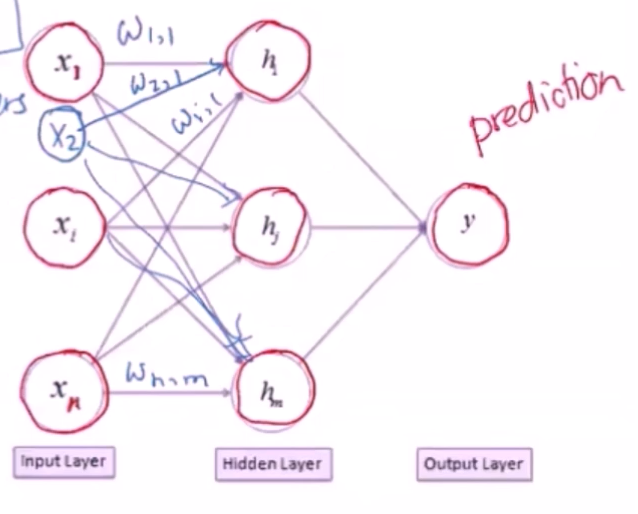
**Machine Learning : Neural Networks (Part 1)**

**Linearly Separable**

* **If the classes can be separated by a hyperplane, then they are linearly separable**

**Adding Layers**

* **Notation as weights from input d to hidden unit i.**
  + **First to second layers.**
* **2-layer neural networks have :**
  + **n inputs**
  + **m hidden units**
  + **1 output unit**



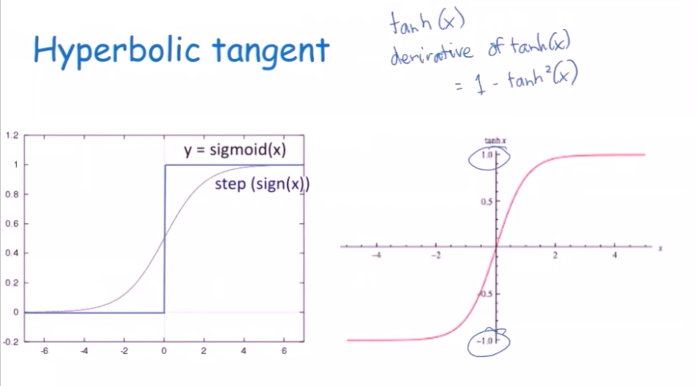
* **Prediction is the similar to the perceptron**
  + **1. Compute activation of hidden nodes**
  + **2. Compute activation of output node(s) Based on hidden nodes activation and second layer of weights**

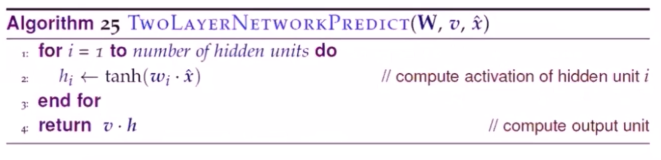
**Activation Function and Gradient Descent**

Text

Description automatically generated with low confidence

* **F is the lost function we are trying to minimize this function.**
* **K is number of iterations of which we will make adjustments.**
* **are the learning rates, it’s a set not just one rate.**





* **W : weights from input node to hidden node**
* **v : Weights from hidden node to output node**
* **x : feature vector for data point**

**Inductive Bias**

* **Smooth interpolation between data points**
* **Prefer sparse weight vectors (small weight values)**

**XOR (with step functions)**

Diagram

Description automatically generated

**How big of a network do I need?**

* **Let F be a continuous function on a bounded subset of D-dimensional space. Then there exists a two-layer neural network with a finite number of hidden units that approximate F arbitartily well. Namely, for all x in the domain of F, .**

Graphical user interface, text, application

Description automatically generated

**Machine Learning : Neural Network (Part 2)**

**Backpropagation**

* **We know how to compute output**
* **How do we learn weights?**
* **Backpropagation = gradient descent + chain rule**
* **Neural Network :**
  + - **V : weights from hidden node to output node.**
    - **W : Weights from input node to hidden node.**
    - **x : Input features.**

Diagram

Description automatically generated

**Optimization Function**

* **Squared Error**

Text, whiteboard

Description automatically generated

* **n : Number of training points**
* **: The expected value for the training data point (ground truth value)**
* **: What the neural network outputs**

**Adjusting weights to output node**

* **Differentiate objective with respect to v**
* **“Inputs” are just hidden node activations**

**Adjusting weights to hidden node**

* **Only compensate for portion of error for that hidden node**
* **Not trying to produce specific output value**
* **Just trying to produce activations that get fed to output layer**
* **Change depends on how output layer uses the weight**
  + **If error is small, make small weight changes**
  + **If weight for a particular hidden node to a output node is small, then that hidden node played a small role and the weight change should be small**

A picture containing text, whiteboard

Description automatically generated

* **The first line indicates the lost function for the first set of weights form input to hidden**
* **The second line is the derivative of the lost function with respect to a set of weights and that needs to be decomposed using the chain rule of calculus**
* **The 4th line is the last term, derivative of the activation function with respect to the weight vector going to the hidden node i.**

Text

Description automatically generated

* **K : Number of hidden nodes**
* **MaxIter : Number of epochs**
* **: Learning Rate**
* **D : Set of training data points**

**Forward Propagation**

Diagram

Description automatically generated

* **Error is 0 – 0.39 = - 0.39**

**Back Propagation**

Diagram

Description automatically generated

Diagram

Description automatically generated

Practical Issues

* Initializing weights
  + Can’t initialize to zero because you can get stuck in a local minimum
    - If ;

**Learning Rate**

Chart, histogram

Description automatically generated

**Hyperparameters**

* # of layers
* # of hidden units
* Learning rate
* Activation function
* Weight initialization
* Stopping Criteria (Fixed epochs? Convergence?)
* Difficult to fine tune to get what you want

Chart, bubble chart

Description automatically generated

* **Popular for leaning Sequences**
* **Sequence Models**
  + **Speech Recognition**
  + **Music Generation**
  + **DNA Sequence Analysis**
  + **Machine Translation**
  + **Video Activity Recognition**
  + **Name Entity Recognition**

Diagram

Description automatically generated

Scatter chart

Description automatically generated with medium confidence