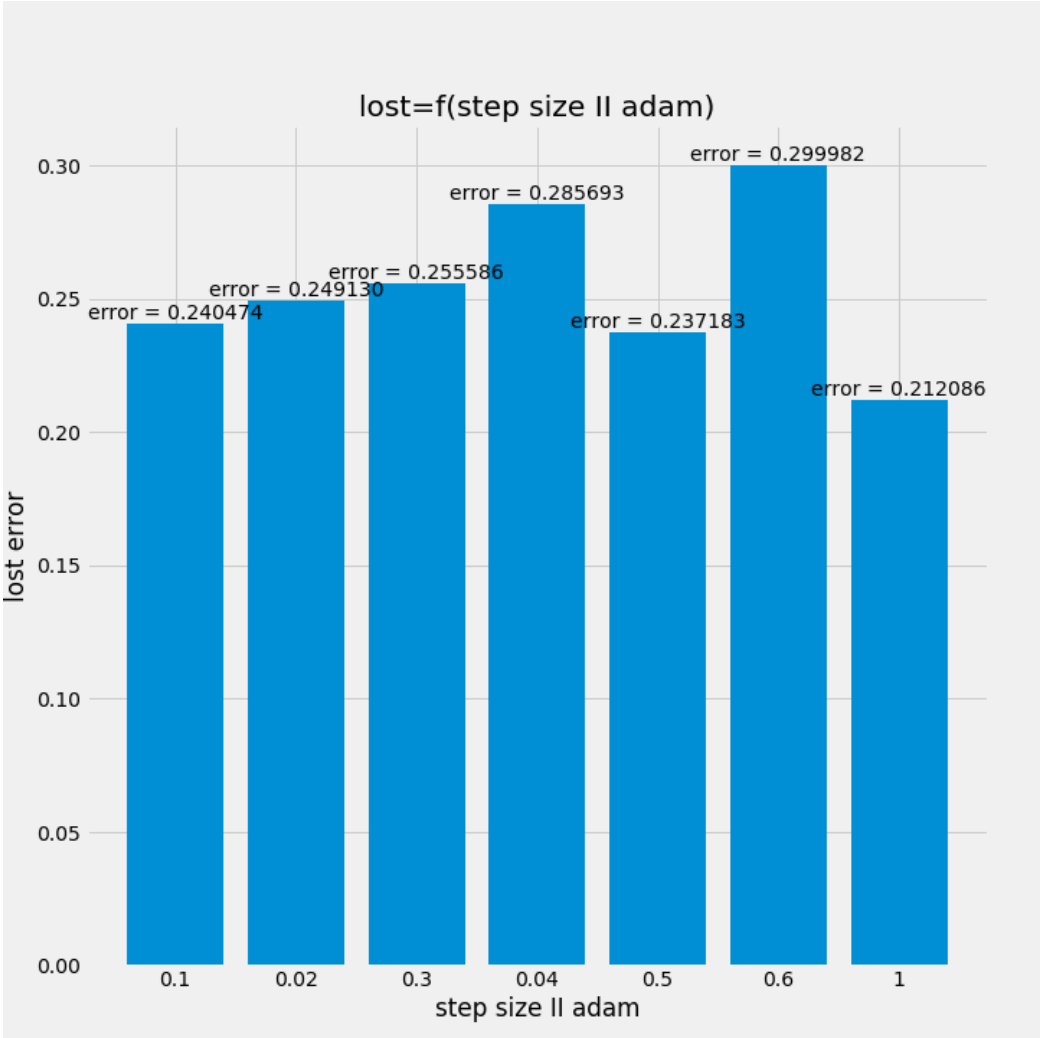
We set the number of neurons in the hidden layer to 5 and we set λ_1 to 0.01, λ_2 to 0.01, β_2 to 0.01 and the step size to 0.01. So by varying β_1 and using cross validation we get the below histogram. So we can say that the best β_1 is 0.0002.



We set the number of neurons in the hidden layer to 5 and we set λ_1 to 0.01, λ_2 to 0.01, β_1 to 0.01 and the step size to 0.01. So by varying β_2 and using cross validation we get the below histogram. So we can say that the best β_2 is 0.01.



We set the number of neurons in the hidden layer to 5 and we set λ_1 to 0.01, λ_2 to 0.01, β_1 to 0.01 and the β_1 to 0.01. So by varying the step size and using cross validation we get the below histogram. So we can say that the best step size is 1.



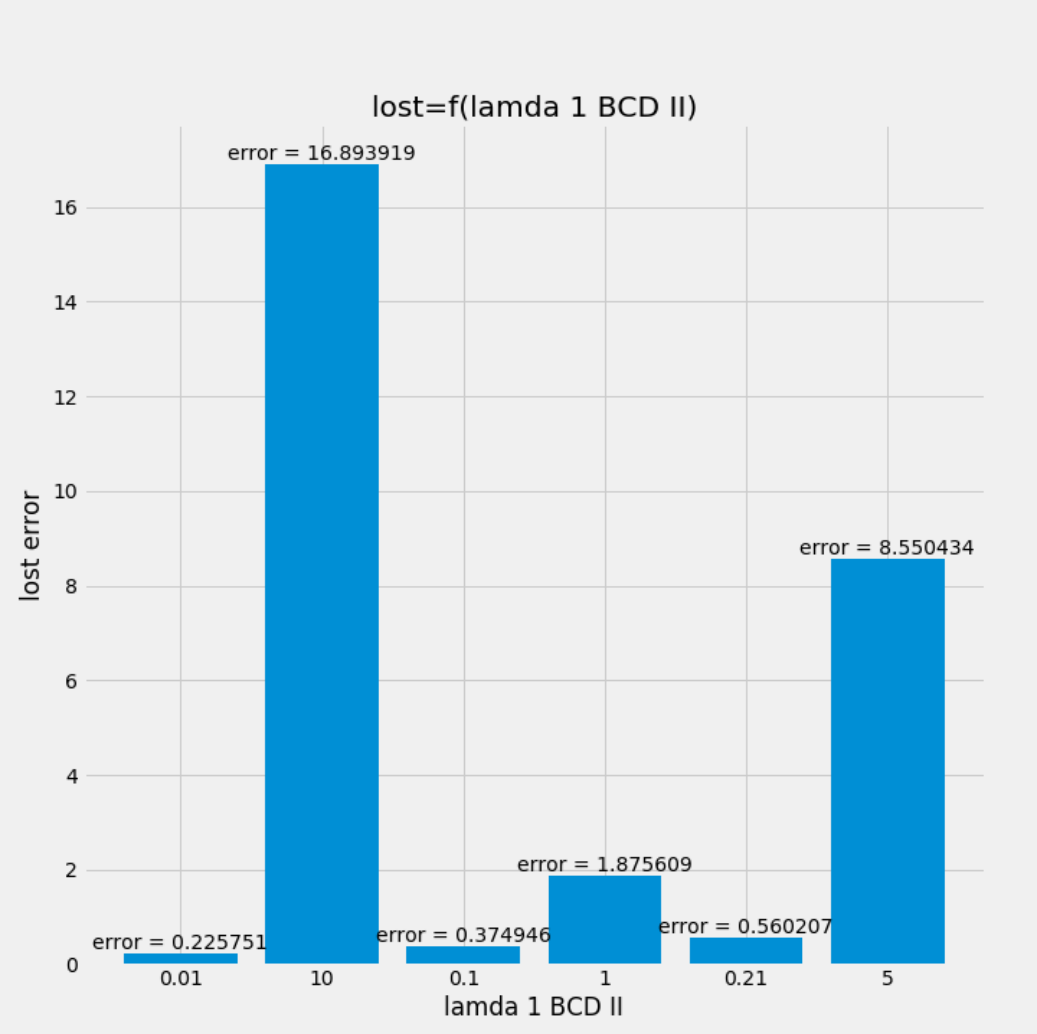
- **BCD** :

The update weight in BCD algorithm is given in thsi case by

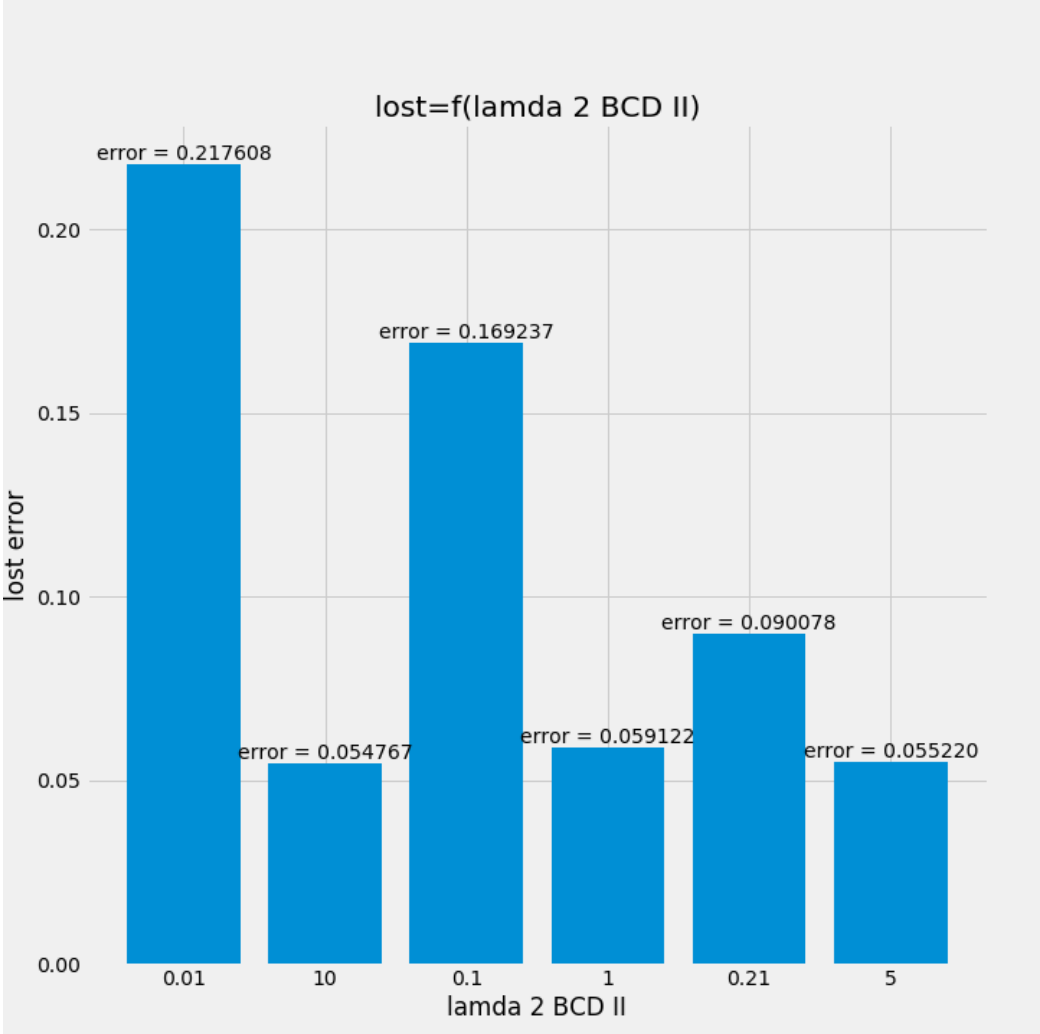
$$W_1^{(k+1)} = W_1^{(k)} - \alpha \nabla_{W_1} f_{|_{W_1 = W_1^{(k)}}} \left(W_1^{(k)}, W_2^{(k)} \right)$$

$$W_2^{(k+1)} = W_2^{(k)} - \alpha \nabla_{W_2} f_{|_{W_2 = W_2^{(k)}}} \left(W_1^{(k+1)}, W_2^{(k)} \right)$$

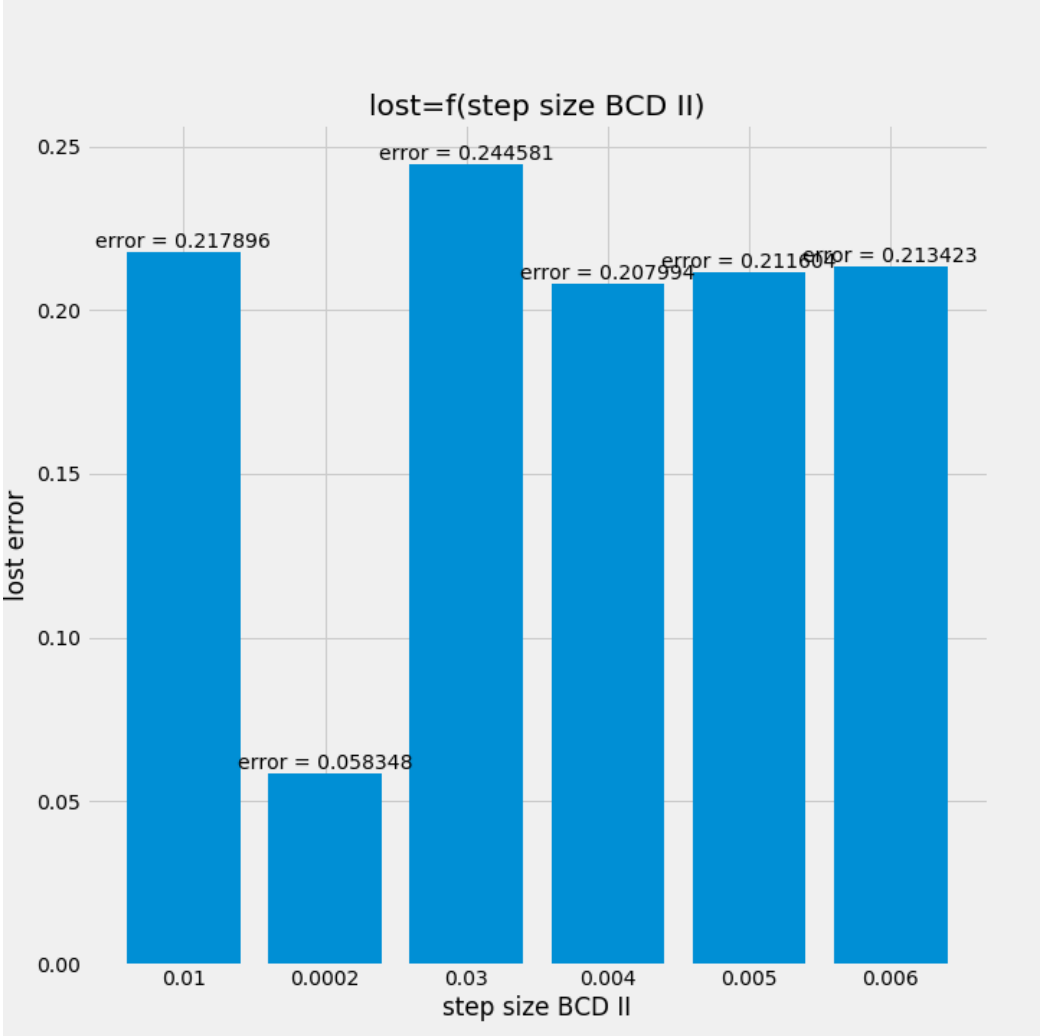
We set the number of neurons in the hidden layer to 5 and we set λ_2 to 0.01 and the step size to 0.01. So by varying λ_1 and using cross validation we get the below histogram. So we can say that the best λ_1 is 0.01.



We set the number of neurons in the hidden layer to 10 and we set λ_1 to 0.01, and the step size to 0.01. So by varying λ_2 and using cross validation we get the below histogram. So we can say that the best λ_2 is 10.

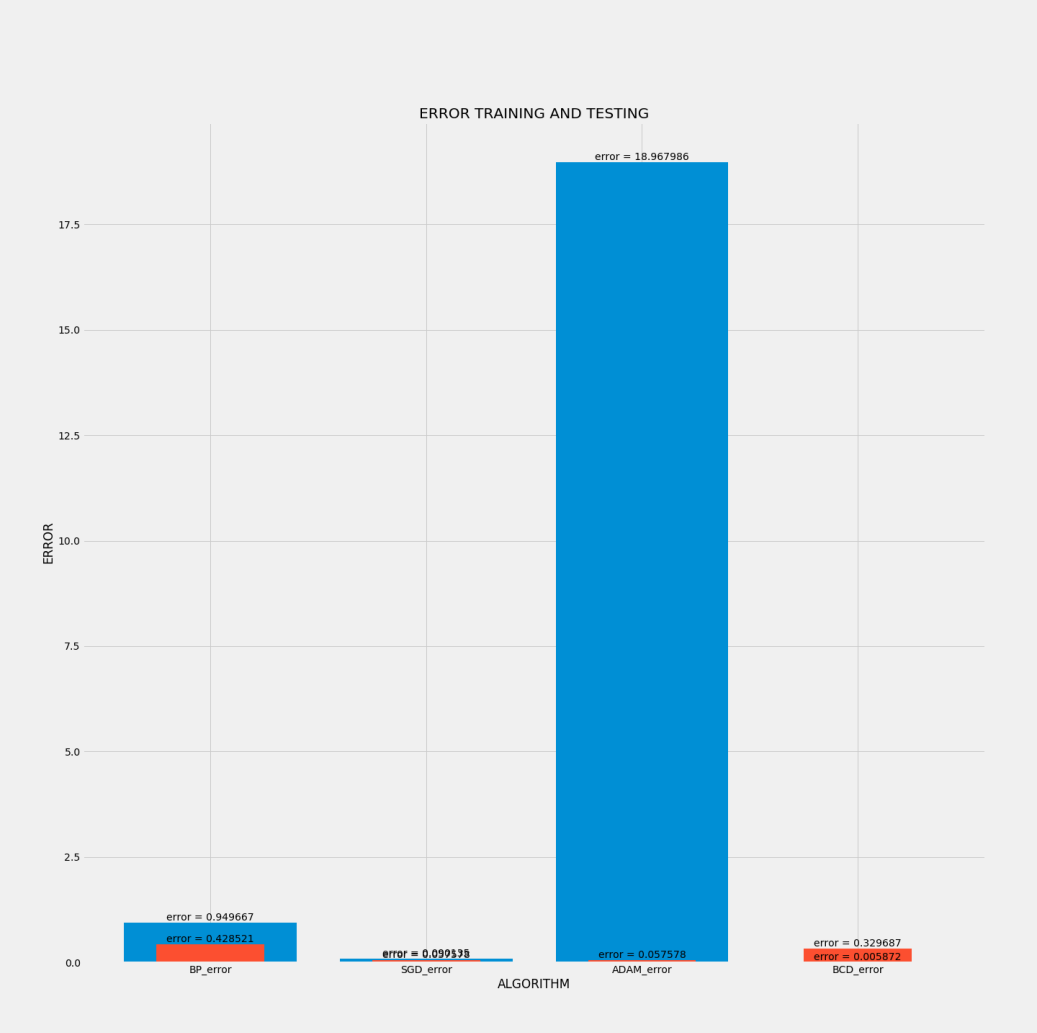


We set the number of neurons in the hidden layer to 10 and we set λ_1 to 0.01, and the λ_2 to 0.01. So by varying the step size and using cross validation we get the below histogram. So we can say that the best step size is 0.0002.



- **Compare the training error and test error.** :

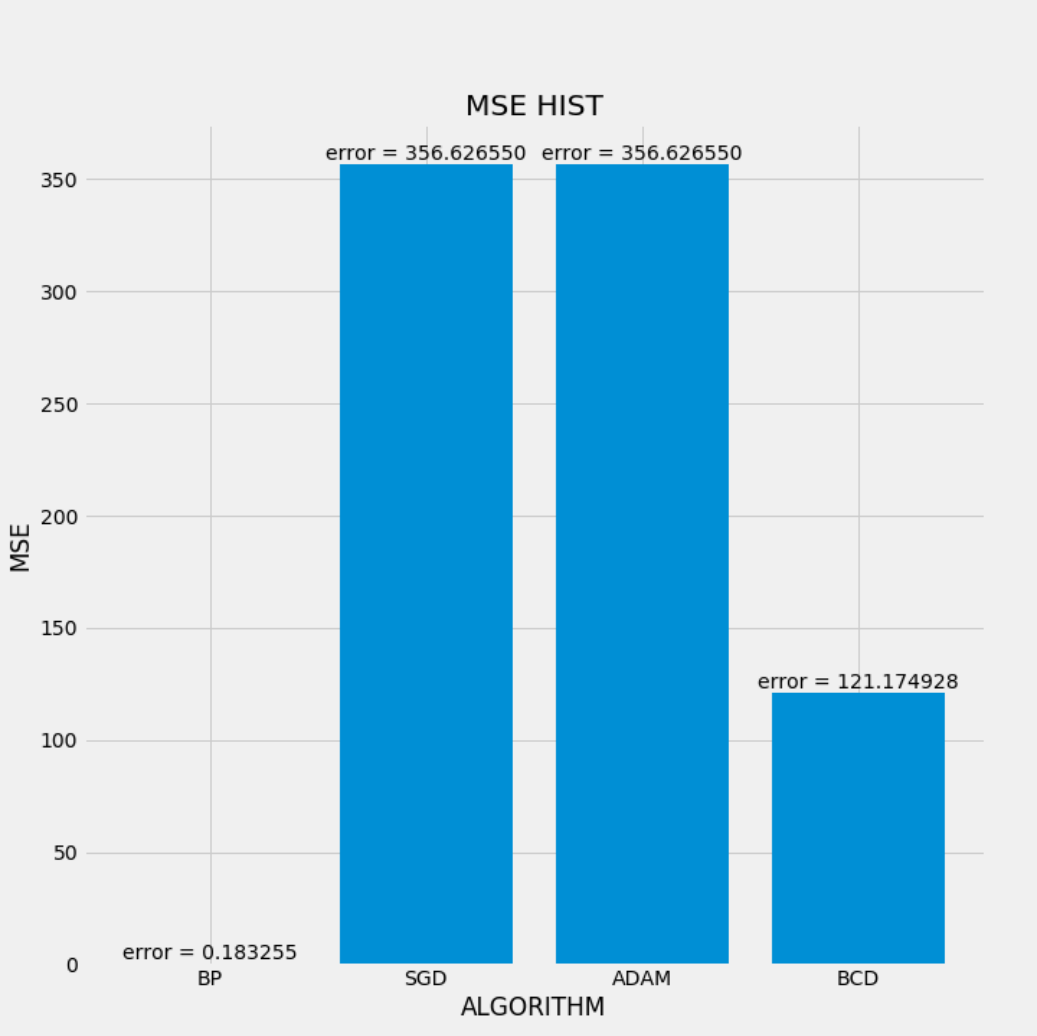
In this figure, we can see in red the average error obtained on the test



data and in blue the average error obtained on the data. It is thus noted that the error on the training data is greater than the error on the test data on all the algorithms with the exception of the BCD algorithm. This can be justified by the fact that the BCD algorithm does not work when we use sigmoid activation in our data.

On the adam algorithm, we can see that the learning error is huge

compared to the error obtained on our test data. This is justified by the error obtained at the beginning of the learning of the model which is very large but once the convergence is reached, the model works correctly.



We can see in this figure that the mse obtained with the BP algorithm is the smallest so we deduce that this algorithm is more optimal for our data. Theoretically, this can also be justified by the fact that the other algorithms are only an approximation of the BP. It is therefore normal that this algorithm is the best.

- **Compare these methods in terms** :

The comparisons made in the first part between the algorithms remain valid, only the calculation of the gradient of the functions is different

and the number of iterations necessary.

b) Compare the test/prediction error of all the method in Parts I and II. Give a qualitative argument to explain this difference..

Answer :
We observe that the errors obtained in the second part are smaller compared to the errors of the algorithms of the first part (see the mse in the previous figures). This is because in the second part the target variable has been normalized and is contained in the interval 0 and 1, while in the first part the target variables can take on larger values.