1. Boosting

(a) Gradient Calculation

The loss function (cost function) $L(y_i, \hat{y}_i)$ in terms of current labels and the true labels can be represented as function

$$L(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$$

The gradient g_i of the cost $L(y_i, \hat{y}_i)$ with respect to the current predictions $\hat{y}_i$ on each instance, can be represented as follows,

$$g_i = \frac{\delta L(y_i, \hat{y}_i)}{\delta y_i}$$

$$= \frac{\delta}{\delta y_i} (y_i - \hat{y}_i)^2$$

$$\boxed{g_i = -2(y_i - \hat{y}_i)}$$

(b) Weak Learner Section

We now choose a learner that can best predict these gradients,

$$h^* = \arg min_{h \epsilon H} \left(\min_{\gamma \epsilon \mathbb{R}} \sum_{i=1}^{n} \left(-g_i - \gamma h(x_i)\right)^2 \right)$$

Given, we can show that the optimal value of step size γ can be computed in closed form in this step, thus the selection rule for h^* can be derived independent of γ.

$$\gamma^* = \arg min_{\gamma \epsilon \mathbb{R}} \left(\min_{\gamma \epsilon \mathbb{R}} \sum_{i=1}^{n} \left(-g_i - \gamma h(x_i)\right)^2 \right)$$

$$= \arg min_{\gamma \epsilon \mathbb{R}} \left(\min_{\gamma \epsilon \mathbb{R}} \sum_{i=1}^{n} \left(2(y_i - \hat{y}_i) - \gamma h(x_i)\right)^2 \right)$$

Differentiating with respect to γ and equating it to zero, we get,

$$\sum_{i=1}^{n} -2h(x_i)[2(y_i - \hat{y}_i) - \gamma h(x_i)] = 0$$

$$\sum_{i=1}^{n} [-4h(x_i)(y_i - \hat{y}_i) + 2\gamma h(x_i)^2] = 0$$

Simplifying the above equation and solving for γ, we get,

$$4 \sum_{i=1}^{n} h(x_i)(y_i - \hat{y}_i) = 2\gamma \sum_{i=1}^{n} h(x_i)^2$$

$$\boxed{\gamma = \frac{2\sum_{i=1}^{n} h(x_i)(y_i - \hat{y}_i)}{\sum_{i=1}^{n} h(x_i)^2}}$$

(c) Step Size Selection

We now select a step size α^* that minimizes the loss,

$$\alpha^* = \arg min_{\alpha \epsilon \mathbb{R}} \sum_{i=1}^{n} L(y_i, \hat{y}_i + \alpha h^*(x_i))$$

Computation of α^* analytically in terms of $y_i, \hat{y}_i, and\ h^*$

$$\alpha^* = \arg min_{\alpha \epsilon \mathbb{R}} \sum_{i=1}^{n} \left(y_i - \left(\hat{y}_i + \alpha h^*(x_i)\right)\right)^2$$

Differentiating with respect to α and equating it to zero, we get,

$$\sum_{i=1}^{n} -2h^*(x_i)(y_i - \hat{y}_i - \alpha h^*(x_i)) = 0$$

$$\sum_{i=1}^{n} [-2h^*(x_i)y_i + 2h^*(x_i)\hat{y}_i + 2\alpha h^*(x_i)^2] = 0$$

$$2\sum_{i=1}^{n} [2h^*(x_i)(y_i - \hat{y}_i) = 2\sum_{i=1}^{n} \alpha h^*(x_i)^2$$

$$\boxed{\alpha = \frac{\sum_{i=1}^{n} [2h^*(x_i)(y_i - \hat{y}_i)}{\sum_{i=1}^{n} \alpha h^*(x_i)^2}}$$

2. Neural Networks

(a)

Given, a neural network with a single logistic output and with linear activation functions in hidden layers. We've to show that it is equivalent to logistic regression.

The prediction function can be represented as, where w is weights of second layer and v are weights of first layer:

$$y = \sigma\left(\sum_k w_k X \left(\sum_i v_{ki} x_i\right)\right)$$

$$= \frac{1}{1 + \exp(-\sum_k w_k X(\sum_i v_{ki} x_i))}$$

$$= \frac{1}{1 + \exp(-\sum_k \sum_i w_k v_{ki} x_i)}$$

$$= \frac{1}{1 + \exp(-\sum_i w_i' x_i)}$$

Where $w_i' = \sum_k w_k v_{ki}$

(b)

Given, a neural network with one hidden layer (see fig 1)

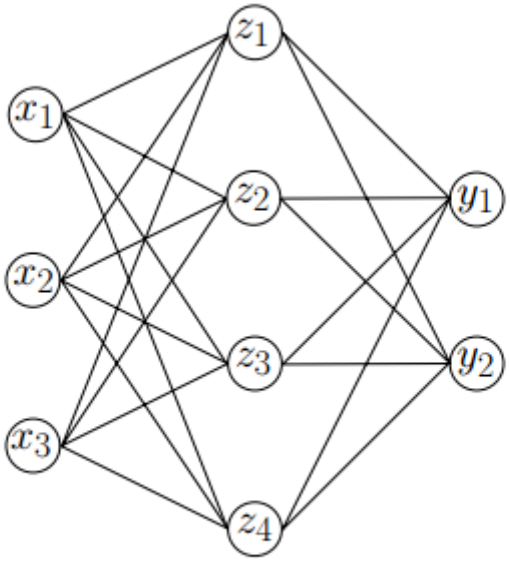


Figure 1: A neural network with one hidden layer

Each hidden layer is defined as $z_k = \tanh(\sum_{i=1}^{3} w_{ki} x_i)\ for\ k = 1, \ldots, 4$ and the outputs are defined as $y_j = \sum_{k=1}^{4} v_{jk} z_k\ for\ j = 1,2$. Consider the square loss function for each pair, we get,

$$L(y, \hat{y}) = \frac{1}{2}((y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2)$$

where $y_j\ and\ \hat{y}_j$ represent true outputs and our estimations.

Finding the gradient for v_{jk}:

$$\frac{\delta L}{\delta v_{jk}} = \frac{\delta L}{\delta b_j} \frac{\delta b_j}{\delta v_{jk}}$$

where $b_j = \hat{y}_j = \sum_k v_{jk} z_k$

$$\frac{\delta L}{\delta v_{jk}} = -(y_j - \hat{y}_j) z_k$$

Now finding gradient for w_{ki}:

$$\frac{\delta L}{\delta w_{ki}} = \frac{\delta L}{\delta a_k}\frac{\delta a_k}{\delta w_{ki}}$$

where $a_k = \sum_k w_{ki} x_i$

$$= \frac{\delta L}{\delta a_k} x_i$$

$$= \sum_j \frac{\delta L}{\delta b_j}\frac{\delta b_j}{\delta a_k}\, x_i$$

$$= \sum_j (y_j - \hat{y}_j)\frac{\delta(v_{jk}\tanh(a_k))}{\delta a_k}\, x_i$$

$$\boxed{= \sum_j^2 (y_j - \hat{y}_j) v_{jk}\left(1 - \tanh^2\left(\sum_i^3 w_{ki} x_i\right)\right) x_i}$$

3. Programming

(d) Linear Activations

(i)

Linear Activation, Architecture I

Score for architecture = [50, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear: 0.820897242504

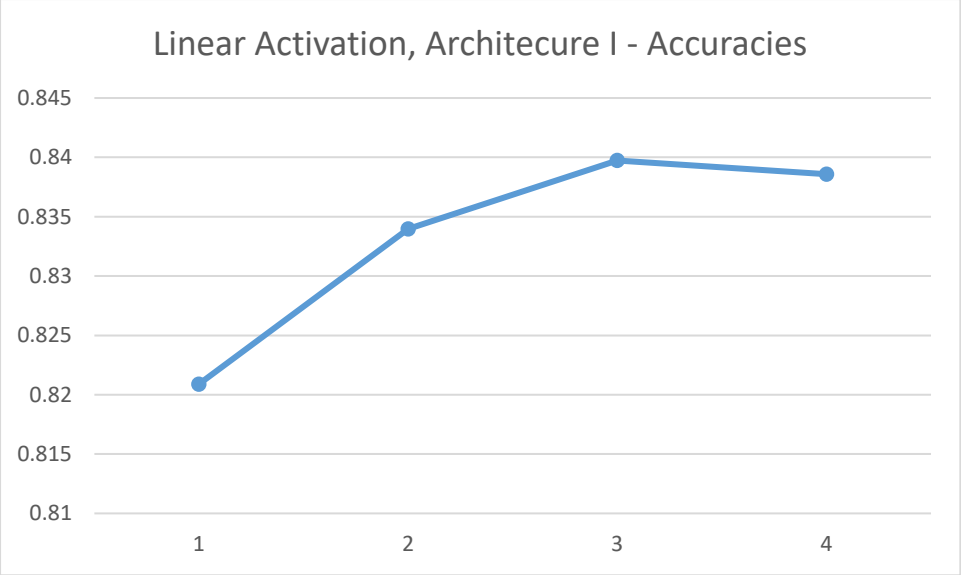
Score for architecture = [50, 50, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear: 0.833967625288

Score for architecture = [50, 50, 50, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear: 0.839733982118

Score for architecture = [50, 50, 50, 50, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear: 0.838580712713

Best Config: architecture = [50, 50, 50, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear, best_acc = 0.839733982118

The training time for Linear Activation, Architecture I 26.685893774 seconds



As we can see that the accuracies keep improving with increase in the complexity of the architecture to a certain extent and then it slightly tends to fluctuate near a certain maximum value for a while. Further increase in complexity of the architecture might lead to better accuracies. But in general, with linear activation we see rise in accuracy with increase in complexity of architecture.

(ii)

Linear Activation, Architecture II

Score for architecture = [50, 50, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear: 0.830815356901

Score for architecture = [50, 500, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear: 0.840464385416

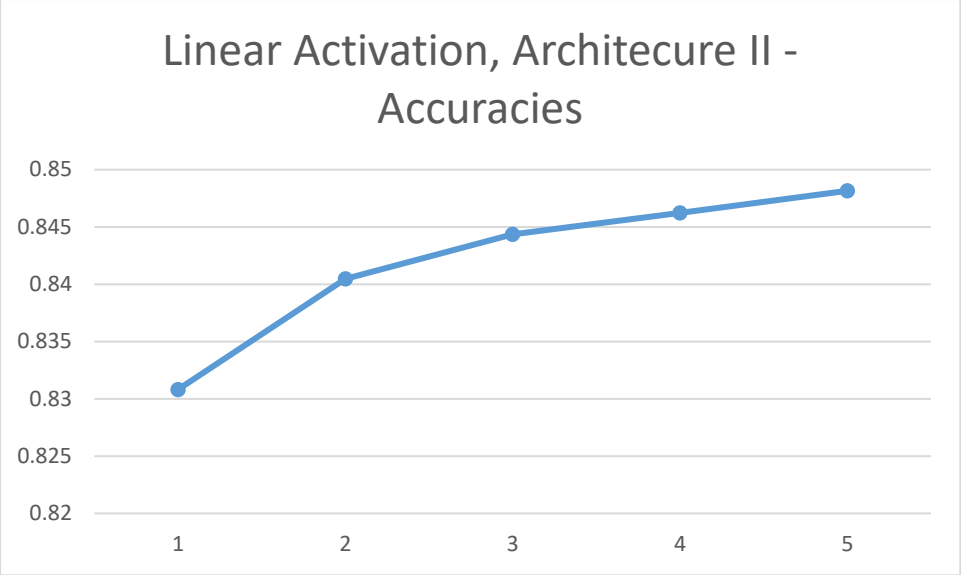
Score for architecture = [50, 500, 300, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear: 0.844347058086

Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear: 0.846230732587

Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear: 0.848152844328

Best Config: architecture = [50, 800, 800, 500, 300, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear, best_acc = 0.848152844328

The training time for Linear Activation, Architecture II 237.228027105 seconds



Here, we have used more complex architectures for linear activation and we can clearly see the rise in accuracies with the increase in complexity. In this as the architectures are more complex compared to the previous one the change in accuracies are easily noted.

(e) Sigmoid Activation

Sigmoid Activation

Score for architecture = [50, 50, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = sigmoid: 0.718525354294

Score for architecture = [50, 500, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = sigmoid: 0.764002621327

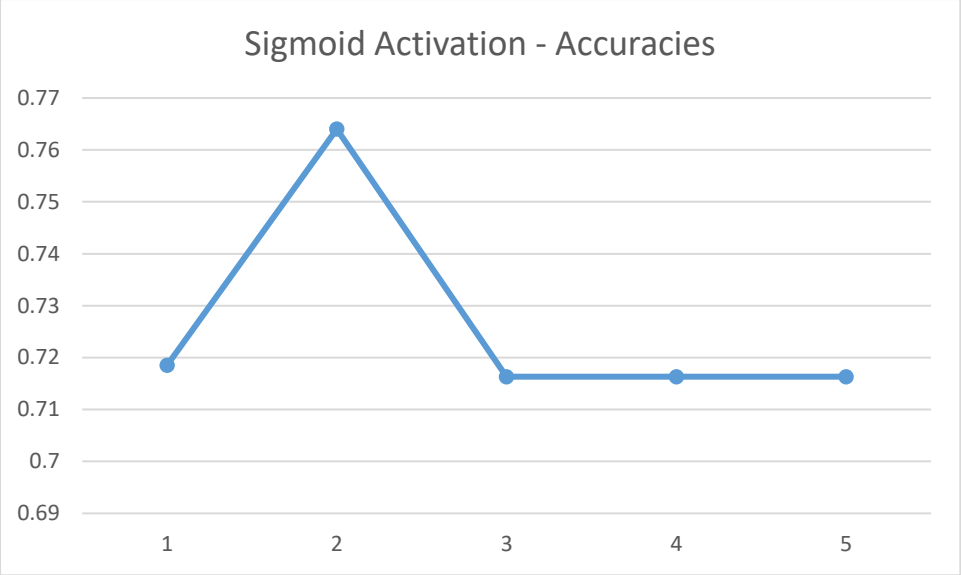
Score for architecture = [50, 500, 300, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = sigmoid: 0.71633413834

Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = sigmoid: 0.71633413834

Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = sigmoid: 0.71633413834

Best Config: architecture = [50, 500, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = sigmoid, best_acc = 0.764002621327

The training time for Sigmoid Activation 821.756697893 seconds



In sigmoid activation, as we can see in the graph above the accuracy increases to a certain extent, then drops suddenly and finally settles down at an almost constant value. This trend is noted with the increase in complexity of the architecture.

We use a sigmoid activation function to introduce a non-linearity, but the sigmoid function returns only values between 0 or 1 as the output, a lot of values have to be approximated (vanishing gradient) to it which in turn reduces the accuracy where as in linear we can use the values directly throughout the input range.

The time taken by sigmoid activation is usually higher than linear activation function because it is more complex.

(f) ReLu Activation

ReLu Activation

Score for architecture = [50, 50, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = relu: 0.816591706682

Score for architecture = [50, 500, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = relu: 0.81589974193

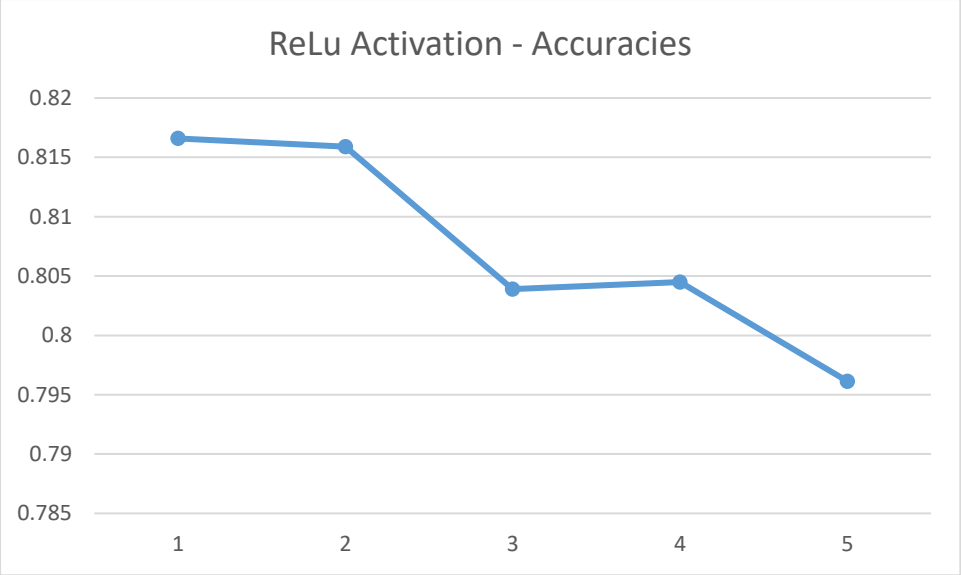
Score for architecture = [50, 500, 300, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = relu: 0.803905733448

Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = relu: 0.804482372241

Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = relu: 0.796140393675

Best Config: architecture = [50, 50, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = relu, best_acc = 0.816591706682

The training time for ReLu Activation 389.922206163 seconds



The accuracy of ReLu decreases with increase in complexity of the architecture because ReLu tends to blow up the activation function (since it is not constrained)

ReLu activation gives better accuracy compared to sigmoid but the accuracy is still lower compared to linear activation (by 1 – 2%). It gives a better accuracy than sigmoid as it takes into consideration the sparsity of the data and the vanishing gradient isn't present.(better in back propogation)

The time taken by ReLu is far lesser than sigmoid, but it takes more time than linear activation in general.

(g) L-2 Regularization

L-2 Regularization

Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-07, decay = 0.0, momentum = 0.0, actfn = relu: 0.810556262664

Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-07, decay = 0.0, momentum = 0.0, actfn = relu: 0.805020564135

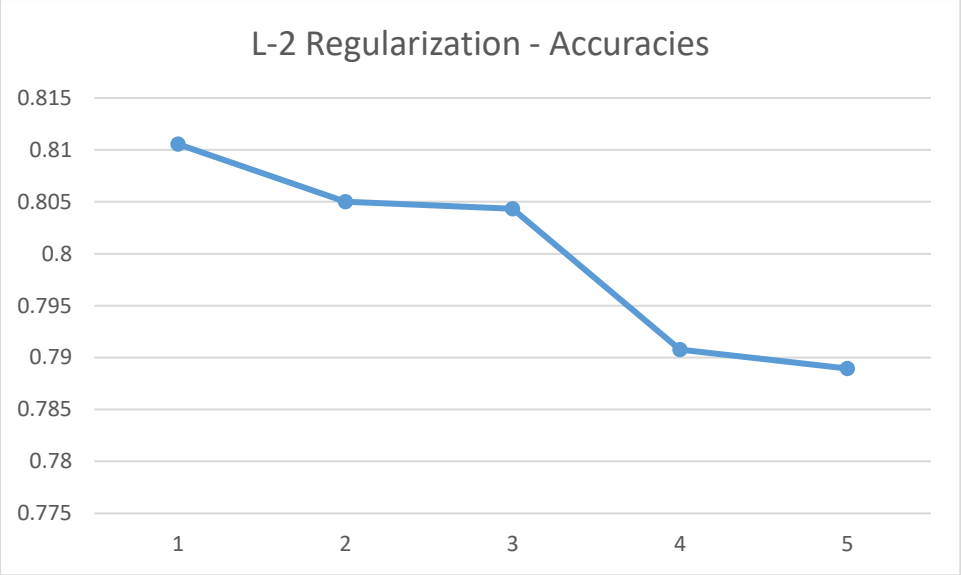
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-06, decay = 0.0, momentum = 0.0, actfn = relu: 0.804328607243

Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-06, decay = 0.0, momentum = 0.0, actfn = relu: 0.790758476677

Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-05, decay = 0.0, momentum = 0.0, actfn = relu: 0.788951676655

Best Config: architecture = [50, 800, 500, 300, 2], lambda = 1e-07, decay = 0.0, momentum = 0.0, actfn = relu, best_acc = 0.810556262664

The training time for L-2 Regularization 624.809979916 seconds



The best value of regularization hyper parameter is 1e-07.

In linear regularization we see that the accuracies tend to slowly decrease with the increase the regularization coefficient. The decrease isn't by a huge factor but it is significant.(prevents over fitting)

(h) L-2 Regularization with Early Stopping

Early Stopping and L2-regularization

Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-07, decay = 0.0, momentum = 0.0, actfn = relu: 0.796486372368

Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-07, decay = 0.0, momentum = 0.0, actfn = relu: 0.796447927762

Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-06, decay = 0.0, momentum = 0.0, actfn = relu: 0.805251222436

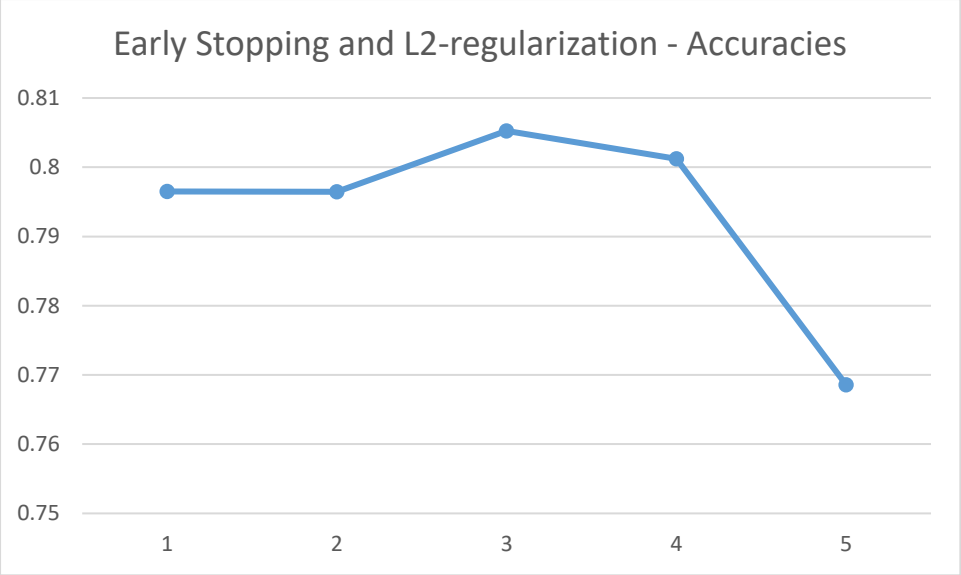
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-06, decay = 0.0, momentum = 0.0, actfn = relu: 0.801214775602

Epoch 00009: early stopping

Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-05, decay = 0.0, momentum = 0.0, actfn = relu: 0.768577251383

Best Config: architecture = [50, 800, 500, 300, 2], lambda = 1e-06, decay = 0.0, momentum = 0.0, actfn = relu, best_acc = 0.805251222436

The training time for Early Stopping and L2-regularization 498.872879028 seconds



The best value of regularization hyper parameter is 1e-06.

In this method we see that initially with the increase the hyper parameter value the accuracies tend to increase to a certain extent and then start dropping with the further increase in the value of hyper parameter. (bias increases to large extent resulting in drop of accuracy). Also since the early stopping is dependent on the validation set, selection of data for validation influences the accuracy as well.

We see that the original L-2 regularization gives a better accuracy as it computes all the data extensively where as in early stopping the calculations are pre-empted in early stopping method to reduce the overall execution time. So, here it is a decision to reduce the calculation time by sacrificing the accuracy.

(i) SGD with weight decay

SGD with weight decay

Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-07, decay = 1e-05, momentum = 0.0, actfn = relu: 0.770883795096

Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-07, decay = 5e-05, momentum = 0.0, actfn = relu: 0.7573905431

Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-07, decay = 0.0001, momentum = 0.0, actfn = relu: 0.715449962677

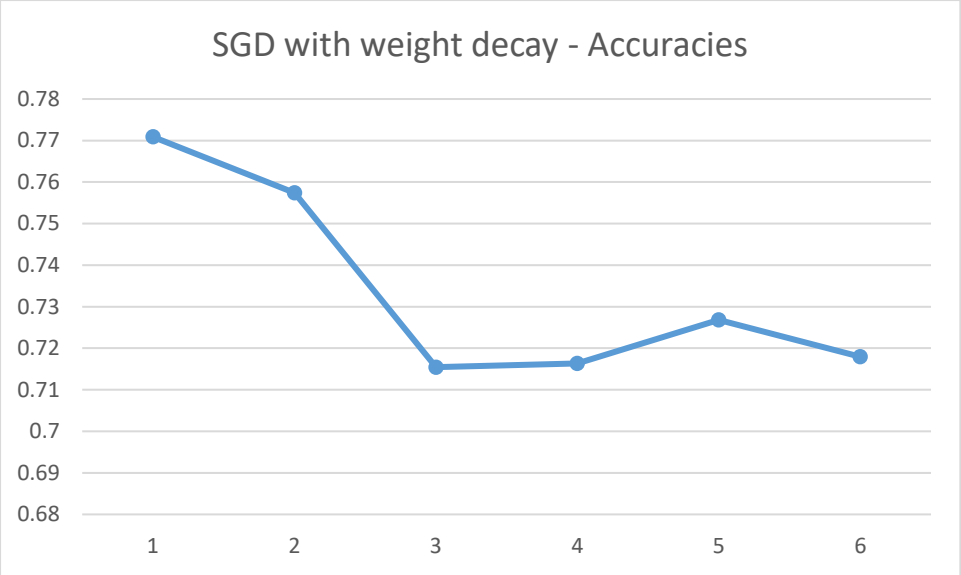
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-07, decay = 0.0003, momentum = 0.0, actfn = relu: 0.71633413834

Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-07, decay = 0.0007, momentum = 0.0, actfn = relu: 0.726790447908

Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-07, decay = 0.001, momentum = 0.0, actfn = relu: 0.717987161907

Best Config: architecture = [50, 800, 500, 300, 2], lambda = 5e-07, decay = 1e-05, momentum = 0.0, actfn = relu, best_acc = 0.770883795096

The training time for SGD with weight decay 2448.58550787 seconds



Best decay value: 1e-05

(j) Momentum

Momentum

Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0, decay = 1e-05, momentum = 0.99, actfn = relu: 0.851074461288

Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0, decay = 1e-05, momentum = 0.98, actfn = relu: 0.82804750723

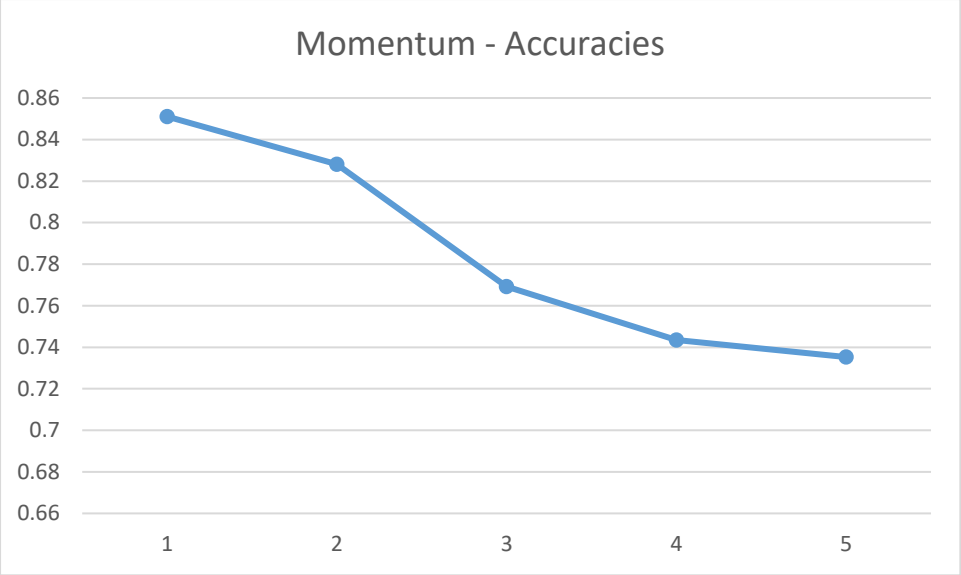
Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0, decay = 1e-05, momentum = 0.95, actfn = relu: 0.769153885593

Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0, decay = 1e-05, momentum = 0.9, actfn = relu: 0.743435978271

Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0, decay = 1e-05, momentum = 0.85, actfn = relu: 0.735286206526

Best Config: architecture = [50, 800, 500, 300, 2], lambda = 0.0, decay = 1e-05, momentum = 0.99, actfn = relu, best_acc = 0.851074461288

The training time for Momentum 971.953459978 seconds



Best momentum value: 0.99

(k) Combining the above

Combination of different coefficents

Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-07, decay = 1e-05, momentum = 0.99, actfn = relu: 0.867912198964

Best Config: architecture = [50, 800, 500, 300, 2], lambda = 1e-07, decay = 1e-05, momentum = 0.99, actfn = relu, best_acc = 0.867912198964

The training time for Combination of different coefficients 381.624815941 seconds

For the above set of values, we see that the combination of the best values is giving us a better accuracy than the rest of the methods. But, this might not always be the case as we are dividing the data in random and running all the activation methods from which we retrieve the best values to be used in this method. Also, the best values in each method combined needn't necessarily give the highest accuracy as they may or may not be independent of each other.

(l) Grid search with cross validation

Grid search with cross-validation

Epoch 00013: early stopping

Score for architecture = [50, 50, 2], lambda = 1e-07, decay = 1e-05, momentum = 0.99, actfn = relu: 0.80848038085

Score for architecture = [50, 50, 2], lambda = 1e-07, decay = 5e-05, momentum = 0.99, actfn = relu: 0.839388007515

Score for architecture = [50, 50, 2], lambda = 1e-07, decay = 0.0001, momentum = 0.99, actfn = relu: 0.834044514008

Score for architecture = [50, 50, 2], lambda = 5e-07, decay = 1e-05, momentum = 0.99, actfn = relu: 0.842501829497

Score for architecture = [50, 50, 2], lambda = 5e-07, decay = 5e-05, momentum = 0.99, actfn = relu: 0.845923197023

Score for architecture = [50, 50, 2], lambda = 5e-07, decay = 0.0001, momentum = 0.99, actfn = relu: 0.841732977011

Epoch 00084: early stopping

Score for architecture = [50, 50, 2], lambda = 1e-06, decay = 1e-05, momentum = 0.99, actfn = relu: 0.841540765115

Score for architecture = [50, 50, 2], lambda = 1e-06, decay = 5e-05, momentum = 0.99, actfn = relu: 0.846615154408

Score for architecture = [50, 50, 2], lambda = 1e-06, decay = 0.0001, momentum = 0.99, actfn = relu: 0.831891742168

Score for architecture = [50, 50, 2], lambda = 5e-06, decay = 1e-05, momentum = 0.99, actfn = relu: 0.844039520231

Epoch 00011: early stopping

Score for architecture = [50, 50, 2], lambda = 5e-06, decay = 5e-05, momentum = 0.99, actfn = relu: 0.815323105429

Score for architecture = [50, 50, 2], lambda = 5e-06, decay = 0.0001, momentum = 0.99, actfn = relu: 0.838465381357

Score for architecture = [50, 50, 2], lambda = 1e-05, decay = 1e-05, momentum = 0.99, actfn = relu: 0.844039515648

Score for architecture = [50, 50, 2], lambda = 1e-05, decay = 5e-05, momentum = 0.99, actfn = relu: 0.83508246105

Score for architecture = [50, 50, 2], lambda = 1e-05, decay = 0.0001, momentum = 0.99, actfn = relu: 0.840003081256

Score for architecture = [50, 500, 2], lambda = 1e-07, decay = 1e-05, momentum = 0.99, actfn = relu: 0.852073962418

Epoch 00057: early stopping

Score for architecture = [50, 500, 2], lambda = 1e-07, decay = 5e-05, momentum = 0.99, actfn = relu: 0.834044513516

Score for architecture = [50, 500, 2], lambda = 1e-07, decay = 0.0001, momentum = 0.99, actfn = relu: 0.84688425224

Score for architecture = [50, 500, 2], lambda = 5e-07, decay = 1e-05, momentum = 0.99, actfn = relu: 0.851689541903

Score for architecture = [50, 500, 2], lambda = 5e-07, decay = 5e-05, momentum = 0.99, actfn = relu: 0.849498327748

Score for architecture = [50, 500, 2], lambda = 5e-07, decay = 0.0001, momentum = 0.99, actfn = relu: 0.845692535125

Epoch 00060: early stopping

Score for architecture = [50, 500, 2], lambda = 1e-06, decay = 1e-05, momentum = 0.99, actfn = relu: 0.839503323324

Score for architecture = [50, 500, 2], lambda = 1e-06, decay = 5e-05, momentum = 0.99, actfn = relu: 0.848075967065

Epoch 00010: early stopping

Score for architecture = [50, 500, 2], lambda = 1e-06, decay = 0.0001, momentum = 0.99, actfn = relu: 0.810248721211

Epoch 00010: early stopping

Score for architecture = [50, 500, 2], lambda = 5e-06, decay = 1e-05, momentum = 0.99, actfn = relu: 0.808672585872

Score for architecture = [50, 500, 2], lambda = 5e-06, decay = 5e-05, momentum = 0.99, actfn = relu: 0.84915235413

Score for architecture = [50, 500, 2], lambda = 5e-06, decay = 0.0001, momentum = 0.99, actfn = relu: 0.844923696386

Score for architecture = [50, 500, 2], lambda = 1e-05, decay = 1e-05, momentum = 0.99, actfn = relu: 0.850766929001

Epoch 00009: early stopping

Score for architecture = [50, 500, 2], lambda = 1e-05, decay = 5e-05, momentum = 0.99, actfn = relu: 0.81443893255

Score for architecture = [50, 500, 2], lambda = 1e-05, decay = 0.0001, momentum = 0.99, actfn = relu: 0.843385998286

Score for architecture = [50, 500, 300, 2], lambda = 1e-07, decay = 1e-05, momentum = 0.99, actfn = relu: 0.858955138478

Score for architecture = [50, 500, 300, 2], lambda = 1e-07, decay = 5e-05, momentum = 0.99, actfn = relu: 0.856725480209

Epoch 00080: early stopping

Score for architecture = [50, 500, 300, 2], lambda = 1e-07, decay = 0.0001, momentum = 0.99, actfn = relu: 0.847845313346

Score for architecture = [50, 500, 300, 2], lambda = 5e-07, decay = 1e-05, momentum = 0.99, actfn = relu: 0.86187675773

Score for architecture = [50, 500, 300, 2], lambda = 5e-07, decay = 5e-05, momentum = 0.99, actfn = relu: 0.858301620131

Epoch 00009: early stopping

Score for architecture = [50, 500, 300, 2], lambda = 5e-07, decay = 0.0001, momentum = 0.99, actfn = relu: 0.797139897096

Epoch 00009: early stopping

Score for architecture = [50, 500, 300, 2], lambda = 1e-06, decay = 1e-05, momentum = 0.99, actfn = relu: 0.801483873434

Epoch 00008: early stopping

Score for architecture = [50, 500, 300, 2], lambda = 1e-06, decay = 5e-05, momentum = 0.99, actfn = relu: 0.795179338458

Score for architecture = [50, 500, 300, 2], lambda = 1e-06, decay = 0.0001, momentum = 0.99, actfn = relu: 0.850459387548

Score for architecture = [50, 500, 300, 2], lambda = 5e-06, decay = 1e-05, momentum = 0.99, actfn = relu: 0.859416444929

Score for architecture = [50, 500, 300, 2], lambda = 5e-06, decay = 5e-05, momentum = 0.99, actfn = relu: 0.860300613718

Epoch 00009: early stopping

Score for architecture = [50, 500, 300, 2], lambda = 5e-06, decay = 0.0001, momentum = 0.99, actfn = relu: 0.789220771211

Epoch 00009: early stopping

Score for architecture = [50, 500, 300, 2], lambda = 1e-05, decay = 1e-05, momentum = 0.99, actfn = relu: 0.799907735311

Score for architecture = [50, 500, 300, 2], lambda = 1e-05, decay = 5e-05, momentum = 0.99, actfn = relu: 0.856071957772

Score for architecture = [50, 500, 300, 2], lambda = 1e-05, decay = 0.0001, momentum = 0.99, actfn = relu: 0.849113900851

Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-07, decay = 1e-05, momentum = 0.99, actfn = relu: 0.870372508167

Epoch 00008: early stopping

Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-07, decay = 5e-05, momentum = 0.99, actfn = relu: 0.761849840322

Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-07, decay = 0.0001, momentum = 0.99, actfn = relu: 0.858993572934

Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-07, decay = 1e-05, momentum = 0.99, actfn = relu: 0.8681428481

Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-07, decay = 5e-05, momentum = 0.99, actfn = relu: 0.865451886656

Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-07, decay = 0.0001, momentum = 0.99, actfn = relu: 0.859570216309

Epoch 00007: early stopping

Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-06, decay = 1e-05, momentum = 0.99, actfn = relu: 0.750086493738

Epoch 00009: early stopping

Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-06, decay = 5e-05, momentum = 0.99, actfn = relu: 0.765578747501

Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-06, decay = 0.0001, momentum = 0.99, actfn = relu: 0.861107902952

Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-06, decay = 1e-05, momentum = 0.99, actfn = relu: 0.868142850883

Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-06, decay = 5e-05, momentum = 0.99, actfn = relu: 0.868065964455

Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-06, decay = 0.0001, momentum = 0.99, actfn = relu: 0.862568717407

Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-05, decay = 1e-05, momentum = 0.99, actfn = relu: 0.865413447125

Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-05, decay = 5e-05, momentum = 0.99, actfn = relu: 0.8657594304

Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-05, decay = 0.0001, momentum = 0.99, actfn = relu: 0.860185285467

Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 1e-07, decay = 1e-05, momentum = 0.99, actfn = relu: 0.872986583675

Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 1e-07, decay = 5e-05, momentum = 0.99, actfn = relu: 0.870257178117

Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 1e-07, decay = 0.0001, momentum = 0.99, actfn = relu: 0.867220241579

Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 5e-07, decay = 1e-05, momentum = 0.99, actfn = relu: 0.875946645734

Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 5e-07, decay = 5e-05, momentum = 0.99, actfn = relu: 0.869219232874

Epoch 00007: early stopping

Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 5e-07, decay = 0.0001, momentum = 0.99, actfn = relu: 0.736247261251

Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 1e-06, decay = 1e-05, momentum = 0.99, actfn = relu: 0.876023531178

Epoch 00008: early stopping

Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 1e-06, decay = 5e-05, momentum = 0.99, actfn = relu: 0.733863838476

Epoch 00008: early stopping

Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 1e-06, decay = 0.0001, momentum = 0.99, actfn = relu: 0.745819403337

Epoch 00008: early stopping

Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 5e-06, decay = 1e-05, momentum = 0.99, actfn = relu: 0.735862836153

Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 5e-06, decay = 5e-05, momentum = 0.99, actfn = relu: 0.869296116519

Epoch 00008: early stopping

Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 5e-06, decay = 0.0001, momentum = 0.99, actfn = relu: 0.740475914413

Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 1e-05, decay = 1e-05, momentum = 0.99, actfn = relu: 0.874139852587

Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 1e-05, decay = 5e-05, momentum = 0.99, actfn = relu: 0.871602660231

Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 1e-05, decay = 0.0001, momentum = 0.99, actfn = relu: 0.864990577913

Best Config: architecture = [50, 800, 800, 500, 300, 2], lambda = 1e-06, decay = 1e-05, momentum = 0.99, actfn = relu, best_acc = 0.876023531178

The training time for Grid search with cross-validation 14497.3963192 seconds.

COLLABORATION

Brain stormed and collaborated with Adarsha Desai and Ravishankar Sivaraman for this assignment.