CSE 473

Chapter 20

Machine Learning Algorithms: Nearest Neighbors & Neural Networks

Machine Learning Recap

- We've looked at:
  - Classification using Decision Trees
  - Combining classifiers using:
    - Majority voting (Bagging)
    - Weighted majority voting (Boosting)
- Last Time: How do we know the classifier function we have learned is good?
  - Look at generalization error on test data
    - Method 1: Split data in training vs test set (the "hold out" method)
    - Method 2: Cross-Validation

Cross-validation

- K-fold cross-validation:
  - Divide data into k subsets of equal size
  - Train learning algorithm K times, leaving out one of the subsets. Compute error on left-out subset
  - Report average error over all subsets
- Leave-1-out cross-validation: K = number of data points.
  - Train on all but 1 data point, test on that data point; repeat for each point
  - Report average error over all points

k-Nearest Neighbors

- Another simple classification algorithm
- Idea:
  - Look around you to see how your neighbors classify data
  - Classify a new data-point according to a *majority vote* of your k nearest neighbors

Distance Metric

- How do we measure what it means to be a neighbor (what is "close")?
- Appropriate distance metric depends on the problem
- Examples:

  x discrete (e.g., strings): Hamming distance

  $d(x_1,x_2)$ = number of features on which x_1 and x_2 differ

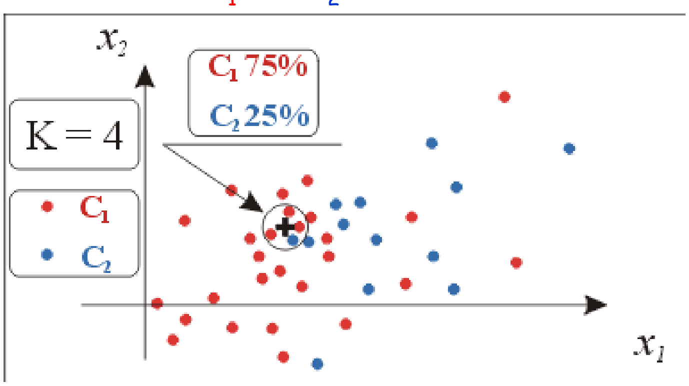
  x continuous (e.g., vectors over reals): Euclidean distance

  $d(x_1,x_2) = || x_1 - x_2 ||$ = square root of sum of squared differences between corresponding elements of data vectors

Example

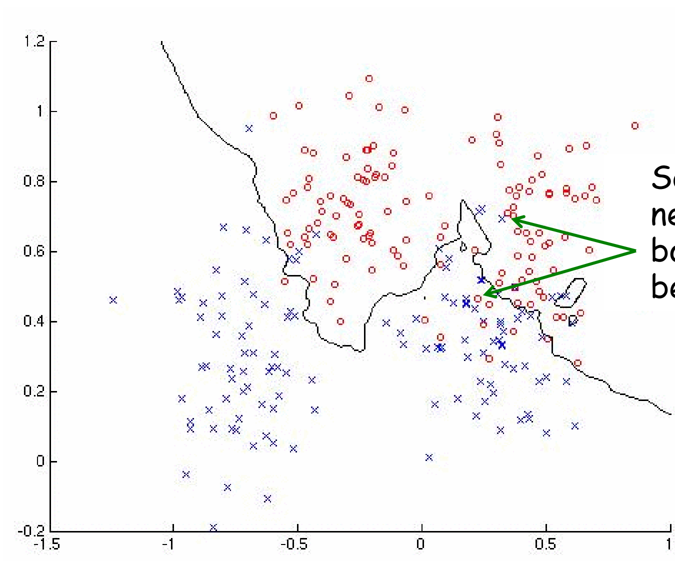
Input Data: 2-D points (x_1,x_2)

Two classes: C_1 and C_2. New Data Point +



K = 4: Look at 4 nearest neighbors of +
3 are in C_1, so classify + as C_1

Decision Boundary using k-NN



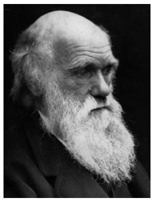
Some points near the boundary may be misclassified

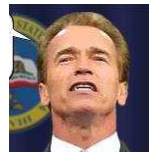
2-D is for "girlie men" – give me high dimensional data

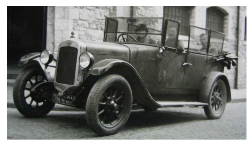
http://www.ipjnet.com/schwarzenegger2/pages/arnold_01.htm

Object Classification in Images

Training Data Set (2 classes):



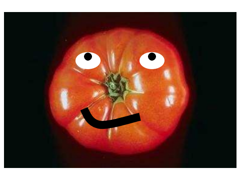



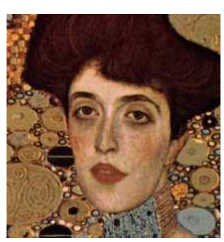



Test Set of Images

Do these belong to one of the classes in the previous slide?





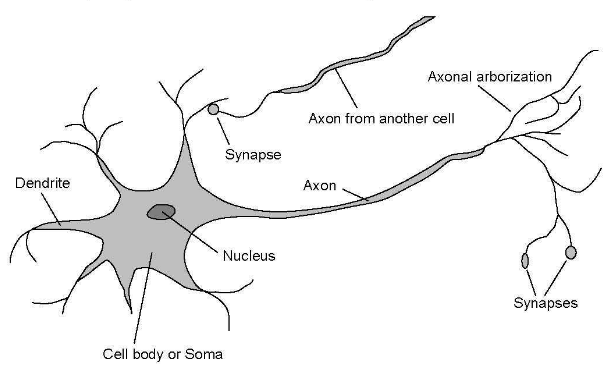



The human brain is extremely good at classifying objects in images

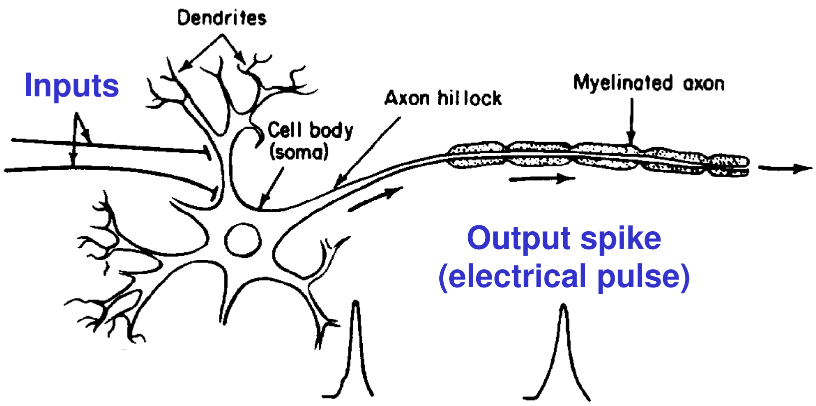
Can we develop classification methods by emulating the brain?

Brains

10^{11} neurons of > 20 types, 10^{14} synapses, 1ms–10ms cycle time
Signals are noisy "spike trains" of electrical potential



Neurons communicate via spikes



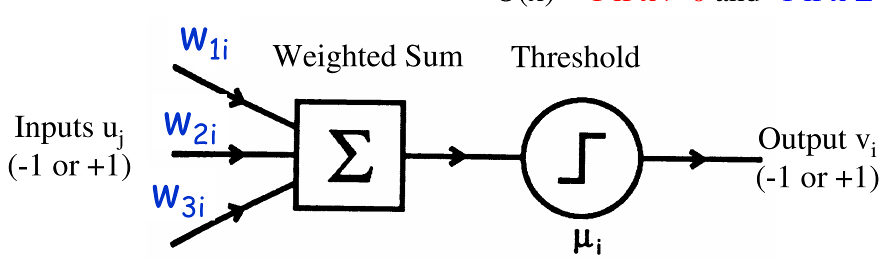
Output spike roughly dependent on whether sum of all inputs reaches a threshold

Neurons as "Threshold Units"

- Artificial neuron:
  - m binary inputs (-1 or 1) and 1 output (-1 or 1)
  - Synaptic weights w_{ji}
  - Threshold μ_i

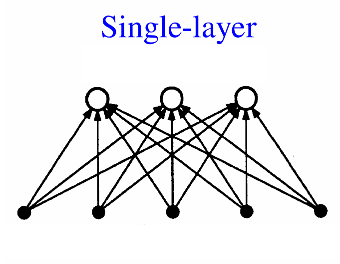
$$v_i = \Theta\left(\sum_j w_{ji} u_j - \mu_i\right)$$

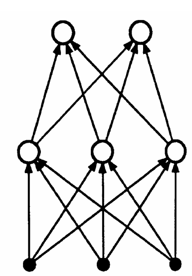
$\Theta(x)$ = 1 if $x > 0$ and -1 if $x \leq 0$



"Perceptrons" for Classification

- Fancy name for a type of layered "feed-forward" networks (no loops)
- Uses artificial neurons ("units") with binary inputs and outputs





Perceptrons and Classification

- Consider a single-layer perceptron

  Weighted sum forms a *linear hyperplane*

$$\sum_j w_{ji} u_j - \mu_i = 0$$

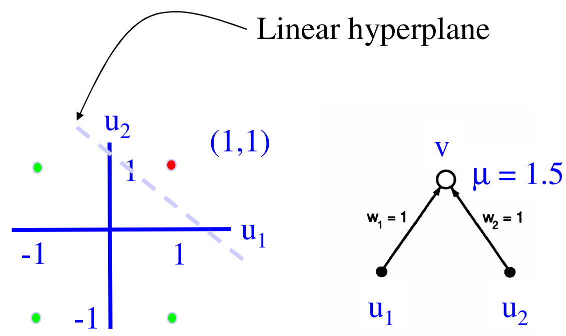
  Everything *on one side* of this hyperplane is in class 1 (output = +1) and everything *on other side* is class 2 (output = -1)

- Any function that is linearly separable can be computed by a perceptron

Linear Separability

- Example: AND is linearly separable

u_1	u_2	AND
-1	-1	-1
1	-1	-1
-1	1	-1
1	1	1

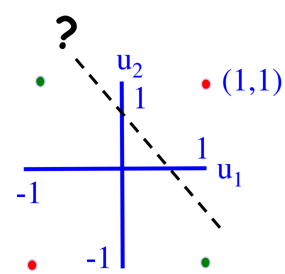


$v = 1$ iff $u_1 + u_2 - 1.5 > 0$

Similarly for OR and NOT

What about the XOR function?

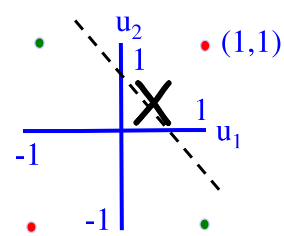
u_1	u_2	XOR
-1	-1	1
1	-1	-1
-1	1	-1
1	1	1



Can a straight line separate the +1 outputs from the -1 outputs?

Linear Inseparability

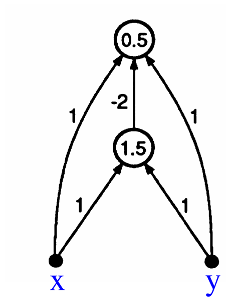
- Single-layer perceptron with threshold units fails if classification task is not linearly separable
  Example: XOR
  No single line can separate the "yes" (+1) outputs from the "no" (-1) outputs!
- Minsky and Papert's book showing such negative results put a damper on neural networks research for over a decade!



How do we deal with linear inseparability?

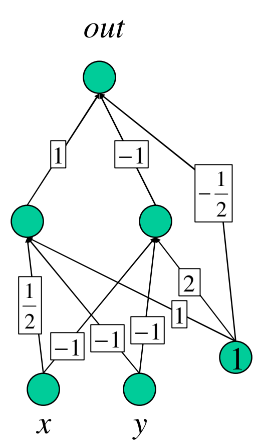
Multilayer Perceptrons

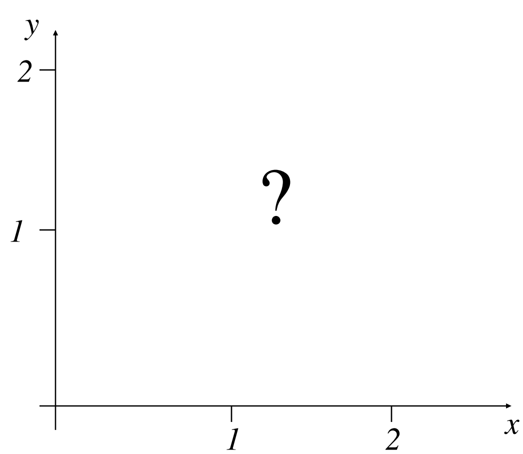
- Removes limitations of single-layer networks
  Can solve XOR
- Example: Two-layer perceptron that computes XOR



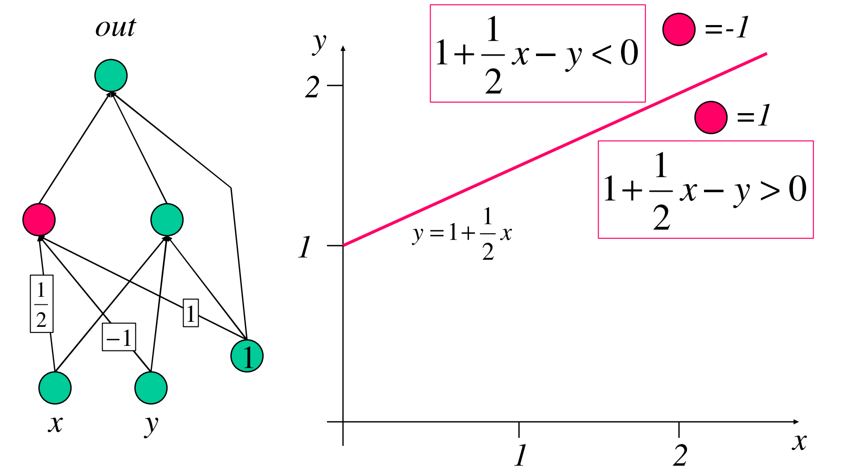
- Output is +1 if and only if $x + y - 2\Theta(x + y - 1.5) - 0.5 > 0$

Multilayer Perceptron: What does it do?

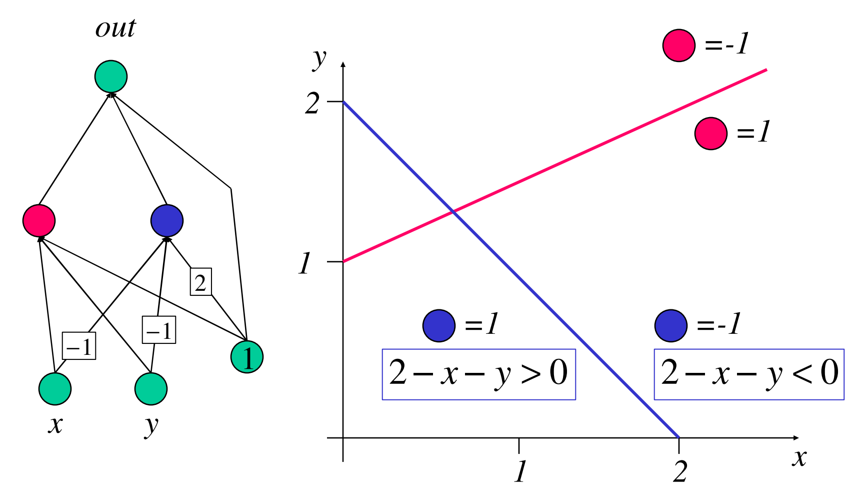




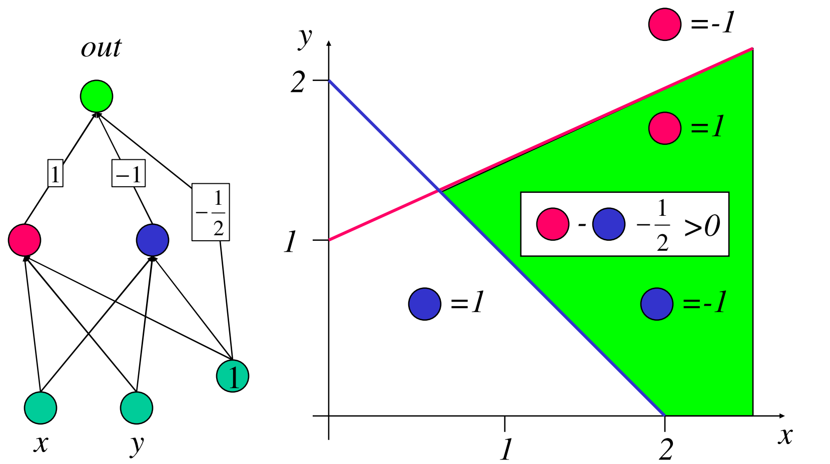
Example: Perceptrons as Constraint Satisfaction Networks



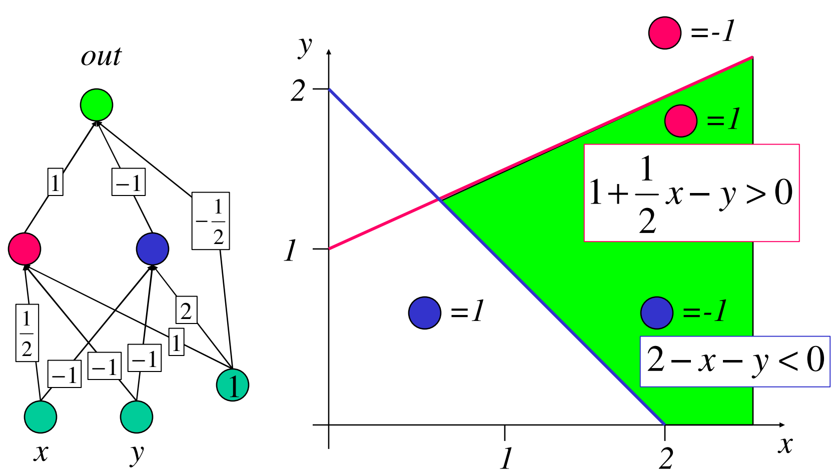
Example: Perceptrons as Constraint Satisfaction Networks



Example: Perceptrons as Constraint Satisfaction Networks



Perceptrons as Constraint Satisfaction Networks



Next Time

- Function Approximation using Neural Networks
  - Gradient Descent