Week 3 - Shallow Neural Network

Neural Networks Overview

- What is a Neural Network? A computation graph with forward passes and backward propagation of the errors.

Network Representation

- Input layer
- Hidden layer. The true values of these layers are not observed in the training set.
- Output layer
- Each layer has its activation function $a^{[i]}$

Computing Neural Network s Output

- Each neuron computes a two step process. The first step is $z = w^T x + b$ and the second step is the activation step $a = \sigma(z)$
- Each layer has its own set of activations with dimensions correspondent to the number of neurons
- Cumulative layers impact on each other as each one become the next one input

Vectorizing across multiple examples

- for $i = 1$ to m
  - $z^{[1](i)} = W^{[1]} x^{(i)} + b^{[1]}$
  - $a^{[1](i)} = \sigma(z^{[1](i)})$
  - $z^{[2](i)} = W^{[2]} x^{(i)} + b^{[2]}$
  - $a^{[2](i)} = \sigma(z^{[2](i)})$
- Becomes:
  - $Z^{[1]} = W^{[1]} X + b^{[1]}$
  - $A^{[1]} = \sigma(Z^{[1]})$
  - $Z^{[2]} = W^{[2]} X + b^{[2]}$
  - $A^{[2]} = \sigma(Z^{[2]})$

Activation functions

- The sigmoid is called an activation function
- tanh is mostly better than sigmoid because it pushes the mean to 0 and make it easier for the next layers to learn
- $a = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
- $\mathrm{ReLU} = \max(0, x)$ much faster to learn because the slope is constant even if it's not well defined in the 0
- $\mathrm{LeakyReLU} = \max(\alpha z, z)$ with small α so the slope in the negative numbers isn't 0

Why do you need non-linear activation functions?

- Linear activation function = identity function
- The composition of two linear functions is itself a linear function
- Linear activations are more or less useless as they don't add any solving power to the neurons
- Linear activation might only be useful in regression problems with real output. And should only be used in the output layer. But even though it might makes more sense to use a ReLU if there's no need for the negative values

Derivative of activation functions

- Sigmoid
- $g(z) = \frac{1}{1+e^{-z}}$
- $\frac{d}{dx}g(x) = g^{'}(z) = g(z)(1-g(z))$
- Tanh
- $g(z) = \tanh(z)$
- $\frac{d}{dx}g(x) = g^{'}(z) = 1-g(z)^2$
- ReLU
- $g(z) = \max(0, z)$
- $g^{'}(z) = \{\begin{matrix} 0 \text{ if } z < 0 \\ 1 \text{ if } z \geq 0 \end{matrix}$
- LeakyReLU
- $g(z) = \max(\alpha z, z)$
- $g^{'}(z) = \{\begin{matrix} \alpha \text{ if } z < 0 \\ 1 \text{ if } z \geq 0 \end{matrix}$

Gradient descent for Neural Networks

- Forward propagation:
  - $Z^{[1]} = W^{[1]}X + b^{[1]}$
  - $A^{[1]} = g^{[1]}(Z^{[1]})$
  - $Z^{[2]} = W^{[2]}X + b^{[2]}$
  - $A^{[2]} = g^{[1]}(Z^{[2]})$
- Backward Propagation:
  - $dZ^{[2]} = A^{[2]} - Y$
  - $dw^{[2]} = \frac{1}{m}dZ^{[2]}A^{[1]T}$
  - $db^{[2]} = \frac{1}{m}np.sum(dZ^{[2]}, axis=1, keepdims=True)$
  - $dZ^{[1]} = W^{[2]}dZ^{[2]} * g^{'[2]}(Z^{[1]})$
  - $dw^{[1]} = \frac{1}{m}dZ^{[1]}X^T$
  - $db^{[1]} = \frac{1}{m}np.sum(dZ^{[1]}, axis=1, keepdims=True)$

Random Initialization

- Initializing the bias to 0 is okay but initializing the weights to 0 can be a problem.

- As different neurons have the same weight matrix they are symmetrical therefore computing the same thing. Zero-initializing the weight matrix on all the neurons easily leads to this problem. A simple solution is initializing the weights randomly
- $W^{[1]} = np.random.randn(n_{neurons}, n_{inputs}) * 0.001$
- $b^{[1]} = np.zeros(n_{neurons}, 1)$
- Usually we want to initialize the weights to small values because functions as $\tanh(x)$ or $\sigma(x)$ tend to saturate in big values

Quiz

1. Which of the following are true? (Check all that apply.)
   - ☐ $a_4^{[2]}$ is the activation output of the 2^{nd} layer for the 4^{th} training example
   - ☐ X is a matrix in which each row is one training example.
   - ✓ $a^{[2]}$ denotes the activation vector of the 2^{nd} layer.
   - ✓ X is a matrix in which each column is one training example.
   - ☐ $a^{[2](12)}$ denotes activation vector of the 2^{th} layer on the 2^{nd} training example.
   - ✓ $a^{[2](12)}$ denotes the activation vector of the 2^{nd} layer for the 12^{th} training example.
   - ✓ $a^{[2]}$_4 is the activation output by the 4^{th} neuron of the 2^{nd} layer
2. The tanh activation usually works better than sigmoid activation function for hidden units because the mean of its output is closer to zero, and so it centers the data better for the next layer. True/False?
   - ✓ True
   - ☐ False
3. Which of these is a correct vectorized implementation of forward propagation for layer ll, where 1 \leq l \leq L1≤l≤L?
   - ☐ $Z^{[l]} = W^{[l-1]}A^{[l]} + b^{[l-1]}$ $A^{[l]} = g^{[l]}(Z^{[l]})$
   - ☐ $Z^{[l]} = W^{[l]}A^{[l]} + b^{[l]}$ $A^{[l+1]} = g^{[l+1]}(Z^{[l]})$
   - ☐ $Z^{[l]} = W^{[l]}A^{[l]} + b^{[l]}$ $A^{[l+1]} = g^{[l]}(Z^{[l]})$
   - ✓ $Z^{[l]} = W^{[l]}A^{[l-1]} + b^{[l]}$ $A^{[l]} = g^{[l]}(Z^{[l]})$
4. You are building a binary classifier for recognizing cucumbers (y=1) vs. watermelons (y=0). Which one of these activation functions would you recommend using for the output layer?
   - ☐ ReLU
   - ☐ Leaky ReLU
   - ✓ sigmoid
   - ☐ tanh
5. Consider the following code:

   ```
   A = np.random.randn(4,3)
   B = np.sum(A, axis = 1, keepdims = True)
   ```

   What will be B.shape? (If you're not sure, feel free to run this in python to find out).

   - ✓ (4, 1)
   - ☐ (1, 3)
   - ☐ (4,)
   - ☐ (, 3)
6. Suppose you have built a neural network. You decide to initialize the weights and biases to be zero. Which of the following statements is true?

- ✓ Each neuron in the first hidden layer will perform the same computation. So even after multiple iterations of gradient descent each neuron in the layer will be computing the same thing as other neurons.
- ☐ Each neuron in the first hidden layer will perform the same computation in the first iteration. But after one iteration of gradient descent they will learn to compute different things because we have "broken symmetry".
- ☐ Each neuron in the first hidden layer will compute the same thing, but neurons in different layers will compute different things, thus we have accomplished "symmetry breaking" as described in lecture.
- ☐ The first hidden layer's neurons will perform different computations from each other even in the first iteration; their parameters will thus keep evolving in their own way.

7. Logistic regression's weights w should be initialized randomly rather than to all zeros, because if you initialize to all zeros, then logistic regression will fail to learn a useful decision boundary because it will fail to "break symmetry", True/False?
   - ☐ True
   - ✓ False
8. You have built a network using the tanh activation for all the hidden units. You initialize the weights to relative large values, using np.random.randn(..,..)*1000. What will happen?
   - ☐ This will cause the inputs of the tanh to also be very large, causing the units to be "highly activated" and thus speed up learning compared to if the weights had to start from small values.
   - ☐ This will cause the inputs of the tanh to also be very large, thus causing gradients to also become large. You therefore have to set \alphaα to be very small to prevent divergence; this will slow down learning.
   - ✓ This will cause the inputs of the tanh to also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow.
   - ☐ It doesn't matter. So long as you initialize the weights randomly gradient descent is not affected by whether the weights are large or small.
9. Consider the following 1 hidden layer neural network. Which of the following statements are True? (Check all that apply).
   - ☐ $W^{[1]}$ will have shape (2, 4)
   - ✓ $b^{[1]}$ will have shape (4, 1)
   - ✓ $W^{[1]}$ will have shape (4, 2)
   - ☐ $b^{[1]}$ will have shape (2, 1)
   - ✓ $W^{[2]}$ will have shape (1, 4)
   - ☐ $b^{[2]}$ will have shape (4, 1)
   - ☐ $W^{[2]}$ will have shape (4, 1)
   - ✓ $b^{[2]}$ will have shape (1, 1)
10. In the same network as the previous question, what are the dimensions of $Z^{[1]}$ and $A^{[1]}$?
    - ☐ $Z^{[1]}$ and $A^{[1]}$ are (1,4)
    - ☐ $Z^{[1]}$ and $A^{[1]}$ are (4,1)
    - ☐ $Z^{[1]}$ and $A^{[1]}$ are (4,2)
    - ✓ $Z^{[1]}$ and $A^{[1]}$ are (4,m)