Neural Networks

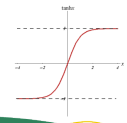
CS 510
Lecture #21
March 27th, 2017

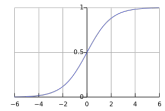
Colorado State University

Review: Non-linear Perceptrons

- $a(x) = \sigma(wx + b) = \sigma(b + \sum_i w_i x_i)$
  - b and w_i are weights to be learned
  - σ is a non-linear transfer (squashing) function
    - Example (logistic): $\sigma(x) = (1 + e^{-x})^{-1}$
    - Example (tanh): $\sigma(x) = tanh(x)$

tanh



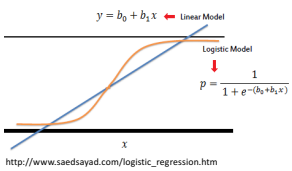


logistic

Colorado State University

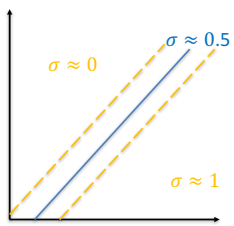
What a Perceptron Learns

1D



http://www.saedsayad.com/logistic_regression.htm

2D

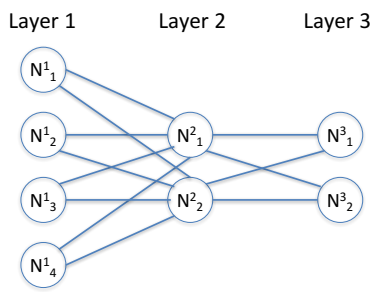


Colorado State University

Review: Combining Perceptrons into a Simple Neural Network

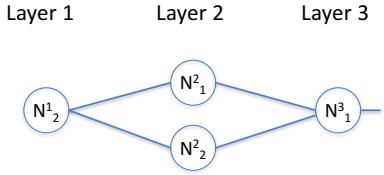
Note that Layer 1 is just the input vector

Layers 2 through N are non-linear perceptrons



Colorado State University

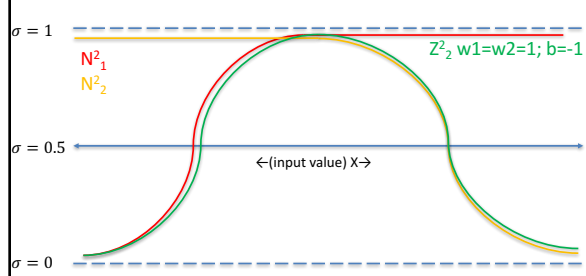
What Can a Neural Network Learn?



- Extremely simple example:
  - 1D input (function of single value)
  - 2 hidden units (perceptrons)
  - 1 outpus unit (perceptron)

Colorado State University

What Can It Learn? (II)



Colorado State University

What Can it Learn (III)?

- With 1D input:
  - Note that z = 0.5 *twice*
  - With one unit, one 0.5 crossing
  - With two hidden units, two are possible
  - More units → more crossings

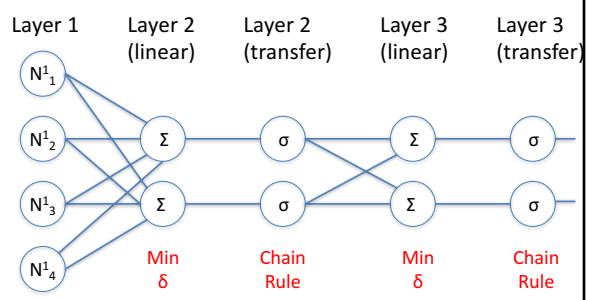
What Can It Learn (IV)?

- In 2D Feature Space
  - Given N hidden units
  - A curve that combines pieces of N line segments
  - (But not a spline – σ governs the combination)
- It can learn curved regions of feature space
  - Including multiple, closed regions

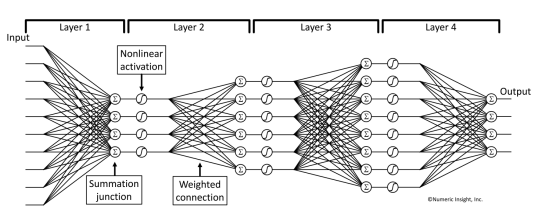
Review: Partial derivatives as error measures

- Imagine you change the output z^l_j, by Δz^l_j
- Then $\Delta C = \frac{\partial C}{\partial z^l_j} \Delta z^l_j$
- If $\left|\frac{\partial C}{\partial z^l_j}\right|$ is large, then C becomes smaller by giving Δz^l_j the opposite sign from the derivative
- But if $\left|\frac{\partial C}{\partial z^l_j}\right|$ is near zero, then Δz^l_j doesn't matter.
  - $\frac{\partial C}{\partial z^l_j}$ is already optimal!
  - $\delta^l_j \equiv \frac{\partial C}{\partial z^l_j}$

Redrawing the Network



Or, at a bigger scale…



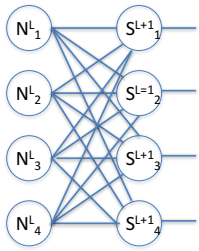
http://numericinsight.com/uploads/A_Gentle_Introduction_to_Backpropagation.pdf

Training a Neural Network

- Step 1: Input & Output Encodings
  - Input Encodings
    - Feature spaces are choices
    - Pre-processing can be very helpful
    - Something of a "black art"
  - Output Encodings
    - For true/false, 1 output value
      - Which you will threshold based on the transfer function
    - For multi-class labels, "one hot" encodings
      - One output per class
      - Select the class with the maximum value

SoftMax

- An optional output layer
- When using one output bit per class
- Favors the more dominant classes
- $\sigma(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_j}}$



Colorado State University

Training a Neural Network (II)

- Step 2: Pick the Architecture
  - For the moment, we will use three layers
    - Input layer
    - One "hidden" layer
    - One output layer
  - Choose the number of hidden units
  - Choose the transfer function
  - Choose the cost function

Colorado State University

Training a Neural Network (III)

- Step 3: train
  - Split data into "train" and "validation"
    - Majority of samples in train
  - Pick a learning rate α
    - May get smaller with the epoch count or training error
    - This is an annealing schedule
  - Perform an "epoch"
    - Randomize the order of your training samples
    - Present every training sample, updating weights
    - Evaluate network on your validation data
      - Producing an accuracy measure
  - Perform more epochs
    - Until convergence or target validation accuracy threshold
  - Return network with highest validation accuracy

Colorado State University

Prepare for PA4

- Download TensorFlow onto a machine you use
  - If not already installed
- Start going through the tutorials
  - Until you get to the MNIST examples
- Do this as a team, not individuals

Colorado State University