Assignment 1 Theory - Aaron Dubale

Tuesday, February 17, 2026 11:01 PM

1. Understanding learning as function approximation
    a. Counting Model Capacity
        i. Linear functions

            $f(x) = w_1 x + w_0$

            Parameters: 2
        ii. Quadratic Polynomials

            $f(x) = w_2 x^2 + w_1 x + w_0$

            Parameters: 3
        iii. Cubic Polynomials

            $f(x) = w_3 x^3 + w_2 x^2 + w_1 x + w_0$

            Parameters: 4
        iv. Two-layer neural networks with 5 hidden units and ReLU Activation

            First layer: 5 weights and 5 biases

            Second layer: 5 weights and 1 bias

            Parameters: 16

    b. Model Capacity and Real-World Constraints
        i. Which hypothesis class is likely to overfit? Why is this dangerous in prod?

            The most likely class to overfit is the neural network. This has more parameters than datapoints, so it is very likely to just memorize the training set, leading ot high variance and low training error.
        ii. Which is most likely to underfit? When might you deliberately choose this simpler model?

            The linear model is most likely to underfit. It can only represent a straight line, so it will not be able to represent any bends or curves in the relationship between price-sqft.
        iii. Suppose the true relationship is y=50+100x+0.5x^2 (quadratic). Explain what will happen if you fit:
            1) A linear model

                It will underfit because it cannot represent the curvature in the 0.5x^2 term. It instead will be a straight line that might look

okay in some range of the line, but as x approaches infinity, the curvature will contribute to a larger amount of error.

2) A degree-10 polynomial

It can, theoretically, perfectly represent the data by setting all coefficients x^3 through x^10 to 0, but more realistically, it'll overfit the data by memorizing the training dataset.

c. Why training error misleads you

i. Which model will perform best on new houses?

The degree-3 polynomial has the lowest Test MSE, meaning it is the best-generalized in the set of models. This means the degree-3 will perform best on new houses.

ii. The degree-10 polynomial achieves near-zero training error but terrible test error. What phenomenon is this? Why is it dangerous?

This is overfitting, the degree-10 polynomial is too powerful for the problem and is able to almost-perfectly memorize the dataset, but since there is no real generalization being built, it cannot perform well on new data compared to some of its more aptly-fit counterparts.

iii. If you could only see training error, which model would you choose? Would this be the right choice? How do practitioners solve this problem?

I would choose the neural net, but it would be wrong. Practitioners would use a validation set and a test set to ensure that they can capture real performance on unseen data. Data separation is an important concept here.

2. Loss Functions and Their Properties

a. Computing Different Losses

i. Mean Squared Error: 0.43

ii. Mean Absolute Error: 0.567

iii. Root Mean Squared Error: 0.656

b. Outlier sensitivity in Production Systems

i. Recalculate MSE and MAE

MSE: 8.43

MAE: 1.9

ii. By what factor did each loss increase?

MSE increased by: 19.6x

MAE increased by: 3.35x

iii. Which loss is more sensitive to outliers? Explain why this makes sense given the mathematical form of each loss.

MSE is more sensitive to outliers. This makes sense intuitively because MSE is squared. It does this to penalize larger errors more harshly than smaller errors, which lets a single bad prediction greatly skew the average.

c. Why Gradient Descent Needs Smooth Losses

i. Why can't we use gradient descent to optimize loss directly?

In gradient descent, the gradient is the derivative of the loss function. The derivative shows rate of change, but in this function, the rate of change is 0 in the whole graph, so there is no way to optimize.

ii. Instead, we use smooth losses like cross-entropy or hinge loss. What property must a loss function have to be optimizable with gradient descent? Why?

A loss function must be smooth and differentiable. This allows for gradients to be calculated and for parameters to be tweaked using the gradient to better tune them toward accuracy.

3. Why Stacking Linear Layers Does Nothing

a. Two Linear Layers

i. Substitute the first equation into the second to eliminate h:

$$y = W_2(W_1x + b_1) + b_2 = W_2W_1x + W_2b_1 + b_2$$

ii. Let $W = W_2W_1$ and $b = W_2b_1 + b_2$

Show that the composition is equivalent to a single linear layer:

$$y = Wx + b$$

iii. What does this imply about the expressive power of stacking linear layers?

Stacking linear layers does not increase expressive power. Even the stacked linear layers simplify to a single linear layer, which will not capture a nonlinear relationship.

b. With Nonlinearity

i. Can you simplify this into a single linear layer? Why or why not?

You cannot, because the nonlinear σ cannot be distributed or collapsed in a way that brings the function back down to linearity. This means that the nonlinearity is now part of the model, which allows for nonlinear expression.

ii. This is why we say "activation functions enable neural networks to learn nonlinear functions." Explain this in your own words.

Nonlinear activation functions allow for each layer in a neural

network to apply a nonlinear transformation to its inputs. These transformations across multiple layers allow for the representation of nonlinear decision boundaries, which would otherwise simplify back down to y=Wx+b if the transformations were not there.

4. Simgoid and Tanh Equivalence

   a. Sigmoid in Exponential Form

      i. Multiply numerator and denominator by e^a and simplify

$$\sigma(a) = \frac{1}{1 + e^{-a}}$$

$$\sigma(a) = \frac{1 * e^{a}}{(1 + e^{-a})e^{a}}$$

$$\sigma(a) = \frac{e^{a}}{e^{a} + 1}$$

      ii. Your answer should be in the form: $\sigma(a) = \frac{e^{a}}{something}$. What is the denominator?

$$e^{a} + 1$$

   b. Scaling the Input

      i. Write $\sigma(2a)$ using the exponnetial form from part (a):

$$\sigma(2a) = \frac{e^{2a}}{e^{2a} + 1}$$

      ii. Multiply num and denom by e^{-a} and simplify to get an expression with both e^{a} and e^{-a}. Show work.

$$\sigma(2a) = \frac{e^{2a}}{e^{2a} + 1}$$

$$\sigma(2a) = \frac{e^{2a}e^{-a}}{(e^{2a} + 1)e^{-a}} = \frac{e^{a}}{e^{a} + e^{-a}}$$

$$\sigma(2a) = \frac{e^{a}}{e^{a} + e^{-a}}$$

      iii. Write the final simplified result for $\sigma(2a)$.

$$\sigma(2a) = \frac{e^{a}}{e^{a} + e^{-a}}$$

   c. Deriving the Tanh Relationship

      i. Compute $2\sigma(2a)$using your result from part b

$$2\sigma(2a) = \frac{2e^{a}}{e^{a} + e^{-a}}$$

      ii. Now compute $2\sigma(2a) - 1$

$$2\sigma(2a) - 1 = \frac{2e^a}{e^a + e^{-a}} - 1$$

$$= \frac{2e^a}{e^a + e^{-a}} - \frac{e^a + e^{-a}}{e^a + e^{-a}} = \frac{e^a - e^{-a}}{e^a + e^{-a}}$$

iii. Compare this with the definition of tanh(a). Write the relationship discovered.

$\text{Tanh(a)}=\frac{e^a - e^{-a}}{e^a + e^{-a}}$

This means $\tanh(a)=2\sigma(2a) - 1$.

d. Neural Network Implications

i. Using the identity relating σ and tanh that was derived in part c, explain how to rewrite this network using tanh activations instead. Show how the weights and scale parameter change.

ii. What does this tell you about the representational power of sigmoid vs tanh networks?

iii. If they have the same representational power, why might we prefer tanh over sigmoid for hidden layers?