# Machine Learning

George Konidaris
gdk@cs.duke.edu

DUKE COMPUTER SCIENCE

**Spring 2015**

## Information

| $A$ | $B$ | $C$ | L |
|---|---|---|---|
| T | F | T | 1 |
| T | T | F | 1 |
| T | F | F | 1 |
| F | T | F | 2 |
| F | T | T | 2 |
| F | T | F | 2 |
| F | F | T | 1 |
| F | F | F | 1 |

$$I(A) = -f_1 \log_2 f_1 - f_2 \log_2 f_2$$

$$Gain(B) = I(A) - \sum_i f_i I(B_i)$$

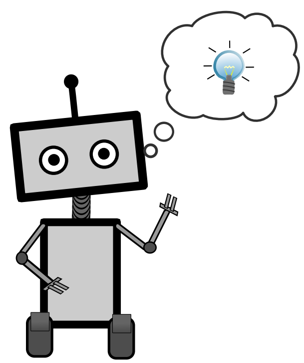
## Machine Learning

Subfield of AI concerned with *learning from data.*

Broadly, using:

- *Experience*
- To Improve *Performance*
- On Some *Task*

*(Tom Mitchell, 1997)*



## Recall: Supervised Learning

Formal definition:

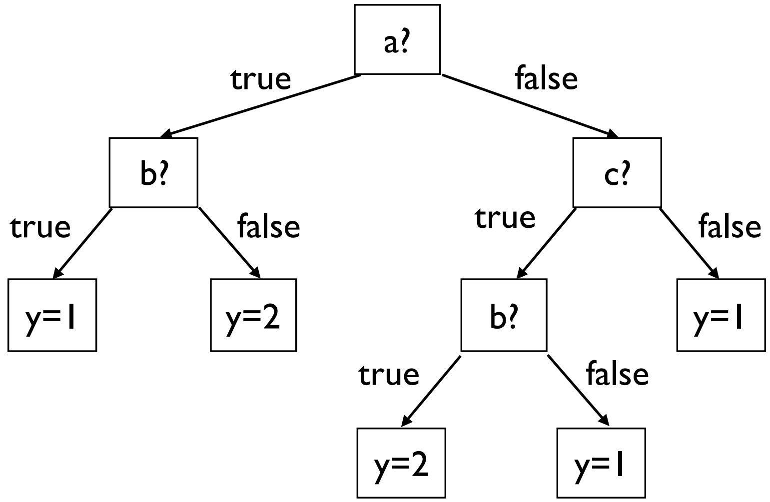
Given training data:
X = {$x_1, \ldots, x_n$} inputs
Y = {$y_1, \ldots, y_n$} labels - if discrete: *classification*

Produce:
Decision function $f : X \to Y$

That minimizes error:

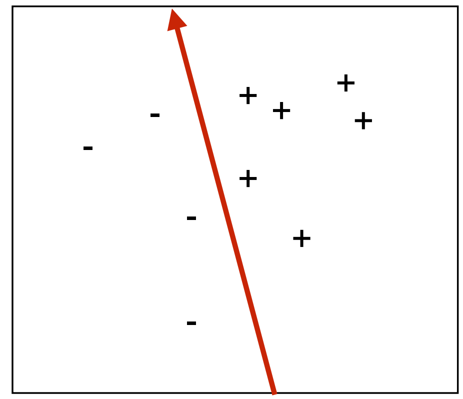
$$\sum_i err(f(x_i), y_i)$$

# Decision Trees




# The Perceptron

If your input ($x_i$) is real-valued … *explicit decision boundary*?




# The Perceptron

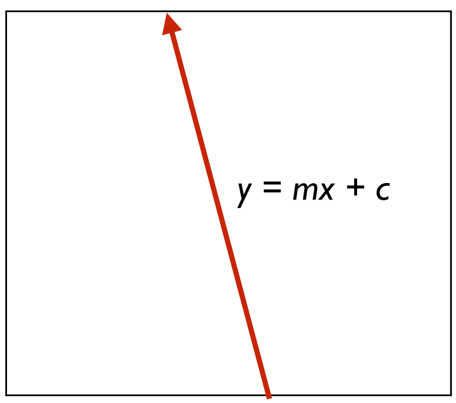
For $x = [x(1), \ldots, x(n)]$:

- Create an *n*-d line
- Slope for each *n*
- Constant offset

$$f(x) = \mathtt{sign}(w \cdot x + c)$$

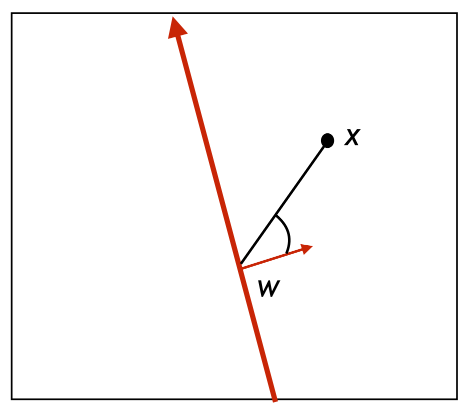
gradient

offset

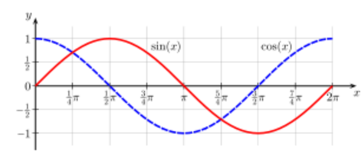



# The Perceptron

Which side of a line are you on?

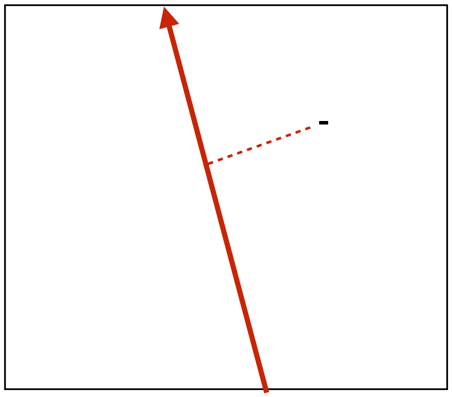



$$w \cdot x = ||w||||x|| \cos(\theta)$$



# The Perceptron

How do you reduce error?




$$e = (y_i - (w \cdot x_i + c))^2$$

$$\frac{\partial e}{\partial w_j} = -2(y_i - (w_i \cdot x_i + c))x_i(j)$$

descend this gradient
to reduce error

# The Perceptron Algorithm

Assume you have a *batch* of data:
$X = \{x_1, \ldots, x_n\}$
$Y = \{y_1, \ldots, y_n\}$

set *w*, *c* to 0.
for each $x_i$:
  **predict** $z_i = \text{sign}(w.x_i + c)$
  if $z_i$ != $y_i$:
    $w = w + a(y_i - z_i)x_i$

learning rate

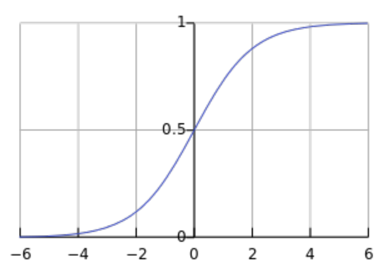
converges if data
is linearly separate

# Probabilities

What if you want a *probabilistic classifier*?

Instead of *sign*, squash output of
linear sum down to [0, 1]:

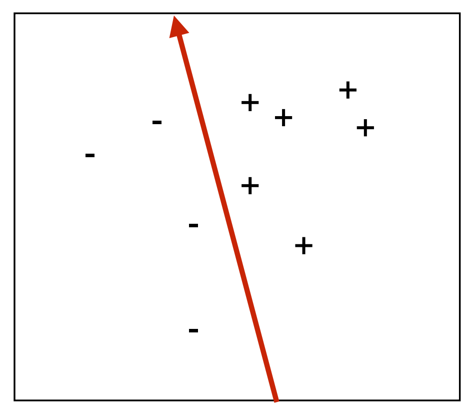
$$\sigma(w \cdot x + c)$$




Resulting algorithm:
**logistic regression.**

# Perceptrons

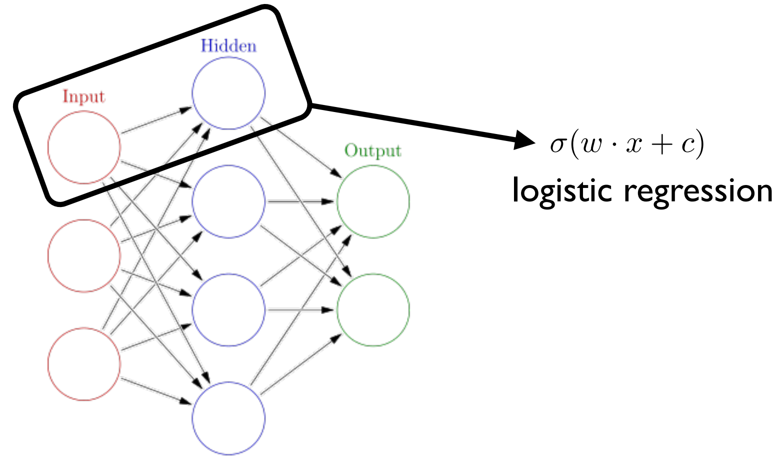
What can't you do?



## Frank Rosenblatt

Built the *Mark I* in 1960.


## Neural Networks




## Nonparametric Methods

Most ML methods can be characterized by setting a few parameters.

Alternative approach:

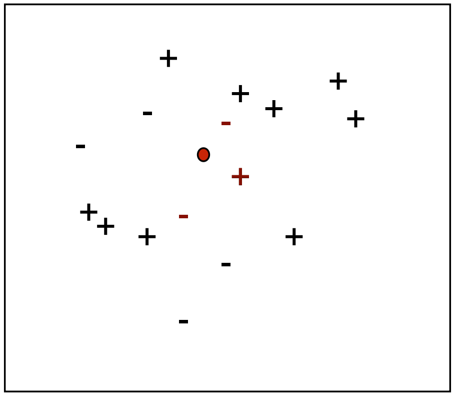
- Let the data speak for itself.
- No finite-sized parameter vector.
- Usually more interesting decision boundaries.

## K-Nearest Neighbors

Given training data:
$X = \{x_1, \ldots, x_n\}$
$Y = \{y_1, \ldots, y_n\}$
Distance metric D(xi, xj)

For a new data point $x_{new}$:
find *k* nearest points in X (measured via D)
set $y_{new}$ is the majority label

# K-Nearest Neighbors



# K-Nearest Neighbors

Properties:

- No learning phase.
- Must store all the data.
- *log(n)* computation per sample - *grows with data*.

Decision boundary: ***any function, given enough data.***

***Classic trade-off:*** memory and compute time for flexibility.