

- Lab 3 due Weds, 11:59.
- Quick app of Image Proc./Rec.:
 - <http://www.seamcarving.com/>
 - Term project ideas due next Monday. More details Thursday
- This week:
 - Today: Neural networks
 - Tuesday: Support Vector Machine (SVM) Introduction and derivation
 - Thursday: SVM demo
 - Friday: SVM lab

Neural networks

- “Biologically inspired” model of computation
- Can model arbitrary real-valued functions for classification and association between patterns
- *Discriminative model*
 - Models decision boundary directly
 - Less memory than nearest neighbor
 - Fast!
- Can be parallelized easily for large problems
- We will take a practical approach to classification

Perceptron model

- Computational model of a single neuron
 - Inputs
 - Outputs
 - Function and threshold
- Will be connected to form a complete network

Modeling logic gates

- We'll do OR together.
 - Inputs: $x_1 = \{0,1\}$, $x_2 = \{0,1\}$
 - We need to pick weights w_i and threshold t such that it outputs 0 or 1 appropriately
- Quiz: You do AND, NOT, and XOR.
- Note that a single perceptron is limited in what it can classify. What is the limitation?

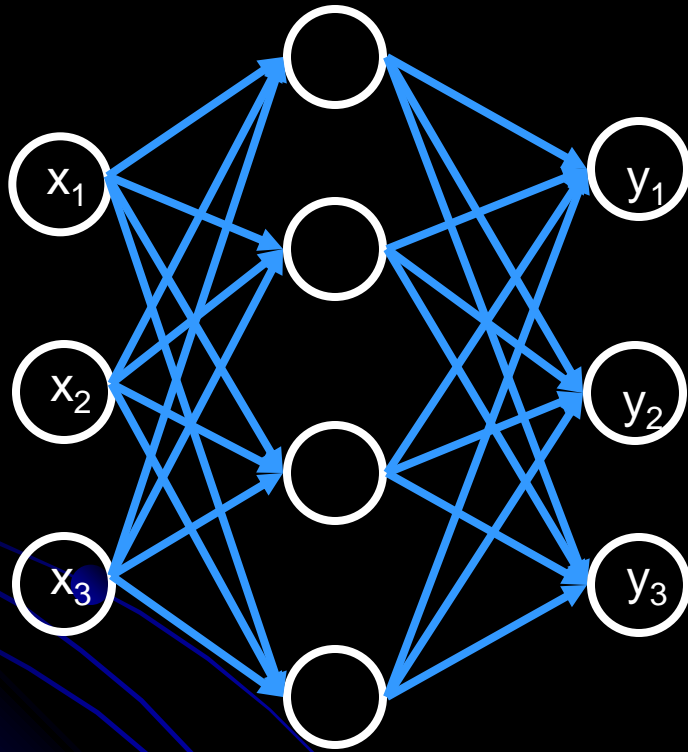
Perceptron training

- Each misclassified sample is used to change the weight “a little bit” so that the classification is better the next time.
- Consider inputs in form $x = [x_1, x_2, \dots, x_n]$
- Target label is $y = \{+1, -1\}$

Algorithm (Hebbian Learning)

- Randomize weights
- Loop until converge
 - If $wx + b > 0$ and y is -1 :
 - $w_i -= \epsilon * x_i$ for all i
 - $b -= \epsilon y$
 - else if $wx + b < 0$ and y is $+1$:
 - $w_i += \epsilon * x_i$ for all i
 - $b += \epsilon y$
 - Else (it's classified correctly, do nothing)
 - ϵ is the learning rate (a parameter that can be tuned).
- Demo perceptronLearning.m

Multilayer feedforward neural nets



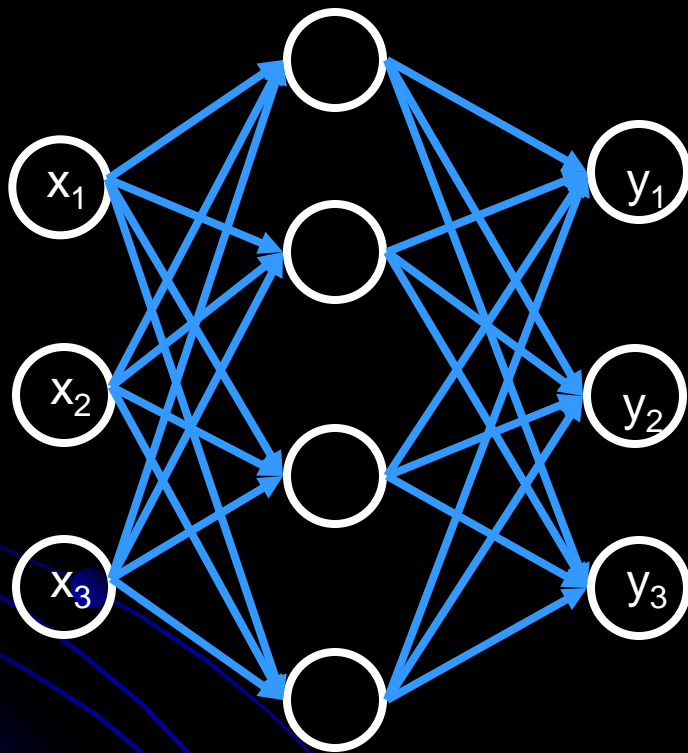
- Many perceptrons
- Organized into layers
 - Input (sensory) layer
 - Hidden layer(s): 2 proven sufficient to model any arbitrary function
 - Output (classification) layer
- Powerful!
- Calculates functions of input, maps to output layers

Sensory
(HSV)

Hidden
(functions)

Classification
• Example
(apple/orange/banana)

Backpropagation algorithm



Initialize all weights randomly

- For each labeled example:
 - Calculate output using current network
 - Update weights across network, from output to input, using Hebbian learning
- Iterate until convergence
 - Epsilon decreases at every iteration
- Matlab does this for you. 😊
- `neuralNetDemo.m`

a. Calculate output (feedforward)



b. Update weights (feedback)



Repeat

Parameters

- Most networks are reasonably robust with respect to learning rate and how weights are initialized
- However, figuring out how to
 - normalize your input
 - determine the architecture of your net
- is a black art. You might need to experiment. One hint:
 - Re-run network with different initial weights and different architectures, and test performance each time on a validation set. Pick best.

References

- This is just the tip of the iceberg! See:
 - Sonka, pp. 404-407
 - Laurene Fausett. *Fundamentals of Neural Networks*. Prentice Hall, 1994.
 - Approachable for beginner.
 - C.M. Bishop. *Neural Networks for Pattern Classification*. Oxford University Press, 1995.
 - Technical reference focused on the art of constructing networks (learning rate, # of hidden layers, etc.)
 - Matlab neural net help

- See slides on term projects
- Come up with idea by the end of the week.
- Lightning talks on Tuesday.

