Assignment 4

Name: Guangbo Yu

ID: 7262891072

1 Boosting

1.1

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

1.2

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

since γ don't appear in the equation, so we can derive h^* independently with γ

1.3

$$L(y_i, \hat{y_i} + \alpha h^*(x_i))$$
$$= (y_i - \hat{y_i} - \alpha h^*(x_i))^2$$
$$\alpha^* = argmin_\alpha \sum_{i=1}^{n} [y_i - \hat{y_i} - \alpha h^*(x_i)]^2$$
$$\frac{\partial \sum_{i=1}^{n} [y_i - \hat{y_i} - \alpha h^*(x_i)]^2}{\partial \alpha} = \sum_{i=1}^{n} (-2h(x_i))[y_i - \hat{y_i} - \alpha h^*(x_i)] = 0$$
$$\sum_{i=1}^{n} h^*(x_i)(y_i - \hat{y_i}) = \alpha \sum_{i=1}^{n} h^{*2}(x_i)$$
$$\alpha^* = \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})h^*(x_i)}{\sum_{i=1}^{n} h^{*2}(x_i)}$$
$$\hat{y_i} \leftarrow \hat{y_i} + \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})h^*(x_i)}{\sum_{i=1}^{n} h^{*2}(x_i)}$$

2 NNET

2.1

$$z^l = \omega^l a^{(l-1)} + b^l$$

since only has one output layer

$$a^{(l-1)} = a^1 = x$$

$$z_{out} = \omega^T x + b$$

$$a^l = \sigma(z) = \sigma(\omega^T x + b)$$

so it is logistic regression

2.2

$$LetL(y, \hat{y}) = \frac{1}{2}\sum_{i=1}^{2}(y_j - \hat{y_j})^2$$

$$\frac{\partial L}{\partial \nu_{jk}} = \frac{\partial L}{\partial \hat{y_j}}\frac{\partial \hat{y_j}}{\partial \nu_{jk}}$$

$$= -\sum_{i=1}^{2}(y_j - \hat{y_j})z_k$$

$$= -[(y_1 - \hat{y_1}) + (y_2 - \hat{y_2})]z_k$$

$$\frac{\partial L}{\partial \omega_{ki}} = \frac{\partial L}{\partial \hat{y_j}}\frac{\partial \hat{y_j}}{\partial z_k}\frac{\partial z_k}{\partial(\sum_{i=1}^{3}\omega_{ki}x_i)}\frac{\partial(\sum_{i=1}^{3}\omega_{ki}x_i)}{\partial \omega_{ki}}$$

$$= -[(y_1 - \hat{y_1}) + (y_2 - \hat{y_2})]\nu_{jk}\tanh'(\sum_{i=1}^{3}\omega_{ki}x_i)x_i$$

$$= -[(y_1 - \hat{y_1}) + (y_2 - \hat{y_2})]\nu_{jk}\frac{1}{\cos^2(\sum_{i=1}^{3}\omega_{ki}x_i)}x_i$$

Programming

(d)

1. [50, 2], accuracy = 0.83143
   [50, 50, 2], accuracy = 0.83477
   [50, 50, 50, 2], accuracy = 0.84100
   [50, 50, 50, 50, 2], accuracy = 0.84211
   training time: 27.245

   as the layer increase, the accuracy also increases.
2. [50, 50, 2], accuracy = 0.84150
   [50, 500, 2], accuracy = 0.84154
   [50, 500, 300, 2], accuracy = 0.84550
   [50, 800, 500, 300, 2], accuracy = 0.84738
   [50, 800, 800, 500, 300, 2], accuracy = 0.84911
   training time: 5438

   almost same accuracy, accuracy increase slowly.

(e)

[50, 500, 2], accuracy = 0.76819
[50, 500, 300, 2], accuracy = 0.72152
[50, 800, 500, 300, 2], accuracy = 0.72152
[50, 800, 800, 500, 300, 2], accuracy = 0.72152
training time:19770.82

the less layer leads to higher accuracy, while the last 3 architectures have almost same accuracy, on the other hand, the training time is much larger than linear architectures while the accuracy is worse than linear model. Also, according to the function graph, it has different shape compared to linear architectures especially near 0 and 1.

(f)

[50, 500, 2], accuracy = 0.81693

[50, 500, 300, 2], accuracy = 0.81739

[50, 800, 500, 300, 2], accuracy = 0.81194

[50, 800, 800, 500, 300, 2], accuracy = 0.79248

training time: 8915.99

the accuracy increases firstly and then decrease, but the first three accuracies are almost the same.

the accuracy is worse than linear and better than sigmoid, and the training time is between linear and sigmoid. Since it greatly accelerates the convergence of stochastic gradient descent compared to sigmoid. For linear, Relu has different function shape.

(g)

10^{-7}	$5 * 10^{-7}$	10^{-6}	$5 * 10^{-6}$	10^{-5}
0.80947	0.81401	0.79944	0.80686	0.80590

Best accuracy: 0.81401 when regularization = $5 * 10^{-7}$

Training time: 14332.90

As the regularization increase, the accuracy increases firstly and begin to decrease in a value, in this case, the value is $5 * 10^{-7}$

(h)

10^{-7}	$5 * 10^{-7}$	10^{-6}	$5 * 10^{-6}$	10^{-5}
0.75666	0.74485	0.73951	0.80109	0.79737

Best accuracy: 0.80109 when regularization = $5 * 10^{-6}$

Training time: 7781.23

The value is decrease as the regularization increase and suddenly increase in some values, the reason is the last two regularizations don’ t early stop, so early stop will decrease the accuracy in this case.

Not the same, not helpful since decreasing accuracy

(i)

10^{-5}	$5 * 10^{-5}$	10^{-4}	$3 * 10^{-4}$	$7 * 10^{-4}$	10^{-3}
0.75942	0.71837	0.71871	0.71940	0.71879	0.71779

Best accuracy: 0.75942 when decay = 10^{-5}

(j)

0.99	0.98	0.95	0.9	0.85
0.84611	0.81739	0.79014	0.75827	0.74378

Best momentum: 0.99

(k)

Regularization: $5 * 10^{-6}$

Decay: 10^{-5}

Momentum: 0.99

Accuracy: 0.86279

Better than the last few parts

(l)

Best parameter:

Architecture: [50, 800, 800, 500, 300, 2]

Regularization: $5 * 10^{-7}$

Decay: 10^{-5}

Momentum: 0.99

Accuracy: 0.87456

Training time: 324933.92