Machine Learning - Homework 4 Solutions:

Name : Achyut Kulkarni

Theory Questions :

1) Boosting :

a) **Gradient Calculation :**

$$L(y_i, y_i') = (y_i - y_i')^2$$

$$g_i = \frac{\partial L(y_i, y_i')}{\partial y_i'} = -2(y_i - y_i')$$

b) **Weak Learner Selection :**

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

We know that :

$$g_i = \frac{\partial L(y_i, y_i')}{\partial y_i'} = -2(y_i - y_i')$$

$$\gamma^* = \frac{\partial h^*}{\partial \gamma} = \frac{\partial (argmin_{h \epsilon H}(min_{\gamma \epsilon R} \sum_{i=1}^{n} (-g_i - \gamma h(x_i))^2)}{\partial \gamma} = \partial (min_{\gamma \epsilon R} \sum_{i=1}^{n} (-g_i - \gamma h(x_i))^2))$$

$$\frac{\partial h^*}{\partial \gamma} = \partial \sum_{i=1}^{n} (2y_i - 2y_i' - \gamma h(x_i))^2) = \sum_{i=1}^{n} (4y_i - 4y_i' - 2\gamma h(x_i))(h(x_i)) = 0$$

$$\gamma^* = \frac{\sum_{i=1}^{n} 2(y_i - y_i')h(x_i)}{\sum_{i=1}^{n} h(x_i))^2}$$

c) **Step Size Selection** :

$$L^* = min_{\alpha \epsilon R} \sum_{i=1}^{n} L(y_i, y_i' + \alpha h^*(x_i))$$

We know that :

$$L(y_i, y_i') = (y_i - y_i')^2 = (y_i - y_i' - \alpha h^*(x_i))^2$$

$$\frac{\partial L}{\partial \alpha} = \sum_{i=1}^{n} 2((y_i - y_i' - \alpha h^*(x_i))(-h^*(x_i)) = 0$$

$$\alpha^* = \frac{\sum_{i=1}^{n} (y_i - y_i')h^*(x_i))}{\sum_{i=1}^{n} h^*(x_i)^2}$$

2)

Soln.:

Given that the neural network has a single logistic output and linear activation functions in the hidden layers, therefore output y can be given by:

$$y = \sigma(z) = \sigma(\sum_k w_k x_k) = \sigma(\sum_k w_k (\sum_i v_{ki} x_i))$$

where $x_k = (\sum_i v_{ki} x_i)$ and x_k is the neurons of the previous hidden layer to output layer

$$\Rightarrow y = \sigma(\sum_k \sum_i w_k v_{ki} x_i)$$

Now, if we keep substituting x_i according to the number of hidden layers, we still will be getting a linear function of inputs; as all hidden layers use linear activation functions.

Putting $\sigma(z) = \frac{1}{1+exp(-z)}$ in the above equation, we get:

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

$$\Rightarrow y = \frac{1}{1+exp(-\sum_i w_i' x_i)} \text{ where } w_i' = \sum_k w_k v_{ki}$$

Hence, the above expression is equivalent to a logistic regression over different combination of weights and would look similar for neural network with multiple hidden layers.

b)

$$L(y, \hat{y}) = \frac{1}{2} \sum_{i=1}^{2} (y_i - \hat{y}_i)^2$$

$$\frac{\partial L}{\partial w_{ki}} = \frac{\partial L}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial z_k} \cdot \frac{\partial z_k}{\partial w_{ki}}$$

$$\frac{\partial L}{\partial \hat{y}_i} = \sum_{i=1}^{2} (y_i - \hat{y}_i)(-1) = \sum_{i=1}^{2} (\hat{y}_i - y_i)$$

$$\frac{\partial \hat{y}_i}{\partial z_k} = v_{jk}$$

and $$\frac{\partial z_k}{\partial w_{ki}} = (tanh'(\sum_{i=1}^{3} w_{ki} x_i))(x_i)$$

$$\Rightarrow \quad = (1 - tanh^2(\sum_{i=1}^{3} w_{ki} x_i))\, x_i$$

Putting the values of all three terms in the equation , we get:

$$\frac{\partial L}{\partial w_{ki}} = \sum_{i=1}^{2} (\hat{y}_i - y_i)\, v_{jk}\, (1 - tanh^2(\sum_{i=1}^{3} w_{ki} x_i))\, x_i$$

Now, the back-propagation updates for estimation of v_{jk} is given by the gradient of v_{jk} :

$$\Rightarrow \quad \frac{\partial L}{\partial v_{jk}} = \frac{\partial L}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial v_{jk}}$$

Now, calculating the terms in above equation, we get:

$$\frac{\partial L}{\partial \hat{y}_i} = (y_i - \hat{y}_i)(-1)$$

$$\frac{\partial \hat{y}_i}{\partial v_{jk}} = z_k$$

Putting the values of both the terms in the equation , we get:

$$\frac{\partial L}{\partial v_{jk}} = -(y_i - \hat{y}_i)\, z_k$$

$$= (\hat{y}_i - y_i)\, z_k$$

Programming : Deep Learning :

(d) Linear activations:

(a) Report :

---------- Part D (1) ----------

Score for architecture = [50, 2] lambda = 0.0 decay = 0.0 momentum = 0.0 actfn = linear: 0.81140198963

Score for architecture = [50, 50, 2] lambda = 0.0 decay = 0.0 momentum = 0.0 actfn = linear: 0.829585209912

Score for architecture = [50, 50, 50, 2] lambda = 0.0 decay = 0.0 momentum = 0.0 actfn = linear: 0.826702033789

Score for architecture = [50, 50, 50, 50, 2] lambda = 0.0 decay = 0.0 momentum = 0.0 actfn = linear: 0.831584205298
Best Config: architecture = [50, 50, 50, 50, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear, best_acc = 0.831584205298
--- 34.4613609314 seconds ---

Here as the number of the Hidden layers increase the accuracies on the test data set is supposed to increase.
As the depth of the neural net increases , accuracy increases ie., the increase in the layers fits the data better and better. As long as the overfitting problem does not arise or the generalisation error is less, the number of increase in the layers in neural network makes sense.

The best architecture here is the deepest one : [50, 50, 50, 50, 2] has the best testing accuracy

---------- Part D (2) ----------
Score for architecture = [50, 2] lambda = 0.0 decay = 0.0 momentum = 0.0 actfn = linear:0.800599704153
Score for architecture = [50, 50, 2] lambda = 0.0 decay = 0.0 momentum = 0.0 actfn = linear: 0.820474379353
Score for architecture = [50, 500, 300, 2] lambda = 0.0 decay = 0.0 momentum = 0.0 actfn = linear: 0.821089454892
Score for architecture = [50, 800, 500, 300, 2] lambda = 0.0 decay = 0.0 momentum = 0.0 actfn = linear: 0.83132010861
Score for architecture = [50, 800, 800,500, 300, 2] lambda = 0.0 decay = 0.0 momentum = 0.0 actfn = linear: 0.841973628756
Best Config: architecture = [50, 800, 800,500, 300, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear, best_acc = 0.841973628756
--- 432.1613609314 seconds ---

As the number of Neurons(Larger) in the Hidden Layers increase, the time required to train the data increases significantly. Also we can say that as the size of each hidden layer decreases with the depth, the network initially has more capacity to converge faster than the smaller hidden layers. The deeper the architecture and the larger the Hidden layers the accuracies improve but also we might have to involve a process called “Pruning” to eliminate excess neurons in the hidden layers which have least weight. Selecting a model in a neural network is mostly trial and error and there is no magic formula to say

The best architecture here is the deepest one : [50, 800, 800,500, 300, 2] has the best testing accuracy

---------- Part E ----------

Score for architecture = [50, 50, 2] lambda = 0.0 decay = 0.0 momentum = 0.0 actfn = sigmoid: 0.758428480163
Score for architecture = [50, 500, 2] lambda = 0.0 decay = 0.0 momentum = 0.0 actfn = sigmoid: 0.796601695544
Score for architecture = [50, 500, 300, 2] lambda = 0.0 decay = 0.0 momentum = 0.0 actfn = sigmoid: 0.758428480163
Score for architecture = [50, 800, 500, 300, 2] lambda = 0.0 decay = 0.0 momentum = 0.0 actfn = sigmoid: 0.768039057197
Score for architecture = [50, 800, 800, 500, 300, 2] lambda = 0.0 decay = 0.0 momentum = 0.0 actfn = sigmoid: 0.784723021202
Best Config: architecture = [50, 500, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = sigmoid, best_acc = 0.796601695544
--- 1762.1613609314 seconds ---

A non-linear activation function is different from Linear activation i., A linear combination of linear classifiers is equivalent to a single linear classifier hence adding new Linear activated Hidden layers is less useful in a neural net. Hence, non-linear functions like sigmoid here gives significant change in the architecture chosen for best testing accuracy. Hence a single hidden layer with sigmoid activation yields better results than multiple layers in this dataset.

The best architecture here : [50,500, 2] has the best testing accuracy

---------- Part F ----------
Score for architecture = [50, 50, 2] lambda = 0.0 decay = 0.0 momentum = 0.0 actfn = relu: 0.801676084837
Score for architecture = [50, 500, 2] lambda = 0.0 decay = 0.0 momentum = 0.0 actfn = relu: 0.816053514295
Score for architecture = [50, 500, 300, 2] lambda = 0.0 decay = 0.0 momentum = 0.0 actfn = relu: 0.809556759243
Score for architecture = [50, 800, 500, 300, 2] lambda = 0.0 decay = 0.0 momentum = 0.0 actfn = relu: 0.805020566919
Score for architecture = [50, 800, 800, 500, 300, 2] lambda = 0.0 decay = 0.0 momentum = 0.0 actfn = relu: 0.785760966445
Best Config: architecture = [50, 500, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = relu, best_acc = 0.816053514295
--- 437.256134033 seconds ---

ReLu gives the best accuracy for the same architecture as sigmoid activation function. However ReLu activation function takes significantly less time compared to sigmoid function and hence is widely used in Hidden layers.
Also the major difference between switching from sigmoid function to a reLu function is “Vanishing gradient” problem i.,e The vanishing gradient problem requires us to use small learning rates with gradient descent which then needs many small steps to converge. Hence, as we switch from sigmoid to reLu functions , the gradient is zero for negative inputs and large for larger inputs which is not the case in sigmoid which causes the significant decrease in time taken to train with reLu activation function.
Also reLu is a non-linear activation function and the accuracy of reLu is better than sigmoid function but performs less than linear function the trend is the same with the amount taken to training.
The trend is due to the the exponential increase in the gradients as the hidden layers increase

The best architecture here : [50,500, 2] has the best testing accuracy

---------- Part G ----------
Score for architecture = [50, 800, 500, 300, 2] lambda = 1e-07 decay = 0.0 momentum = 0.0 actfn = relu: 0.795294662126
Score for architecture = [50, 800, 500, 300, 2] lambda = 5e-07 decay = 0.0 momentum = 0.0 actfn = relu: 0.806289158514
Score for architecture = [50, 800, 500, 300, 2] lambda = 1e-06 decay = 0.0 momentum = 0.0 actfn = relu: 0.800599697772
Score for architecture = [50, 800, 500, 300, 2] lambda = 5e-06 decay = 0.0 momentum = 0.0 actfn = relu: 0.80344443158
Score for architecture = [50, 800, 500, 300, 2] lambda = 1e-05 decay = 0.0 momentum = 0.0 actfn = relu: 0.799254222532
Best Config: architecture = [50, 800, 500, 300, 2] , lambda = 5e-07, decay = 0.0, momentum = 0.0, actfn = relu, best_acc =0.806289158514

As the regularization parameter increases there is a fluctuation in the test accuracies obtained. This could be due to the underfitting and optimal fitting of data. Also it could be due to the behaviour of the combination of reLu activation function and the regularisation parameter.

The best hyperparameter Lambda : 5e-07

---------- Part H ----------
Epoch 00008: early stopping
Score for architecture = [50, 800, 500, 300, 2] lambda = 1e-07 decay = 0.0 momentum = 0.0 actfn = relu: 0.784723021202
Score for architecture = [50, 800, 500, 300, 2] lambda = 5e-07 decay = 0.0 momentum = 0.0 actfn = relu: 0.758428480163
Score for architecture = [50, 800, 500, 300, 2] lambda = 1e-06 decay = 0.0 momentum = 0.0 actfn = relu: 0.801022571566
Epoch 00008: early stopping
Score for architecture = [50, 800, 500, 300, 2] lambda = 5e-06 decay = 0.0 momentum = 0.0 actfn = relu: 0.768039057197
Score for architecture = [50, 800, 500, 300, 2] lambda = 1e-05 decay = 0.0 momentum = 0.0 actfn = relu: 0.796601695544

Early stopping and L2-regularisation are the methods for reducing overfitting . it is 'redundant' to use both L2-regularisation and early stopping together .Early stopping essentially works because the weights start small, and L2 reg tries to keep them small. Hence we can see the trend that for same L2-regularisation parameter the accuracy with the early stopping is reducing hence it does not help in improving the accuracy.

The changes L2-regularisation parameter is : 1e-06

---------- Part I ----------
Score for architecture = [50, 800, 500, 300, 2] lambda = 5e-07 decay = 1e-05 momentum = 0.0 actfn = relu: 0.784588475223
Score for architecture = [50, 800, 500, 300, 2] lambda = 5e-07 decay = 5e-05 momentum = 0.0 actfn = relu: 0.811017563547
Score for architecture = [50, 800, 500, 300, 2] lambda = 5e-07 decay = 0.0001 momentum = 0.0 actfn = relu: 0.668857886392
Score for architecture = [50, 800, 500, 300, 2] lambda = 5e-07 decay = 0.0003 momentum = 0.0 actfn = relu: 0.700111487966,
Score for architecture = [50, 800, 500, 300, 2] lambda = 5e-07 decay = 0.0007 momentum = 0.0 actfn = relu: 0.744743015286
Score for architecture = [50, 800, 500, 300, 2] lambda = 5e-07 decay = 0.001 momentum = 0.0 actfn = relu: 0.798100954926
Best Config: architecture = [50, 800, 500, 300, 2], lambda = 5e-07, decay = 5e-05, momentum = 0.0, actfn = relu, best_acc =0.811017563547

The best decay value for the architecture is : 5e-05

---------- Part J ----------
Score for architecture = [50, 800, 500, 300, 2] lambda = 0.0 decay = 5e-05 momentum = 0.99 actfn = relu: 0.848806364474
Score for architecture = [50, 800, 500, 300, 2] lambda = 0.0 decay = 5e-05 momentum = 0.98 actfn = relu: 0.829431435749
Score for architecture = [50, 800, 500, 300, 2] lambda = 0.0 decay = 5e-05 momentum = 0.95 actfn = relu: 0.780148384765

Score for architecture = [50, 800, 500, 300, 2] lambda = 0.0 decay = 5e-05 momentum = 0.9 actfn = relu: 0.755276208992
Score for architecture = [50, 800, 500, 300, 2] lambda = 0.0 decay = 5e-05 momentum = 0.85 actfn = relu: 0.772613700509
Best Config: architecture = [50, 800, 500, 300, 2], lambda = 0.0, decay = 5e-05, momentum = 0.99, actfn = relu, best_acc = 0.848806364474

The best Momentum value for the architecture is : 0.99

---------- Part K ----------
Epoch 00008: early stopping
Score for architecture = [50, 800, 500, 300, 2] lambda = 1e-07 decay = 5e-05 momentum = 0.99 actfn = relu: 0.748933226304
Best Config: architecture = [50, 800, 500, 300, 2], lambda = 1e-07, decay = 5e-05, momentum = 0.99, actfn = relu, best_acc = 0.748933226304

The test accuracies has reduced when we combine all the parts .
---------- Part L ----------
Score for architecture = [50, 50, 2] lambda = 1e-07 decay = 1e-05 momentum = 0.99 actfn = relu: 0.849113905927
Score for architecture = [50, 500, 2] lambda = 1e-07 decay = 1e-05 momentum = 0.99 actfn = relu: 0.850805366733
Score for architecture = [50, 500, 300, 2] lambda = 1e-07 decay = 1e-05 momentum = 0.99 actfn = relu: 0.863645104965
Score for architecture = [50, 800, 500, 300, 2] lambda = 1e-07 decay = 1e-05 momentum = 0.99 actfn = relu: 0.869219237949
Score for architecture = [50, 800, 800, 500, 300, 2] lambda = 1e-07 decay = 1e-05 momentum = 0.99 actfn = relu: 0.873640102514
Score for architecture = [50, 50, 2] lambda = 1e-07 decay = 5e-05 momentum = 0.99 actfn = relu: 0.848345058022
Score for architecture = [50, 500, 2] lambda = 1e-07 decay = 5e-05 momentum = 0.99 actfn = relu: 0.849075459522
Score for architecture = [50, 500, 300, 2] lambda = 1e-07 decay = 5e-05 momentum = 0.99 actfn = relu: 0.85564910051
Score for architecture = [50, 800, 500, 300, 2] lambda = 1e-07 decay = 5e-05 momentum = 0.99 actfn = relu: 0.861838307235
Score for architecture = [50, 800, 800, 500, 300, 2] lambda = 1e-07 decay = 5e-05 momentum = 0.99 actfn = relu: 0.866951138351
Epoch 00015: early stopping
Score for architecture = [50, 50, 2] lambda = 1e-07 decay = 0.0001 momentum = 0.99 actfn = relu: 0.823434432568
Epoch 00098: early stopping
Score for architecture = [50, 500, 2] lambda = 1e-07 decay = 0.0001 momentum = 0.99 actfn = relu: 0.845615656555
Score for architecture = [50, 500, 300, 2] lambda = 1e-07 decay = 0.0001 momentum = 0.99 actfn = relu: 0.855418444993
Score for architecture = [50, 800, 500, 300, 2] lambda = 1e-07 decay = 0.0001 momentum = 0.99 actfn = relu: 0.859032019339
Epoch 00007: early stopping
Score for architecture = [50, 800, 800, 500, 300, 2] lambda = 1e-07 decay = 0.0001 momentum = 0.99 actfn = relu: 0.720409020122
Score for architecture = [50, 50, 2] lambda = 5e-07 decay = 1e-05 momentum = 0.99 actfn = relu: 0.847268670957
Score for architecture = [50, 500, 2] lambda = 5e-07 decay = 1e-05 momentum = 0.99 actfn = relu: 0.850574713508
Score for architecture = [50, 500, 300, 2] lambda = 5e-07 decay = 1e-05 momentum = 0.99 actfn = relu: 0.865797869931

Epoch 00008: early stopping
Score for architecture = [50, 800, 500, 300, 2] lambda = 5e-07 decay = 1e-05 momentum = 0.99 actfn = relu: 0.760350596486
Epoch 00007: early stopping
Score for architecture = [50, 800, 800, 500, 300, 2] lambda = 5e-07 decay = 1e-05 momentum = 0.99 actfn = relu: 0.735017107881
Score for architecture = [50, 50, 2] lambda = 5e-07 decay = 5e-05 momentum = 0.99 actfn = relu: 0.84242494127
Epoch 00011: early stopping
Score for architecture = [50, 500, 2] lambda = 5e-07 decay = 5e-05 momentum = 0.99 actfn = relu: 0.815553759147
Score for architecture = [50, 500, 300, 2] lambda = 5e-07 decay = 5e-05 momentum = 0.99 actfn = relu: 0.855649105092
Score for architecture = [50, 800, 500, 300, 2] lambda = 5e-07 decay = 5e-05 momentum = 0.99 actfn = relu: 0.866912696528
Epoch 00007: early stopping
Score for architecture = [50, 800, 800, 500, 300, 2] lambda = 5e-07 decay = 5e-05 momentum = 0.99 actfn = relu: 0.726828891529
Epoch 00009: early stopping
Score for architecture = [50, 50, 2] lambda = 5e-07 decay = 0.0001 momentum = 0.99 actfn = relu: 0.808095960827
Score for architecture = [50, 500, 2] lambda = 5e-07 decay = 0.0001 momentum = 0.99 actfn = relu: 0.847999083912
Score for architecture = [50, 500, 300, 2] lambda = 5e-07 decay = 0.0001 momentum = 0.99 actfn = relu: 0.853573214605
Score for architecture = [50, 800, 500, 300, 2] lambda = 5e-07 decay = 0.0001 momentum = 0.99 actfn = relu: 0.858762922492
Epoch 00008: early stopping
Score for architecture = [50, 800, 800, 500, 300, 2] lambda = 5e-07 decay = 0.0001 momentum = 0.99 actfn = relu: 0.734824889111
Score for architecture = [50, 50, 2] lambda = 1e-06 decay = 1e-05 momentum = 0.99 actfn = relu: 0.845039017763
Score for architecture = [50, 500, 2] lambda = 1e-06 decay = 1e-05 momentum = 0.99 actfn = relu: 0.8535347682
Score for architecture = [50, 500, 300, 2] lambda = 1e-06 decay = 1e-05 momentum = 0.99 actfn = relu: 0.862684038784
Score for architecture = [50, 800, 500, 300, 2] lambda = 1e-06 decay = 1e-05 momentum = 0.99 actfn = relu: 0.869565216642
Score for architecture = [50, 800, 800, 500, 300, 2] lambda = 1e-06 decay = 1e-05 momentum = 0.99 actfn = relu: 0.875446891079
Score for architecture = [50, 50, 2] lambda = 1e-06 decay = 5e-05 momentum = 0.99 actfn = relu: 0.845807862883
Score for architecture = [50, 500, 2] lambda = 1e-06 decay = 5e-05 momentum = 0.99 actfn = relu: 0.848960138637
Score for architecture = [50, 500, 300, 2] lambda = 1e-06 decay = 5e-05 momentum = 0.99 actfn = relu: 0.858724492126
Score for architecture = [50, 800, 500, 300, 2] lambda = 1e-06 decay = 5e-05 momentum = 0.99 actfn = relu: 0.864875257029
Epoch 00008: early stopping
Score for architecture = [50, 800, 800, 500, 300, 2] lambda = 1e-06 decay = 5e-05 momentum = 0.99 actfn = relu: 0.744243266691
Epoch 00019: early stopping
Score for architecture = [50, 50, 2] lambda = 1e-06 decay = 0.0001 momentum = 0.99 actfn = relu: 0.823934185916
Epoch 00010: early stopping
Score for architecture = [50, 500, 2] lambda = 1e-06 decay = 0.0001 momentum = 0.99 actfn = relu: 0.812516823101
Epoch 00094: early stopping

Score for architecture = [50, 500, 300, 2] lambda = 1e-06 decay = 0.0001 momentum = 0.99 actfn = relu: 0.849805865111
Score for architecture = [50, 800, 500, 300, 2] lambda = 1e-06 decay = 0.0001 momentum = 0.99 actfn = relu: 0.861377007658
Score for architecture = [50, 800, 800, 500, 300, 2] lambda = 1e-06 decay = 0.0001 momentum = 0.99 actfn = relu: 0.86372198861
Score for architecture = [50, 50, 2] lambda = 5e-06 decay = 1e-05 momentum = 0.99 actfn = relu: 0.851920188747
Score for architecture = [50, 500, 2] lambda = 5e-06 decay = 1e-05 momentum = 0.99 actfn = relu: 0.8501902925
Score for architecture = [50, 500, 300, 2] lambda = 5e-06 decay = 1e-05 momentum = 0.99 actfn = relu: 0.863875753115
Score for architecture = [50, 800, 500, 300, 2] lambda = 5e-06 decay = 1e-05 momentum = 0.99 actfn = relu: 0.865990083626
Epoch 00008: early stopping
Score for architecture = [50, 800, 800, 500, 300, 2] lambda = 5e-06 decay = 1e-05 momentum = 0.99 actfn = relu: 0.74105255419
Score for architecture = [50, 50, 2] lambda = 5e-06 decay = 5e-05 momentum = 0.99 actfn = relu: 0.848614157653
Score for architecture = [50, 500, 2] lambda = 5e-06 decay = 5e-05 momentum = 0.99 actfn = relu: 0.848767922651
Score for architecture = [50, 500, 300, 2] lambda = 5e-06 decay = 5e-05 momentum = 0.99 actfn = relu: 0.857455891366
Score for architecture = [50, 800, 500, 300, 2] lambda = 5e-06 decay = 5e-05 momentum = 0.99 actfn = relu: 0.865567216213
Score for architecture = [50, 800, 800, 500, 300, 2] lambda = 5e-06 decay = 5e-05 momentum = 0.99 actfn = relu: 0.869642100287
Score for architecture = [50, 50, 2] lambda = 5e-06 decay = 0.0001 momentum = 0.99 actfn = relu: 0.834198283589
Score for architecture = [50, 500, 2] lambda = 5e-06 decay = 0.0001 momentum = 0.99 actfn = relu: 0.847806865635
Epoch 00009: early stopping
Score for architecture = [50, 500, 300, 2] lambda = 5e-06 decay = 0.0001 momentum = 0.99 actfn = relu: 0.797485868915
Score for architecture = [50, 800, 500, 300, 2] lambda = 5e-06 decay = 0.0001 momentum = 0.99 actfn = relu: 0.860838817562
Epoch 00008: early stopping
Score for architecture = [50, 800, 800, 500, 300, 2] lambda = 5e-06 decay = 0.0001 momentum = 0.99 actfn = relu: 0.761926729534
Score for architecture = [50, 50, 2] lambda = 1e-05 decay = 1e-05 momentum = 0.99 actfn = relu: 0.848729485412
Score for architecture = [50, 500, 2] lambda = 1e-05 decay = 1e-05 momentum = 0.99 actfn = relu: 0.850805371316
Score for architecture = [50, 500, 300, 2] lambda = 1e-05 decay = 1e-05 momentum = 0.99 actfn = relu: 0.865105907471
Score for architecture = [50, 800, 500, 300, 2] lambda = 1e-05 decay = 1e-05 momentum = 0.99 actfn = relu: 0.867835311722
Epoch 00008: early stopping
Score for architecture = [50, 800, 800, 500, 300, 2] lambda = 1e-05 decay = 1e-05 momentum = 0.99 actfn = relu: 0.734440469088
Score for architecture = [50, 50, 2] lambda = 1e-05 decay = 5e-05 momentum = 0.99 actfn = relu: 0.84684581091
Score for architecture = [50, 500, 2] lambda = 1e-05 decay = 5e-05 momentum = 0.99 actfn = relu: 0.846845804036
Epoch 00008: early stopping
Score for architecture = [50, 500, 300, 2] lambda = 1e-05 decay = 5e-05 momentum = 0.99 actfn = relu: 0.785837845508
Score for architecture = [50, 800, 500, 300, 2] lambda = 1e-05 decay = 5e-05 momentum = 0.99 actfn = relu: 0.863414440283

Score for architecture = [50, 800, 800, 500, 300, 2] lambda = 1e-05 decay = 5e-05 momentum = 0.99 actfn = relu: 0.867643100319
Score for architecture = [50, 50, 2] lambda = 1e-05 decay = 0.0001 momentum = 0.99 actfn = relu: 0.843808865699
Epoch 00009: early stopping
Score for architecture = [50, 500, 2] lambda = 1e-05 decay = 0.0001 momentum = 0.99 actfn = relu: 0.813324105954
Score for architecture = [50, 500, 300, 2] lambda = 1e-05 decay = 0.0001 momentum = 0.99 actfn = relu: 0.855110908122
Score for architecture = [50, 800, 500, 300, 2] lambda = 1e-05 decay = 0.0001 momentum = 0.99 actfn = relu: 0.859916202368
Score for architecture = [50, 800, 800, 500, 300, 2] lambda = 1e-05 decay = 0.0001 momentum = 0.99 actfn = relu: 0.865797869931
Best Config: architecture = [50, 800, 800, 500, 300, 2], lambda = 1e-06, decay = 1e-05, momentum = 0.99, actfn = relu, best_acc =0.875446891079

Here, the best parameters found :

L2-regularisation : 1e-06
decay = 1e-05
momentum = 0.99
best_acc =0.875446891079