CSCI 567 Assignment 4
Fall 2016

Snehal Adsule
2080872073
adsule@usc.edu

November 5,2016

1 Problem 1

1.1 1 (a)

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

$$\frac{dL}{d\hat{y}_i} = g_i = 2(\hat{y}_i - y_i) \tag{1}$$

1.2 1 (b)

Given that h^*= arg min (min $\sum_{i=1}^{n}(-g_i - \gamma h(x_i))^2$)

Differentiating wrt γ for optimal h^*

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

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

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

To prove the optimal we should take the second derivative

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

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

As the above equation is always going to be positive, we can say that h^* is optimal.

1.3 1 (c)

Given that a^*= arg min $\sum_{i=1}^{n} L(y_i, \hat{y}_i + \alpha h^*(x_i))$ differntiating wrt α

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

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

Differentiating again , for optimal

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

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

As, second derivative is positive it will be optimal solution.

2 Problem 2

2.1 2 (a)

Conside the neural network with the linear activation for hidden layer and sigmoid output. We can think of it as a one single layer input , which is the output from the hidden layer with j units $a_j = \sum h(w_{ji}x_i)$.
$z_j = \sum w_j i x_i$,
$a_j = h(z_j)$
$y_k = \sum \sigma(v_{kj} z_j)$,where h is the linear activation function.

Hence, we can think of it as reduced input a_j along with weights v_{kj} , as linear input to the final sigmoid unit. We can represent the output in terms of the logistic regression, with a_j as the input .

$f(x) = \frac{1}{1+e^{\sum v_{jk} a_j}}$

Therefore, it is equivalent to the logistic regression.

2.2 2 (b)

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

The error in the output layer is given as

$$\frac{dL}{dy_j} = \delta_j = (y_j - t_j)$$

using chain rule, we can back propagate the error as

$$\begin{aligned}\frac{dL}{da_k} &= \delta_k \\ &= \frac{dL}{da_j}\frac{da_j}{da_k}, \\ \delta_k &= \delta_j(1-z^2)\sum v_{jk}\end{aligned}$$

Now, derivative for weight update of v_{jk}, using chain rule and $\sigma^{'}(a) = 1 - z^2$ for tanh function

$$\begin{aligned}\frac{dL}{dv_{jk}} &= \frac{dL}{dy_j}\frac{dy_j}{dv_{jk}} \\ &= \delta_j \frac{d\sum v_{jk}z_k}{dv_{jk}} \\ \frac{dL}{dv_{jk}} &= \delta_j z_k\end{aligned}$$

Now, derivative for weight update update of w_{ki}, using chian rule

$$\begin{aligned}\frac{dL}{dw_{ki}} &= \frac{dL}{da_k}\frac{da_k}{dw_{ki}} \\ &= \delta_k \frac{d\sum w_{ki}x_i}{dw_{ki}} \\ \frac{dL}{dw_{ki}} &= \delta_k x_i\end{aligned}$$

Problem 3 -Programming

3 (a) Completed the installation of the required packages on local and Google Cloud Platform

3 (b) Checked the verbose and ran the experiment multiple times.

3 (c) 1 Load Data -(104051, 50) (26013, 50) (104051, 2) (26013, 2)

3 (c) 2 Normalization -(104051, 50) (26013, 50)

3 (d) 1 Linear Activation

Score for architecture = [50, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear: test accu = 0.590051125298 ,time =6.44198489189
Score for architecture = [50, 50, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear: test accu = 0.505554916757 ,time =10.7836301327
Score for architecture = [50, 50, 50, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear: test accu = 0.439280360475 ,time =13.2830498219
Score for architecture = [50, 50, 50, 50, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear: test accu = 0.656748545077 ,time =15.7790420055
Best Config: test accu = architecture = [50, 50, 50, 50, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear, best_acc = 0.656748545077
Overall Time taken = 46.2878129482 seconds

The accuracy improves with the number of hidden layer increases in linear activation. Large hidden layers can allow the neural network to fit the training data better as observed with architecture = [50, 50, 50, 50, 2], but it may also result in overfitting.

Time taken for individual architecture is mentioned above with accuracy , overall time taken for this set of architectures was 46 seconds.

3 (d) 2 Linear Activation

Score for architecture = [50, 50, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear: test accu = 0.626340680926 ,time =11.1874220371
Score for architecture = [50, 500, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear: test accu = 0.611540374077 ,time =24.2037501335
Score for architecture = [50, 500, 300, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear: test accu = 0.73840002671 ,time =48.7018830776
Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear: test accu = 0.826086962339 ,time =118.230014086
Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear: test accu = 0.825548759481 ,time =216.175589085
Best Config: test accu = architecture = [50, 800, 500, 300, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = linear, best_acc = 0.826086962339
Overall Time taken = 418.498801947seconds

However, using a first hidden layer which is larger than the input layer tends to work better, as observed in this simulation , giving best config for architecture = [50, 800, 500, 300, 2], compared to other in this run.

We are using the same size for all hidden layers (50) in and it may work better or the same as using a decreasing or increasing size {800,500,300}. Comparing acurracy for architecture = [50, 50, 50, 50, 2] being lower than of architecture = [50, 800, 500, 300, 2].

Overall time taken to train this set of architecture is 418 sec, which is much larger than same size hidden layer.

3 (e) 1 Sigmoid Activation

Score for architecture = [50, 50, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = sigmoid: test accu = 0.268673353967 ,time =17.7105529308
Score for architecture = [50, 500, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = sigmoid: test accu = 0.719755513725 ,time =86.0492258072
Score for architecture = [50, 500, 300, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = sigmoid: test accu = 0.719755513725 ,time =148.651331902
Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = sigmoid: test accu = 0.719755513725 ,time =317.584743977
Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = sigmoid: test accu = 0.719755513725 ,time =515.921478033
Best Config: test accu = architecture = [50, 500, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = sigmoid, best_acc = 0.719755513725
Overall Time taken = 1085.91745496seconds

With the sigmoid function for activation, increasing the number of hidden layer was not effective in improving the test accuracy, as clearly observed in the linear activation.

However, larger size for 1 hidden layer showed significant improvement in the accuracy, same as linear activation.

The time taken by the sigmoid activation is 1085 seconds, which is more than twice as in linear.

Due to the non lineraity of the sgmoid function it is able to better handle the larger positive and negative values and perform better than linear function. It smoothly squashes out the output. However, for the configuration [50, 50, 2] linear activation performed very well compared to the sigmoid run and the best test accuracy of linear was observed better than sigmoid.

3 (f) 1

Score for architecture = [50, 50, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = relu: test accu = 0.830546264144 ,time =12.3142368793
Score for architecture = [50, 500, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = relu: test accu = 0.819474872655 ,time =36.8106482029

Score for architecture = [50, 500, 300, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = relu: test accu = 0.813670088736 ,time =69.7732560635
Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = relu: test accu = 0.806904240928 ,time =159.945312023
Score for architecture = [50, 800, 800, 500, 300, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = relu: test accu = 0.781109441288 ,time =280.3577981
Best Config: test accu = architecture = [50, 50, 2], lambda = 0.0, decay = 0.0, momentum = 0.0, actfn = relu, best_acc = 0.830546264144
Overall Time taken = 559.201347113seconds

ReLu performed better than sigmoid and linear activation with best accuracy = 0.8305 , however, the larger size of hidden layer showed no effect here compared to linear and sigmoid.

It is very effective due to its linear and non linear nature, and it accelerates quickly , with better convergence of stochastic gradient descent compared to the sigmoid functions.

Also, the number of hidden layer in the architecture only gave negative impact in this simulation. As the ReLu is more expensive in evaluating the threshold for max, it took more times than linear but less than sigmoid, with overall 559 seconds.

3 (g) 1 L2-regularization

Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-06, decay = 0.0, momentum = 0.0, actfn = relu: test accu = 0.807749974768 ,time =170.483810902
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-06, decay = 0.0, momentum = 0.0, actfn = relu: test accu = 0.812901243124 ,time =169.765723944
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-05, decay = 0.0, momentum = 0.0, actfn = relu: test accu = 0.810633145816 ,time =169.8930161
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-05, decay = 0.0, momentum = 0.0, actfn = relu: test accu = 0.820781917529 ,time =169.453432083
Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0001, decay = 0.0, momentum = 0.0, actfn = relu: test accu = 0.807673088832 ,time =169.302447081
Best Config: test accu = architecture = [50, 800, 500, 300, 2], lambda = 5e-05, decay = 0.0, momentum = 0.0, actfn = relu, best_acc = 0.820781917529
Overall Time taken = 848.898546934seconds

The best value for the lambda is $5x 10_{-05}$ for the given architecture. The larger regularization coefficient penalize more and larger wieght values, as a method used for the avoiding overfitting. With the decrease in the regularized coefficient the test accuracy improves but then decreases, after a particular value.

3 (h) Early Stopping and L2-regularization

Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-06, decay = 0.0, momentum = 0.0, actfn = relu: test accu = 0.795217777989 ,time =161.92662096
Epoch 00007: test accu = early stopping
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-06, decay = 0.0, momentum = 0.0, actfn = relu: test accu = 0.719755513725 ,time =42.8527059555
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-05, decay = 0.0, momentum = 0.0, actfn = relu: test accu = 0.79856225778 ,time =156.673537016
Epoch 00009: test accu = early stopping
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-05, decay = 0.0, momentum = 0.0, actfn = relu: test accu = 0.768308154044 ,time =58.5802140236
Epoch 00008: test accu = early stopping
Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0001, decay = 0.0, momentum = 0.0, actfn = relu: test accu = 0.773113445998 ,time =48.0944261551
Best Config: test accu = architecture = [50, 800, 500, 300, 2], lambda = 1e-05, decay = 0.0, momentum = 0.0, actfn = relu, best_acc = 0.79856225778
Overall Time taken = 468.127600193seconds

Early stopping is very effective with the time taken for the architecture to run. It determines when to stop training, once performance on a held-out validation set stops increasing

The value of the best regularization coefficient is not the same without early stopping, it helps determin the number of iterations to prevent the overfitting.

It is also useful in reducing the execution time almost half with overall time taken was 468.127600193 seconds.

3 (i) SGD with weight decay

Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-05, decay = 1e-05, momentum = 0.0, actfn = relu: test accu = 0.749471412138,time =560.327785015
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-05, decay = 5e-05, momentum = 0.0, actfn = relu: test accu = 0.721792949454,time =540.752099037
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-05, decay = 0.0001, momentum = 0.0, actfn = relu: test accu = 0.647022639776,time =480.0944261551
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-05, decay = 0.0003, momentum = 0.0, actfn = relu: test accu = 0.733325640521,time =571.92662096
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-05, decay = 0.0007, momentum = 0.0, actfn = relu: test accu = 0.57186791189,time =501.8527059555
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-05, decay = 0.001, momentum = 0.0, actfn = relu: test accu = 0.532195440231,time =487.652362108
Best Config: architecture = [50, 800, 500, 300, 2], lambda = 1e-05, decay = 1e-05, momentum = 0.0, actfn = relu, best_acc = 0.749471412138

Overall Time taken = 3059.14172316seconds

There is no particular trend with the increase in the weight decay value and the best test accuracies was observed for decay = 1e-05, with no early stopping and no momentum.

3 (j) Momentum

Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0, decay = 1e-05, momentum = 0.99, actfn = relu: test accu = 0.8657935646369 ,time =270.466249943
Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0, decay = 1e-05, momentum = 0.98, actfn = relu: test accu = 0.857633105076 ,time =269.752099037
Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0, decay = 1e-05, momentum = 0.95, actfn = relu: test accu = 0.85424273594 ,time =270.576807976
Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0, decay = 1e-05, momentum = 0.9, actfn = relu: test accu = 0.8454184445 ,time =270.979725122
Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0, decay = 1e-05, momentum = 0.85, actfn = relu: test accu = 0.8411563446 ,time =271.081454992
Best Config: test accu = architecture = [50, 800, 500, 300, 2], lambda = 0.0, decay =1e-05, momentum = 0.99, actfn = relu, best_acc = 0.8657935646369
Overall Time taken = 1352.85644412seconds

Best momentum was observed for value of 0.99, there is a decrease in the accuracy observed with the decrease in the momentum but no significant impact on the execution time.

3 (k) Combining the above

Epoch 00092: test accu = early stopping

Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-05, decay = 1e-05, momentum = 0.99, actfn = relu: test accu = 0.871817323098 ,time =488.006507874
Best Config: test accu = architecture = [50, 800, 500, 300, 2], lambda = 1e-05, decay = 1e-05, momentum = 0.99, actfn = relu, best_acc = 0.870817323098
Overall Time taken = 488.006630182seconds

In this run we have taken the tuned parameters (lambda = 1e-05, decay = 1e-05, momentum = 0.99) which has led to higher accuracy of 0.87 than the part (i), (j) and (k).

3 (l) Grid search with cross-validation

Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-06, decay = 0.0001, momentum = 0.99, actfn = relu: test accu = 0.87124168892 ,time =530.31359601
Epoch 00082: test accu = early stopping
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-06, decay = 0.0005, momentum = 0.99, actfn = relu: test accu = 0.8650939145335 ,time =436.688696861
Epoch 00057: test accu = early stopping

Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-06, decay = 0.001, momentum = 0.99, actfn = relu: test accu = 0.852097416205 ,time =304.498915911
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-06, decay = 0.0001, momentum = 0.99, actfn = relu: test accu = 0.871049475226 ,time =524.638388157
Epoch 00084: test accu = early stopping
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-06, decay = 0.0005, momentum = 0.99, actfn = relu: test accu = 0.8650170290557 ,time =449.755156994
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-06, decay = 0.001, momentum = 0.99, actfn = relu: test accu = 0.855134361417 ,time =525.990138054
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-05, decay = 0.0001, momentum = 0.99, actfn = relu: test accu = 0.872048976355 ,time =530.16048193
Epoch 00094: test accu = early stopping
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-05, decay = 0.0005, momentum = 0.99, actfn = relu: test accu = 0.8651592658607 ,time =499.917109013
Epoch 00085: test accu = early stopping
Score for architecture = [50, 800, 500, 300, 2], lambda = 1e-05, decay = 0.001, momentum = 0.99, actfn = relu: test accu = 0.854596157573 ,time =454.010499001
Epoch 00079: test accu = early stopping
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-05, decay = 0.0001, momentum = 0.99, actfn = relu: test accu = 0.8658973596195 ,time =419.859564066
Epoch 00087: test accu = early stopping
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-05, decay = 0.0005, momentum = 0.99, actfn = relu: test accu = 0.8652130855577 ,time =463.250978231
Epoch 00066: test accu = early stopping
Score for architecture = [50, 800, 500, 300, 2], lambda = 5e-05, decay = 0.001, momentum = 0.99, actfn = relu: test accu = 0.853865760656 ,time =355.305943966
Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0001, decay = 0.0001, momentum = 0.99, actfn = relu: test accu = 0.870088413627 ,time =526.634424925
Epoch 00051: test accu = early stopping
Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0001, decay = 0.0005, momentum = 0.99, actfn = relu: test accu = 0.855672551512 ,time =274.630086899
Epoch 00044: test accu = early stopping
Score for architecture = [50, 800, 500, 300, 2], lambda = 0.0001, decay = 0.001, momentum = 0.99, actfn = relu: test accu = 0.850482838551 ,time =238.652362108
Best Config: test accu = architecture = [50, 800, 500, 300, 2], lambda = 1e-05, decay = 0.0001, momentum = 0.99, actfn = relu, best_acc = 0.872048976355
Overall Time taken = 6534.30647278seconds

Grid search took the maximum time as it does cross validation and Best test accuracy of 0.872048976355 was observed for architecture = [50, 800, 500, 300, 2], with best hyper parameters lambda = 1e-05, decay = 0.0001, momentum = 0.99, activation function = ReLu .