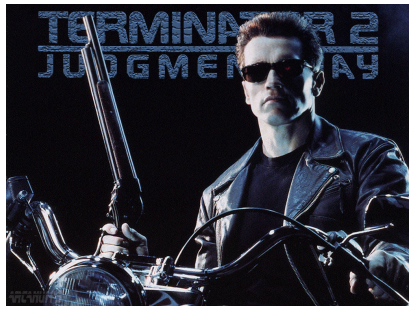
Machine Learning

CMPSCI 383

Learning Machines
and Hollywood



HAL plays chess

Outline

- Motivation: why should agents learn?
- Different models of learning
- Learning from observation
  - classification and regression
- Learning decision trees
- Linear regression

Machine Learning is everywhere!

- Every time you speak on the phone to an automated program (travel, UPS, Fedex, ...)
- Google, Amazon, Facebook all extensively use machine learning to predict user behavior (they hire many of our PhD's)
- Machine learning is one of the most sought after disciplines for prospective employers

IBM Watson Jeopardy Quiz Program

- Watson uses machine learning to select answers to a wide range of questions

Stanley: DARPA Grand Challenge Winner

- Autonomous car that drove several hundred miles through the desert using machine learning

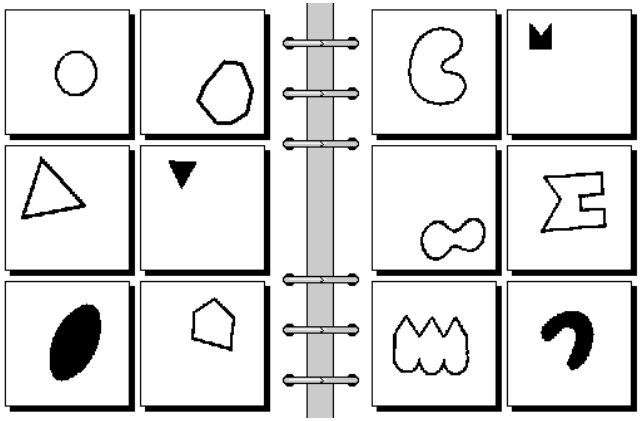
The Future: ML everywhere!

- Hand-held devices will have terabytes of RAM and petabytes of disk space
- Massive use of machine learning across all smartphones, web software, OS, desktops
- Cars will increasingly use machine learning to drive autonomously
- Hard to overestimate the impact of ML

Human Learning

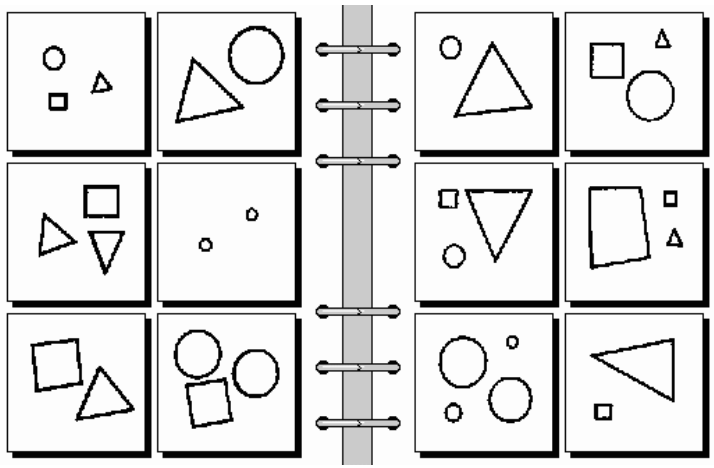
- Learning is a hallmark of intelligence
- Human abilities depend on learning
  - Learning a language (e.g., English, French)
  - Learning to drive
  - Learning to recognize people (faces)
  - Learning in the classroom

Bongard Problem

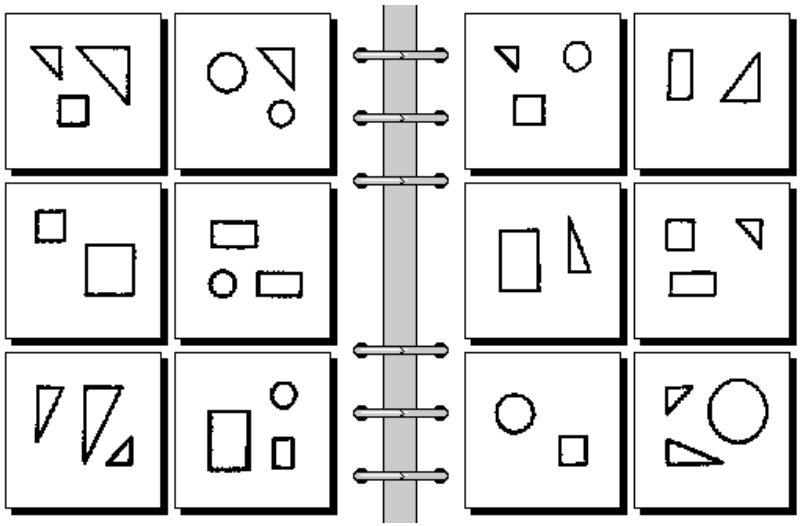


Identify a rule that separates the figures on the left from those on the right

Bongard Problem



Bongard Problem



Types of Learning

- There are many types of learning
  - Supervised learning
  - Unsupervised learning
  - Reinforcement learning
  - Evolutionary (genetic) learning

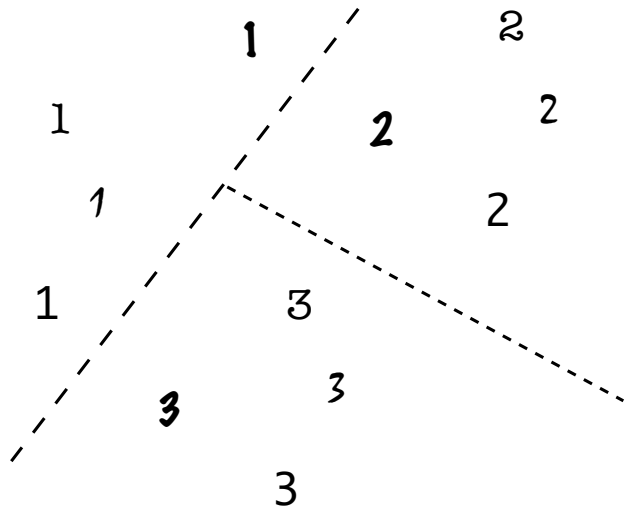
Supervised Learning

- Simplest model of learning
- An agent is given positive and negative examples of some concept or function
- The goal is to learn an approximation of the desired concept or function
- Classification: discrete concept spaces
- Regression: real-valued functions

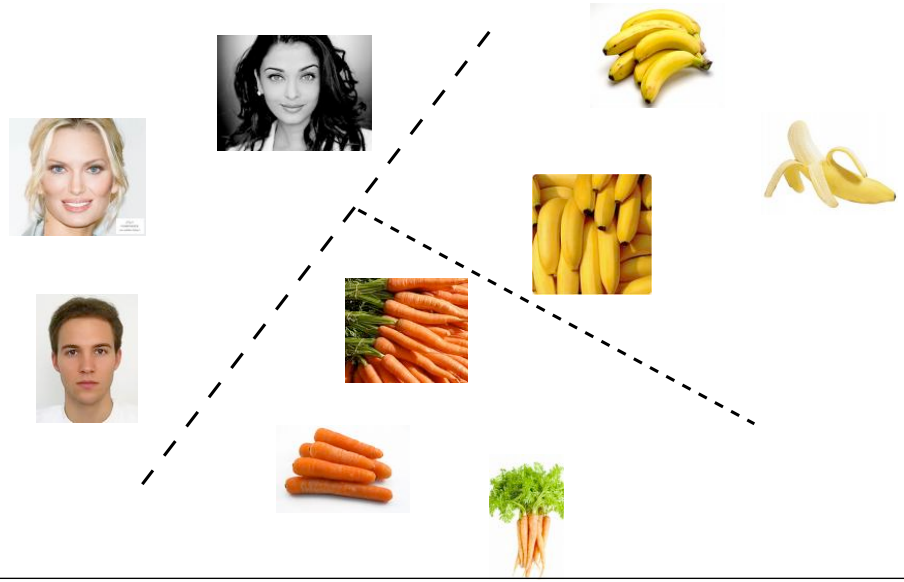
Character Recognition

- Humans can effortlessly recognize complex visual patterns (characters, faces, text)
- This apparently simple problem is formidably difficult for machines

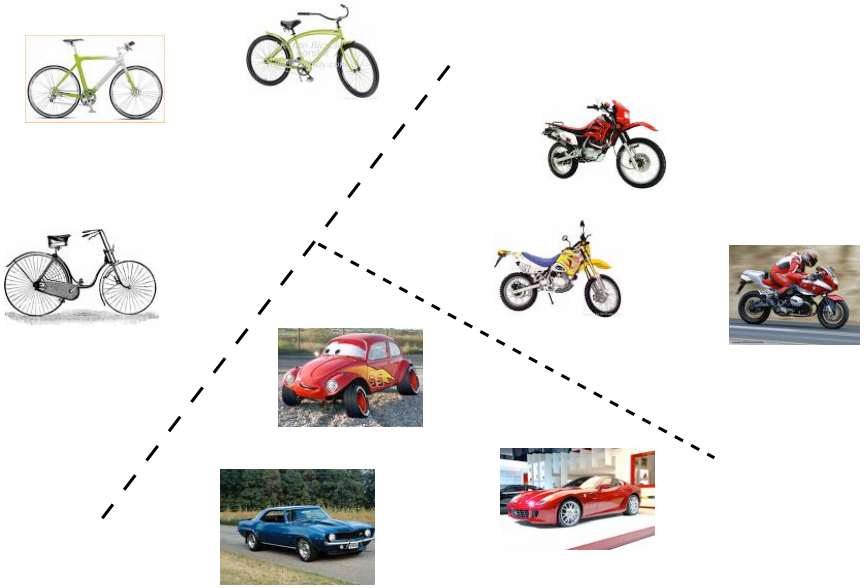
Classification



Classification



Classification



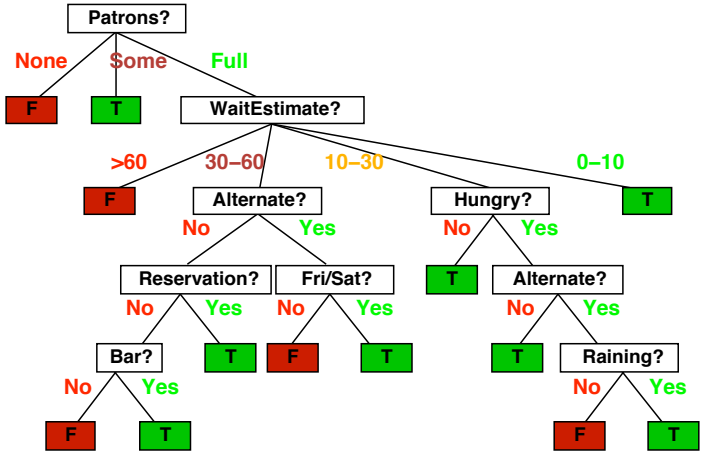
Attribute-Value Data

Examples described by attribute values (Boolean, discrete, continuous, etc.)
E.g., situations where I will/won't wait for a table:

Example	Attributes										Target
	Alt	Bar	Fri	Hun	Pat	$Price$	$Rain$	Res	$Type$	Est	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*10–30*	*T*
X_5	*T*	*F*	*T*	*F*	*Full*	*$$$*	*F*	*T*	*French*	*>60*	*F*
X_6	*F*	*T*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Italian*	*0–10*	*T*
X_7	*F*	*T*	*F*	*F*	*None*	*$*	*T*	*F*	*Burger*	*0–10*	*F*
X_8	*F*	*F*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Thai*	*0–10*	*T*
X_9	*F*	*T*	*T*	*F*	*Full*	*$*	*T*	*F*	*Burger*	*>60*	*F*
X_{10}	*T*	*T*	*T*	*T*	*Full*	*$$$*	*F*	*T*	*Italian*	*10–30*	*F*
X_{11}	*F*	*F*	*F*	*F*	*None*	*$*	*F*	*F*	*Thai*	*0–10*	*F*
X_{12}	*T*	*T*	*T*	*T*	*Full*	*$*	*F*	*F*	*Burger*	*30–60*	*T*

Classification of examples is positive (T) or negative (F)

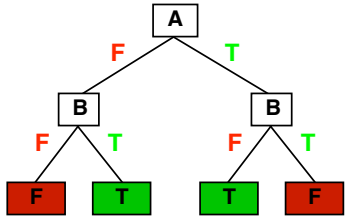
Decision Trees



Boolean Functions

Decision trees can express any function of the input attributes.
E.g., for Boolean functions, truth table row $\rightarrow$ path to leaf:

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



Trivially, there is a consistent decision tree for any training set
w/ one path to leaf for each example (unless f nondeterministic in x)
but it probably won't generalize to new examples

Prefer to find more **compact** decision trees

Hypothesis Spaces

How many distinct decision trees with n Boolean attributes??

$=$ number of Boolean functions
$=$ number of distinct truth tables with 2^n rows $= 2^{2^n}$

E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

How many purely conjunctive hypotheses (e.g., $Hungry \wedge \neg Rain$)??

Each attribute can be in (positive), in (negative), or out
$\Rightarrow$ 3^n distinct conjunctive hypotheses

More expressive hypothesis space
– increases chance that target function can be expressed
– increases number of hypotheses consistent w/ training set
$\Rightarrow$ may get worse predictions

Learning Decision Trees

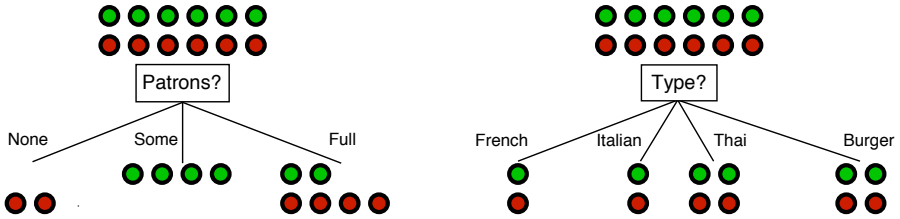
Aim: find a small tree consistent with the training examples

Idea: (recursively) choose "most significant" attribute as root of (sub)tree

```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

Attribute Selection

Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



Patrons? is a better choice—gives **information** about the classification

Entropy

- We can apply ideas from the field of information theory to attribute selection
- Given a set of events, each of which occurs with probability p_i , entropy measures the surprise associated with a particular outcome
- Low frequency events are more surprising

$$H(P) = -\sum_i p_i \log_2 p_i$$

Information Theory

Suppose we have p positive and n negative examples at the root
$\Rightarrow$ $H(\langle p/(p+n), n/(p+n)\rangle)$ bits needed to classify a new example
E.g., for 12 restaurant examples, $p = n = 6$ so we need 1 bit

An attribute splits the examples E into subsets E_i, each of which (we hope) needs less information to complete the classification

Let E_i have p_i positive and n_i negative examples
$\Rightarrow$ $H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$ bits needed to classify a new example
$\Rightarrow$ **expected** number of bits per example over all branches is

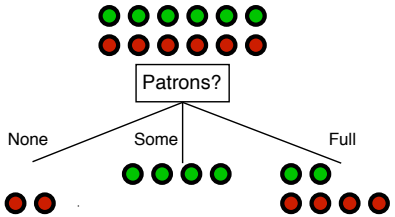
$$\sum_i \frac{p_i + n_i}{p + n} H(\langle p_i/(p_i + n_i), n_i/(p_i + n_i)\rangle)$$

For $Patrons?$, this is 0.459 bits, for $Type$ this is (still) 1 bit

$\Rightarrow$ choose the attribute that minimizes the remaining information needed

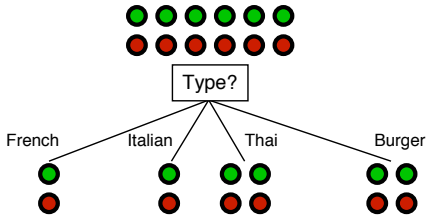
Entropy reduction

Entropy = 1 bit



Entropy = 0.459 bits

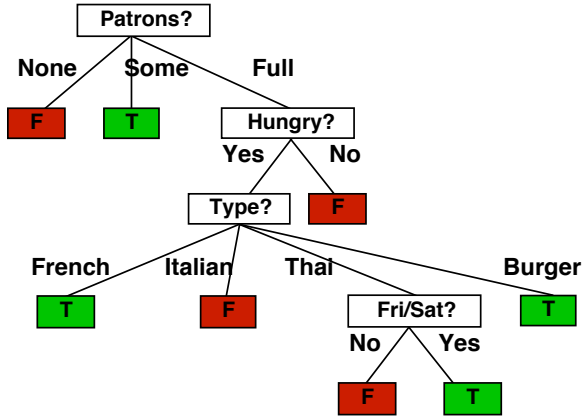
Entropy = 1 bit



Entropy = 1 bit

Learned Decision Tree

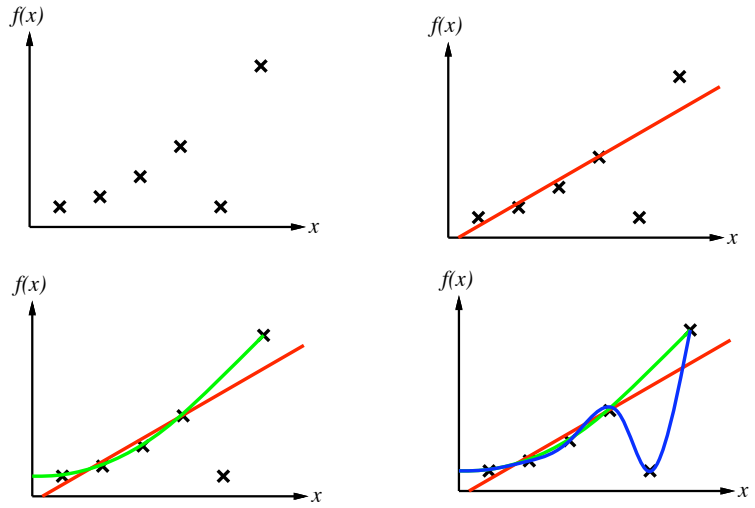
Decision tree learned from the 12 examples:



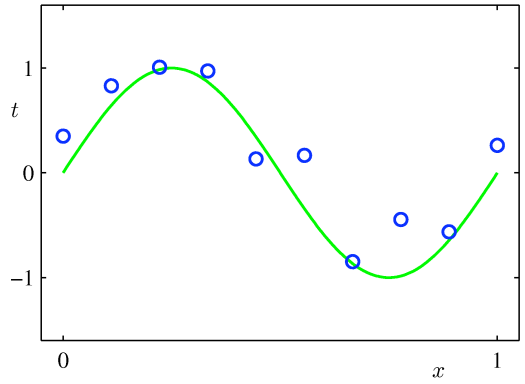
Regression

- Another common type of learning involves making continuous real-valued predictions
  - How much does it cost to fly to Europe?
  - How long to drive to Northampton?
  - How much money will I make when I graduate?

Regression



Polynomial Curve Fitting



$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

Linear Models

Generally

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})$$

where $\phi_j(\mathbf{x})$ are known as *basis functions*.

Typically, $\phi_0(\mathbf{x}) = 1$, so that w_0 acts as a bias.

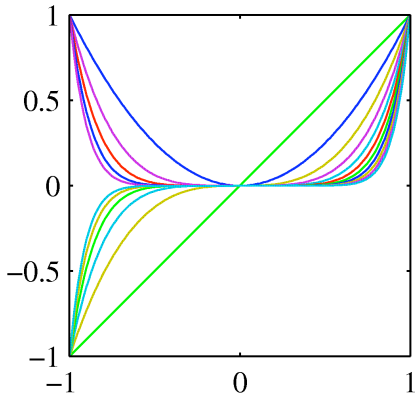
In the simplest case, we use linear basis functions : $\phi_d(\mathbf{x}) = x_d$.

Polynomial Basis

Polynomial basis functions:

$$\phi_j(x) = x^j.$$

These are global; a small change in x affect all basis functions.

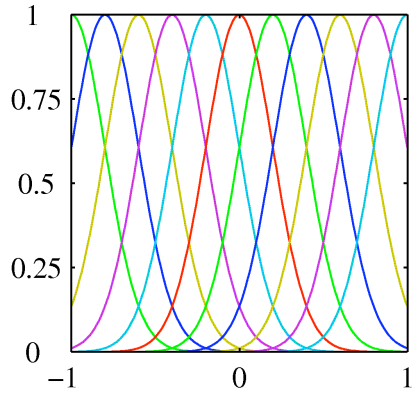


Gaussian Bases

Gaussian basis functions:

$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$$

These are local; a small change in x only affect nearby basis functions. μ_j and s control location and scale (width).



Minimize Squared Error

- Process of curve fitting is based on minimizing a loss function
- One example is minimizing sum of squared errors (between predicted and actual values)

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2$$

Widrow-Hoff Algorithm

- Incremental algorithm that modifies the weights based on the gradient of the error
- For each example i in a dataset:

$$w^{t+1} \leftarrow w^t + \alpha_t(t_i - \phi(x_i)^T w^t)\phi(x_i)$$

- until the error is small enough

Matrix Approach

- Taking gradient of the error function gives

$$\frac{\partial}{\partial w} E_D(w) = \beta \sum_{n=1}^{N} \{t_n - w^T \phi(x_n)\} \phi(x_n)^T = 0$$

Solving for $\mathbf{w}$, we get

$$\mathbf{w}_{\mathrm{ML}} = \overbrace{\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}}\mathbf{t}$$

The Moore-Penrose pseudo-inverse, $\mathbf{\Phi}^{\dagger}$.

where

$$\mathbf{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}.$$

Summary

- Learning is a fundamental component of intelligence
- Classification is a way of discriminating among categories
- Decision trees: simple classification method
- Regression is the estimation of functions
- Least-squares is a standard method for regression