ML Assignment3

Arpita Agrawal, USC ID: 8100884538

November 2016

1 Boosting

(a) Gradient Calculation

$\mathcal{L}(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$, gradient g_i w.r.t the current predictions $\hat{y}_i$ on each instance =

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

(b) Weak Learner Selection

$h^* = argmin_{h \in H}(min_{\in R} \sum_{i=1}^{n}(-g_i - \gamma h(x_i))^2$, $\frac{\partial h^*}{\partial \gamma} = 0$

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

So, with the above value of γ, we get a closed form solution for γ and thus the selection rule for h^* can be derived independent of γ

(c) Step Size Selection

Step size that minimizes the loss:

$\alpha^* = argmin_{\alpha \in \mathcal{R}} \sum_{i=1}^{n}(y_i - \hat{y}_i - \alpha h^*(x_i))^2,$ $\frac{\partial \mathcal{L}}{\partial \alpha}(\sum_{i=1}^{n}(y_i - \hat{y}_i - \alpha h^*(x_i))^2) = \sum_{i=1}^{n} -2(y_i - \hat{y}_i - \alpha h^*(x_i))(h^*(x_i))$

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

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

2 Neural Networks

(a) Neural Network with Single logistic output Neuron

Suppose we have, N input nodes in our Input Layer, M hidden Layers with k nodes in each of the M hidden layer, and 1 logistic node in our output Layer.

Let $z_k^{M_j}$ be the kth node in our hidden Layer $M_j, w_k^{M_j}$ be the weight of kth node.

$z_k^{M_1} = \sum_{i=1}^{N} w_{ki}^{M_1} x_i$ - for hidden layer 1

$z_k^{M_2} = \sum_{i=1}^{N} w_{ki}^{M_2} z_i$ - for hidden layer 2

$z_k^{M_j} = \sum_{i=1}^{N} w_{ki}^{M_j} z_i$ - for hidden layer j

$y = \sigma(\sum_{j=0}^{L_{M-1}} w_{1j}^{L_M} z_i)$ - output y, given by one logistic node (i.e Node 1 in output layer) depends on Mth hidden Layer

$y = \sigma(\sum_{j=0}^{L_{M-1}} w_{1j}^{L_M} z_i) = \sigma(\sum_{j=0}^{L_{M-1}} w_{1j}^{L_M})(\sum_{p=0}^{L_{M-2}} w_{jp}^{L_{M-1}} z_p)$ - and so on z can be replaced again and again till we reach the input layer

$y = \sigma(WX)$ - which is in turn equal to how logistic regression predicts. Therefore a neural network with a single logistic output and with linear activation functions in the hidden layers (possibly with multiple hidden layers) is equivalent to the logistic regression.

(b) Neural Network backpropogation updates

We have a neural network with 3 layers. i.e. Input Layer with 3 inputs x_i, 1 hidden Layer(4 inputs z_k) and 1 output Layer with 2 outputs y_j

$z_k = tanh(\sum_{i=1}^{3} w_{ki}x_i), \quad y_j =_{k=1}^{4} v_{jk}z_k, \quad \mathcal{L}(y, \hat{y}) = \frac{1}{2}((y_1 - \hat{y}_2)^2 + (y_2 - \hat{y}_2)^2)$

where w_{ki} are weights going from input layer ith node to kth node in hidden layer. and v_{jk} are the weights going from kth node of hidden layer to jth node of output layer.

First differentiating the Cost function $\mathcal{L}(y, \hat{y})$ w.r.t v_{jk}, we get,

Our cost function can also be written as $\mathcal{L}(y, \hat{y}) = \sum_{j=1}^{2} \frac{1}{2}(y_j - \hat{y}_j)^2$

$\frac{\partial \mathcal{L}}{\partial v_{jk}} = -\frac{2}{2}(y_j - \hat{y}_j)\frac{\partial(\sum_{j=1}^{2} v_{jk}z_k)}{\partial v_{jk}}$

Since, the derivation is w.r.t v_{jk}, we are differentiating w.r.t to one specific jth node in the output layer, so the summation can be removed as all the other j terms will become zero.

$\frac{\partial \mathcal{L}}{\partial v_{jk}} = \frac{2}{2}(y_j - \hat{y}_j)\frac{\partial(v_{jk}z_k)}{\partial v_{jk}}$ where $z_k = tanh(\sum_{i=1}^{3} w_{ki}x_i)$ which is independent of v_{jk}

So, $\frac{\partial \mathcal{L}}{\partial v_{jk}} = -(y_j - \hat{y}_j)z_k$

Then differentiating the Cost function $\mathcal{L}(y, \hat{y})$ w.r.t w_{ki}, we get,

$\mathcal{L}(y, \hat{y}) = \sum_{j=1}^{2} \frac{1}{2}(y_j - \hat{y}_j)^2, \quad$ Using, $\frac{\partial tanh(x)}{\partial x} = 1 - tan^2(x)$

$\frac{\partial L}{\partial w_{ki}} = \frac{\partial L}{\partial \hat{y}_j}\frac{\partial \hat{y}_j}{\partial wS_{jk}}\frac{\partial wS_{jk}}{\partial z_k}\frac{\partial z_k}{\partial w_{ki}}$

$\frac{\partial \mathcal{L}}{\partial w_{ki}} = \sum_{j=1}^{2} -\frac{2}{2}(y_j - \hat{y}_j)\frac{\partial(v_{jk}z_k)}{\partial v_{jk}}$, where $z_k = tanh(\sum_{i=1}^{3} w_{ki}x_i)$, which is dependent on w_{ki}

$\frac{\partial \mathcal{L}}{\partial w_{ki}} = -(y_j - \hat{y}_j)v_{jk}\frac{\partial z_k}{\partial w_{ki}} \quad = \quad -(y_j - \hat{y}_j)v_{jk}\frac{\partial tanh(\sum_{i=1}^{3} w_{ki}x_i)}{\partial w_{ki}} \quad = \quad$ -$\sum_{j=1}^{2}(y_j - \hat{y}_j)v_{jk}(1 - tan^2(\sum_{i=1}^{3} w_{ki}x_i))x_i$

Thus, the update rule becomes :

$w_{ki} = w_{ki} + \eta \sum_{j=1}^{2} v_{jk}(y_j - \hat{y}_j)(1 - tan^2(\sum_{i=1}^{3} w_{ki}x_i))x_i$

$v_{jk} = v_{jk} + \eta(y_j - \hat{y}_j)z_k$

3 Programming: Deep Learning

```
d) Linear Activations


 Score for architecture = [50, 2], lambda = 0.0 , decay = 0.0, momentum = 0.0, actfn = linear:
 0.830623146483
Score for architecture = [50, 50, 2], lambda = 0.0 , decay = 0.0, momentum = 0.0, actfn = linear:
0.840848810513
Score for architecture = [50, 50, 50, 2], lambda = 0.0 , decay = 0.0, momentum = 0.0, actfn = linear:
0.840349060441
Score for architecture = [50, 50, 50, 50, 2], lambda = 0.0 , decay = 0.0, momentum = 0.0, actfn = linear:
0.848614150286
Best Config: architecture = [50, 50, 50, 50, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear,
best_acc = 0.848614150286
Time taken to train: 28.702406168

With the above observation, we see that the test accuracy increases very minutely and decreases a bit
in the middle. Effects of Adding more number of layers in the network depends on problem to problem.
Generally the netwroks are kept shallow going upto 2 layers with more number of neuron in each layer.
Though, some problems require deep networks, where more complex features can be developed. The problem,
in our hand seems to be working with more number of layers.



Experiment part d(b) with linear activation and different architectures
Score for architecture = [50, 50, 2], lambda = 0.0 , decay = 0.0, momentum = 0.0, actfn = linear:
0.834813364696

Score for architecture = [50, 500, 2], lambda = 0.0 , decay = 0.0, momentum = 0.0, actfn = linear:
0.839118903794
Score for architecture = [50, 500, 300, 2], lambda = 0.0 , decay = 0.0, momentum = 0.0, actfn = linear:
0.845846309781
```

Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0 , decay = 0.0, momentum = 0.0, actfn = linear:
0.846922684897
Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 0.0 , decay = 0.0, momentum = 0.0, actfn = lin
0.848268173885
Best Config: architecture = [50, 800, 800, 500, 300, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = l
best_acc = 0.848268173885
Time taken to train: 246.955076933 increase with more number of neurons in hidden layers as expected.
It takes more time to train more number of neurons in comparisions to lesser number of neurons in each layer.

With the above observation, we see that the test accuracy follows a similar pattern like the one in the above observation with lesser number of neurons .Adding more number of nodes in the hidden layers increases the accuracy a bit. The reason behind it can be, more number of combinations can now be tried at the same time on our model now. So, adding more number of neurons in the network help in the problem in our hand. Though the difference isnt too much.

Experiment part e with sigmoid activation (e) part
Score for architecture = [50, 50, 2], lambda = 0.0 , decay = 0.0, momentum = 0.0, actfn = sigmoid:
0.731749504861
Score for architecture = [50, 500, 2], lambda = 0.0 , decay = 0.0, momentum = 0.0, actfn = sigmoid:
0.761542311803
Score for architecture = [50, 500, 300, 2], lambda = 0.0 , decay = 0.0, momentum = 0.0, actfn = sigmoid:
0.718832896732
Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0 , decay = 0.0, momentum = 0.0, actfn = sigmoid:
0.718832896732
Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 0.0 , decay = 0.0, momentum = 0.0, actfn = sig
0.718832896732
Best Config: architecture = [50, 500, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = sigmoid,
best_acc = 0.761542311803
Time taken to train: 907.781808138. Sigmoid is a complex function and hence it takes a lot of computation to calculate the gradient descent.

Using sigmoid activation function reduces down the accuracy in comparison to the above observations because it tries to over fit the data on training data set. The accuracies increase upto the second hidden layer addition after which it reduces again. As the number of layers increase in the neural network, the over-fitting increases in the network. In linear activation function we were trying to fit the data using simpler curves, but when we use sigmoid function we are trying to fit the data with more complex function and we are not using any regularisation to prevent the over-fitting. Hence a decrease in accuracy is observed.

Experiment part e with Relu activation (f) part
Score for architecture = [50, 50, 2], lambda = 0.0 , decay = 0.0, momentum = 0.0, actfn = relu:
0.815361550528
Score for architecture = [50, 500, 2], lambda = 0.0 , decay = 0.0, momentum = 0.0, actfn = relu:
0.817129901853
Score for architecture = [50, 500, 300, 2], lambda = 0.0 , decay = 0.0, momentum = 0.0, actfn = relu:
0.809787412961
Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0 , decay = 0.0, momentum = 0.0, actfn = relu:
0.802022064023
Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 0.0 , decay = 0.0, momentum = 0.0, actfn = rel
0.762272705936
Best Config: architecture = [50, 500, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = relu,
best_acc = 0.817129901853
Time taken to train: 395.356348991 which is lesser than the time taken to train with Sigmoid activation function. Sigmoid is a more complex function and hence takes a lot of computation to calculate the gradient descent. So, the time taken in Relu is less in comparisions to Sigmoid.
But, will take more time than the time taken to train for linear activation functions.

Using relu activation function works well with less deeper neural network - i.e. accuracies are good upto 2 layers of hidden layers after which it starts reducing.
Here also, i feel overfitting is happening, as the accuracy reduces down with more complex models.

Experiment part e with L2 Regularization activation (g) part
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-07 , decay = 0.0, momentum = 0.0, actfn = relu: 0.797639636697
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-07 , decay = 0.0, momentum = 0.0, actfn = relu: 0.80890323959
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-06 , decay = 0.0, momentum = 0.0, actfn = relu: 0.810171835768
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-06 , decay = 0.0, momentum = 0.0, actfn = relu: 0.813362548269
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-05 , decay = 0.0, momentum = 0.0, actfn =

relu: 0.808057512624
Best Config: architecture = [50, 800, 500, 300, 2], lambda = 5e-06, decay = 0.0, momentum = 0.0, actfn = relu, best_acc = 0.813362548269

After applying L2 Regularization for different lamda values the accuracy seems to be increasing with increasing lambda value, which just gives us an insight into the how with increase in regularisation parameter the accuracy improves upto a point and then it starts decreasing.

Experiment part e with Early Stopping and L2 Regularization activation (h) part
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-07 , decay = 0.0, momentum = 0.0, actfn = relu: 0.793449436814
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-07 , decay = 0.0, momentum = 0.0, actfn = relu: 0.808941683704
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-06 , decay = 0.0, momentum = 0.0, actfn = relu: 0.803213778847
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-06 , decay = 0.0, momentum = 0.0, actfn = relu: 0.802175838186
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-05 , decay = 0.0, momentum = 0.0, actfn = relu: 0.805558762091
Best Config: architecture = [50, 800, 500, 300, 2], lambda = 5e-07, decay = 0.0, momentum = 0.0, actfn = relu, best_acc = 0.808941683704
Time taken to train: 582.724961996

After applying early stopping the accuracy reduced a bit. So, even though using early stopping we were trying to reduce down the overfitting, we ended up stoppin earlier than desired. The best lamda value now obtained changed from before, it comes out to be 5e-07.

Experiment part SGD with weight decay (i) part
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-07 , decay = 1e-05, momentum = 0.0, actfn = relu: 0.779763967034
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-07 , decay = 5e-05, momentum = 0.0, actfn = relu: 0.747203325967
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-07 , decay = 0.0001, momentum = 0.0, actfn = relu: 0.773728526613
^[Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-07 , decay = 0.0003, momentum = 0.0, actfn = relu: 0.725675621312
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-07 , decay = 0.0007, momentum = 0.0, actfn = relu: 0.638719104034
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-07 , decay = 0.001, momentum = 0.0, actfn = relu: 0.71879445491
Best Config: architecture = [50, 800, 500, 300, 2], lambda = 5e-07, decay = 1e-05, momentum = 0.0, actfn =
relu, best_acc = 0.779763967034
Time taken to train: 2506.75295496

In general when the value of weight decay increased on SGD, the accuracy reduces as deacy increases.

Experiment part Momentum (j) part
moment 1e-05
Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0 , decay = 1e-05, momentum = 0.99, actfn = relu: 0.85030561846
Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0 , decay = 1e-05, momentum = 0.98, actfn = relu: 0.829700528998
Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0 , decay = 1e-05, momentum = 0.95, actfn = relu: 0.779225772355
Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0 , decay = 1e-05, momentum = 0.9, actfn = relu: 0.742474917657
Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0 , decay = 1e-05, momentum = 0.85, actfn = relu: 0.729788949006
Best Config: architecture = [50, 800, 500, 300, 2], lambda = 0.0, decay = 1e-05, momentum = 0.99, actfn = relu, best_acc = 0.85030561846
Time taken to train: 1036.32893181

The best momentum that gives us the best accuracies is 0.99

Combining the above: (k) part

Grid search with cross-validation
Score for architecture = [50, 50, 2], lambda = 1e-07 , decay = 1e-05, momentum = 0.99, actfn = relu: 0.837350550178
Score for architecture = [50, 50, 2], lambda = 1e-07 , decay = 5e-05, momentum = 0.99, actfn = relu: 0.848537268933
Epoch 00015: early stopping
Score for architecture = [50, 50, 2], lambda = 1e-07 , decay = 0.0001, momentum = 0.99, actfn = relu: 0.82101257174
Epoch 00082: early stopping

```
Score for architecture = [50, 50, 2], lambda = 5e-07 , decay = 1e-05, momentum = 0.99, actfn = relu:
0.847345556893
Score for architecture = [50, 50, 2], lambda = 5e-07 , decay = 5e-05, momentum = 0.99, actfn = relu:
0.835543763904
Score for architecture = [50, 50, 2], lambda = 5e-07 , decay = 0.0001, momentum = 0.99, actfn = relu:
0.846038522983
Epoch 00078: early stopping
Score for architecture = [50, 50, 2], lambda = 1e-06 , decay = 1e-05, momentum = 0.99, actfn = relu:
0.848921696322
Score for architecture = [50, 50, 2], lambda = 1e-06 , decay = 5e-05, momentum = 0.99, actfn = relu:
0.851535771016
Epoch 00077: early stopping
Score for architecture = [50, 50, 2], lambda = 1e-06 , decay = 0.0001, momentum = 0.99, actfn = relu:
0.843731977964
Score for architecture = [50, 50, 2], lambda = 5e-06 , decay = 1e-05, momentum = 0.99, actfn = relu:
0.841233232014
Score for architecture = [50, 50, 2], lambda = 5e-06 , decay = 5e-05, momentum = 0.99, actfn = relu:
0.838388495421
Score for architecture = [50, 50, 2], lambda = 5e-06 , decay = 0.0001, momentum = 0.99, actfn = relu:
0.839387994752
Score for architecture = [50, 50, 2], lambda = 1e-05 , decay = 1e-05, momentum = 0.99, actfn = relu:
0.836697039198
Score for architecture = [50, 50, 2], lambda = 1e-05 , decay = 5e-05, momentum = 0.99, actfn = relu:
0.841310119749
Epoch 00084: early stopping
Score for architecture = [50, 50, 2], lambda = 1e-05 , decay = 0.0001, momentum = 0.99, actfn = relu:
0.839349552437
Score for architecture = [50, 500, 2], lambda = 1e-07 , decay = 1e-05, momentum = 0.99, actfn = relu:
0.850997573554
Score for architecture = [50, 500, 2], lambda = 1e-07 , decay = 5e-05, momentum = 0.99, actfn = relu:
0.84623073717
Score for architecture = [50, 500, 2], lambda = 1e-07 , decay = 0.0001, momentum = 0.99, actfn =
relu: 0.845039020054
Score for architecture = [50, 500, 2], lambda = 5e-07 , decay = 1e-05, momentum = 0.99, actfn = relu:
0.8483066162
Score for architecture = [50, 500, 2], lambda = 5e-07 , decay = 5e-05, momentum = 0.99, actfn = relu:
0.848345055731
Score for architecture = [50, 500, 2], lambda = 5e-07 , decay = 0.0001, momentum = 0.99, actfn =
relu: 0.846153842068
Epoch 00009: early stopping
Score for architecture = [50, 500, 2], lambda = 1e-06 , decay = 1e-05, momentum = 0.99, actfn = relu:
0.811709529284
Epoch 00084: early stopping
Score for architecture = [50, 500, 2], lambda = 1e-06 , decay = 5e-05, momentum = 0.99, actfn = relu:
0.84353976836
Epoch 00011: early stopping
Score for architecture = [50, 500, 2], lambda = 1e-06 , decay = 0.0001, momentum = 0.99, actfn =
relu: 0.813362543686
Score for architecture = [50, 500, 2], lambda = 5e-06 , decay = 1e-05, momentum = 0.99, actfn = relu:
0.849921197452
Epoch 00010: early stopping
Score for architecture = [50, 500, 2], lambda = 5e-06 , decay = 5e-05, momentum = 0.99, actfn = relu:
0.808980127817
Score for architecture = [50, 500, 2], lambda = 5e-06 , decay = 0.0001, momentum = 0.99, actfn =
relu: 0.843731979763
Score for architecture = [50, 500, 2], lambda = 1e-05 , decay = 1e-05, momentum = 0.99, actfn = relu:
0.852266172023
Score for architecture = [50, 500, 2], lambda = 1e-05 , decay = 5e-05, momentum = 0.99, actfn = relu:
0.849459886911
Epoch 00079: early stopping
Score for architecture = [50, 500, 2], lambda = 1e-05 , decay = 0.0001, momentum = 0.99, actfn =
relu: 0.842232728068
Score for architecture = [50, 500, 300, 2], lambda = 1e-07 , decay = 1e-05, momentum = 0.99, actfn =
relu: 0.859877744506
Score for architecture = [50, 500, 300, 2], lambda = 1e-07 , decay = 5e-05, momentum = 0.99, actfn =
relu: 0.857878753703
Score for architecture = [50, 500, 300, 2], lambda = 1e-07 , decay = 0.0001, momentum = 0.99, actfn =
relu: 0.854265176574
Score for architecture = [50, 500, 300, 2], lambda = 5e-07 , decay = 1e-05, momentum = 0.99, actfn =
relu: 0.854880252605
Epoch 00008: early stopping
Score for architecture = [50, 500, 300, 2], lambda = 5e-07 , decay = 5e-05, momentum = 0.99, actfn =
relu: 0.787798401855
Score for architecture = [50, 500, 300, 2], lambda = 5e-07 , decay = 0.0001, momentum = 0.99, actfn =
relu: 0.855841305039
Score for architecture = [50, 500, 300, 2], lambda = 1e-06 , decay = 1e-05, momentum = 0.99, actfn =
relu: 0.864798368802
```

```
Score for architecture = [50, 500, 300, 2], lambda = 1e-06 , decay = 5e-05, momentum = 0.99, actfn =
relu: 0.857494330897
Score for architecture = [50, 500, 300, 2], lambda = 1e-06 , decay = 0.0001, momentum = 0.99, actfn =
relu: 0.852227732492
Score for architecture = [50, 500, 300, 2], lambda = 5e-06 , decay = 1e-05, momentum = 0.99, actfn =
relu: 0.860454383299
Score for architecture = [50, 500, 300, 2], lambda = 5e-06 , decay = 5e-05, momentum = 0.99, actfn =
relu: 0.861377005366
Score for architecture = [50, 500, 300, 2], lambda = 5e-06 , decay = 0.0001, momentum = 0.99, actfn =
relu: 0.8524583903
Score for architecture = [50, 500, 300, 2], lambda = 1e-05 , decay = 1e-05, momentum = 0.99, actfn =
relu: 0.858686038847
Epoch 00009: early stopping
Score for architecture = [50, 500, 300, 2], lambda = 1e-05 , decay = 5e-05, momentum = 0.99, actfn =
relu: 0.791450436846
Score for architecture = [50, 500, 300, 2], lambda = 1e-05 , decay = 0.0001, momentum = 0.99, actfn =
relu: 0.852035514214
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-07 , decay = 1e-05, momentum = 0.99, actfn =
relu: 0.868296627338
Epoch 00008: early stopping
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-07 , decay = 5e-05, momentum = 0.99, actfn =
relu: 0.777880285659
Epoch 00008: early stopping
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-07 , decay = 0.0001, momentum = 0.99, actfn =
relu: 0.751162877207
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-07 , decay = 1e-05, momentum = 0.99, actfn =
relu: 0.870526273658
Epoch 00008: early stopping
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-07 , decay = 5e-05, momentum = 0.99, actfn =
relu: 0.780302154838
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-07 , decay = 0.0001, momentum = 0.99, actfn =
relu: 0.860685038815
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-06 , decay = 1e-05, momentum = 0.99, actfn =
relu: 0.868527275981
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-06 , decay = 5e-05, momentum = 0.99, actfn =
relu: 0.865182793899
Epoch 00007: early stopping
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-06 , decay = 0.0001, momentum = 0.99, actfn =
relu: 0.772152385385
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-06 , decay = 1e-05, momentum = 0.99, actfn =
relu: 0.869296123885
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-06 , decay = 5e-05, momentum = 0.99, actfn =
relu: 0.862453387849
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-06 , decay = 0.0001, momentum = 0.99, actfn =
relu: 0.859647105029
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-05 , decay = 1e-05, momentum = 0.99, actfn =
relu: 0.869603658464
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-05 , decay = 5e-05, momentum = 0.99, actfn =
relu: 0.862645601544
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-05 , decay = 0.0001, momentum = 0.99, actfn =
relu: 0.857263679963
Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 1e-07 , decay = 1e-05, momentum = 0.99, actfn
relu: 0.872294620401
Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 1e-07 , decay = 5e-05, momentum = 0.99, actfn
relu: 0.8704109459
Epoch 00007: early stopping
Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 1e-07 , decay = 0.0001, momentum = 0.99, actfn
relu: 0.730211818218
Epoch 00008: early stopping
Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 5e-07 , decay = 1e-05, momentum = 0.99, actfn
relu: 0.743089993689
Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 5e-07 , decay = 5e-05, momentum = 0.99, actfn
relu: 0.869834309399
Epoch 00008: early stopping
Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 5e-07 , decay = 0.0001, momentum = 0.99, actfn
relu: 0.732210820477
Epoch 00008: early stopping
Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 1e-06 , decay = 1e-05, momentum = 0.99, actfn
relu: 0.744781456787
Epoch 00007: early stopping
Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 1e-06 , decay = 5e-05, momentum = 0.99, actfn
relu: 0.742090487977
Epoch 00007: early stopping
Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 1e-06 , decay = 0.0001, momentum = 0.99, actfn
relu: 0.738361583089
Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 5e-06 , decay = 1e-05, momentum = 0.99, actfn
relu: 0.872256176287
```

```
Epoch 00007: early stopping
Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 5e-06 , decay = 5e-05, momentum = 0.99, actfn
relu: 0.731672621216
Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 5e-06 , decay = 0.0001, momentum = 0.99, actfn
relu: 0.865182791608
Epoch 00008: early stopping
Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 1e-05 , decay = 1e-05, momentum = 0.99, actfn
relu: 0.740629684979
Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 1e-05 , decay = 5e-05, momentum = 0.99, actfn
relu: 0.868950138319
Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 1e-05 , decay = 0.0001, momentum = 0.99, actfn
relu: 0.864759924688

The best configuration received with grid search is as below, :

Best Config: architecture = [50, 800, 800, 500, 300, 2], lambda = 1e-07, decay = 1e-05, momentum = 0.99, actf
relu, best_acc = 0.872294620401
Time taken to train: 13124.1046131
```

The best accuracy observed was 87.22%