CSCI 567 HW # 4

Mohmmad Suhail Ansari
USC ID: 8518586692
e-mail: mohmmada@usc.edu

November 2, 2016

Sol. 1.1 Given,

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

then, its derivative w.r.t g_i will be

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

Sol. 1.2 Optimal h^* can be found by calculating the gradient of

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

w.r.t γ and equating it to 0, i.e.

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

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

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

Sol. 1.3 Similar to 1.2 we again we select α^* which will minimize

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

$\therefore$, we get

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

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

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

Finally, the update rule can be given as

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

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

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

Sol. 2.1 Let our neural network consist of N input nodes, M linear activation hidden layers and K output nodes. Let $z_j^{L_i}$ be the j^{th} node in the L_i layer and $w_j^{L_i}$ be the weight, the we can write the output of the first hidden layer

$$z_j = \sum_{i=0}^{N} w_{ji}^{L_1} x_i$$

Then for the second hidden layer, we have

$$z_j = \sum_{i=0}^{L_1} w_{ji}^{L_2} z_i$$

and now, finally the output of the final output layer can be given as

$$y_j = \sigma(\sum_{i=0}^{L_{M-1}} w_{ji}^{L_M} z_i)$$

$$= \sigma(\sum_{i=0}^{L_{M-1}} w_{ji}^{L_M} \sum_{i=0}^{L_{M-2}} z_i)$$

$$= \sigma(\sum_{i=0}^{L_{M-1}} w_{ji}^{L_M} \sum_{i=0}^{L_{M-2}} w_{ji}^{L_M-1} \dots \sum_{i=0}^{N} w_j^{L_1} x_i)$$

Which can be written as

$$y_j = \sigma(WX)$$

where

$$W = \sum_{i=0}^{L_{M-1}} w_{ji}^{L_M} \sum_{i=0}^{L_{M-2}} w_{ji}^{L_M-1} \dots \sum_{i=0}^{N} w_{ji}^{L_1}$$

and

$$X = [x_1 \quad x_2 \quad x_3 \dots x_N]$$

Which is similar to Logistic Regression.

Sol. 2.2 Given

$$L(y, \hat{y}) = \frac{1}{2}((y_1 - \hat{y_1})^2 + (y_2 - \hat{y_2})^2) = \frac{1}{2}\sum_{j=1}^{2}(y_j - \hat{y}_j)^2$$

where

$$\hat{y}_j = \sum_{k=1}^{4} v_{jk} z_k$$

and

$$z_k = tanh(\sum_{i=3}^{3} w_{ki} x_i)$$

taking partial differential of L w.r.t v_{jk}, we get

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

we know that

$$\frac{\partial tanh(x)}{\partial x} = 1 - tanh^2(x)$$

$\therefore$

$$\frac{\partial tanh(\sum_{i=3}^{3} w_{ki}x_i)}{\partial w_{ki}} = (1 - tanh^2(\sum_{i=3}^{3} w_{ki}x_i))x_i$$

then, we have the partial differential of L w.r.t w_{ki} as

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

Therefore we get the update rules as

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

and

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

Sol. 3.1 Linear Activations

Architecture	Accuracy	Time(seconds)
[d_{in}, d_{out}]	83.4198%	3.0391190000
[d_{in}, 50, d_{out}]	83.5544%	5.9478230000
[d_{in}, 50, 50, d_{out}]	84.0503%	8.1254930000
[d_{in}, 50, 50, 50, d_{out}]	84.4424%	10.2770940000
[d_{in}, 50, d_{out}]	83.9965%	5.3710550000
[d_{in}, 500, d_{out}]	84.1925%	22.6269160000
[d_{in}, 500, 300, d_{out}]	84.7153%	35.4776900000
[d_{in}, 800, 500, 300, d_{out}]	85.1228%	66.4187950000
[d_{in}, 800, 800, 500, 300, d_{out}]	85.1882%	103.6037080000

Both accuracy and training time for each architecture increases with increasing number of hidden layers and increasing number of nodes in each hidden layer.

Sol. 3.2 Sigmoid Activations

Architecture	Accuracy	Time(seconds)
$[d_{in}, 50, d_{out}]$	74.8280%	9.4715190000
$[d_{in}, 500, d_{out}]$	76.7309%	77.9355170000
$[d_{in}, 500, 300, d_{out}]$	72.2639%	122.3402530000
$[d_{in}, 800, 500, 300, d_{out}]$	72.2639%	241.0532810000
$[d_{in}, 800, 800, 500, 300, d_{out}]$	72.2639%	363.2464260000

The accuracy first increases when one hidden layer is used, but then decreases and remains stable for rest of the models with increasing number of hidden layers.

I believe that the the lower accuracy (compared to linear activation function) for sigmoid activation function is the result of the sigmoid activation function's property of "killing the gradient", i.e. when a sigmoid neurons activation saturates at either tail of 0 or 1, the gradient at these regions is almost zero. Recall that during backpropagation, this (local) gradient will be multiplied to the gradient of this gates output for the whole objective.

Sol. 3.3 ReLu Activations

Architecture	Accuracy	Time(seconds)
$[d_{in}, 50, d_{out}]$	82.3165%	5.90191
$[d_{in}, 500, d_{out}]$	82.0282%	35.21581
$[d_{in}, 500, 300, d_{out}]$	81.8706%	56.89574
$[d_{in}, 800, 500, 300, d_{out}]$	81.0326%	109.04267
$[d_{in}, 800, 800, 500, 300, d_{out}]$	80.1253%	166.93421

The accuracy seems to be decreasing with increasing hidden layers and number of nodes in hidden layers.

The accuracy compared to linear activation is still lower, which I believe is because of ReLu neurons can "die", i.e. a large gradient flowing through a ReLU neuron could cause the weights to update in such a way that the neuron will never activate on any data point again. If this happens, then the gradient flowing through the unit will forever be zero from that point on.

The training time for ReLu neurons is more than linear neurons but shorter than sigmoid neurons. Its shorter than sigmoid neurons because of ReLu's linear and non-saturating form which can be implemented by simply thresholding a matrix of activations at zero.

Sol. 3.4 L2 Regularization

Architecture	λ	Accuracy	Time(seconds)
$[d_{in}, 800, 500, 300, d_{out}]$	10^{-7}	80.5559%	117.7750190000
$[d_{in}, 800, 500, 300, d_{out}]$	5^{-7}	80.5059%	117.2375600000
$[d_{in}, 800, 500, 300, d_{out}]$	10^{-6}	81.3901%	117.9980010000
$[d_{in}, 800, 500, 300, d_{out}]$	5^{-6}	80.8019%	118.1240910000
$[d_{in}, 800, 500, 300, d_{out}]$	10^{-5}	80.6135%	118.6769920000

Best $\lambda = 10^{-6}$

The accuracy increases and decreases alternatively with increasing λ (although the difference seems to be really small). We get this trend because we are trying to find the best λ that generalizes our data the best and since the difference between the different λ is small, so is the difference in their accuracies.

Sol. 3.5 Early Stopping and L2-regularization

Architecture	λ	Accuracy	Time(seconds)
$[d_{in}, 800, 500, 300, d_{out}]$	10^{-7}	79.8370%	110.0074800000
$[d_{in}, 800, 500, 300, d_{out}]$	5^{-7}	78.7606%	112.0828270000
$[d_{in}, 800, 500, 300, d_{out}]$	10^{-6}	80.5251%	111.0474780000
$[d_{in}, 800, 500, 300, d_{out}]$	5^{-6}	79.7832%	110.7920260000

Best $\lambda = 10^{-6}$

The trend is similar to the L2 regularization without early-stopping. Since, the λ for both cases is the same and so is the trend in accuracy, I believe the early stopping did help, since it takes little to achieve similar results.

Sol. 3.5 SGD with weight decay

Architecture	Lambda	Decay Rate	Accuracy	Time(seconds)
$[d_{in}, 800, 500, 300, d_{out}]$	5^{-7}	10^{-5}	72.4791%	424.3529670000
$[d_{in}, 800, 500, 300, d_{out}]$	5^{-7}	5^{-5}	73.2941%	413.6669020000
$[d_{in}, 800, 500, 300, d_{out}]$	5^{-7}	10^{-4}	74.1322%	429.5304860000
$[d_{in}, 800, 500, 300, d_{out}]$	5^{-7}	3^{-4}	72.5445%	409.1038830000
$[d_{in}, 800, 500, 300, d_{out}]$	5^{-7}	7^{-4}	65.3097%	391.0522660000
$[d_{in}, 800, 500, 300, d_{out}]$	5^{-7}	10^{-3}	72.3446%	393.0722790000

Best Decay Rate = 10^{-4}

Sol. 3.6 Momentum

Architecture	Decay Rate	Momentum	Accuracy	Time(seconds)
$[d_{in}, 800, 500, 300, d_{out}]$	10^{-4}	0.99	84.6384%	188.7084680000
$[d_{in}, 800, 500, 300, d_{out}]$	10^{-4}	0.98	82.3704%	188.8004400000
$[d_{in}, 800, 500, 300, d_{out}]$	10^{-4}	0.95	78.8606%	189.2771110000
$[d_{in}, 800, 500, 300, d_{out}]$	10^{-4}	0.90	74.0822%	189.9112340000
$[d_{in}, 800, 500, 300, d_{out}]$	10^{-4}	0.85	75.6929%	189.8105580000

Best Momentum = 0.99

Sol. 3.7 Combination

Architecture	Lambda	Decay Rate	Momentum	Accuracy	Time(seconds)
$[d_{in}, 800, 500, 300, d_{out}]$	10^{-6}	10^{-4}	0.99	85.6264%	366.8762140000

Yes, the accuracy for this combination is the best we have achieved so far.

Sol. 3.8 Grid search with Cross-Validation:

Architecture	Lambda	Decay Rate	Momentum	Accuracy	Time(seconds)
$[d_{in}, 50, d_{out}]$	10^{-7}	10^{-5}	0.99	84.6807%	20.2832450000
$[d_{in}, 50, d_{out}]$	10^{-7}	5^{-5}	0.99	80.1753%	3.3854140000
$[d_{in}, 50, d_{out}]$	10^{-7}	10^{-4}	0.99	84.3002%	20.1343490000
$[d_{in}, 50, d_{out}]$	5^{-7}	10^{-5}	0.99	85.0459%	20.6533200000
$[d_{in}, 50, d_{out}]$	5^{-7}	5^{-5}	0.99	84.2848%	20.4618000000
$[d_{in}, 50, d_{out}]$	5^{-7}	10^{-4}	0.99	84.0541%	20.3314530000
$[d_{in}, 50, d_{out}]$	10^{-6}	10^{-5}	0.99	84.2002%	19.6094510000
$[d_{in}, 50, d_{out}]$	10^{-6}	5^{-5}	0.99	84.0234%	19.9445860000
$[d_{in}, 50, d_{out}]$	10^{-6}	10^{-4}	0.99	83.3737%	19.9657800000
$[d_{in}, 50, d_{out}]$	5^{-6}	10^{-5}	0.99	85.3765%	19.4346490000
$[d_{in}, 50, d_{out}]$	5^{-6}	5^{-5}	0.99	84.1118%	19.6701670000
$[d_{in}, 50, d_{out}]$	5^{-6}	10^{-4}	0.99	82.9009%	19.8740320000
$[d_{in}, 50, d_{out}]$	10^{-5}	10^{-5}	0.99	85.2036%	19.9823050000
$[d_{in}, 50, d_{out}]$	10^{-5}	5^{-5}	0.99	83.9888%	20.1908340000
$[d_{in}, 50, d_{out}]$	10^{-5}	10^{-4}	0.99	83.3237%	20.2327580000
$[d_{in}, 500, d_{out}]$	10^{-7}	10^{-5}	0.99	85.1228%	109.6329600000
$[d_{in}, 500, d_{out}]$	10^{-7}	5^{-5}	0.99	84.6846%	110.6199690000
$[d_{in}, 500, d_{out}]$	10^{-7}	10^{-4}	0.99	84.5039%	110.8651400000
$[d_{in}, 500, d_{out}]$	5^{-7}	10^{-5}	0.99	81.2363%	12.2082780000
$[d_{in}, 500, d_{out}]$	5^{-7}	5^{-5}	0.99	84.7230%	110.2654860000
$[d_{in}, 500, d_{out}]$	5^{-7}	10^{-4}	0.99	81.4862%	12.9311620000
$[d_{in}, 500, d_{out}]$	10^{-6}	10^{-5}	0.99	85.0613%	109.4635540000
$[d_{in}, 500, d_{out}]$	10^{-6}	5^{-5}	0.99	81.4900%	10.8740570000
$[d_{in}, 500, d_{out}]$	10^{-6}	10^{-4}	0.99	84.4847%	110.2498010000
$[d_{in}, 500, d_{out}]$	5^{-6}	10^{-5}	0.99	85.1267%	109.9373850000
$[d_{in}, 500, d_{out}]$	5^{-6}	5^{-5}	0.99	84.7422%	112.1203260000
$[d_{in}, 500, d_{out}]$	5^{-6}	10^{-4}	0.99	80.5405%	12.0689210000
$[d_{in}, 500, d_{out}]$	10^{-5}	10^{-5}	0.99	85.0998%	110.8005040000
$[d_{in}, 500, d_{out}]$	10^{-5}	5^{-5}	0.99	84.7615%	109.8176930000
$[d_{in}, 500, d_{out}]$	10^{-5}	10^{-4}	0.99	84.5270%	110.5633620000

Architecture	Lambda	Decay Rate	Momentum	Accuracy	Time(seconds)
$[d_{in}, 500, 300, d_{out}]$	10^{-7}	10^{-5}	0.99	85.6072%	182.9771570000
$[d_{in}, 500, 300, d_{out}]$	10^{-7}	5^{-5}	0.99	79.6717%	19.3831510000
$[d_{in}, 500, 300, d_{out}]$	10^{-7}	10^{-4}	0.99	78.8990%	19.4750520000
$[d_{in}, 500, 300, d_{out}]$	5^{-7}	10^{-5}	0.99	85.8071%	182.3738550000
$[d_{in}, 500, 300, d_{out}]$	5^{-7}	5^{-5}	0.99	79.4795%	19.3143720000
$[d_{in}, 500, 300, d_{out}]$	5^{-7}	10^{-4}	0.99	85.1920%	182.6694480000
$[d_{in}, 500, 300, d_{out}]$	10^{-6}	10^{-5}	0.99	86.4568%	181.8394820000
$[d_{in}, 500, 300, d_{out}]$	10^{-6}	5^{-5}	0.99	85.3958%	182.5855900000
$[d_{in}, 500, 300, d_{out}]$	10^{-6}	10^{-4}	0.99	85.2843%	182.2573850000
$[d_{in}, 500, 300, d_{out}]$	5^{-6}	10^{-5}	0.99	86.1608%	183.5565350000
$[d_{in}, 500, 300, d_{out}]$	5^{-6}	5^{-5}	0.99	85.5726%	184.1752500000
$[d_{in}, 500, 300, d_{out}]$	5^{-6}	10^{-4}	0.99	78.7529%	18.0114930000
$[d_{in}, 500, 300, d_{out}]$	10^{-5}	10^{-5}	0.99	86.1069%	181.5487010000
$[d_{in}, 500, 300, d_{out}]$	10^{-5}	5^{-5}	0.99	85.4880%	181.8470030000
$[d_{in}, 500, 300, d_{out}]$	10^{-5}	10^{-4}	0.99	85.0344%	182.3495300000
$[d_{in}, 800, 500, 300, d_{out}]$	10^{-7}	10^{-5}	0.99	86.8796%	370.1537460000
$[d_{in}, 800, 500, 300, d_{out}]$	10^{-7}	5^{-5}	0.99	86.3107%	372.6882040000
$[d_{in}, 800, 500, 300, d_{out}]$	10^{-7}	10^{-4}	0.99	85.9647%	375.3052920000
$[d_{in}, 800, 500, 300, d_{out}]$	5^{-7}	10^{-5}	0.99	77.2498%	31.2794710000
$[d_{in}, 800, 500, 300, d_{out}]$	5^{-7}	5^{-5}	0.99	86.3453%	367.1678460000
$[d_{in}, 800, 500, 300, d_{out}]$	5^{-7}	10^{-4}	0.99	85.9493%	371.9213370000
$[d_{in}, 800, 500, 300, d_{out}]$	10^{-6}	10^{-5}	0.99	86.5260%	369.9097790000
$[d_{in}, 800, 500, 300, d_{out}]$	10^{-6}	5^{-5}	0.99	86.3376%	369.4920700000
$[d_{in}, 800, 500, 300, d_{out}]$	10^{-6}	10^{-4}	0.99	78.1340%	35.0205110000
$[d_{in}, 800, 500, 300, d_{out}]$	5^{-6}	10^{-5}	0.99	86.9681%	373.8522750000
$[d_{in}, 800, 500, 300, d_{out}]$	5^{-6}	5^{-5}	0.99	86.5490%	381.4162190000
$[d_{in}, 800, 500, 300, d_{out}]$	5^{-6}	10^{-4}	0.99	85.8263%	394.7567470000
$[d_{in}, 800, 500, 300, d_{out}]$	10^{-5}	10^{-5}	0.99	77.4036%	32.6861520000
$[d_{in}, 800, 500, 300, d_{out}]$	10^{-5}	5^{-5}	0.99	86.3453%	391.6142450000
$[d_{in}, 800, 500, 300, d_{out}]$	10^{-5}	10^{-4}	0.99	76.4733%	33.5267370000

Architecture	Lambda	Decay Rate	Momentum	Accuracy	Time(seconds)
[d_{in}, 800, 800, 500, 300, d_{out}]	10^{-7}	10^{-5}	0.99	87.5716%	587.6533990000
[d_{in}, 800, 800, 500, 300, d_{out}]	10^{-7}	5^{-5}	0.99	74.4282%	48.7787180000
[d_{in}, 800, 800, 500, 300, d_{out}]	10^{-7}	10^{-4}	0.99	73.1865%	48.3710490000
[d_{in}, 800, 800, 500, 300, d_{out}]	5^{-7}	10^{-5}	0.99	74.5627%	48.9536650000
[d_{in}, 800, 800, 500, 300, d_{out}]	5^{-7}	5^{-5}	0.99	72.6444%	54.2774970000
[d_{in}, 800, 800, 500, 300, d_{out}]	5^{-7}	10^{-4}	0.99	86.0800%	583.3533200000
[d_{in}, 800, 800, 500, 300, d_{out}]	10^{-6}	10^{-5}	0.99	87.3409%	575.2738650000
[d_{in}, 800, 800, 500, 300, d_{out}]	10^{-6}	5^{-5}	0.99	74.6896%	48.5039770000
[d_{in}, 800, 800, 500, 300, d_{out}]	10^{-6}	10^{-4}	0.99	74.9779%	48.4011350000
[d_{in}, 800, 800, 500, 300, d_{out}]	5^{-6}	10^{-5}	0.99	87.5293%	580.0695650000
[d_{in}, 800, 800, 500, 300, d_{out}]	5^{-6}	5^{-5}	0.99	73.9938%	47.8524650000
[d_{in}, 800, 800, 500, 300, d_{out}]	5^{-6}	10^{-4}	0.99	86.1761%	575.5397880000
[d_{in}, 800, 800, 500, 300, d_{out}]	10^{-5}	10^{-5}	0.99	87.7100%	568.8467470000
[d_{in}, 800, 800, 500, 300, d_{out}]	10^{-5}	5^{-5}	0.99	73.8708%	47.4766680000
[d_{in}, 800, 800, 500, 300, d_{out}]	10^{-5}	10^{-4}	0.99	86.5952%	569.0146650000

The best configuration we could obtain is

$$Architecture = [d_{in}, \quad 800, \quad 800, \quad 500, \quad 300, \quad d_{out}]$$

$$\lambda = 10^{-05}$$

$$Decay = 10^{-05}$$

$$Momentum = 0.99$$

$$Accuracy = 87.71\%$$