QA3

Lec5 – Bias and Variance, ensemble

- How do we link model performance to bias and variance?
    - Answer: if a model has high bias, it produces high training error and high validation error. In this case we might need to increase model complexity. If a model has high variance, it has low training error, but high validation error, indicating an overfitting problem. We might need to reduce model complexity for it.
- What is different when comparing boosting to bagging?
    - Answer: boosting reduces both bias and variance while bagging prevents overfitting (i.e. reduces variance). It is done by sequential training, as opposed to bagging, which trains data in parallel (like random forests).

Lec6 – Neural Networks

- What do different activation functions do and how would one choose which one to use?
    - Answer: common activation functions include
        - ReLU: f(x) = max(0,x). This is simple but effective (by setting all negative values to 0), and is quick to compute. It is commonly used in hidden layers and works well for most tasks, while the downside is that sometimes it can suffer from the dying ReLU problem where neurons get stuck and output zeros when inputs are negative.
        - Leaky ReLU: f(x) = max(alpha*x, x). This corrects the possible dying problem in ReLU by allowing a small, non-zero gradient when $x < 0$.
        - Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$. It maps the input to a range between 0 and 1, which can be used in probability/binary classification tasks. The concern is that it can cause vanishing gradients during backpropagation due to very small gradients for extreme values.

        - Softmax: $f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$. It converts logits into probabilities that sum to 1, which can be used in the output layer for multi-class classification problems (i.e. it produces a probability distribution over classes).
    - Decision rules:
        - For classification tasks, in the output layer, if its binary → sigmoid; multi-class → softmax. For regression tasks, just use identity function (y=z).
        - For hidden layers, ReLU is a good start, while if encountering dead neuron issues, one may change to Leaky ReLU.
        - Considering the depth of the network, if it is shallow, sigmoid works fine. If it is deep, consider using ReLU to avoid vanishing gradients.
- What are some common hyperparameters in neural networks?
    - Answer: some hyperparameters include:
        - Learning rate: controls how much the model weights are updated during training. Higher means faster updates, but too high may cause the model not able to converge overshoot the optimal solutions.
        - Depth (number of hidden layers): adding more layers increases the model's capacity to learn complex patterns but may also lead to overfitting.
        - Width (number of neurons per layer): similar to depths, more neurons = more capacity while too much → overfitting.
        - Activation function: this has been addressed in the question above. It has impact in training speed and model performance.
        - Epochs: the number of complete passes through the entire training dataset during training. Too many can lead to overfitting.
        - Batch size: the number of training samples used in one iteration before updating the weights. A small batch size → frequent update, faster convergence, but can introduce more noise (while larger batch is more stable but slower).
        - Dropout rate: a regularization technique that randomly drops a fraction of neurons during training to prevent overfitting.
        - And more