USC CSCI 567 HOMEWORK 4 SOLUTIONS

ThammeGowda Narayanaswamy
EMail: tnarayan@usc.edu USCID: 2074-6694-39

November 1, 2016

1 BOOSTING

Given :
Labels: $y \in \{+1, -1\}$
Input features, $x_i \in \mathbb{R}^d$, for $i = 1, 2, ...n$
Weak learners, $\mathcal{H} = \{h_j, j = 1, 2, ...M\}$
Loss, $L(y_i, \hat{y_i}) = (y_i - \hat{y_i})^2$

1.a Gradient Calculation

$$g_i = \frac{\partial L(y_i, \hat{y_i})}{\partial \hat{y_i}} = \frac{\partial}{\partial \hat{y_i}}(y_i - \hat{y_i})^2 = -2(y_i - \hat{y_i}) \tag{1}$$

1.b Weak Learner Selection

Given that the next learner is selected by,

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

Let $J = \sum_{i=1}^{n} (-g_i - \gamma h(x_i))^2$

The minimum of J w.r.t to γ is when its first order derivative is zero

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

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

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

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

By substituting the value of $\gamma*$ in 2, we get

$$\begin{aligned} h^* &= \arg\min_{h \in \mathcal{H}} \left(\sum_{i=1}^{n} (-g_i - \gamma^* h(x_i))^2 \right) \\ &= \arg\min_{h \in \mathcal{H}} \left(\sum_{i=1}^{n} \big[- g_i - \frac{2 \sum_{j=1}^{n} h(x_j)(y_j - \hat{y_j})}{\sum_{k=1}^{n} (h(x_k))^2} \times h(x_i) \big]^2 \right) \end{aligned} \tag{3}$$

Thus equation 3 can be derived independent of γ

1.c Step Size Selection

Given that

$$\alpha^* = \arg\min_{\alpha \in \mathbb{R}} \sum_{i=1}^{n} \big[y_i - (\hat{y}_i + \alpha h^*(x_i))\big]^2 \tag{4}$$

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

J is minimum w.r.t α when its first order derivative is zero.

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

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

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

Thus, we can compute the step size analytically and perform the following update:

$$\hat{y}_i \leftarrow \hat{y}_i + \alpha^* h^*(x_i)$$

2 NEURAL NETWORKS

2.a

Formulation:
Let us consider a neural network with L layers.
The number of neurons in each layer are $\{n_1, n_2, ...n_{L_1}, 1\}$. Note that the input size is n_1 and output size is 1. As stated in the problem, the last layer has a single neuron with logistic activation.
All the neurons in the hidden layers $2, 3, ...L-1$ are having linear activation function
The following notation is used in the proof:
$a_k^{(l)}$ - activation of k^{th} neuron in l^{th} layer
$b_k^{(l)}$ - bias of k^{th} neuron in l^{th} layer
$w_{ij}^{(l)}$ - weight for i^{th} neuron in l^{th} layer, has input from j^{th} neuron of $(l-1)$ layer

In the first layer, $l = 1$, activation is same as input, i.e., $a_j^{(1)} = x_j, j = 1, 2, ...n_1$

In the last layer, $l = L$, activation is output y, from a logistic function.
i.e.,

$$y = a_1^{(L)} = \sigma(\sum_{j=1}^{N_{L-1}} w_{1j}^{(L)} a_j^{(L-1)} + b_1^{(L)}) \tag{5}$$

$\forall$ Hidden layers, we have linear activations

$$a_j^{(l)} = \sum_{k=1}^{N_{l-1}} w_{jk}^{(l)} a_k^{(l-1)} + b_j^{(l)}, \text{ s.t. } 2 \le l \le L-1; \forall j \in \{1, ...N_l\} \tag{6}$$

Specifically, for $l = 2$, we have much nicer activation equation:

$$a_j^{(2)} = \sum_{k=1}^{N_1} w_{jk}^{(2)} a_k^{(1)} + b_j^{(2)} = \sum_{k=1}^{N_1} w_{jk}^{(2)} x_k + b_j^{(2)} \tag{7}$$

By using the fact 6 recursively in 5 until we reach the base case 7

$$y = a_1^{(L)} = \sigma\Big(\sum_{j=1}^{N_{L-1}} w_{1j}^{(L)} \big[\sum_{k=1}^{N_{L-2}} w_{jk}^{(L-1)} a_k^{(L-2)} + b_j^{(L-1)} \big] + b_1^{(L)} \Big)$$
$$= \sigma\Big(\sum_{j=1}^{N_{L-1}} w_{1j}^{(L)} \big[\sum_{k=1}^{N_{L-2}} w_{jk}^{(L-1)} [...(\sum_{i=1}^{N_1} w_{j'i}^{(2)} x_i + b_{j'}^{(2)}) + ...] + b_j^{(L-1)} \big] + b_1^{(L)} \Big)$$

NOTE: j' is the connecting neuron's indice in 3rd layer

With algebraic manipulations we can expand the series and group the terms such that all terms with x_i are at one place and similarly biases can also be grouped together. In the next step we note that all the terms common to x_i can be reduced to $R_i \in \mathbb{R}$ and all the biases can be reduced to a real number $R_b \in \mathbb{R}$ This forms the equation:

$$y = a_1^{(L)} = \sigma(\sum_{i=1}^{N_1} R_i x_i + R_b) \tag{8}$$

Thus the whole network is equivalent to a single logistic neuron.

2.b Back Propagation

Given: A network with 1 hidden layer
The input layer, x_i for $i = 1, 2, 3$ Activation of the hidden layer, $z_k = \tanh(\sum_{i=1}^{3} w_{ki} x_i)$
Activation of the output layer, $\hat{y_j} = \sum_{k=1}^{4} v_{jk} z_k$ for $j = 1, 2$
The optimization function is a squared error function, given by

$$L(y_i, \hat{y_i}) = \frac{1}{2}\big(\sum_{j=1}^{2} (y_j - \hat{y_j})^2\big) \tag{9}$$

2.b.1 Updates for the last layer

Lets find the gradient of equation 9 w.r.t parameters of the last layer neurons.

$$\frac{\partial L}{\partial v_{jk}} = \sum_{j=1}^{2} (y_j - \hat{y_j}) \times \frac{\partial \hat{y_j}}{\partial v_{jk}} = \sum_{j=1}^{2} (y_j - \hat{y_j}) \times \frac{\partial}{\partial v_{jk}} \sum_{k=1}^{4} v_{jk} z_k = -\sum_{k=1}^{4} z_k \sum_{j=1}^{2} (y_j - \hat{y_j}) \tag{10}$$

Update to be made using step size α:

$$\Delta v_{jk} = -\alpha \frac{\partial L}{\partial v_{jk}}$$
$$v_{jk} \leftarrow v_{jk} + \Delta v_{jk} \tag{11}$$

2.b.2 Updates for the hidden layer

Lets find the gradient of equation 9 w.r.t parameters of the hidden layer neurons.

$$\frac{\partial L}{\partial w_{ki}} = \sum_{j=1}^{2} (y_j - \hat{y_j}) \times \frac{\partial \hat{y_j}}{\partial w_{ki}} = \sum_{j=1}^{2} (y_j - \hat{y_j}) \times \frac{\partial}{\partial w_{ki}} \sum_{k=1}^{4} v_{jk} z_k = \sum_{j=1}^{2} (y_j - \hat{y_j}) \times \sum_{k=1}^{4} v_{jk} \frac{\partial z_k}{\partial w_{ki}} \tag{12}$$

$$\frac{\partial z_k}{\partial w_{ki}} = \frac{\partial}{\partial w_{ki}} \tanh(\sum_{i=1}^{3} w_{ki} x_i) = [1 - \tanh^2(\sum_{i=1}^{3} w_{ki} x_i)] \times \frac{\partial}{\partial w_{ki}} \sum_{i=1}^{3} w_{ki} x_i = [1 - \tanh^2(\sum_{i=1}^{3} w_{ki} x_i)] \times \sum_{i=1}^{3} x_i \tag{13}$$

Substituting 13 in 12, we get:

$$\frac{\partial L}{\partial w_{ki}} = \sum_{j=1}^{2}(y_j - \hat{y_j})\Big[\sum_{k=1}^{4} v_{jk}\big(1 - \tanh^2(\sum_{i=1}^{3} w_{ki}x_i)\big) \times \sum_{i=1}^{3} x_i\Big] \tag{14}$$

Update to be made using step size α :

$$\begin{aligned} \Delta w_{ki} =& -\alpha \frac{\partial L}{\partial w_{ki}} \\ w_{ki} \leftarrow& w_{ki} + \Delta w_{ki} \end{aligned} \tag{15}$$

3 PROGRAMMING

The programming assignment was performed on a Ubuntu 16.10 machine on Google Cloud. This machine had 24CPU cores and enough RAM to run smoothly. The unit for reported time below is Seconds.

3.d Linear Activation

Report the observations and explain the pattern of test set accuracies obtained.

```
## (d) Linear Activation
#                 Architecture    Lambda     Decay     Momtm    Act_Fn     Score      Time
1                      [50, 2]         0         0         0    linear  0.832622   6.32704
2                   [50, 50, 2]         0         0         0    linear  0.836159   5.44851
3               [50, 50, 50, 2]         0         0         0    linear  0.839004   7.55398
4           [50, 50, 50, 50, 2]         0         0         0    linear  0.841618   9.63156

Best Config: architecture = [50, 50, 50, 50, 2], lambda = 0.0, decay = 0.0,
                momentum = 0.0, actfn = linear, best_acc = 0.841617658418

5                   [50, 50, 2]         0         0         0    linear  0.837581   5.44945
6                  [50, 500, 2]         0         0         0    linear  0.840042   22.7603
7            [50, 500, 300, 2]         0         0         0    linear  0.843386   35.7083
8       [50, 800, 500, 300, 2]         0         0         0    linear  0.845231   67.9852
9  [50, 800, 800, 500, 300, 2]         0         0         0    linear  0.847499    105.25

Best Config: architecture = [50, 800, 800, 500, 300, 2], lambda = 0.0, decay = 0.0,
                momentum = 0.0, actfn = linear, best_acc = 0.847499330564
```

Observations:

- As the number of layers increased the test set accuracy increased and the time taken to train the network also increased.
- As the number of neurons increased the test set accuracy increased and the time taken to train the network also increased.

3.e Sigmoid Activation

```
#                 Architecture    Lambda     Decay     Momtm    Act_Fn     Score      Time
1                   [50, 50, 2]         0         0         0   sigmoid  0.746588   8.94158
2                  [50, 500, 2]         0         0         0   sigmoid  0.754892   76.4045
3            [50, 500, 300, 2]         0         0         0   sigmoid  0.717564   124.218
4       [50, 800, 500, 300, 2]         0         0         0   sigmoid  0.717564   244.296
5  [50, 800, 800, 500, 300, 2]         0         0         0   sigmoid  0.717564   370.078
Best Config: architecture = [50, 500, 2], lambda = 0.0, decay = 0.0,
               momentum = 0.0, actfn = sigmoid, best_acc = 0.754891784386
```

Observations :

- As the number of layers increased the test set accuracy fluctuated initially then it stopped changing.
- The reason behind no change in accuracy is due to 'The vanishing gradient problem' with sigmoid function.
- As the number of layers increased, the time taken to train the network increased.
- Network with sigmoid activations took more time to train than the network with linear activations. This is because sigmoid activation has an exponent term which takes more time to compute and the gradients are smaller during the back propagation.

3.f ReLu Activation

```
#                Architecture    Lambda     Decay     Momtm    Act_Fn     Score      Time
1                 [50, 50, 2]         0         0         0      relu  0.822089   6.61757
2                [50, 500, 2]         0         0         0      relu  0.818283   36.0388
3           [50, 500, 300, 2]         0         0         0      relu  0.805328   58.1833
4      [50, 800, 500, 300, 2]         0         0         0      relu  0.805597   113.869
5 [50, 800, 800, 500, 300, 2]         0         0         0      relu  0.791835   173.752
Best Config: architecture = [50, 50, 2], lambda = 0.0, decay = 0.0,
               momentum = 0.0, actfn = relu, best_acc = 0.82208895373
```

- As the number of layers increased the test set accuracy decreased. This decrement was very small.
- Unlike sigmoid, the relu does not suffer from the 'Vanishing gradients problem' so it continued to vary with more layers.
- As the number of layers increased, the time taken to train the network also increased.
- The time taken by relu network is lesser than the time taken by sigmoid network. This is because ReLu activation is faster to compute and its gradients are larger during the back propagation.

3.g L2-Regularization

```
#           Architecture    Lambda     Decay     Momtm    Act_Fn     Score      Time
1 [50, 800, 500, 300, 2]     1e-07         0         0      relu  0.802868   125.633
2 [50, 800, 500, 300, 2]     5e-07         0         0      relu   0.80479   125.408
3 [50, 800, 500, 300, 2]     1e-06         0         0      relu  0.806712   126.387
4 [50, 800, 500, 300, 2]     5e-06         0         0      relu  0.805213   125.781
5 [50, 800, 500, 300, 2]     1e-05         0         0      relu  0.798793   124.815
Best Config: architecture = [50, 800, 500, 300, 2], lambda = 1e-06, decay = 0.0,
            momentum = 0.0, actfn = relu, best_acc = 0.8067120346
Best Regularization Coefficient= 1e-06
```

- The accuracy is a concave function with maxima at $\lambda = 10^{-6}$. i.e. It increased until the maxima and then start to decrease.
- The best $\lambda = 10^{-6}$

3.h Early Stopping and L2-regularization

```
#           Architecture    Lambda     Decay     Momtm    Act_Fn     Score      Time
1 [50, 800, 500, 300, 2]     1e-07         0         0      relu  0.802868   116.136
2 [50, 800, 500, 300, 2]     5e-07         0         0      relu  0.803137   116.616
3 [50, 800, 500, 300, 2]     1e-06         0         0      relu  0.790835   116.909
4 [50, 800, 500, 300, 2]     5e-06         0         0      relu  0.754507   40.0303
```

```
5 [50, 800, 500, 300, 2]     1e-05         0         0      relu  0.766847   35.3188
Epoch 00009: early stopping
Best Config: architecture = [50, 800, 500, 300, 2], lambda = 5e-07, decay = 0.0,
              momentum = 0.0, actfn = relu, best_acc = 0.80313689471
Best Regularization Coefficient with early stopping= 5e-07
```

- The accuracy has a maxima at $\lambda = 5 \times 10^{-7}$. i.e. It increased until the maxima and then start to decrease. However, the accuracy function has some fluctuations and thus it is not a smooth function.
- With early stop, the best $\lambda = 5 \times 10^{-7}$. No, this is not same as the previous experiment.
- With early stopping didnt really help in terms of accuracy. The training time is cut off when early stop is triggered.

3.i SGD with weight decay

```
#           Architecture    Lambda     Decay     Momtm    Act_Fn     Score      Time
1 [50, 800, 500, 300, 2]     5e-07     1e-05         0      relu  0.797564   409.972
2 [50, 800, 500, 300, 2]     5e-07     5e-05         0      relu  0.784603   410.182
3 [50, 800, 500, 300, 2]     5e-07    0.0001         0      relu  0.717141   413.814
4 [50, 800, 500, 300, 2]     5e-07    0.0003         0      relu  0.718141    410.25
5 [50, 800, 500, 300, 2]     5e-07    0.0007         0      relu  0.295199   413.014
6 [50, 800, 500, 300, 2]     5e-07     0.001         0      relu  0.778688   411.389
Best Config: architecture = [50, 800, 500, 300, 2], lambda = 5e-07, decay = 1e-05,
             momentum = 0.0,  actfn = relu, best_acc = 0.0.797564
```

Best Decay $= 1 \times 10^{-5}$

3.j Momentum

```
#           Architecture    Lambda     Decay     Momtm    Act_Fn     Score      Time
1 [50, 800, 500, 300, 2]          0     5e-05      0.99      relu  0.848345   194.788
2 [50, 800, 500, 300, 2]          0     5e-05      0.98      relu  0.818937   195.591
3 [50, 800, 500, 300, 2]          0     5e-05      0.95      relu  0.785761   195.349
4 [50, 800, 500, 300, 2]          0     5e-05       0.9      relu  0.766655   194.508
5 [50, 800, 500, 300, 2]          0     5e-05      0.85      relu  0.724292   194.438
Best Config: architecture = [50, 800, 500, 300, 2], lambda = 0.0, decay = 5e-05,
            momentum = 0.99, actfn = relu, best_acc = 0.848345062112
```

Best momentum $= 0.99$

3.k Combining the above

```
#           Architecture    Lambda     Decay     Momtm    Act_Fn     Score      Time
1 [50, 800, 500, 300, 2]     5e-07     1e-05      0.99      relu  0.870218   36.4348
```

- Test set accuracy $= 0.870218$
- It is better than the previous values.

3.l Grid search with cross-validation

```
 #               Architecture    Lambda     Decay     Momtm    Act_Fn     Score      Time
 1                [50, 50, 2]     1e-07     1e-05      0.99      relu  0.845154   20.4715
 2                [50, 50, 2]     1e-07     5e-05      0.99      relu  0.842771   20.3143
 3                [50, 50, 2]     1e-07    0.0001      0.99      relu  0.805674   3.80574
```

4	[50, 50, 2]	5e-07	1e-05	0.99	relu	0.845231	20.3806
5	[50, 50, 2]	5e-07	5e-05	0.99	relu	0.845001	20.5079
6	[50, 50, 2]	5e-07	0.0001	0.99	relu	0.842771	20.161
7	[50, 50, 2]	1e-06	1e-05	0.99	relu	0.850229	20.5862
8	[50, 50, 2]	1e-06	5e-05	0.99	relu	0.844424	20.2211
9	[50, 50, 2]	1e-06	0.0001	0.99	relu	0.847768	20.3865
10	[50, 50, 2]	5e-06	1e-05	0.99	relu	0.847269	20.7574
11	[50, 50, 2]	5e-06	5e-05	0.99	relu	0.841579	20.5422
12	[50, 50, 2]	5e-06	0.0001	0.99	relu	0.836505	20.1935
13	[50, 50, 2]	1e-05	1e-05	0.99	relu	0.844462	20.7328
14	[50, 50, 2]	1e-05	5e-05	0.99	relu	0.844501	20.9433
15	[50, 50, 2]	1e-05	0.0001	0.99	relu	0.842656	20.1146
16	[50, 500, 2]	1e-07	1e-05	0.99	relu	0.851228	111.938
17	[50, 500, 2]	1e-07	5e-05	0.99	relu	0.810402	12.2399
18	[50, 500, 2]	1e-07	0.0001	0.99	relu	0.844693	112.215
19	[50, 500, 2]	5e-07	1e-05	0.99	relu	0.848729	112.914
20	[50, 500, 2]	5e-07	5e-05	0.99	relu	0.847153	112.21
21	[50, 500, 2]	5e-07	0.0001	0.99	relu	0.847192	111.397
22	[50, 500, 2]	1e-06	1e-05	0.99	relu	0.850575	112.272
23	[50, 500, 2]	1e-06	5e-05	0.99	relu	0.807865	13.1088
24	[50, 500, 2]	1e-06	0.0001	0.99	relu	0.843924	112.289
25	[50, 500, 2]	5e-06	1e-05	0.99	relu	0.850921	112.793
26	[50, 500, 2]	5e-06	5e-05	0.99	relu	0.845116	113.816
27	[50, 500, 2]	5e-06	0.0001	0.99	relu	0.843501	113.023
28	[50, 500, 2]	1e-05	1e-05	0.99	relu	0.85019	114.008
29	[50, 500, 2]	1e-05	5e-05	0.99	relu	0.847653	112.867
30	[50, 500, 2]	1e-05	0.0001	0.99	relu	0.844847	112.733
31	[50, 500, 300, 2]	1e-07	1e-05	0.99	relu	0.861608	188.814
32	[50, 500, 300, 2]	1e-07	5e-05	0.99	relu	0.855572	187.028
33	[50, 500, 300, 2]	1e-07	0.0001	0.99	relu	0.790797	18.3032
34	[50, 500, 300, 2]	5e-07	1e-05	0.99	relu	0.861569	188.087
35	[50, 500, 300, 2]	5e-07	5e-05	0.99	relu	0.789951	18.1336
36	[50, 500, 300, 2]	5e-07	0.0001	0.99	relu	0.80602	20.3596
37	[50, 500, 300, 2]	1e-06	1e-05	0.99	relu	0.861954	188.927
38	[50, 500, 300, 2]	1e-06	5e-05	0.99	relu	0.851305	190.002
39	[50, 500, 300, 2]	1e-06	0.0001	0.99	relu	0.851074	189.165
40	[50, 500, 300, 2]	5e-06	1e-05	0.99	relu	0.797447	18.0872
41	[50, 500, 300, 2]	5e-06	5e-05	0.99	relu	0.857533	187.847
42	[50, 500, 300, 2]	5e-06	0.0001	0.99	relu	0.856802	188.897
43	[50, 500, 300, 2]	1e-05	1e-05	0.99	relu	0.858801	188.426
44	[50, 500, 300, 2]	1e-05	5e-05	0.99	relu	0.856572	189.707
45	[50, 500, 300, 2]	1e-05	0.0001	0.99	relu	0.855957	189.398
46	[50, 800, 500, 300, 2]	1e-07	1e-05	0.99	relu	0.773113	36.0215
47	[50, 800, 500, 300, 2]	1e-07	5e-05	0.99	relu	0.86649	384.123
48	[50, 800, 500, 300, 2]	1e-07	0.0001	0.99	relu	0.752547	36.385
49	[50, 800, 500, 300, 2]	5e-07	1e-05	0.99	relu	0.869411	384.686
50	[50, 800, 500, 300, 2]	5e-07	5e-05	0.99	relu	0.862915	382.734
51	[50, 800, 500, 300, 2]	5e-07	0.0001	0.99	relu	0.861492	387.472
52	[50, 800, 500, 300, 2]	1e-06	1e-05	0.99	relu	0.868719	385.912
53	[50, 800, 500, 300, 2]	1e-06	5e-05	0.99	relu	0.86526	385.432
54	[50, 800, 500, 300, 2]	1e-06	0.0001	0.99	relu	0.859801	386.415
55	[50, 800, 500, 300, 2]	5e-06	1e-05	0.99	relu	0.775497	40.4417
56	[50, 800, 500, 300, 2]	5e-06	5e-05	0.99	relu	0.861492	384.428
57	[50, 800, 500, 300, 2]	5e-06	0.0001	0.99	relu	0.863991	384.855

```
58      [50, 800, 500, 300, 2]     1e-05     1e-05      0.99      relu  0.776381   36.3755
59      [50, 800, 500, 300, 2]     1e-05     5e-05      0.99      relu  0.866682   385.443
60      [50, 800, 500, 300, 2]     1e-05    0.0001      0.99      relu  0.860723   384.931
61 [50, 800, 800, 500, 300, 2]     1e-07     1e-05      0.99      relu  0.754315   56.1611
62 [50, 800, 800, 500, 300, 2]     1e-07     5e-05      0.99      relu  0.867874   602.568
63 [50, 800, 800, 500, 300, 2]     1e-07    0.0001      0.99      relu  0.861877   600.322
64 [50, 800, 800, 500, 300, 2]     5e-07     1e-05      0.99      relu  0.875293   595.138
65 [50, 800, 800, 500, 300, 2]     5e-07     5e-05      0.99      relu  0.870334   600.348
66 [50, 800, 800, 500, 300, 2]     5e-07    0.0001      0.99      relu  0.738746   56.2818
67 [50, 800, 800, 500, 300, 2]     1e-06     1e-05      0.99      relu  0.874371   598.163
68 [50, 800, 800, 500, 300, 2]     1e-06     5e-05      0.99      relu  0.870488    597.95
69 [50, 800, 800, 500, 300, 2]     1e-06    0.0001      0.99      relu  0.861723   601.545
70 [50, 800, 800, 500, 300, 2]     5e-06     1e-05      0.99      relu  0.724561   50.2382
71 [50, 800, 800, 500, 300, 2]     5e-06     5e-05      0.99      relu   0.87487   598.255
72 [50, 800, 800, 500, 300, 2]     5e-06    0.0001      0.99      relu  0.869219   599.841
73 [50, 800, 800, 500, 300, 2]     1e-05     1e-05      0.99      relu  0.734364   56.2232
74 [50, 800, 800, 500, 300, 2]     1e-05     5e-05      0.99      relu  0.734825   55.9429
75 [50, 800, 800, 500, 300, 2]     1e-05    0.0001      0.99      relu  0.741245   55.9157

Best Config: architecture = [50, 800, 800, 500, 300, 2], lambda = 5e-07, decay = 1e-05,
             momentum = 0.99, actfn = relu, best_acc = 0.875293123297
```

The best test set accuracy = 0.875293123297