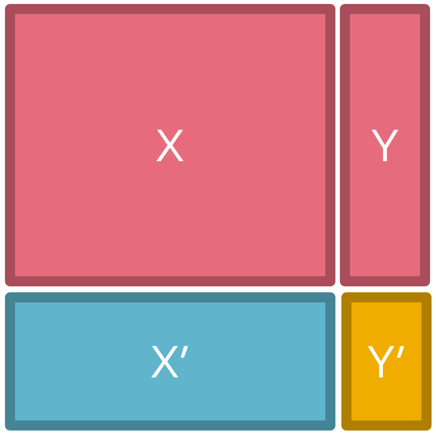
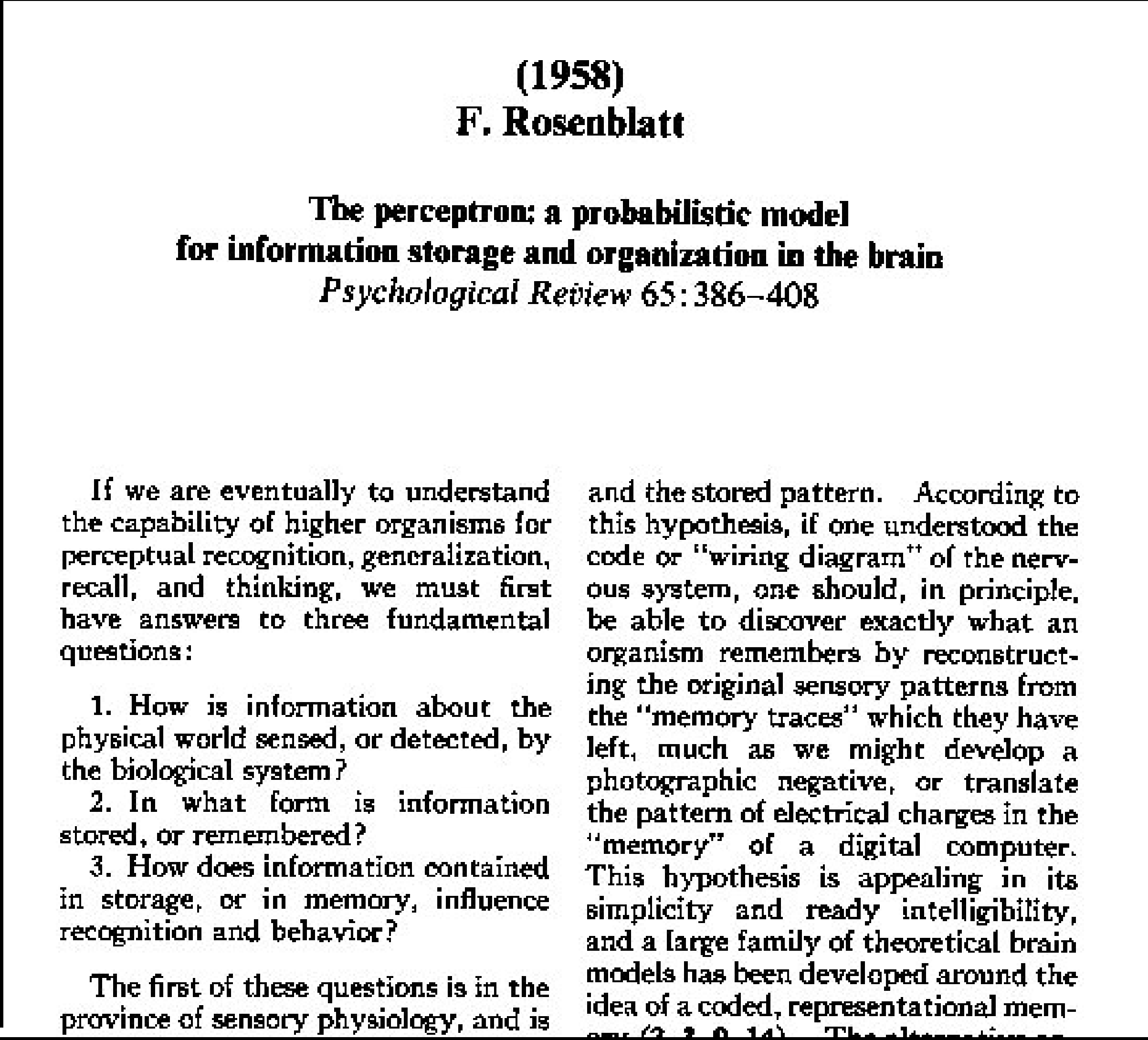
Supervised Learning

- **Would like to do prediction:**
  estimate a function f(x) so that y = f(x)
- **Where *y* can be:**
  - Real number: Regression
  - Categorical: Classification
  - Complex object:
    - Ranking of items, Parse tree, etc.
- **Data is labeled:**
  - Have many pairs {(x, y)}
    - x … vector of real valued features
    - y … class ({+1, -1}, or a real number)



Training and test set


(1958)
F. Rosenblatt

The perceptron: a probabilistic model for information storage and organization in the brain
Psychological Review 65:386–408

If we are eventually to understand the capability of higher organisms for perceptual recognition, generalization, recall, and thinking, we must first have answers to three fundamental questions:

1. How is information about the physical world sensed, or detected, by the biological system?
2. In what form is information stored, or remembered?
3. How does information contained in storage, or in memory, influence recognition and behavior?

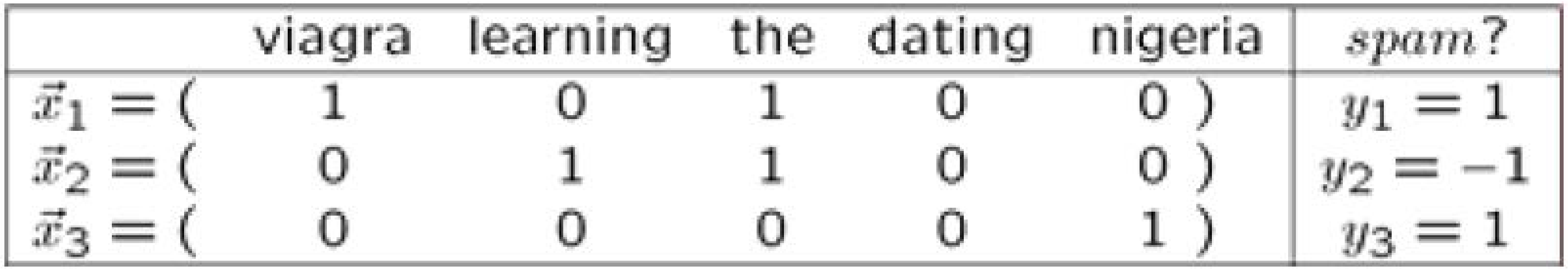
The first of these questions is in the province of sensory physiology, and is

and the stored pattern. According to this hypothesis, if one understood the code or "wiring diagram" of the nervous system, one should, in principle, be able to discover exactly what an organism remembers by reconstructing the original sensory patterns from the "memory traces" which they have left, much as we might develop a photographic negative, or translate the pattern of electrical charges in the "memory" of a digital computer. This hypothesis is appealing in its simplicity and ready intelligibility, and a large family of theoretical brain models has been developed around the idea of a coded, representational mem-

Perceptron

Linear models: Perceptron

- **Example:** **Spam filtering**



	viagra	learning	the	dating	nigeria	*spam?*
$\vec{x}_1 = ($	1	0	1	0	0 $)$	$y_1 = 1$
$\vec{x}_2 = ($	0	1	1	0	0 $)$	$y_2 = -1$
$\vec{x}_3 = ($	0	0	0	0	1 $)$	$y_3 = 1$

- **Instance space x ∈ X** (|X|= n data points)
  - Binary feature vector *x* of word occurrences
  - *d* features (words + other things, d~100,000)
- **Class y ∈ Y:**
  - *y*: Spam (+1), Ham (-1)

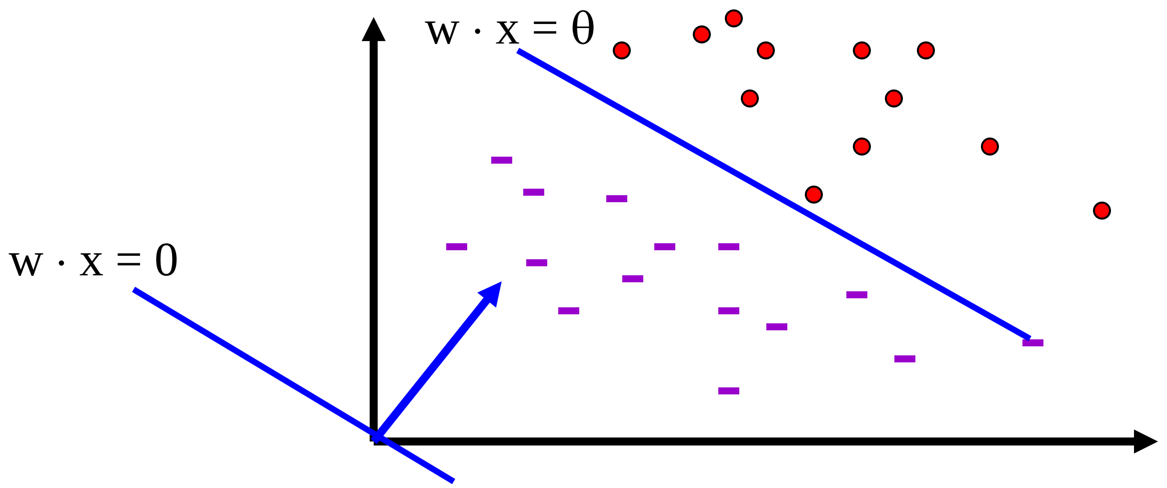
Linear models for classification

- Binary classification:

$$f(x) = \begin{cases} 1 & \text{if } \mathrm{w}_1\,\mathrm{x}_1 + \mathrm{w}_2\,\mathrm{x}_2 + \ldots\, \mathrm{w}_d\,\mathrm{x}_d \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

Decision boundary is **linear**

- Input: Vectors x_i and labels y_i
- Goal: Find vector $w = (w_1, w_2, \ldots, w_d)$
  - Each w_i is a real number


$\mathrm{w} \cdot \mathrm{x} = \theta$
$\mathrm{w} \cdot \mathrm{x} = 0$
–

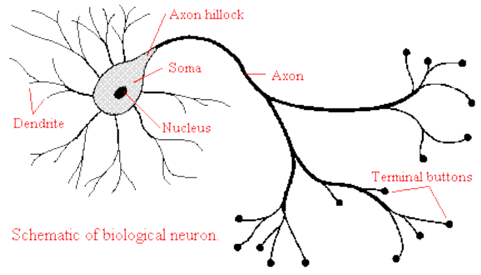
Note:

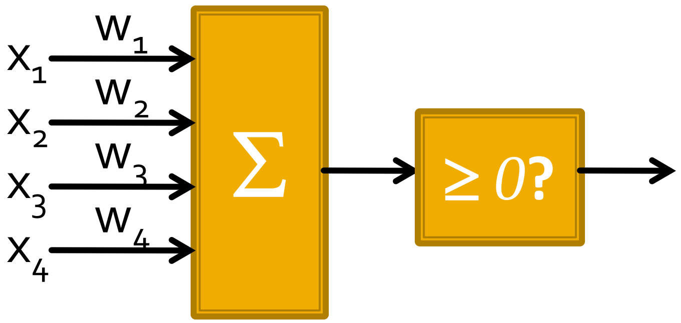
$$\mathbf{x} \Leftrightarrow \langle \mathbf{x}, 1 \rangle \quad \forall \mathbf{x}$$

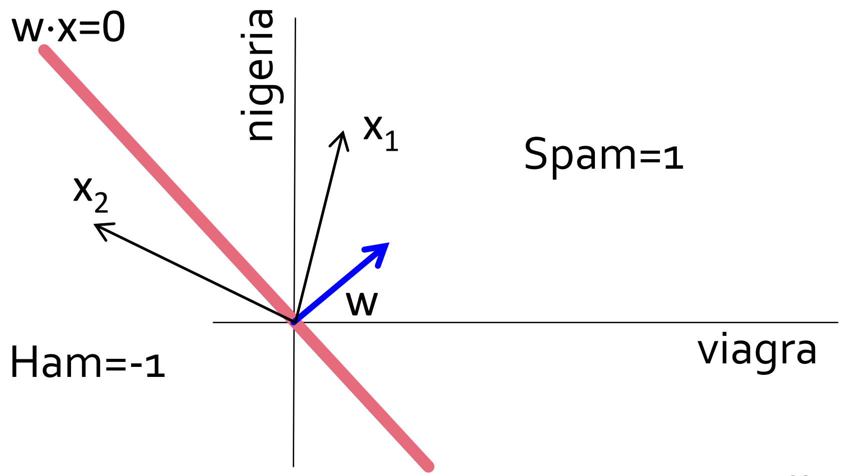
$$\mathbf{w} \Leftrightarrow \langle \mathbf{w}, -\theta \rangle$$

Perceptron [Rosenblatt '57]

- (very) Loose motivation: **Neuron**
- Inputs are feature values
- Each feature has a weight w_i
- Activation is the sum:
  - $f(x) = \Sigma_i w_i x_i = w \cdot x$
- If the *f(x)* is:
  - Positive: predict +1
  - Negative: predict -1





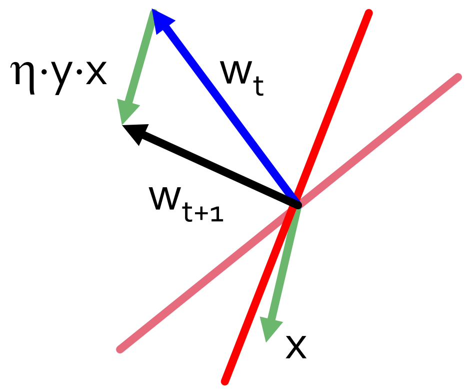


Perceptron: Estimating w

- **Perceptron: *$y' = sign(w \cdot x)$***
- **How to find parameters *w*?**
  - Start with $w_0 = 0$
  - Pick training examples x_t one by one (from disk)
  - Predict class of x using current weights
    - $y' = sign(w_t \cdot x_t)$
  - If y' is correct (i.e., $y_t = y'$)
    - No change: $w_{t+1} = w_t$
  - If y' is wrong: adjust w

$$w_{t+1} = w_t + \eta \cdot y_t \cdot x_t$$

- η is the learning rate parameter
- x_t is the training example
- y_t is true class label ({+1, -1})



Perceptron Convergence

- Perceptron Convergence Theorem:
  - If there exist a set of weights that are consistent (i.e., the data is linearly separable) the perceptron learning algorithm will converge
- How long would it take to converge?
- Perceptron Cycling Theorem:
  - If the training data is not linearly separable the perceptron learning algorithm will eventually repeat the same set of weights and therefore enter an infinite loop
- How to provide robustness, more expressivity?

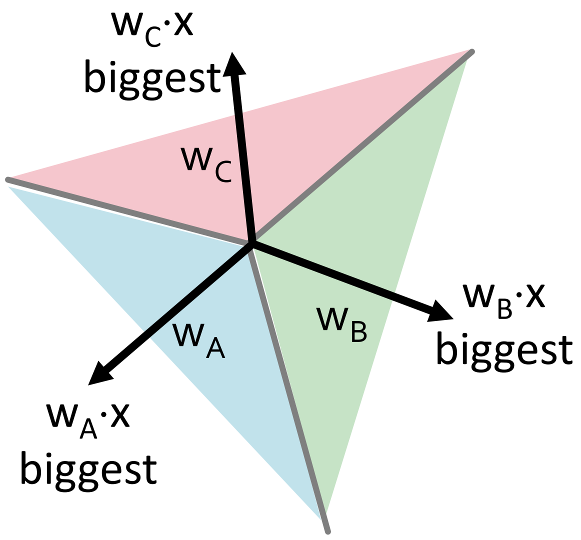
Multiclass Perceptron

- **If more than 2 classes:**
  - Weight vector w_c for each class
    - Train one class vs. the rest
      - Example: 3-way classification y = {A, B, C}
      - Train 3 classifies: w_A: A vs. B,C; w_B: B vs. A,C; w_C: C vs. A,B
  - **Calculate activation for each class**

$$f(x,c) = \Sigma_i \ w_{c,i} \ x_i \ = \ w_c \cdot x$$

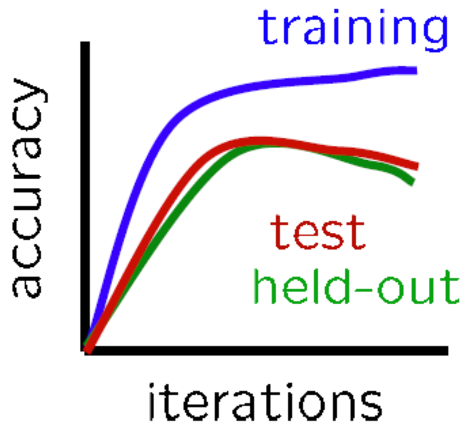
  - **Highest activation wins:**

$$c = \arg\max_c f(x,c)$$

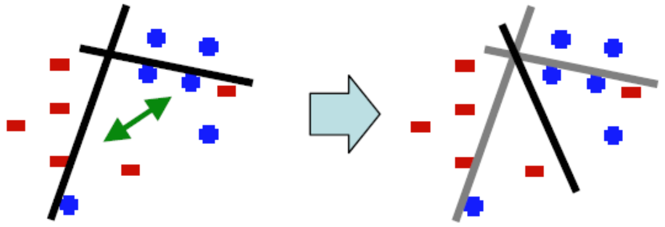


Issues with Perceptrons

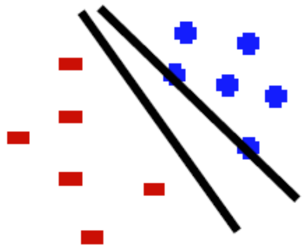
- **Overfitting:**



- **Regularization:** if the data is not separable weights dance around



- **Mediocre generalization:**
  - Finds a "barely" separating solution



Winnow Algorithm

- **Winnow algorithm**
  - Similar to perceptron, just different updates

```
Initialize: $\theta = n;\ w_i = 1$
Prediction is 1 iff $w \bullet x \geq \theta$
If no mistake: do nothing
If $f(x) = 1$ but $w \bullet x < \theta$, $w_i \leftarrow 2w_i$ (if $x_i = 1$) (promotion)
If $f(x) = 0$ but $w \bullet x \geq \theta$, $w_i \leftarrow w_i/2$ (if $x_i = 1$) (demotion)
```

  - Learns linear threshold functions

Summary of Algorithms

$\textbf{Examples}: x \in \{0,1\}^n; \quad \textbf{Hypothesis}: w \in R^n$

$\textbf{Prediction is 1 iff } w \bullet x \geq \theta$

- Additive weight update algorithm [Perceptron, Rosenblatt, 1958]

$w \leftarrow w + \eta_i \, y_j x_j$

$\text{If Class} = 1 \text{ but } w \bullet x \leq \theta, \quad w_i \leftarrow w_i + 1 \text{ (if } x_i = 1) \text{ (promotio}$

$\text{If Class} = 0 \text{ but } w \bullet x \geq \theta, \quad w_i \leftarrow w_i - 1 \text{ (if } x_i = 1) \text{ (demotion)}$

- Multiplicative weight update algorithm [Winnow, Littlestone, 1988]

$w \leftarrow w \, \eta_i \exp\{y_j x_j\}$

$\text{If Class} = 1 \text{ but } w \bullet x \leq \theta, \quad w_i \leftarrow 2w_i \text{ (if } x_i = 1) \text{ (promotion)}$

$\text{If Class} = 0 \text{ but } w \bullet x \geq \theta, \quad w_i \leftarrow w_i/2 \text{ (if } x_i = 1) \text{ (demotion)}$

Perceptron vs. Winnow

- **Perceptron**
- Online: can adjust to changing target, over time
- Advantages
  - Simple
  - Guaranteed to learn a linearly separable problem
- Limitations
  - only linear separations
  - only converges for linearly separable data
  - not really “efficient with many features”

- **Winnow**
- Online: can adjust to changing target, over time
- Advantages
  - Simple
  - Guaranteed to learn a linearly separable problem
  - **Suitable for problems with many irrelevant attributes**
- Limitations
  - only linear separations
  - only converges for linearly separable data
  - not really “efficient with many features”