# ML Concepts - Neural Nets, Training Neural Nets and Multi Class Neural Nets

- Nonlinear problems can't be accurately predicted with a linear model.

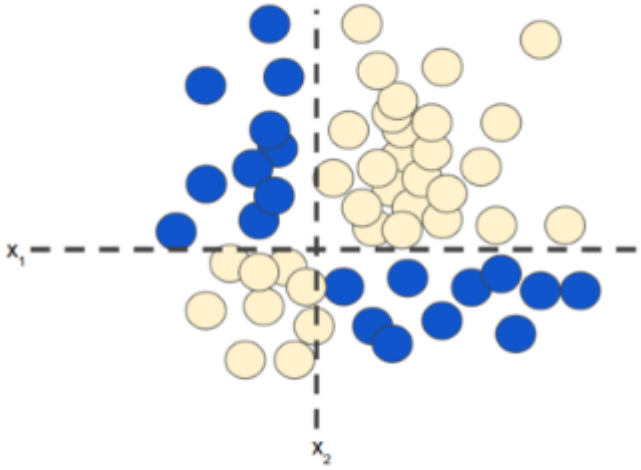


Figure 1. Nonlinear classification problem.

- Consider the below dataset. This data cannot be effectively classified using a linear model.

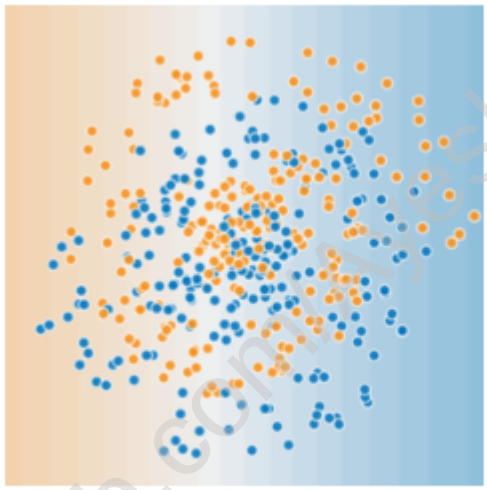

Figure 2. A more difficult nonlinear classification problem.

- A linear model is represented as a graph with blue circles as input features and a green circle representing the weighted sum of inputs

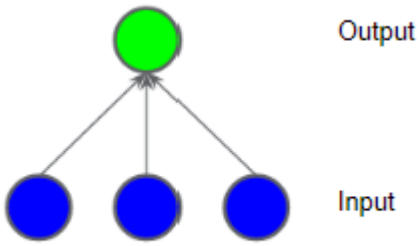


**Figure 3. Linear model as graph.**

- Introduce a "hidden layer" with yellow nodes, each being a weighted sum of blue input node values. The output remains a weighted sum of the yellow nodes (Figure 4).
- Despite the addition of a hidden layer, this model is still linear because its output is a linear combination of its inputs.

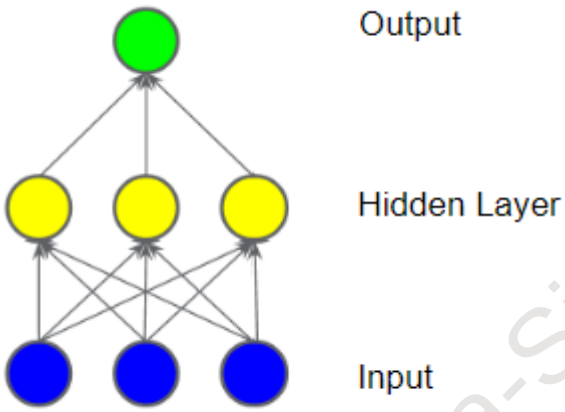


**Figure 4. Graph of two-layer model.**

- Add another hidden layer of weighted sums (Figure 5). However, even with multiple hidden layers, the model remains linear.

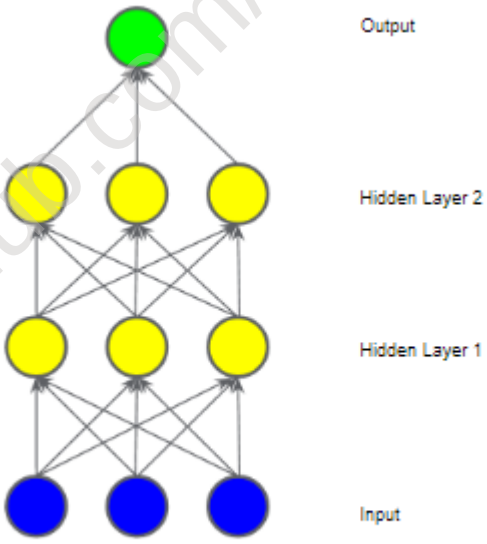


**Figure 5. Graph of three-layer model.**

- To introduce nonlinearity, pass each node in the hidden layer through a nonlinear function. This is known as the activation function (Figure 6).
- The transformation layer applies a nonlinear function before passing values to the next layer.
- With activation functions, adding layers becomes more impactful. Stacking nonlinearities allows modeling of complex relationships between inputs and predicted outputs.

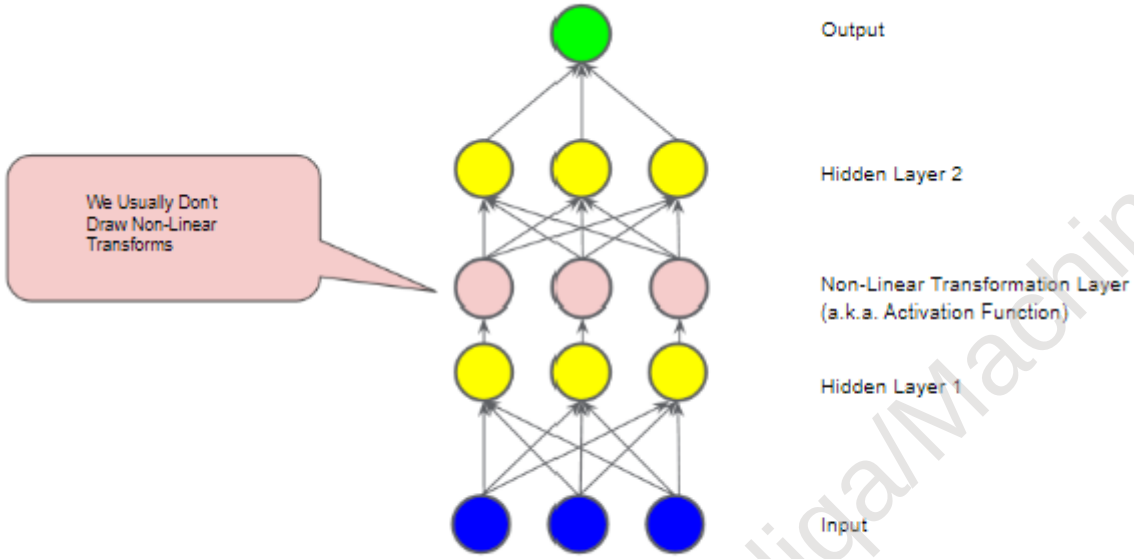


**Figure 6. Graph of three-layer model with activation function.**

# *Activation Functions

1. **Sigmoid Activation Function**:
   - Converts the weighted sum to a value between 0 and 1.
   - It has a characteristic S-shaped curve.
   - Figure 7 depicts the sigmoid activation function.
2. **Rectified Linear Unit (ReLU)**:
   - Often more effective than sigmoid for many applications.
   - Provides a simpler and faster computation.
   - ReLU returns 0 for negative inputs and the input value for positive inputs.
   - Figure 8 illustrates the ReLU activation function.
3. **Choosing an Activation Function**:
   - Empirical findings suggest that ReLU tends to work better than sigmoid in many scenarios.
   - ReLU's responsiveness doesn't diminish as quickly as sigmoid, making it more suitable for deep learning models.
4. **Flexibility in Activation Functions**:
   - While Sigmoid and ReLU are common, any mathematical function can serve as an activation function.
   - The chosen function affects how the network learns and generalizes.
5. **Node Value Calculation**:

- For a given activation function, the value of a node in the network is determined by applying the function to the weighted sum of its inputs.

6. **TensorFlow Support for Activation Functions**:
   - TensorFlow offers a range of activation functions as part of its neural network operations.
   - It's recommended to start with ReLU due to its effectiveness.

$$F(x) = \frac{1}{1 + e^{-x}}$$

Here's a plot:

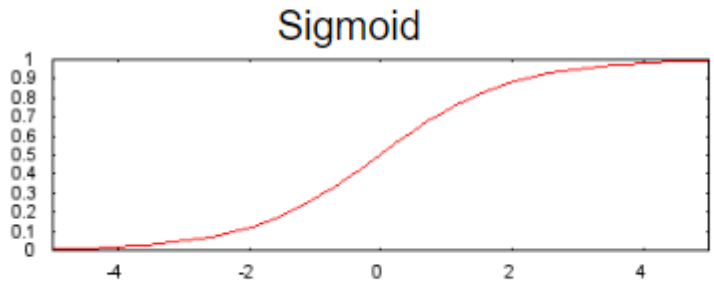


**Figure 7. Sigmoid activation function.**

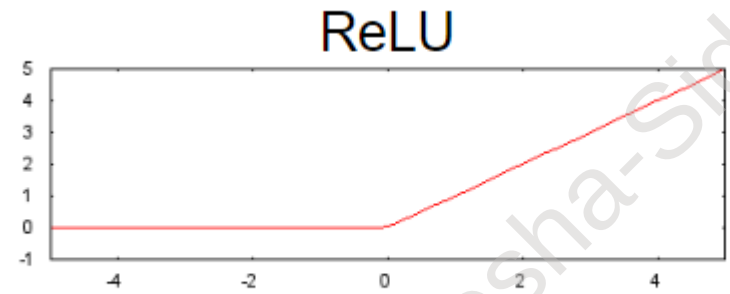


**Figure 8. ReLU activation function.**

Now our model has all the standard components of what people usually mean when they say "neural network":

- A set of nodes, analogous to neurons, organized in layers.
- A set of weights representing the connections between each neural network layer and the layer beneath it. The layer beneath may be another neural network layer, or some other kind of layer.
- A set of biases, one for each node.
- An activation function that transforms the output of each node in a layer. Different layers may have different activation functions.

## *Best Practices in Training Neural Nets*

**Failure Cases in Backpropagation**:

1. **Vanishing Gradients**:
   - Gradients for lower layers can become very small, especially in deep networks.

- This causes the lower layers to train very slowly or not at all.
- The ReLU activation function can help mitigate vanishing gradients.

2. **Exploding Gradients**:
   - When weights are very large, gradients for lower layers involve products of many large terms.
   - This can lead to gradients that become too large to converge.
   - Techniques like batch normalization and lowering the learning rate can help prevent exploding gradients.
3. **Dead ReLU Units**:
   - If the weighted sum for a ReLU unit falls below 0, the unit can become "dead".
   - It outputs 0 activation, contributing nothing to the network's output, and gradients can't flow through it during backpropagation.
   - Lowering the learning rate can help prevent ReLU units from dying.

**Dropout Regularization**:

- Dropout is a form of regularization used in neural networks.
- It randomly "drops out" unit activations in a network for a single gradient step.
- The dropout rate determines the proportion of activations to drop out:
  - 0.0 means no dropout regularization.
  - 1.0 means drop out everything, which leads to the model learning nothing.
  - Values between 0.0 and 1.0 are more useful and provide varying degrees of regularization.

In summary, understanding and addressing failure cases in backpropagation is crucial for effectively training neural networks. Techniques like using appropriate activation functions, batch normalization, adjusting learning rates, and applying dropout regularization can help mitigate these issues and improve the performance of the network.

# *Multi Class*

**One vs. All Approach**:

- **Concept**:
  - In the one-vs.-all approach, for a classification problem with N possible solutions, N separate binary classifiers are used.
  - Each binary classifier is designed to answer a separate classification question.
- **Training Process**:
  - During training, the model goes through a sequence of binary classifiers, training each to distinguish one class from the rest.
  - For example, if there are 5 possible classes, there will be 5 binary classifiers.
- **Example**:
  - Given an image of a dog, the classifiers might be trained as follows:
    1. Is this image an apple? No.
    2. Is this image a bear? No.

3. Is this image candy? No.
4. Is this image a dog? Yes.
5. Is this image an egg? No.

- **Efficiency**:
  - While reasonable for a small number of classes, this approach becomes inefficient as the number of classes increases.

**Improved Approach with Deep Neural Network**:

- **Neural Network Representation**:
  - An efficient alternative is to use a deep neural network in which each output node represents a different class.
- **Representation Example**:
  - If there are 5 classes, there will be 5 output nodes, each corresponding to one class.
- **Efficiency Benefits**:
  - This approach is significantly more efficient compared to the traditional one-vs.-all approach, especially as the number of classes increases.
- **Advantages**:
  - It leverages the capabilities of deep neural networks for multi-class classification.

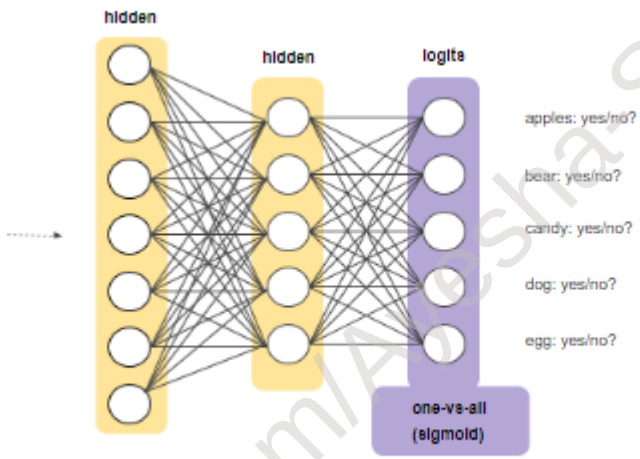


Figure 1. A one-vs.-all neural network.

## *Softmax*

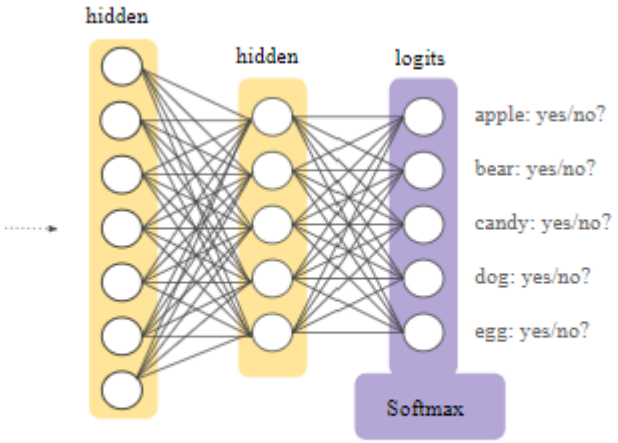


**Figure 2. A Softmax layer within a neural network.**

- **Logistic Regression Recap**:
  - Logistic regression produces a decimal between 0 and 1.0, representing the probability of a given class.
- **Extension to Multi-Class Classification**:
  - Softmax extends this idea into a multi-class world, assigning decimal probabilities to each class.
  - The probabilities must add up to 1.0, ensuring a valid probability distribution.
- **Training Convergence**:
  - The additional constraint of probabilities summing to 1.0 in Softmax helps training converge more quickly.
- **Example**:
  - In an image analysis problem with classes like apple, bear, candy, dog, and egg, Softmax might assign probabilities like:
    - Apple: 0.001
    - Bear: 0.04
    - Candy: 0.008
    - Dog: 0.95
    - Egg: 0.001
- **Implementation in Neural Networks**:
  - Softmax is implemented as a neural network layer just before the output layer.
  - The Softmax layer must have the same number of nodes as the output layer.
- **Variants of Softmax**:
  - **Full Softmax** calculates probabilities for every possible class.
  - **Candidate Sampling** calculates probabilities for positive labels and a random sample of negative labels.
    - Useful when determining between a few classes out of many.
- **One Label vs. Many Labels**:
  - Softmax assumes each example belongs to exactly one class.
  - For cases where examples can belong to multiple classes simultaneously, Softmax is not appropriate.

    - Multiple logistic regressions must be used instead.
- **Efficiency Considerations**:
    - Full Softmax is efficient for a small number of classes but can become computationally expensive with a large number of classes.
    - Candidate sampling can improve efficiency for problems with a large number of classes.

In summary, Softmax is a crucial component in multi-class classification, providing a probability distribution over possible classes. It helps ensure the model's predictions are consistent and converge effectively during training. However, it's important to choose the appropriate Softmax variant based on the specific problem and dataset characteristics.