**Neural Nets**

* Neural net intro
  + Data and output -> Program
  + Complicated function made of simple parts
    - Universal approximator property
  + Recent successes due to better architectures for and increased ability to retain neural nets
  + Not a lot of good theory behind them
  + A neuron
    - Inputs -> Put weight
    - Bias -> Put weight
    - Apply nonlinear function (activation function)
  + Train a network by minimizing a loss function
    - Loss functions measure how well the classifier is doing
      * 0-1 Loss
      * Word Edit Distance
      * Mean-squared error
      * Cross-Entropy Loss -> used in neural networks
    - Loss functions may not capture what you want!
  + Math
    - Relu
      * f(x) = w \* x + b
      * f(x) = Relu(w\*x+b)
      * Relu(x) {x if x > 0, 0 otherwise}
    - Sigmoid
    - Relu is used more
    - One layer
      * f(x→) = Relu(Wx→ + b→)
    - Mean-squared error
      * 1/N sigma i = N to N [(yi→-yi)^2}
  + Modern Deep Learning
    - Programming by gradient descent
    - Gradient – derivative for equations larger than one layer
    - Move parameters in a direction that reduces loss function
    - Function needs to be differentiable
  + Backpropagation - Efficiently computing gradients
    - Assigns weights to the neurons
    - Chain Rule
  + Overfitting
    - Putting too many inputs and layers
    - Model may be fitting to noise instead of true trend
    - You’re going to memorize the original data
    - Ways to address:
      * Use unseen data to measure how good your classifier is performing
      * Regularization
      * Visualizing Decision Boundary
* Programming
  + Machine Learning APIs
    - Google Cloud Machine Learning
    - Azure Machine Learning
  + High level interfaces (handle almost everything)
    - Sklearn
    - Keras
  + Lower-level interfaces
    - Tensorflow
    - PyTorch
  + We’re going to use Sklearn
* Code Demo
  + <https://github.com/chanlaw/nn-presentation>