EGN4060c Intro to Robotics

Lecture 9:

Machine Learning:
Supervised Classifiers

Instructor: Dr. Gita Sukthankar
Email: gitars@eecs.ucf.edu

Upcoming Due Dates

- Much happening in the next few weeks!
- Extended date for lab 2 report due on Wed; no further extensions
- Homework 2 (architectures) due on Wed
- Lab 3 report due next Wed Oct 1st
- Midterm exam: Wed Oct 8th (exam review on the 6th)

Wavefront Summary

- Maintain 2 data structures (also can track current cost)
  - Old cost map
  - New cost map
- Iterate over the grid until the start square has a non-zero value
- For each cell:
  - Find lowest cost neighboring, unoccupied square and add 1 to the cost
  - If the current cost is 0 or if the new cost is lower than the current cost, annotate the cell (new cost map) with the new cost.
- Your path is defined by any uninterrupted sequence of decreasing numbers reaching the goal

Wavefront (Complete)

- You’re done
  - Remember, 0’s should only remain if unreachable regions exist

7	**18**	17	16	15	14	13	12	11	10	9	9	9	9	9	9	9
6	17	17	16	15	14	13	12	11	10	9	8	8	8	8	8	8
5	17	16	16	15	14	13	12	11	10	9	8	7	7	7	7	7
4	17	16	15	15	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	6	6	6	6
3	17	16	15	14	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	5	5	5	5
2	17	16	15	14	13	12	11	10	9	8	7	6	5	4	4	4
1	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	3
0	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	**2**
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Following the Path

- To find the shortest path, according to your metric, simply always move toward a cell with a lower number
  - The numbers generated by the Wavefront planner are roughly proportional to their distance from the goal

Two possible shortest paths shown

Java: Bounds Checking

- Each array object has a public constant called `length` that stores the size of the array
- It is referenced using the array name:

```
scores.length
```

- Note that `length` holds the number of elements, not the largest index

Java: Initializer Lists

- An *initializer list* can be used to instantiate and fill an array in one step
- The values are delimited by braces and separated by commas
- Examples:

```
int[] units = {147, 323, 89, 933, 540,
               269, 97, 114, 298, 476};

char[] letterGrades = {'A', 'B', 'C', 'D', ’F'};
```

Two-Dimensional Arrays

- A two-dimensional array is declared by specifying the size of each dimension separately:

```
int[][] scores = new int[12][50];
```

- A array element is referenced using two index values:

```
value = scores[3][6]
```

- The array stored in one row can be specified using one index

MapGUI: class for map display

```
MapGUI map = new MapGUI();
map.getMap();
map.moveRobot(row, column, angle);
map.moveRobot(row, column, direction);
map.getRobotLocation();
```

Example map file format: 0=empty, 1=wall, 2=goal

```
   6 8
00100010
00000000
00000100
01101100
00011000
10010002
 2 1 120
```

Overview

- You’ve learned how to write scripts for the robot.
- You’ve learned how to have the robot create and execute a plan based on a pre-specified world model.
- But what if you want your robot to be able to learn from experiences?

- The answer----machine learning!

Problem Formulation

- Input: robot sensor reading
- Output: instructions for robot
- Additional information:
  - Example pairs of input and output (supervised learning)
  - Fitness function (evolutionary computing)
  - Reward function (reinforcement learning)
- For the next lab, you'll be doing reinforcement learning.

Classification

- Assign object/event to one of a given finite set of categories.
  - Medical diagnosis
  - Credit card applications or transactions
  - Fraud detection in e-commerce
  - Worm detection in network packets
  - Spam filtering in email
  - Recommended articles in a newspaper
  - Recommended books, movies, music, or jokes
  - Financial investments
  - DNA sequences
  - Spoken words
  - Handwritten letters
  - Astronomical images

Computer vision and machine learning are closely related disciplines.

Planning / Control

- Performing actions in an environment in order to achieve a goal.
  - Solving calculus problems
  - Playing checkers, chess, or backgammon
  - Balancing a pole
  - Driving a car or a jeep
  - Flying a plane, helicopter, or rocket
  - Controlling an elevator
  - Controlling a character in a video game
  - Controlling a mobile robot

Measuring Performance

- Classification accuracy
- Solution correctness
- Solution quality (length, efficiency)
- Speed of performance

The Time is Ripe for ML!

- Many basic effective and efficient algorithms available.
- Large amounts of on-line data available.
- Large amounts of computational resources available.

Supervised Classifiers

Learning

Training Data
(labeled by humans) ➡ Decision Function

Classification

New Data + Learned Decision Function =

Answer

How to learn the decision function?

Regression

- Problem: from set of exemplars and known *x*, predict value for unknown *y*
- Assume a model for the data based on parameters $y=f(x;p)$
  - x is data
  - p are parameters
  - f(x) is the model
- For example:
  - You could model your data with a linear model: $f(x;p)=ax+b$
  - The parameters of this model are: $p=(a,b)$

Regression Basics

- Learning: estimate the model parameters *p* from training data *(x,y)*
- For *f(x;p)=a*

$$a = \frac{1}{N}\sum_i y_i$$

Linear Regression

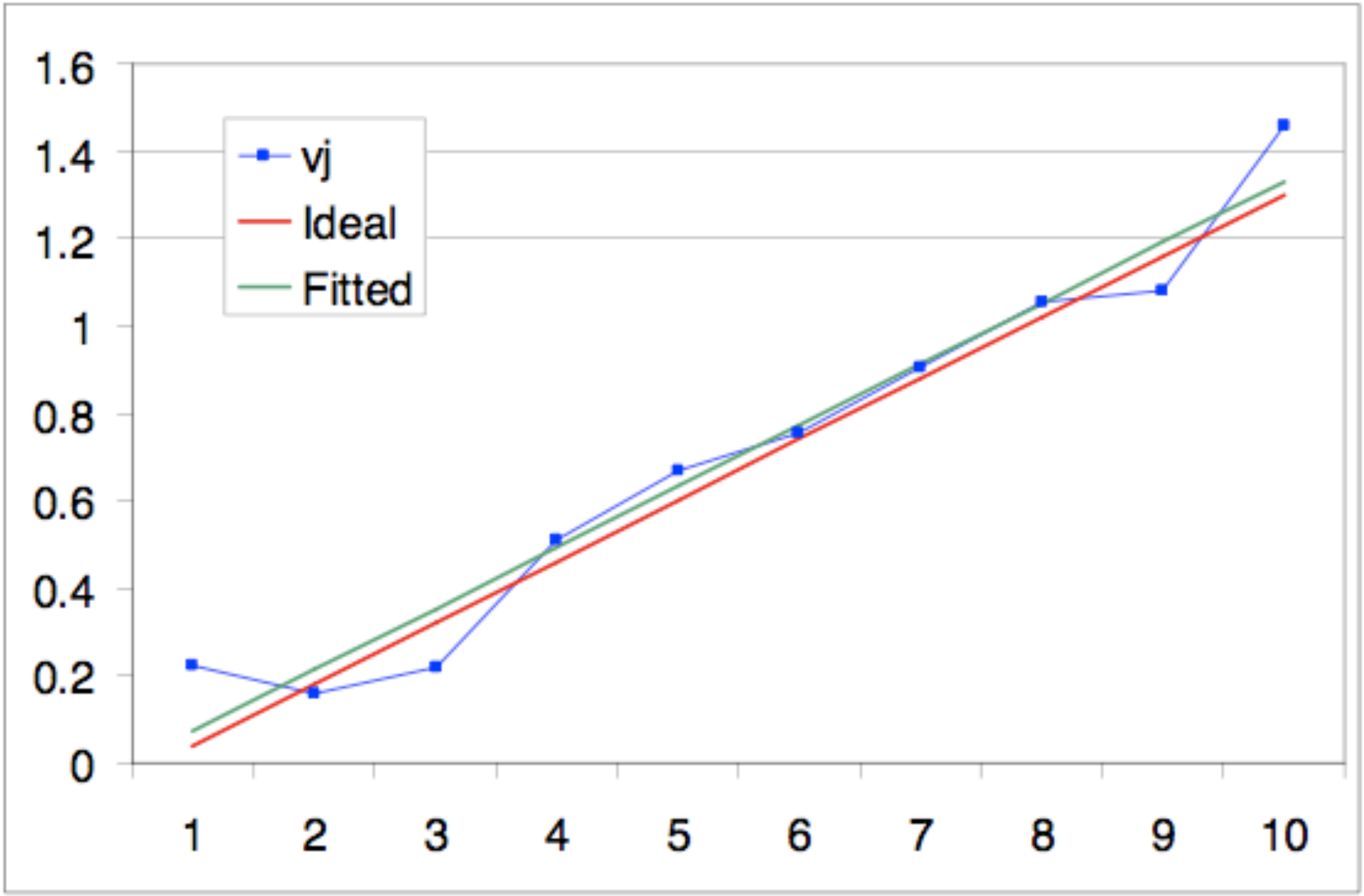
- Linear model: *f(x;p)=ax+b*
- Ordinary least squares method: minimize the residual which can be done in a closed form way

$$S_x = n \cdot \bar{x} = \sum_{j=1}^{n} x_j \qquad S_y = n \cdot \bar{y} = \sum_{j=1}^{n} y_j$$

$$S_{xx} = \sum_{j=1}^{n} x_j^2 \qquad S_{xy} = \sum_{j=1}^{n} x_j y_j$$

$$a = \frac{nS_{xy} - S_x S_y}{nS_{xx} - (S_x)^2} \qquad b = \tfrac{1}{n}\left(S_y - aS_x\right)$$

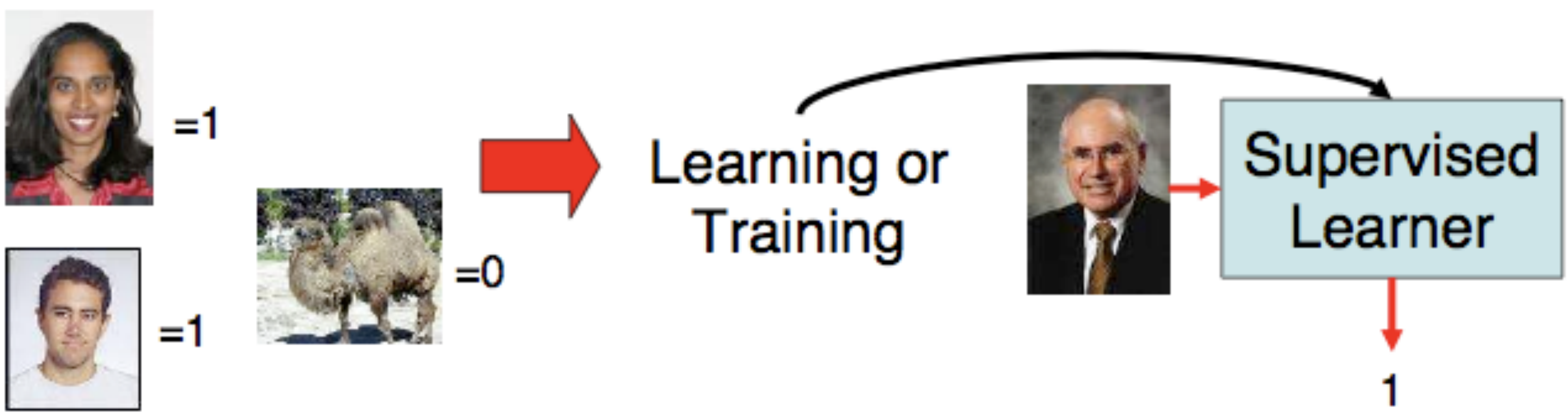
Graph the Result



Once the parameters have been "learned" they can be used to predict the answers for unseen values.

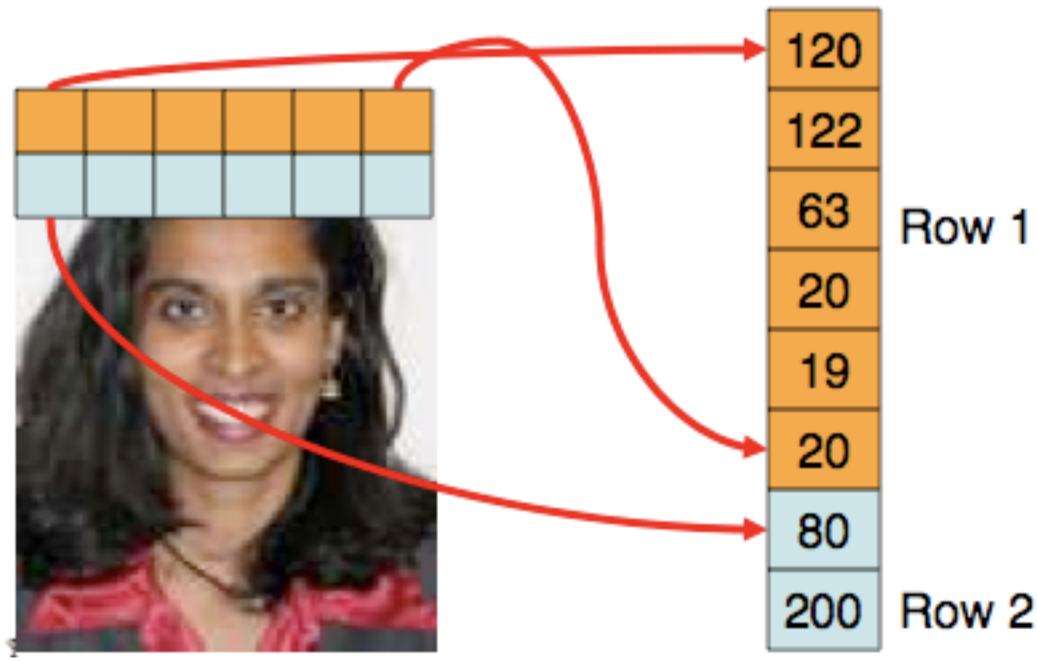
Supervised Learning

- Given training data sequence of inputs *x* and outputs *y* $(x_1,y_1), (x_2, y_2), \ldots (x_N,y_N)$
- Learn to predict the output *y*
- Input *x* can be real, discrete, or multi-dimensional



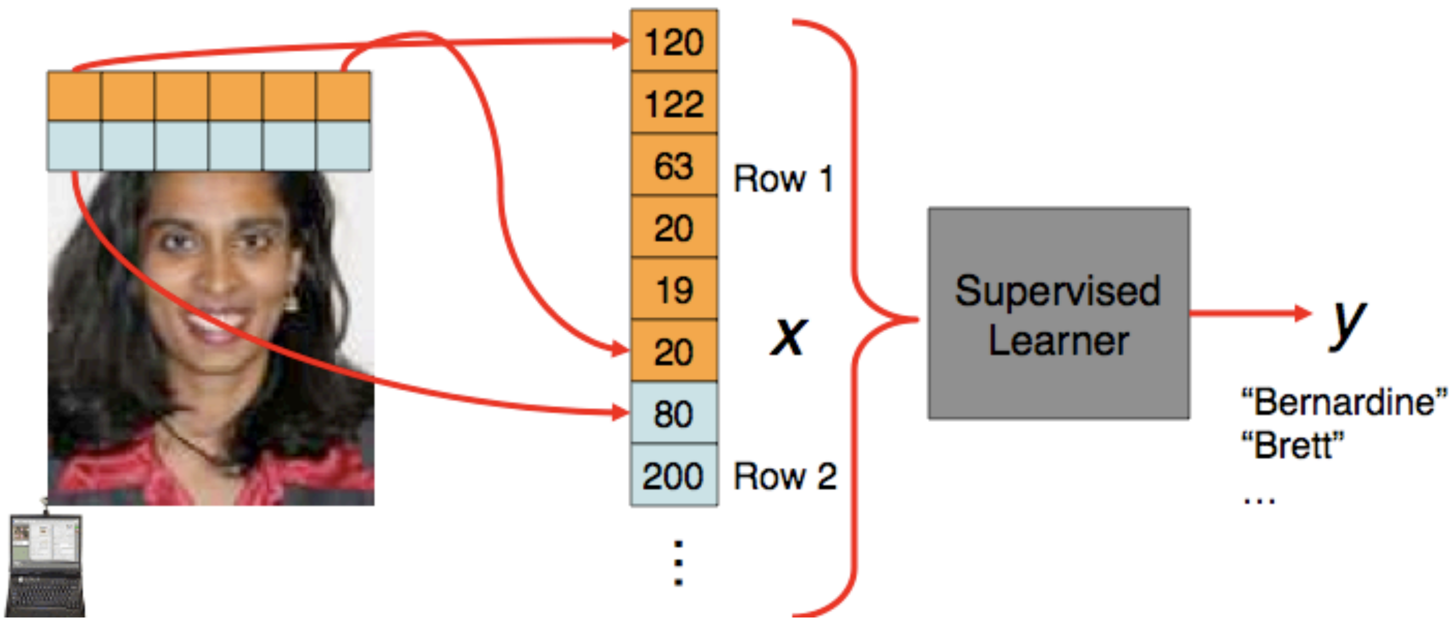
Face Detection

- *X* is a vector of pixel intensities



Supervised Learning

Y is a discrete output class just binary



Supervised Learning

- Labeled examples
- Real input, with desired output

"Bernardine"

"Unknown"

"Brett"

Binary Classification

- Output has 2 classes: true or not true
  - Usually defined as {0,1} or {-1,+1}

"Face" = 1

"Not face" = -1

"Face"= 1

Multi-Class Classification

- Output has many classes
- Usually defined as {1….k}

"Bernardine" = 1

"Camel" = 3

"Brett"= 2

Regression and Classification

- Regression and classification are very similar
- We can often define a binary classification as

$$y = \operatorname{sgn}\left(f\left(x\right)\right)$$

$$y = \begin{cases} 1 & f(x) > 0 \\ -1 & f(x) \le 0 \end{cases}$$

Classification Approaches

- Nearest neighbor
- Naïve Bayes
- Decision trees
- Neural networks
  - Perceptrons
  - Multi-layer perceptron
- All of these different methods answer the same question.
  - Given a known input and training data, what is the unknown class label of a new example?

Definitions

- C is the set of classes, for binary classification
  - C={-1,+1}
- X is the input space
  - Typically a real d-dimensional vector

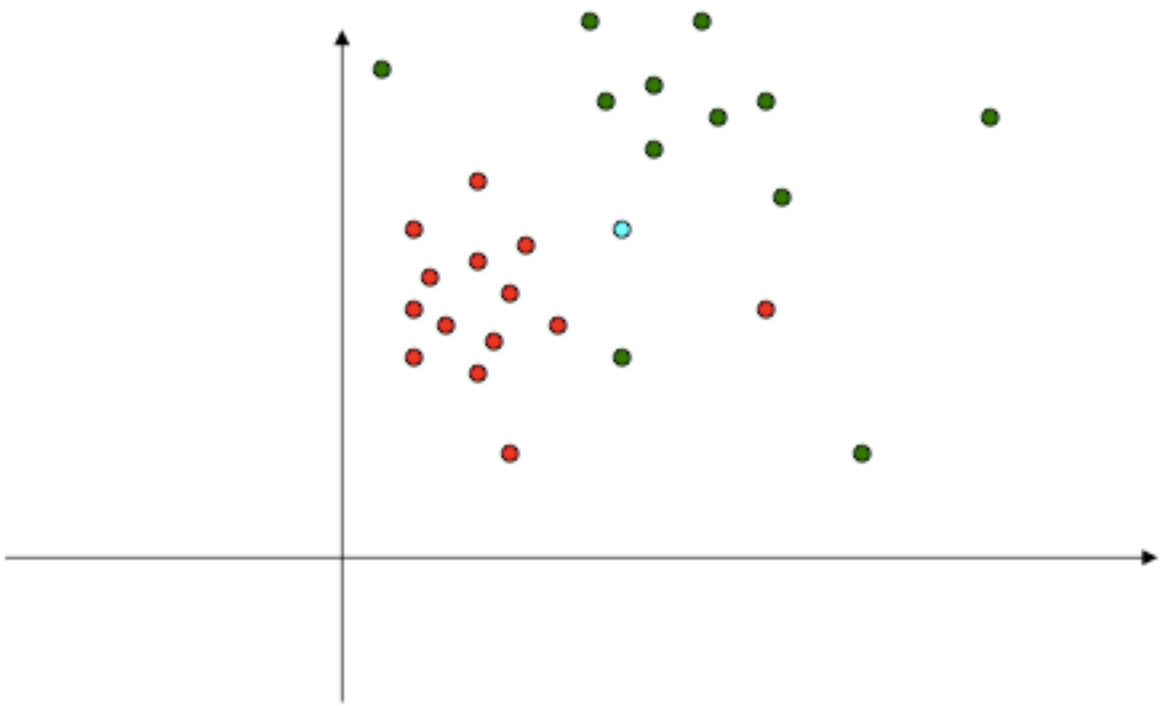
$$x \in \Re^D$$

  - Y is the output so

$$y \in C$$

Problem

- Green dots are one class, red dots are the other
- Given an example (blue) which class does it belong to?



Nearest Neighbor

- Simple idea:
  - Look for closest example x in training data and use its output y as the output
- Mathematically

$$i^* = \arg\min_i \left(D\left(x, x_i\right) \right)$$

$$y = y_{i^*}$$

Similar in concept to k-means clustering (which is an unsupervised learning technique)

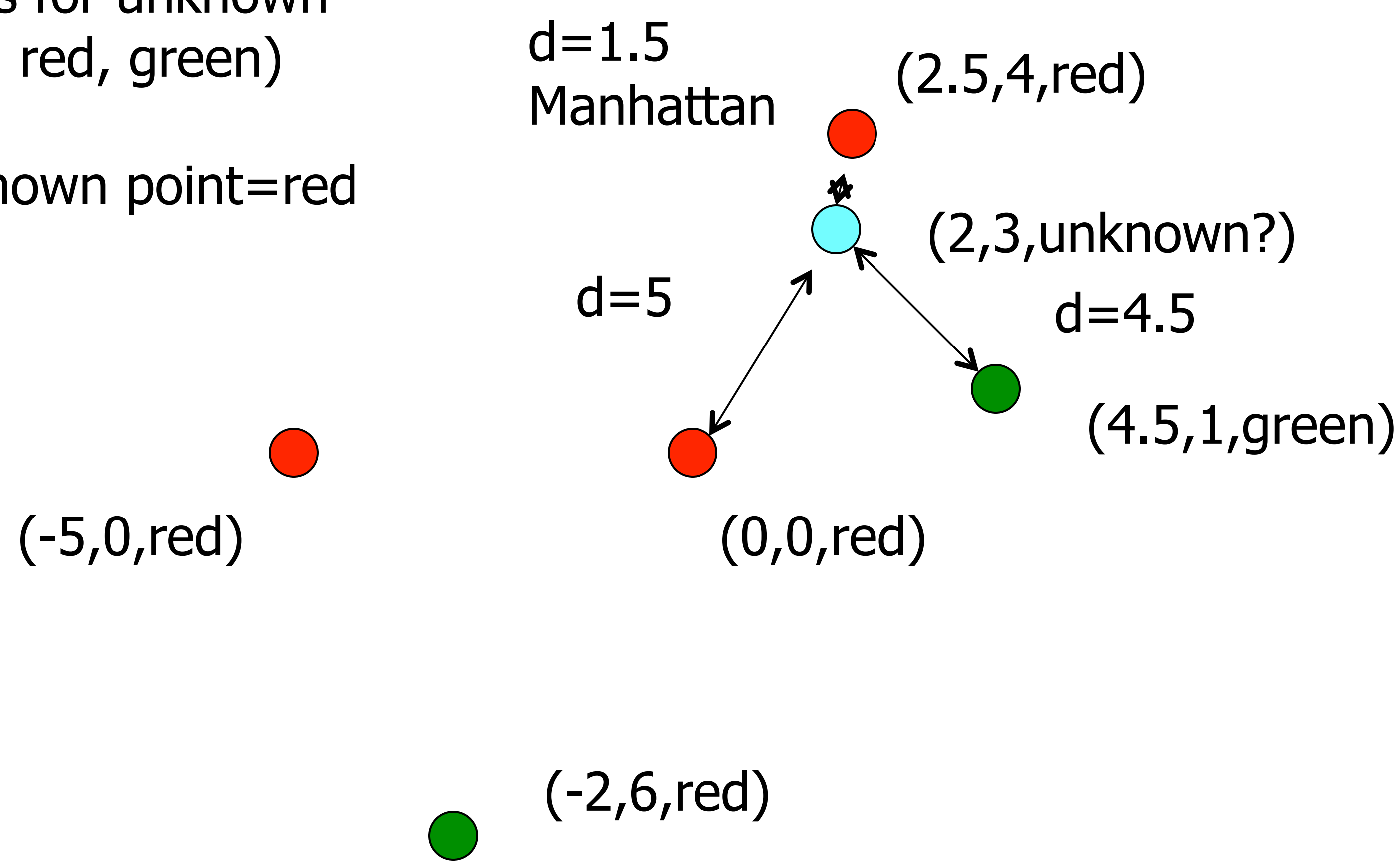
Distance Function

- Euclidean $$D(a,b) = \sqrt{\sum_j (a_j - b_j)^2}$$
- Manhattan $$D(a,b) = \sum_j |a_j - b_j|$$
- Choice in distance function affects answer

3 Nearest Neighbor

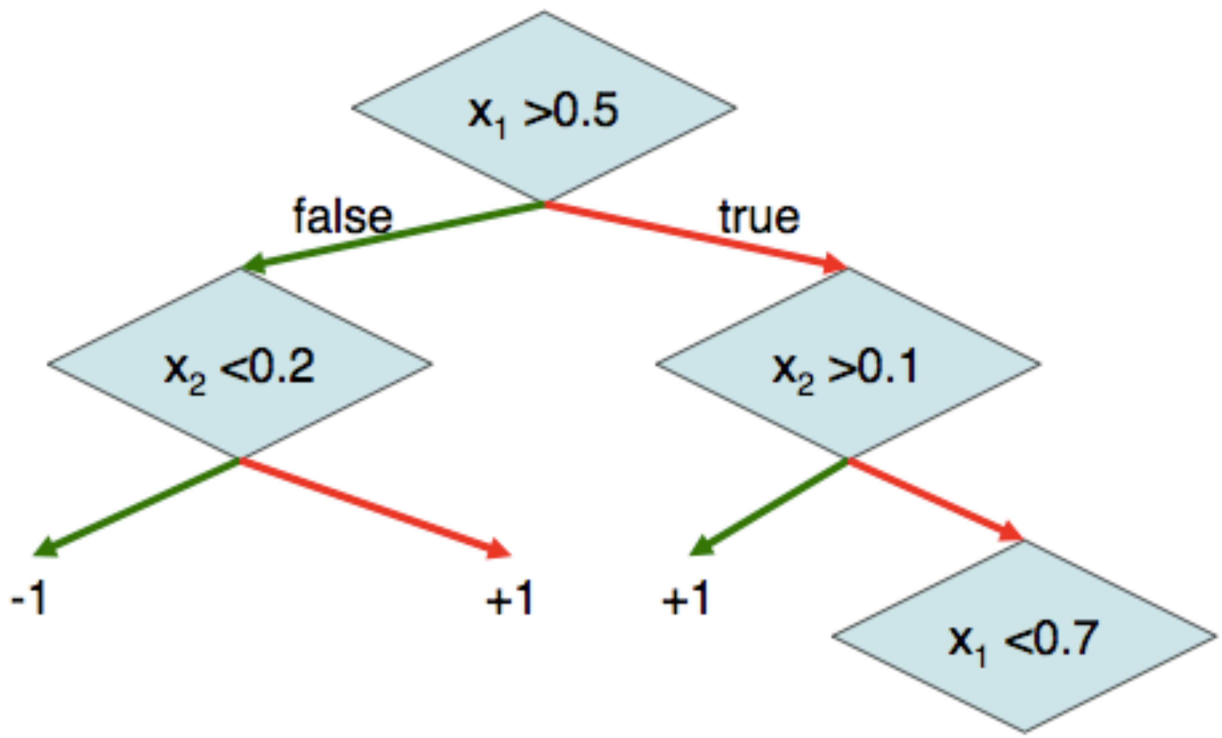
Votes for unknown
(red, red, green)

Unknown point=red



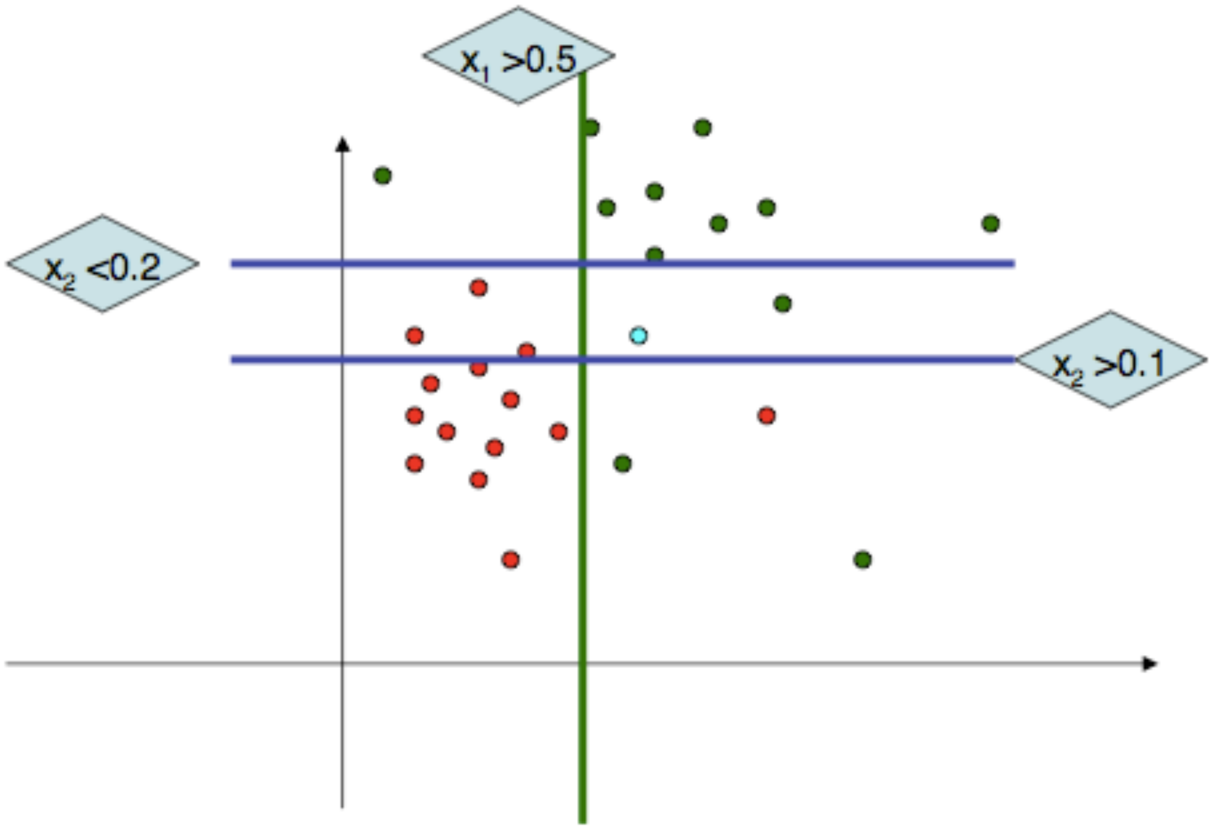
Decision Trees

- Basic concept:
  - Learn a nested set of if/else rules
  - Tree of decisions or splitting points



Decision-Tree

- Axis aligned splitting planes



Decision Trees

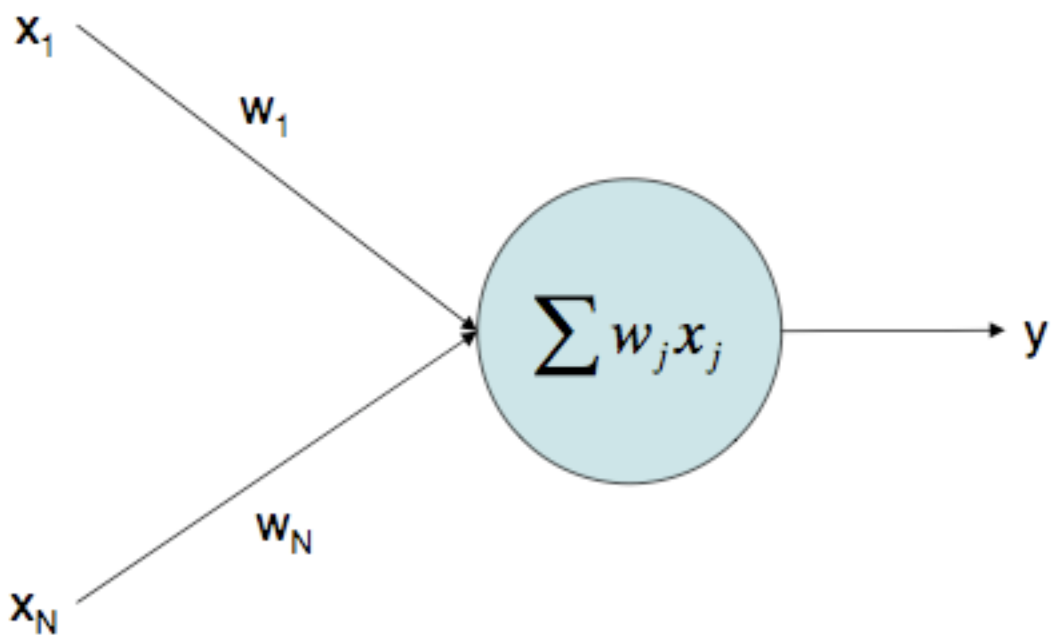
- Best example is C4.5 algorithm
  - Works on discrete and continuous data sets
- Key idea to building trees
  - Decising where to introduce a split and when to start splitting

Neural Networks

- Modeled on biological networks
- Activation related to strength of outputs
- First proposed in 1940s by McCulloch and Pitts
- Shot down by Minsky around 1969
- Returned to popularity in 1980s with the backpropagation algorithm

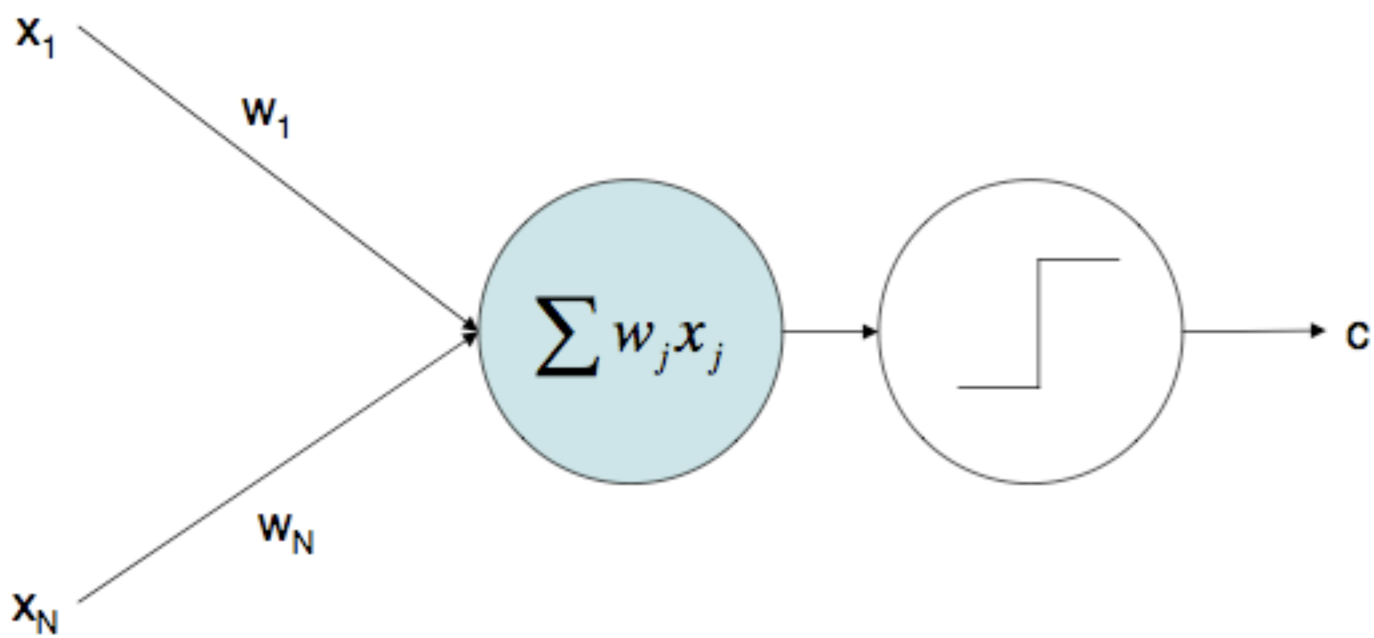
Perceptron Learning

- Single linear "neuron"



Perceptrons for Classification

- Output through step function



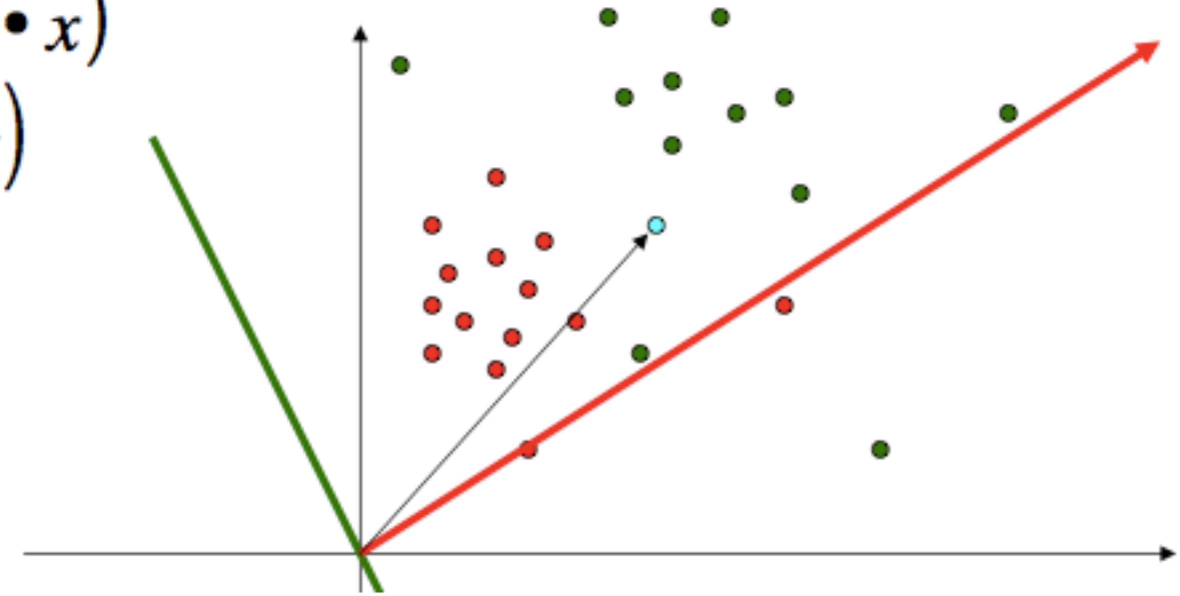
Perceptron Learning

- Operation is a dot product

$$y = \mathrm{sgn}\left(\sum w_j x_j\right)$$

$$y = \mathrm{sgn}(w \bullet x)$$

$$= \mathrm{sgn}(w^T x)$$

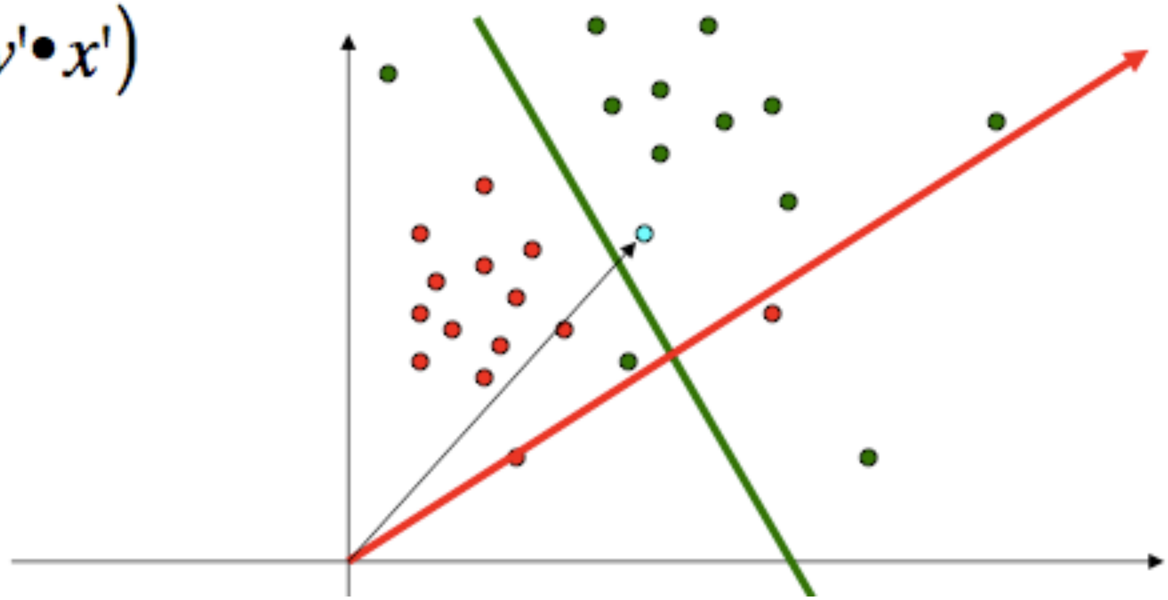


Perceptron Classification

- Adding bias term

$$y = \operatorname{sgn}\left(\sum w_j x_j - b\right)$$

$$y = \operatorname{sgn}\left(w' \bullet x'\right)$$



Perceptron Learning

- Very simple rule:

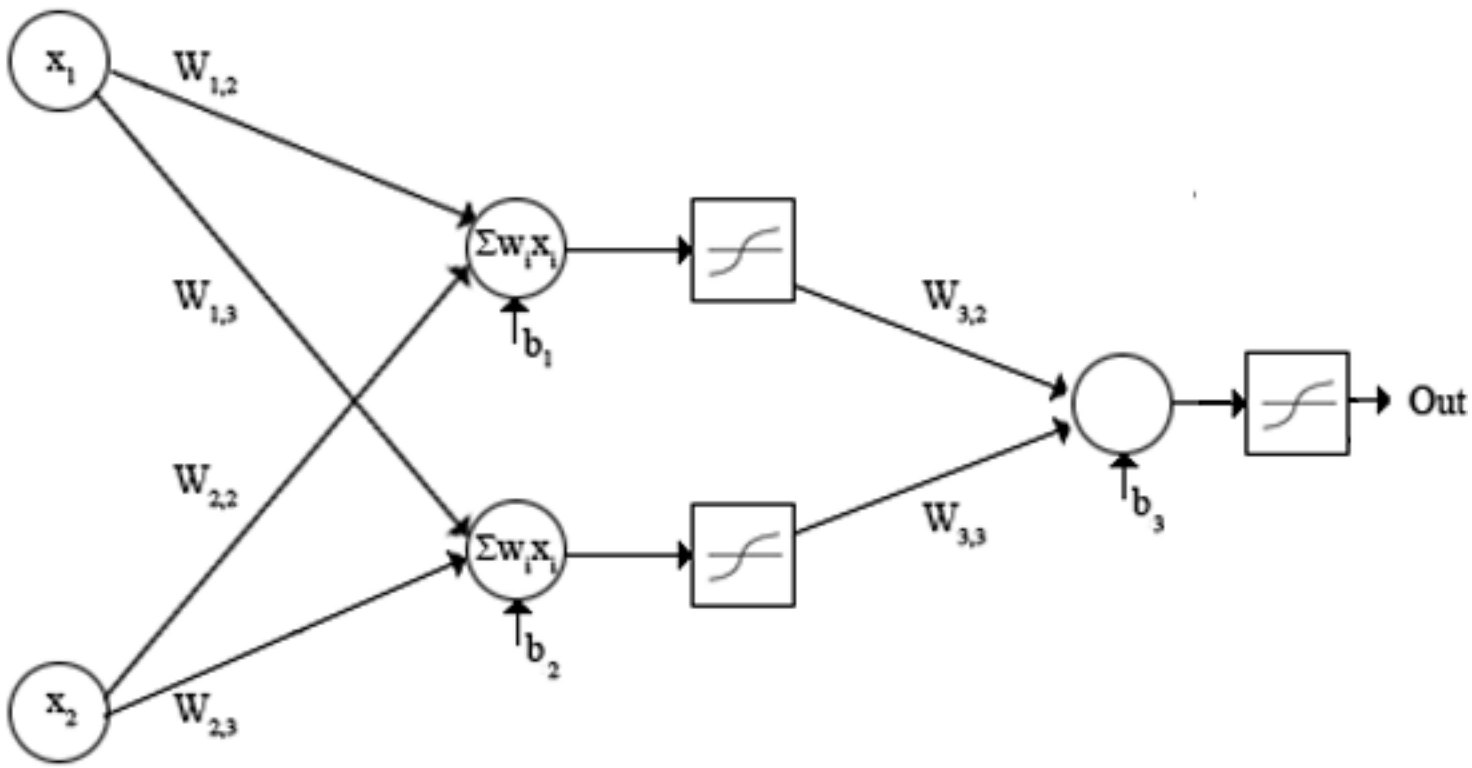
Difference in desired
and actual answer

$$w' = w + \alpha\left(y_i - \operatorname{sgn}\left(w^T x_i\right)\right)x_i$$

Alpha is a
learning rate

- This gives us a method of modifying the weights based on misclassifications.

Multilayer Perceptron



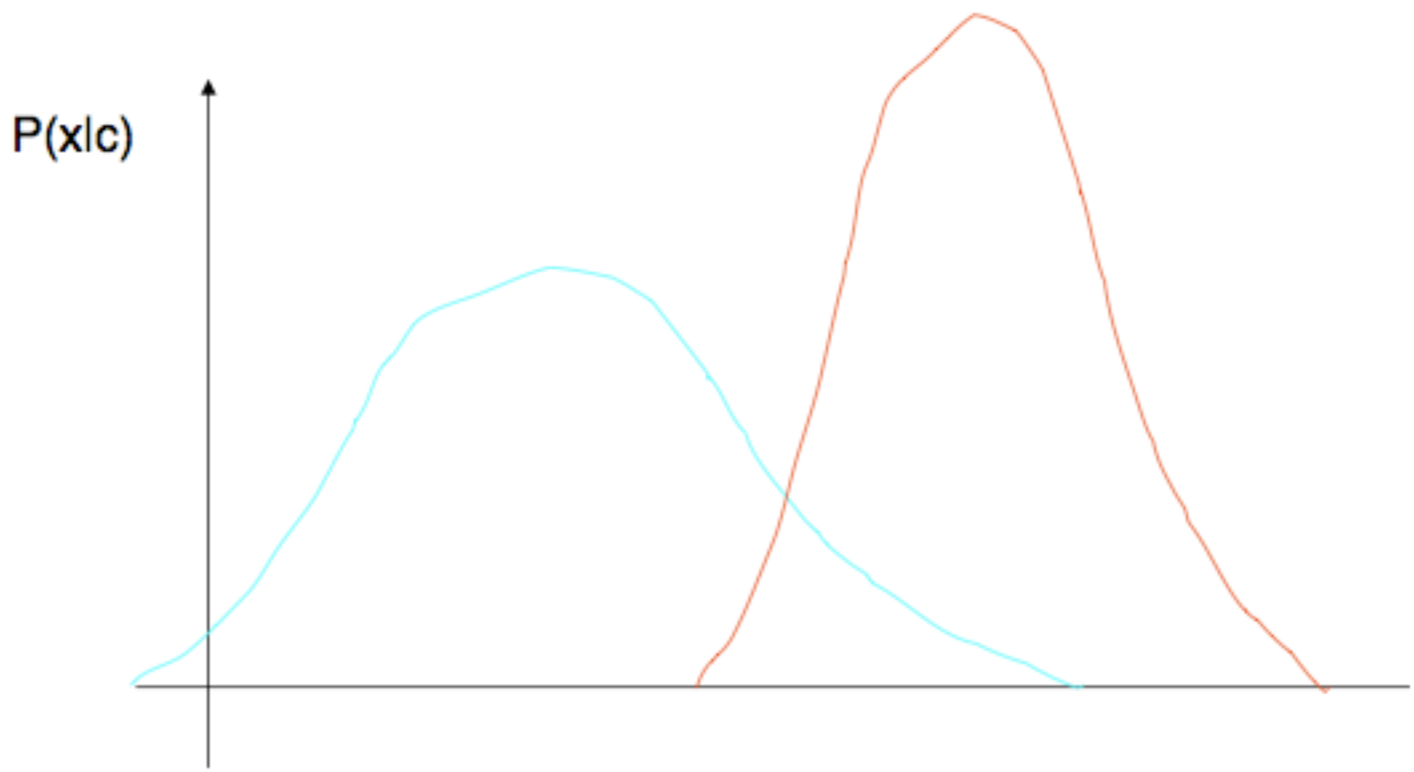
Naïve Bayes

- Another alternative is to look at the problem probabilistically
- Define class probabilities

$$P(x|c=1), P(x|c=-1)$$

Consider a 1-D example with the distributions plotted….

Naïve Bayes



Best choice will be: $$c^* = \arg\max_c P(c|x)$$

Naïve Bayes

Best choice will be:

Prior

$$c^* = \arg\max_c P(x|c)P(c)$$

- How does this work?
- Let's review some probability theory?

Axioms of Probability Theory

- All probabilities between 0 and 1

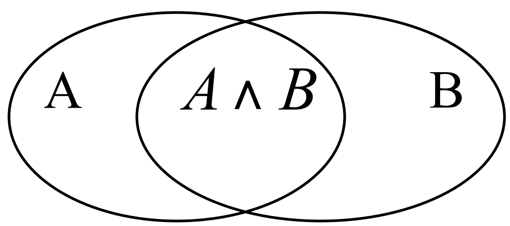
$$0 \le P(A) \le 1$$

- True proposition has probability 1, false has probability 0.

P(true) = 1 P(false) = 0.

- The probability of disjunction is:

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$



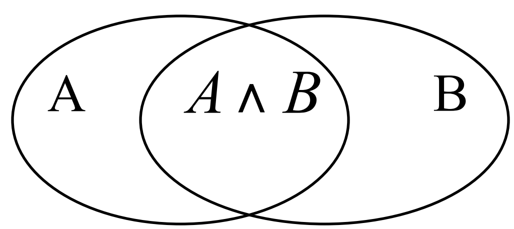
- A and B are independent if the joint probability

P(A and B)=P(A)P(B)

Conditional Probability

- P(*A* | *B*) is the probability of *A* given *B*
- Assumes that *B* is all and only information known.
- Defined by:

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)}$$



Bayes Theorem

$$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}$$

Simple proof from definition of conditional probability:

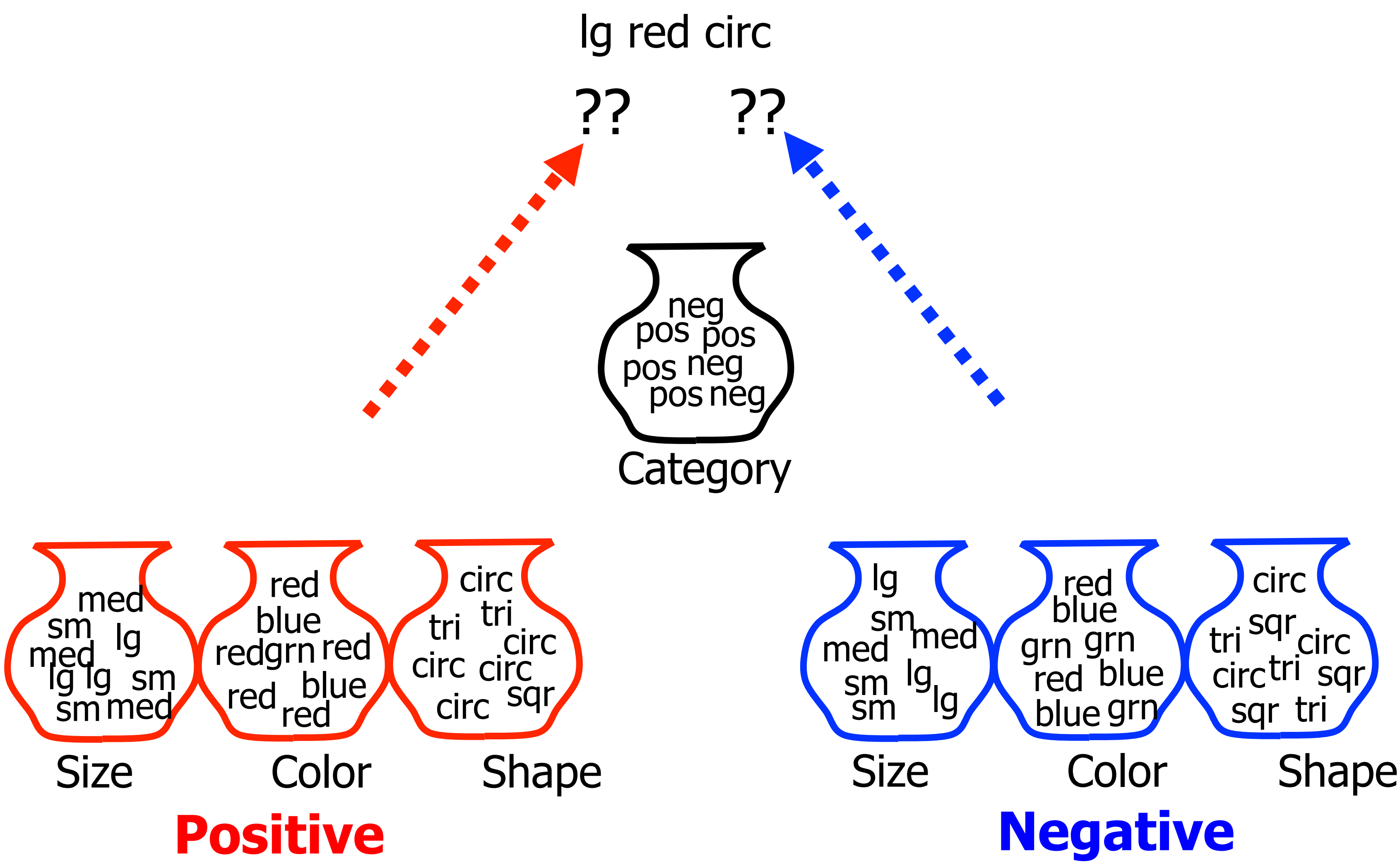
$$P(H \mid E) = \frac{P(H \wedge E)}{P(E)} \quad \text{(Def. cond. prob.)}$$

$$P(E \mid H) = \frac{P(H \wedge E)}{P(H)} \quad \text{(Def. cond. prob.)}$$

$$P(H \wedge E) = P(E \mid H)P(H)$$

QED: $$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}$$

Naïve Bayes Inference Problem



Naïve Bayesian Categorization

- If we assume features of an instance are independent **given the category** (***conditionally independent***).

$$P(X \mid Y) = P(X_1, X_2, \cdots X_n \mid Y) = \prod_{i=1}^{n} P(X_i \mid Y)$$

- Therefore, we then only need to know P(X_i | Y) for each possible pair of a feature-value and a category.
- If Y and all X_i are binary, this requires specifying only 2n parameters:
  - P(X_i=true | Y=true) and P(X_i=true | Y=false) for each X_i
  - P(X_i=false | Y) = 1 – P(X_i=true | Y)
- Compared to specifying 2^n parameters without any independence assumptions.

Naïve Bayes Example

Probability	positive	negative
P(*Y*)	0.5	0.5
P(small | *Y*)	0.4	0.4
P(medium | *Y*)	0.1	0.2
P(large | *Y*)	0.5	0.4
P(red | *Y*)	0.9	0.3
P(blue | *Y*)	0.05	0.3
P(green | *Y*)	0.05	0.4
P(square | *Y*)	0.05	0.4
P(triangle | *Y*)	0.05	0.3
P(circle | *Y*)	0.9	0.3

We learn these probabilities by employing frequency counting techniques on the training data.

Test Instance:
<medium ,red, circle>

Naïve Bayes Example

Probability	positive	negative
$P(Y)$	0.5	0.5
$P(\text{medium} \mid Y)$	0.1	0.2
$P(\text{red} \mid Y)$	0.9	0.3
$P(\text{circle} \mid Y)$	0.9	0.3

Test Instance:
<medium ,red, circle>

Answer:
Drawn from the positive urn

$P(\text{positive} \mid X) = P(\text{positive})*P(\text{medium} \mid \text{positive})*P(\text{red} \mid \text{positive})*P(\text{circle} \mid \text{positive}) / P(X)$

$0.5 * 0.1 * 0.9 * 0.9$

$= 0.0405 / P(X) = 0.0405 / 0.0495 = 0.8181$

$P(\text{negative} \mid X) = P(\text{negative})*P(\text{medium} \mid \text{negative})*P(\text{red} \mid \text{negative})*P(\text{circle} \mid \text{negative}) / P(X)$

$0.5 * 0.2 * 0.3 * 0.3$

$= 0.009 / P(X) = 0.009 / 0.0495 = 0.1818$

$P(\text{positive} \mid X) + P(\text{negative} \mid X) = 0.0405 / P(X) + 0.009 / P(X) = 1$

$P(X) = (0.0405 + 0.009) = 0.0495$

For purposes of making a decision, we can ignore the denominator since it is the same for both Classes.

What can we do with learning?

- Learn parameters that we normally have to specify
- Find good color thresholds for our vision algorithms
- Use reinforcement learning to do maze solving or to identify good kicks for Robocup players
- Use genetic algorithms to find good controller parameters
- Use Bayesian reasoning for localization
- Also these algorithms can be combined in interesting ways.