1 Boosting (20 Points)

a) Gradient Calculation

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

b) Weak Learner Selection

Vector notation way If we denote $\mathbf{h} = [h(\mathbf{x}_1), \ldots, h(\mathbf{x}_n)]$, we can write:

$$\sum_{i=1}^{n}(2(y_i - \hat{y}_i) - \gamma h(\boldsymbol{x}_i))^2 = \|2\mathbf{y} - 2\widehat{\mathbf{y}} - \gamma\mathbf{h}\|_2^2 = \|2\mathbf{y} - 2\widehat{\mathbf{y}}\|_2^2 - 2\gamma\left\langle\|2\mathbf{y} - 2\widehat{\mathbf{y}}, \mathbf{h}\right\rangle + \gamma^2\|\mathbf{h}\|_2^2$$

Setting the derivative with respect to γ to zero, we find the optimal γ as follows:

$$\gamma^* = \frac{\langle 2\mathbf{y} - 2\widehat{\mathbf{y}}, \gamma\mathbf{h}\rangle}{\|\mathbf{h}\|_2^2}.$$

Using this optimal value, the decision rule becomes:

$$h^* = \arg\max_{h\in\mathcal{H}} \frac{\langle 2\mathbf{y} - 2\widehat{\mathbf{y}}, \gamma\mathbf{h}\rangle^2}{\|\mathbf{h}\|_2^2}$$

Summation way

$$\sum_{i=1}^{n}(2(y_i - \hat{y}_i) - \gamma h(\boldsymbol{x}_i))^2 = \sum_{i}^{n}\gamma^2 h(\boldsymbol{x_i})^2 - \gamma 2(y_i - \hat{y}_i)h(\boldsymbol{x}_i) + 4(y_i - \hat{y}_i)^2$$

$$\gamma^* = \frac{\sum_i^n (y_i - \hat{y}_i)h(\boldsymbol{x}_i)}{\sum_i^n h(\boldsymbol{x}_i)^2}$$

c) Step Size Selection

$$\begin{aligned}
\alpha^* &= \arg\min_\alpha \sum_{i=1}^{n} L(y_i, \hat{y}_i + \alpha h^*(\boldsymbol{x}_i)) = \arg\min_\alpha \sum_{i=1}^{n}(y_i - \hat{y}_i - \alpha h^*(\boldsymbol{x}_i))^2 \\
&= \arg\min_\alpha \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 - \alpha 2(y_i - \hat{y}_i)h^*(\boldsymbol{x}_i) + \alpha^2 h^*(\boldsymbol{x}_i)^2 \\
&= \arg\min_\alpha \sum_{i=1}^{n} -\alpha 2(y_i - \hat{y}_i)h^*(\boldsymbol{x}_i) + \alpha^2 h^*(\boldsymbol{x}_i)^2 \\
&= \arg\min_\alpha \alpha \sum_{i=1}^{N} -2(y_i - \hat{y}_i)h^*(\boldsymbol{x}_i) + \alpha^2 \sum_{i=1} h^*(\boldsymbol{x}_i)^2 \\
\alpha^* &= \frac{\sum_i^n (y_i - \hat{y}_i)h^*(\boldsymbol{x}_i)}{\sum_i^n h^*(\boldsymbol{x}_i)^2}
\end{aligned}$$

Similarly, vector notation will give the following alternative result:

$$\alpha^* = \frac{\langle \mathbf{y} - \widehat{\mathbf{y}}, \mathbf{h}^* \rangle}{\|\mathbf{h}^*\|_2^2}$$

2 Neural Networks (20 Points)

a) Logistic Regression

Suppose that our neural network has edge weights $\{w_{ij}^k\}$ from each previous (k^{th}) layer's node i to each subsequent hidden layer's node j. Then the transformation from one layer to another is a linear transform W^k. Further suppose that at each layer, the node's activation is a linear function $f_j^k(x) = a_j^k x + b_j^k$. Stacking f_j^k in order into a vector function F^k with corresponding A^k and B^k, we can write the entire network as one (biased) linear function $W^K A^{K-1} W^{K-1} A^{K-2} \dots A^1 W^1 x + \sum_{i=1}^{N-1} W^K W^{K-1} \dots W^{i+1} B^i$ where x is the raw input. Rewrite this $\tilde{W} X + \tilde{B}$. Note that the final leftmost dimension of W must be 1 (stipulated by the problem). Then the input to the final layer is $wx + b$.

Taking this as input into our logistic output layer, we see that our final output is exactly $\frac{1}{1+\exp(wx+b)}$, which is logistic regression (with bias).

b) Back-propagation

First, we note that the derivative of $h(a) = \tanh(a)$ function, can be written as $h'(a) = 1 - h(a)^2$. Thus, we can write:

Forward Propagation

$$z_k = \tanh\left(\sum_{i=1}^{3} w_{ki} x_i\right)$$
$$\widehat{y}_j = \sum_{k=1}^{4} v_{jk} z_k.$$

Backpropagation Defining $\delta_j = \widehat{y}_j - y_j$, we backpropagate these to obtain δs for the hidden units:

$$\delta_k = (1 - z_k^2) \sum_{j=1}^{2} v_{jk} \delta_j.$$

Computing the Derivatives Finally, the derivatives with respect to the coefficients can be obtained as:

$$\frac{\partial L}{\partial w_{ki}} = \delta_k x_i, \qquad \frac{\partial L}{\partial v_{jk}} = \delta_j z_j.$$

You can verify that these equation give you the correct derivatives.

Programming

Deep Learning (60 Points)

The numerical tables for this programming assignment can differ a lot because of random initializations of neural networks. The best values of parameters reported can also differ a lot. Please be very generous in grading for the tables in this problem.

(d) **Linear activations**: *[5 points]*

Architecture	Test accuracy
50,2	0.8222
50,50,2	0.8331
50,50,50,2	0.8411
50,50,50,50,2	0.8441 (best)

Table 1: Linear activation with thin architectures

Pattern: Accuracy ideally should not change with depth, but in this case it increases slightly with increasing depth. Can show a lot of fluctuation if run many times.

Architecture	Test acc.
50,50,2	0.8338
50,500,2	0.8397
50,500,300,2	0.8446
50,800,500,300,2	0.8465
50,800,800,500,300,2	0.8479 (best)

Table 2: Linear activation with growing architectures

Pattern: Accuracy ideally should not change much, but can increase in this case slightly with increasing depth. Can show fluctuation in values if run many times.
Time taken: 308.50 secs

(e) **Sigmoid activations**: *[5 points]*

Architecture	Test acc.
50,50,2	0.7281
50,500,2	0.7642 (best)
50,500,300,2	0.7136
50,800,500,300,2	0.7136
50,800,800,500,300,2	0.7136

Table 3: Sigmoid activation with growing architectures

Pattern: Accuracy first increases but later decreases with growing depth and width.
Possible cause: Overfitting for bigger networks; Also shown by best network being [50,500,2]. Overfitting doesn't happen for linear layers since combination of linear layers is also linear.
Time taken: 869.73 secs. Much larger than linear layer, since computing sigmoid needs computation of exponential

In this part, though accuracy values and best architecture might differ across students, the pattern for accuracies and time should be clearly visible.

(f) **ReLu activations**: *[5 points]*

Architecture	Test acc.
50,50,2	0.8038
50,500,2	0.8213 (best)
50,500,300,2	0.8071
50,800,500,300,2	0.8124
50,800,800,500,300,2	0.7950

Table 4: ReLu activation with growing architectures

Pattern: Accuracy first increases but later decreases with growing depth and width.
Possible cause: Overfitting for bigger networks, similar to sigmoid.
Doesn't overfit as much as sigmoid though and has accuracy comparable to linear network which also doesn't overfit much.
Time taken: 387.85 secs. Comparable to linear layer, since derivative is either 0 or 1 and much smaller time than sigmoid.
In this part, though accuracy values and best architecture might differ across students, the pattern for accuracies and time should be clearly visible.

(g) **L2-Regularization**: *[5 points]*

λ	Test acc.
10^{-7}	0.8040
5×10^{-7}	0.80678890
10^{-6}	0.80678891 (best)
5×10^{-6}	0.8048
10^{-5}	0.8046

Table 5: L2-Regularization

Pattern: Accuracy first increases due to control on overfitting but later decreases because of excessively constrained parameters.
The test accuracies can differ quite a lot. Since they are very similar for almost all values of λ, the best λ could be different for different initializations of the network.

(h) **Early stopping and L2-Regularization**: *[5 points]*

λ	Test acc.
10^{-7}	0.7605
5×10^{-7}	0.8516 (best)
10^{-6}	0.7812
5×10^{-6}	0.8512
10^{-5}	0.8455

Table 6: Early Stopping and L2-Regularization

Pattern: No apparent trend, except that the best accuracy has improved (over only L2-regularization). Best L2-regularization parameter value can change or remain the same. Early stopping helps by stopping in time before the model overfits the training set.

(i) **SGD with weight decay**: *[5 points]*
Pattern: Best value: 5×10^{-5}. Can differ sometimes but most students will get this result.

β	Test acc.
10^{-5}	0.7180
5×10^{-5}	0.7989 (best)
10^{-4}	0.7488
3×10^{-4}	0.7693
7×10^{-4}	0.6444
10^{-3}	0.7725

Table 7: SGD with weight decay

(j) **SGD with momentum**:
Best value: 0.99 *[5 points]* . Mostly 0.99 is best, sometimes but rarely 0.98 might be the best.

(k) **Combining the above**:
Test accuracy: 0.862. Can be more or less than previous parts, either is fine. *[10 points]*

(l) **Grid search with cross-validation**:
Best Config: arch = [50, 800, 800, 500, 300, 2], $\lambda = 10^{-7}$, $\beta = 10^{-5}$, momentum = 0.99, actfn = 'relu', best accuracy = 0.8781 *[15 points]* Results can differ across different runs, although the best accuracy seems somewhat consistent. Still, be generous to different values of best accuracy as long as it is better than test accuracy of all the other previous parts.