

# Perceptron

Instructor: Alan Ritter

Many Slides from Luke Zettemoyer and Ian Murray

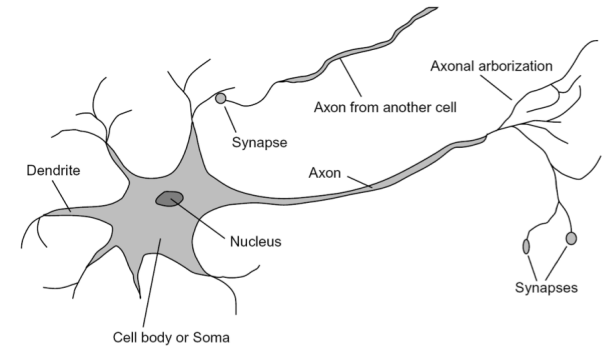
# Who needs probabilities?

- Previously: model data with distributions
- Joint:  $P(X,Y)$ 
  - e.g. Naïve Bayes
- Conditional:  $P(Y|X)$ 
  - e.g. Logistic Regression
- But wait, why probabilities?
- Lets try to be error-driven!

mpg	cylinders	displacemen	horsepower	weight	acceleration	modelyear	make
good	4	97	75	2265	18.2	77	asia
bad	6	199	90	2648	15	70	ameri
bad	4	121	110	2600	12.8	77	europ
bad	8	350	175	4100	13	73	ameri
bad	6	198	95	3102	16.5	74	ameri
bad	4	108	94	2379	16.5	73	asia
bad	4	113	95	2228	14	71	asia
bad	8	302	139	3570	12.8	78	ameri
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
good	4	120	79	2625	18.6	82	ameri
bad	8	455	225	4425	10	70	ameri
good	4	107	86	2464	15.5	76	europ
bad	5	131	103	2830	15.9	78	europ

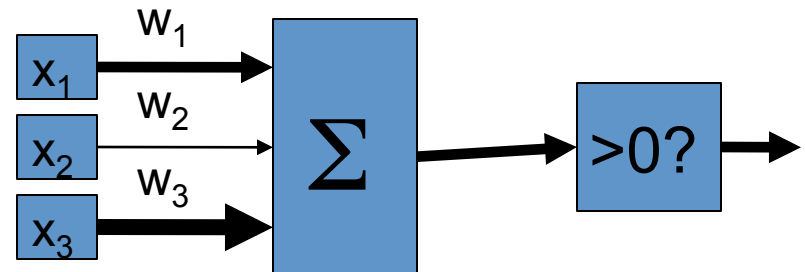
# Linear Classifiers

- Inputs are **feature values**
- Each feature has a **weight**
- Sum is the **activation**



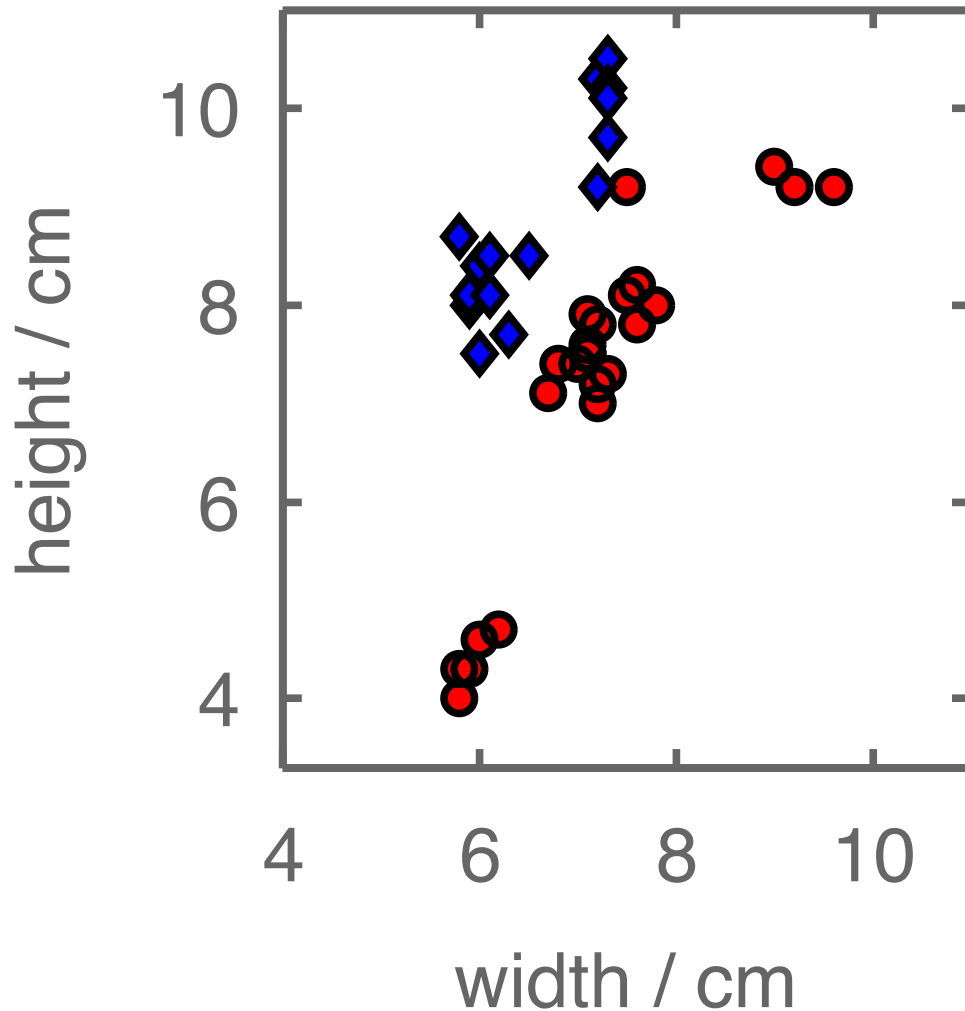
$$\text{activation}_w(x) = \sum_i w_i x_i = w \cdot x$$

- If the activation is:
  - Positive, output *class 1*
  - Negative, output *class 2*

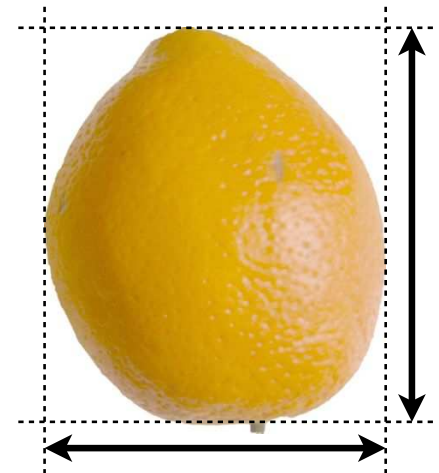


# A two-dimensional space

---



- Oranges
- ◆ Lemons



# The weight vector

---

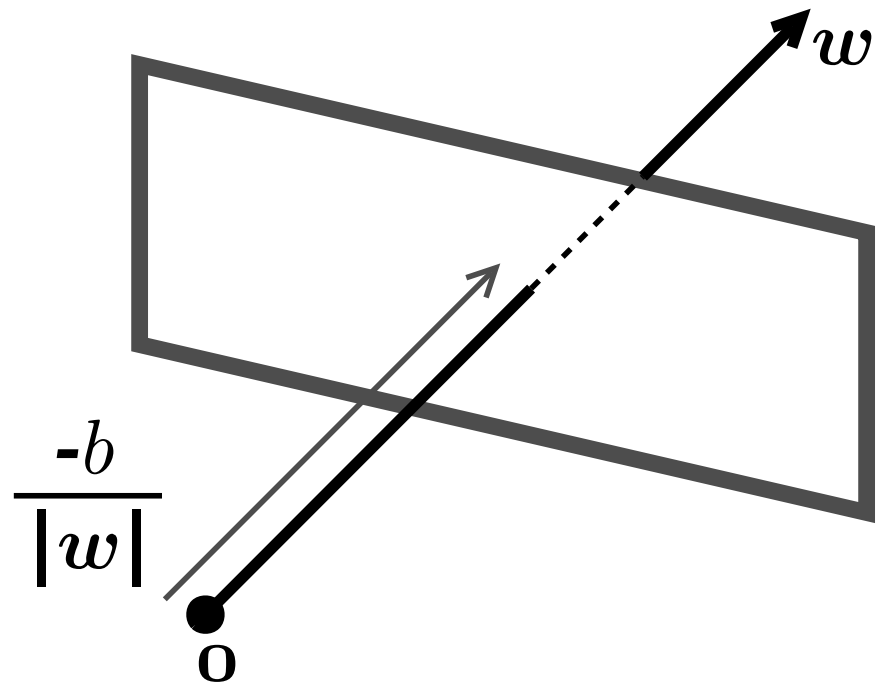
Define positive class region:  $\mathbf{w}^\top \mathbf{x} + b > 0$

$$w_1x_1 + w_2x_2 + w_3x_3 + \dots b > 0$$

$$\sum_d w_d x_d + b > 0$$

We will set  $b = 0$ :

$$\mathbf{w}^\top \mathbf{x} > 0$$



# Learning the weights

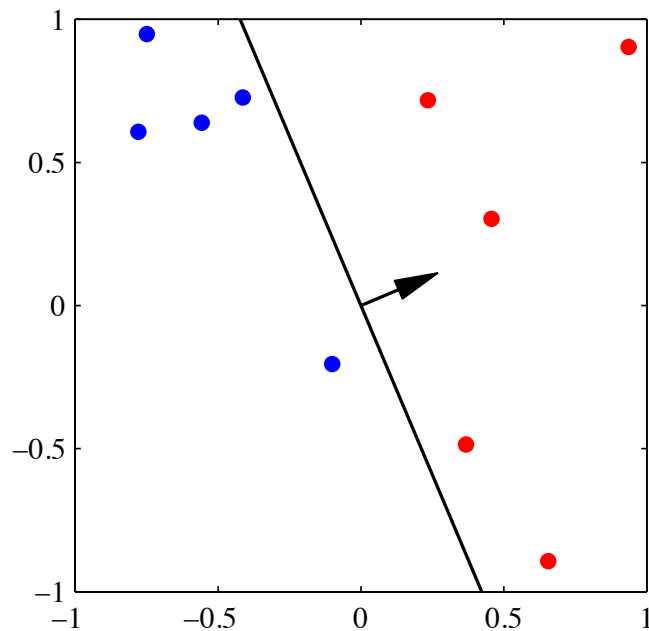
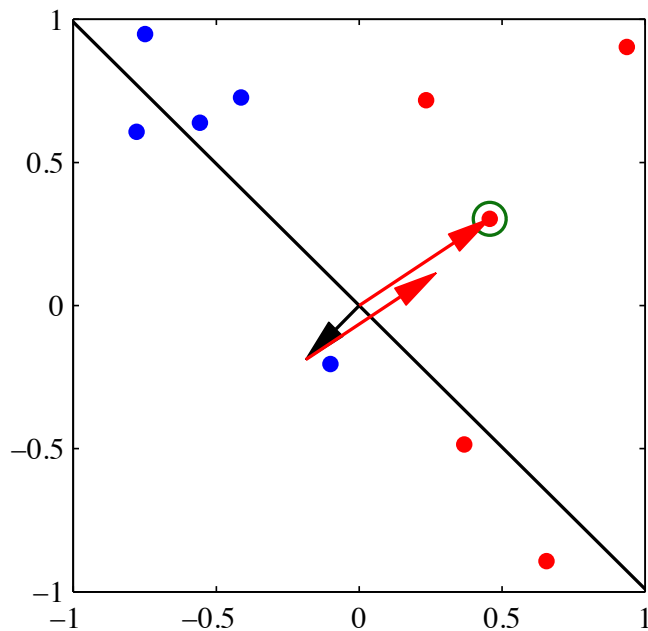
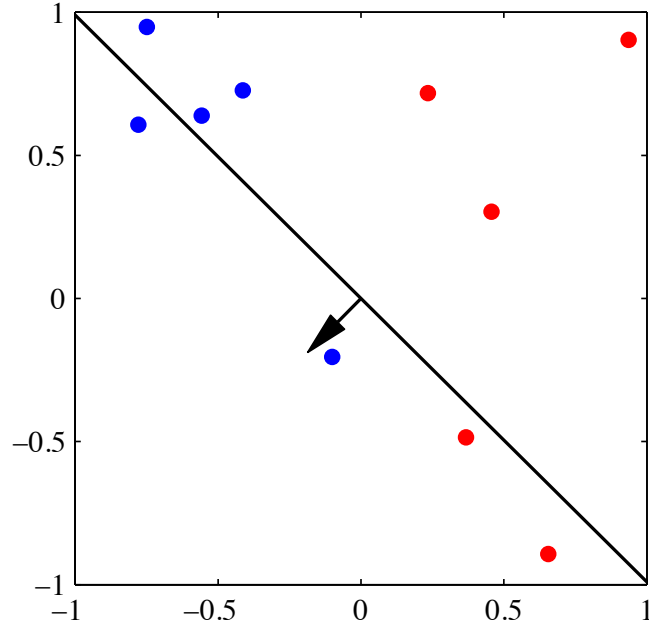
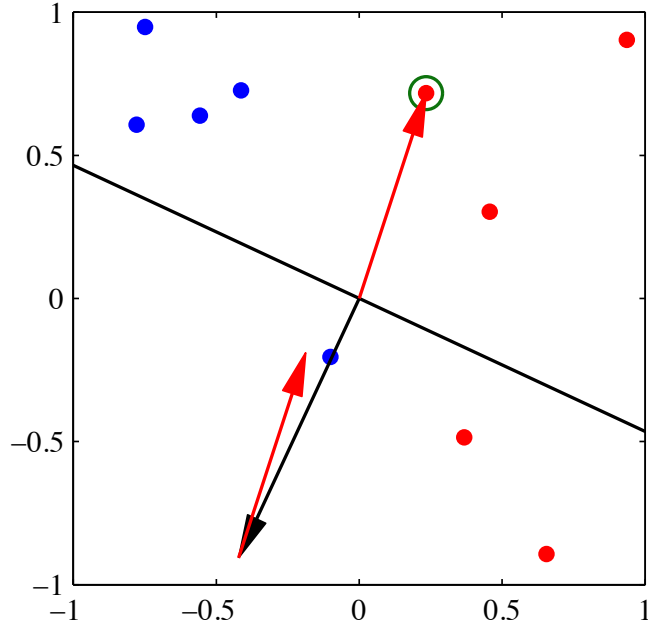
---

## A ‘perceptron’ learning rule:

$$\hat{y} \leftarrow \text{sgn}(\mathbf{w}^\top \mathbf{x})$$

$$\mathbf{w} \leftarrow \mathbf{w} + (y - \hat{y})\mathbf{x}$$

```
% Matlab/Octave code
old_ww = []; ww = zeros(size(xx,2), 1);
while ~isequal(ww, old_ww)
    old_ww = ww;
    for ii = 1:N
        pred = sign(xx(ii, :)*ww);
        ww = ww + (yy(ii) - pred)*xx(ii, :);
    end
end
end
```



# Example: Spam

- Imagine 3 features (spam is “positive” class):
  - free (number of occurrences of “free”)
  - money (occurrences of “money”)
  - BIAS (intercept, always has value 1)

“free money”

$x$	$w$
BIAS : 1	BIAS : -3
free : 1	free : 4
money : 1	money : 2
...	...

$$\begin{aligned} & (1)(-3) + \\ & (1)(4) + \\ & (1)(2) + \\ & \dots \\ & = 3 \end{aligned}$$

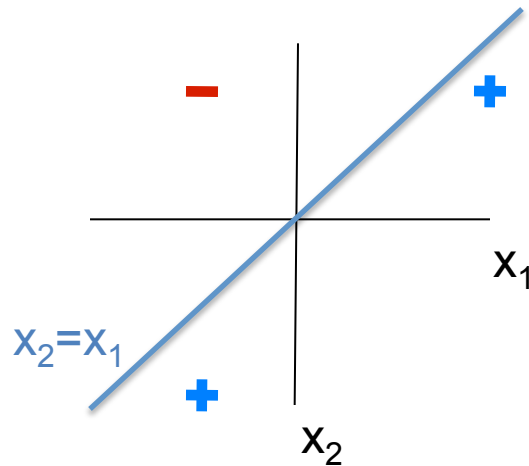
$w \bullet x > 0 \Rightarrow \text{SPAM!!!}$



- For  $t=1..T$ ,  $i=1..n$ :
  - $y = \text{sign}(w \cdot x^i)$
  - if  $y \neq y^i$ 

$$w = w + y^i x^i$$

$x_1$	$x_2$	$y$
3	2	1
-2	2	-1
-2	-3	1



Initial:

- $w = [0,0]$

$t=1, i=1$

- $[0,0] \cdot [3,2] = 0$ ,  $\text{sign}(0) = -1$

- $w = [0,0] + [3,2] = [3,2]$

$t=1, i=2$

- $[3,2] \cdot [-2,2] = -2$ ,  $\text{sign}(-2) = -1$

$t=1, i=3$

- $[3,2] \cdot [-2,-3] = -12$ ,  $\text{sign}(-12) = -1$

- $w = [3,2] + [-2,-3] = [1,-1]$

$t=2, i=1$

- $[1,-1] \cdot [3,2] = 1$ ,  $\text{sign}(1) = 1$

$t=2, i=2$

- $[1,-1] \cdot [-2,2] = -4$ ,  $\text{sign}(-4) = -1$

$t=2, i=3$

- $[1,-1] \cdot [-2,-3] = 1$ ,  $\text{sign}(1) = 1$

Converged!!!

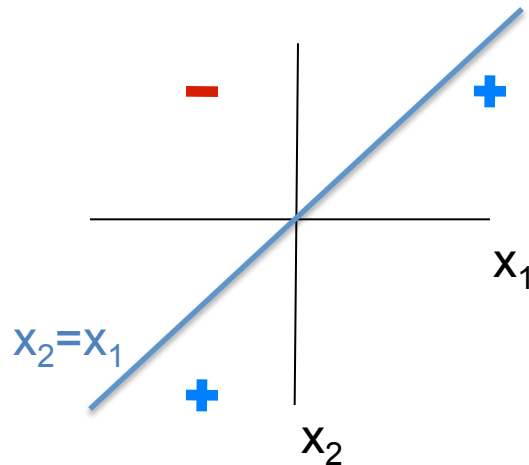
- $y = w_1 x_1 + w_2 x_2 \rightarrow y = x_1 - x_2$

- So, at  $y=0 \rightarrow x_2 = x_1$

- For  $t=1..T$ ,  $i=1..n$ :
  - $y = \text{sign}(w \cdot x^i)$
  - if  $y \neq y^i$ 

$$w = w + y^i x^i$$

$x_1$	$x_2$	$y$
3	2	1
-2	2	-1
-2	-3	1



Initial:

- $w = [0,0]$

$t=1, i=1$

- $[0,0] \cdot [3,2] = 0$ ,  $\text{sign}(0) = -1$

- $w = [0,0] + [3,2] = [3,2]$

$t=1, i=2$

- $[3,2] \cdot [-2,2] = -2$ ,  $\text{sign}(-2) = -1$

$t=1, i=3$

- $[3,2] \cdot [-2,-3] = -12$ ,  $\text{sign}(-12) = -1$

- $w = [3,2] + [-2,-3] = [1,-1]$

$t=2, i=1$

- $[1,-1] \cdot [3,2] = 1$ ,  $\text{sign}(1) = 1$

$t=2, i=2$

- $[1,-1] \cdot [-2,2] = -4$ ,  $\text{sign}(-4) = -1$

$t=2, i=3$

- $[1,-1] \cdot [-2,-3] = 1$ ,  $\text{sign}(1) = 1$

Converged!!!

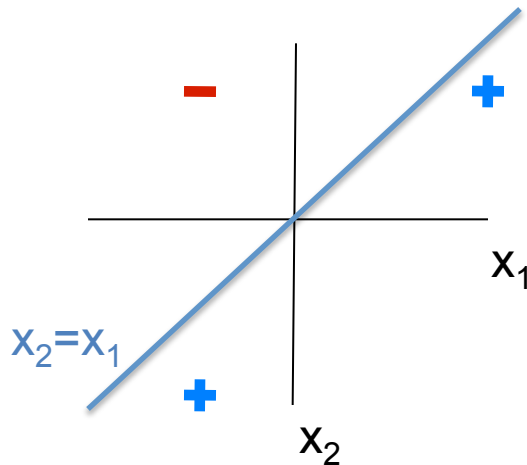
- $y = w_1 x_1 + w_2 x_2 \rightarrow y = x_1 - x_2$

- So, at  $y=0 \rightarrow x_2 = x_1$

- For  $t=1..T$ ,  $i=1..n$ :
  - $y = \text{sign}(w \cdot x^i)$
  - if  $y \neq y^i$ 

$$w = w + y^i x^i$$

$x_1$	$x_2$	$y$
3	2	1
-2	2	-1
-2	-3	1



Initial:

- $w = [0,0]$

$t=1, i=1$

- $[0,0] \cdot [3,2] = 0$ ,  $\text{sign}(0) = -1$

- $w = [0,0] + [3,2] = [3,2]$

$t=1, i=2$

- $[3,2] \cdot [-2,2] = -2$ ,  $\text{sign}(-2) = -1$

$t=1, i=3$

- $[3,2] \cdot [-2,-3] = -12$ ,  $\text{sign}(-12) = -1$

- $w = [3,2] + [-2,-3] = [1,-1]$

$t=2, i=1$

- $[1,-1] \cdot [3,2] = 1$ ,  $\text{sign}(1) = 1$

$t=2, i=2$

- $[1,-1] \cdot [-2,2] = -4$ ,  $\text{sign}(-4) = -1$

$t=2, i=3$

- $[1,-1] \cdot [-2,-3] = 1$ ,  $\text{sign}(1) = 1$

Converged!!!

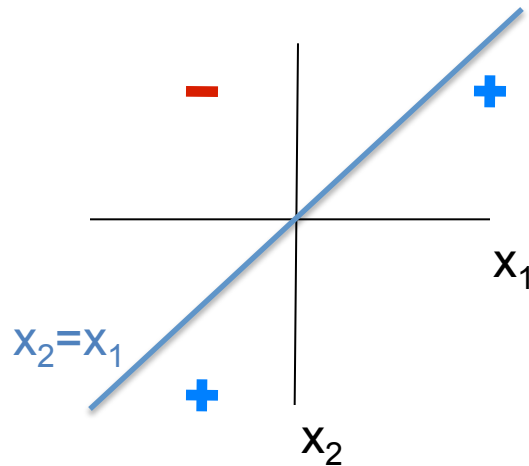
- $y = w_1 x_1 + w_2 x_2 \rightarrow y = x_1 - x_2$

- So, at  $y=0 \rightarrow x_2 = x_1$

- For  $t=1..T$ ,  $i=1..n$ :
  - $y = \text{sign}(w \cdot x^i)$
  - if  $y \neq y^i$ 

$$w = w + y^i x^i$$

$x_1$	$x_2$	$y$
3	2	1
-2	2	-1
-2	-3	1



Initial:

- $w = [0,0]$

$t=1, i=1$

- $[0,0] \cdot [3,2] = 0$ ,  $\text{sign}(0) = -1$

- $w = [0,0] + [3,2] = [3,2]$

$t=1, i=2$

- $[3,2] \cdot [-2,2] = -2$ ,  $\text{sign}(-2) = -1$

$t=1, i=3$

- $[3,2] \cdot [-2,-3] = -12$ ,  $\text{sign}(-12) = -1$

- $w = [3,2] + [-2,-3] = [1,-1]$

$t=2, i=1$

- $[1,-1] \cdot [3,2] = 1$ ,  $\text{sign}(1) = 1$

$t=2, i=2$

- $[1,-1] \cdot [-2,2] = -4$ ,  $\text{sign}(-4) = -1$

$t=2, i=3$

- $[1,-1] \cdot [-2,-3] = 1$ ,  $\text{sign}(1) = 1$

Converged!!!

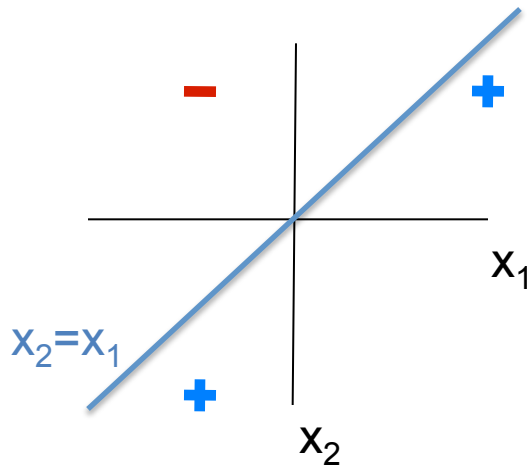
- $y = w_1 x_1 + w_2 x_2 \rightarrow y = x_1 - x_2$

- So, at  $y=0 \rightarrow x_2 = x_1$

- For  $t=1..T$ ,  $i=1..n$ :
  - $y = \text{sign}(w \cdot x^i)$
  - if  $y \neq y^i$ 

$$w = w + y^i x^i$$

$x_1$	$x_2$	$y$
3	2	1
-2	2	-1
-2	-3	1



Initial:

- $w = [0,0]$

$t=1, i=1$

- $[0,0] \cdot [3,2] = 0$ ,  $\text{sign}(0) = -1$

- $w = [0,0] + [3,2] = [3,2]$

$t=1, i=2$

- $[3,2] \cdot [-2,2] = -2$ ,  $\text{sign}(-2) = -1$

$t=1, i=3$

- $[3,2] \cdot [-2,-3] = -12$ ,  $\text{sign}(-12) = -1$

- $w = [3,2] + [-2,-3] = [1,-1]$

$t=2, i=1$

- $[1,-1] \cdot [3,2] = 1$ ,  $\text{sign}(1) = 1$

$t=2, i=2$

- $[1,-1] \cdot [-2,2] = -4$ ,  $\text{sign}(-4) = -1$

$t=2, i=3$

- $[1,-1] \cdot [-2,-3] = 1$ ,  $\text{sign}(1) = 1$



Converged!!!

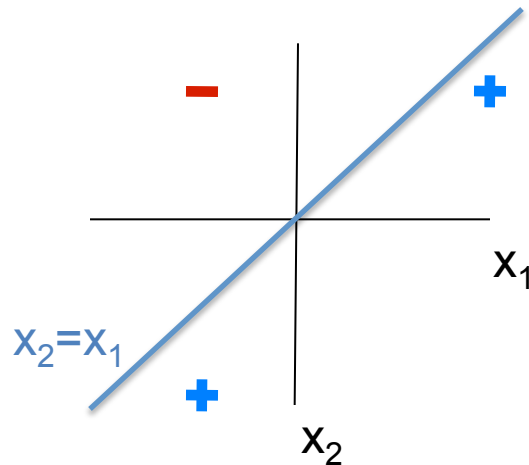
- $y = w_1 x_1 + w_2 x_2 \rightarrow y = x_1 - x_2$

- So, at  $y=0 \rightarrow x_2 = x_1$

- For  $t=1..T$ ,  $i=1..n$ :
  - $y = \text{sign}(w \cdot x^i)$
  - if  $y \neq y^i$ 

$$w = w + y^i x^i$$

$x_1$	$x_2$	$y$
3	2	1
-2	2	-1
-2	-3	1



Initial:

- $w = [0,0]$

$t=1, i=1$

- $[0,0] \cdot [3,2] = 0$ ,  $\text{sign}(0) = -1$

- $w = [0,0] + [3,2] = [3,2]$

$t=1, i=2$

- $[3,2] \cdot [-2,2] = -2$ ,  $\text{sign}(-2) = -1$

$t=1, i=3$

- $[3,2] \cdot [-2,-3] = -12$ ,  $\text{sign}(-12) = -1$

- $w = [3,2] + [-2,-3] = [1,-1]$

$t=2, i=1$

- $[1,-1] \cdot [3,2] = 1$ ,  $\text{sign}(1) = 1$

$t=2, i=2$

- $[1,-1] \cdot [-2,2] = -4$ ,  $\text{sign}(-4) = -1$

$t=2, i=3$

- $[1,-1] \cdot [-2,-3] = 1$ ,  $\text{sign}(1) = 1$

Converged!!!

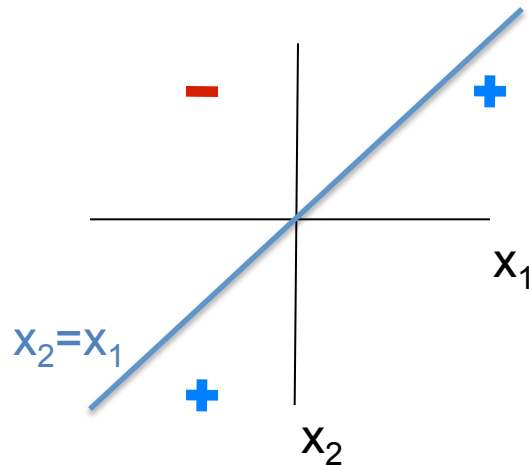
- $y = w_1 x_1 + w_2 x_2 \rightarrow y = x_1 - x_2$

- So, at  $y=0 \rightarrow x_2 = x_1$

- For  $t=1..T$ ,  $i=1..n$ :
  - $y = \text{sign}(w \cdot x^i)$
  - if  $y \neq y^i$ 

$$w = w + y^i x^i$$

$x_1$	$x_2$	$y$
3	2	1
-2	2	-1
-2	-3	1



Initial:

- $w = [0,0]$

$t=1, i=1$

- $[0,0] \cdot [3,2] = 0$ ,  $\text{sign}(0) = -1$

- $w = [0,0] + [3,2] = [3,2]$

$t=1, i=2$

- $[3,2] \cdot [-2,2] = -2$ ,  $\text{sign}(-2) = -1$

$t=1, i=3$

- $[3,2] \cdot [-2,-3] = -12$ ,  $\text{sign}(-12) = -1$

- $w = [3,2] + [-2,-3] = [1,-1]$

$t=2, i=1$

- $[1,-1] \cdot [3,2] = 1$ ,  $\text{sign}(1) = 1$

$t=2, i=2$

- $[1,-1] \cdot [-2,2] = -4$ ,  $\text{sign}(-4) = -1$

$t=2, i=3$

- $[1,-1] \cdot [-2,-3] = 1$ ,  $\text{sign}(1) = 1$



Converged!!!

- $y = w_1 x_1 + w_2 x_2 \rightarrow y = x_1 - x_2$

- So, at  $y=0 \rightarrow x_2 = x_1$

# Multiclass Decision Rule

- If we have more than two classes:

- Have a weight vector for each class:  $w_y$
- Calculate an activation for each class

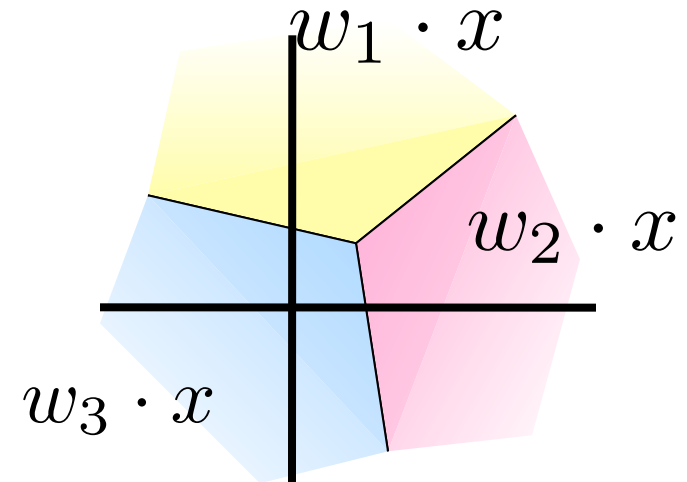
$$\text{activation}_w(x, y) = w_y \cdot x$$

- Highest activation wins

$$y^* = \arg \max_y (\text{activation}_w(x, y))$$

Example:  $y$  is  $\{1, 2, 3\}$

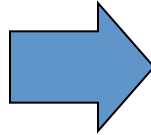
- We are fitting three planes:  $w_1, w_2, w_3$
- Predict  $i$  when  $w_i \cdot x$  is highest





# Example

“win the vote”

 $x$ 

BIAS	:	1
win	:	1
game	:	0
vote	:	1
the	:	1
...		

 $w_{SPORTS}$ 

BIAS	:	-2
win	:	4
game	:	4
vote	:	0
the	:	0
...		

 $w_{POLITICS}$ 

BIAS	:	1
win	:	2
game	:	0
vote	:	4
the	:	0
...		

 $w_{TECH}$ 

BIAS	:	2
win	:	0
game	:	2
vote	:	0
the	:	0
...		

$$x \cdot w_{SPORTS} = 2$$

$$x \cdot w_{POLITICS} = 7$$

$$x \cdot w_{TECH} = 2$$

**POLITICS wins!!!**

# The Multi-class Perceptron Alg.

- Start with zero weights
- For  $t=1..T$ ,  $i=1..n$  (T times over data)

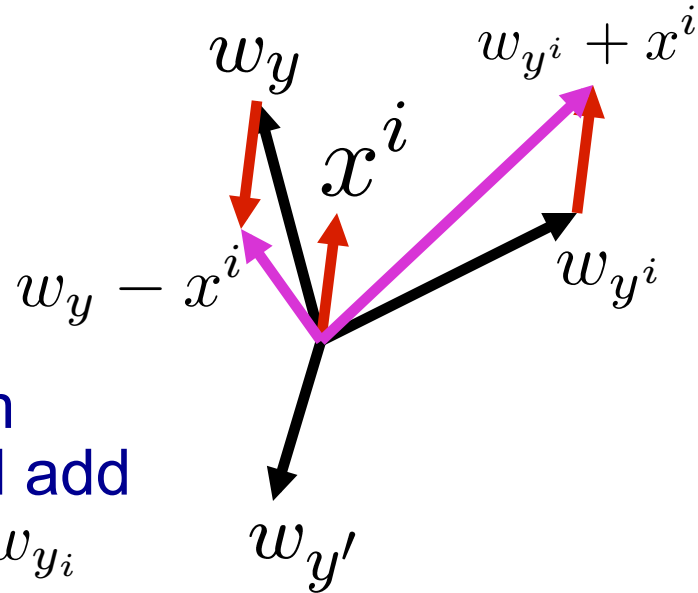
- Classify with current weights

$$y = \arg \max_y w_y \cdot x^i$$

- If correct ( $y=y_i$ ), no change!

- If wrong: subtract features  $x^i$  from weights for predicted class  $w_y$  and add them to weights for correct class  $w_{y_i}$

$$w_y = w_y - x^i$$
$$w_{y_i} = w_{y_i} + x^i$$



# Interpreting Perceptron Weights

$$f(x) = \text{sign}\left(\sum_d w_d x_d + b\right)$$

$$\frac{\partial}{\partial x_7} \left( \sum_d w_d x_d + b \right) = ?$$

# Interpreting Perceptron Weights

$$f(x) = \text{sign}\left(\sum_d w_d x_d + b\right)$$

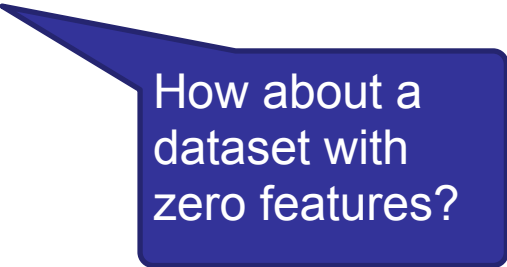
$$\frac{\partial}{\partial x_7} \left( \sum_d w_d x_d + b \right) = w_7$$

# Questions

- » Is the perceptron guaranteed to converge?
- » Under what circumstances?
- » What does it converge to?
- » How long does it take?

# Questions

- » Is the perceptron guaranteed to converge?
- » Under what circumstances?
- » What does it converge to?
- » How long does it take?



How about a dataset with zero features?

# Questions

- » Is the perceptron guaranteed to converge?
- » Under what circumstances?
- » What does it converge to?
- » How long does it take?

How about a dataset with zero features?

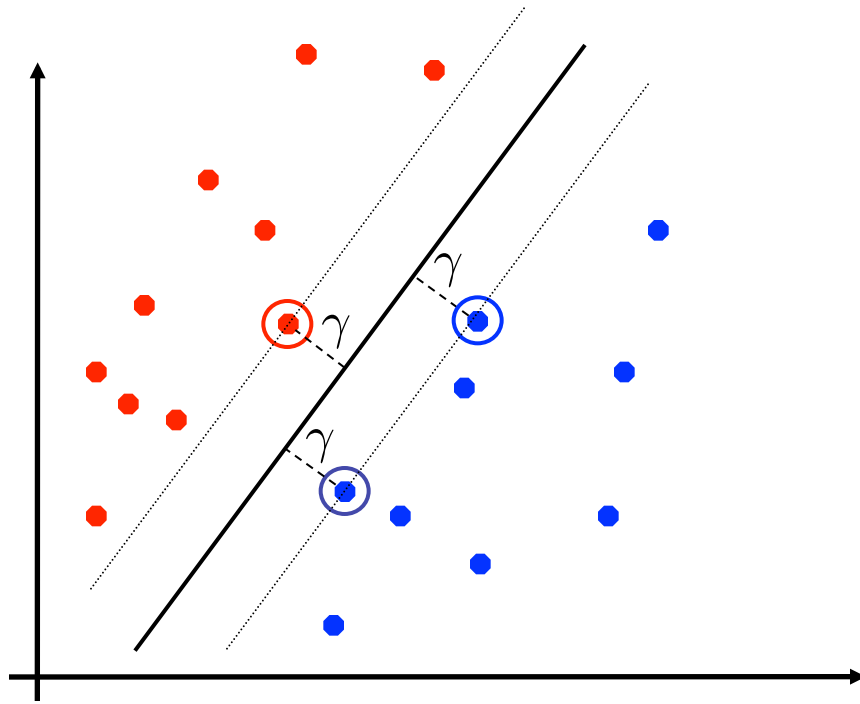
Linearly Separable -> Perceptron Guaranteed to Converge

Not Linearly Separable -> Never Converges

# Linearly Separable (binary case)

- The data is linearly separable with *margin*  $\gamma$ , if:

$$\exists w. \forall t. y^t (w \cdot x^t) \geq \gamma > 0$$



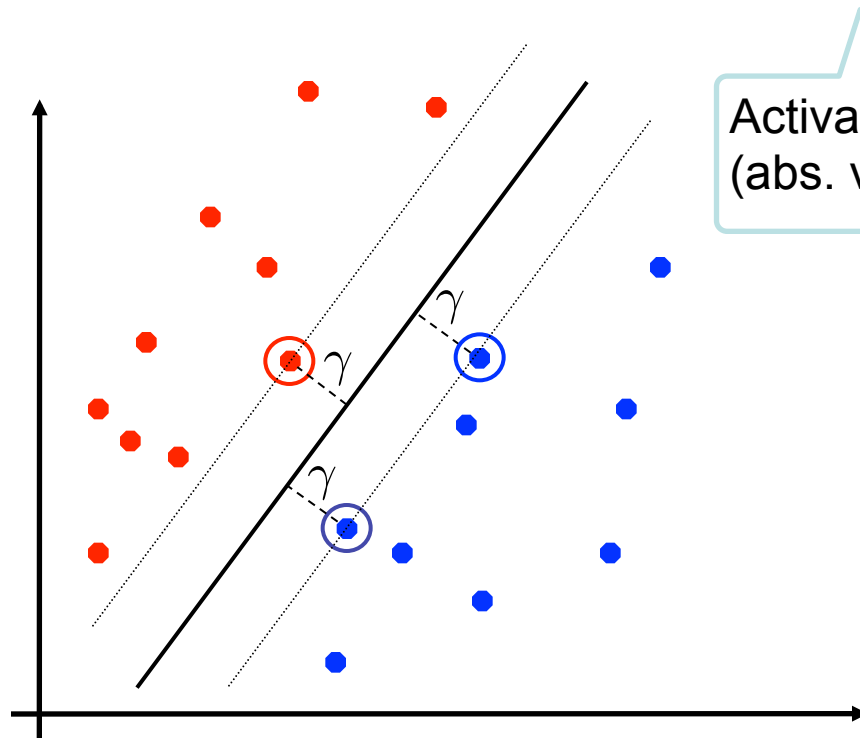
- For  $y^t = 1$   
 $w \cdot x^t \geq \gamma$
- For  $y^t = -1$   
 $w \cdot x^t \leq -\gamma$



# Linearly Separable (binary case)

- The data is linearly separable with *margin*  $\gamma$ , if:

$$\exists w. \forall t. y^t (w \cdot x^t) \geq \gamma > 0$$



Activation  
(abs. val.)

- For  $y^t = 1$   
 $w \cdot x^t \geq \gamma$
- For  $y^t = -1$   
 $w \cdot x^t \leq -\gamma$

# Mistake Bound for Perceptron

$$\|x\|_2 = \sqrt{\sum_i x_i^2}$$

- Assume data is separable with margin  $\gamma$ :

$$\exists w^* \text{ s.t. } \|w^*\|_2 = 1 \text{ and } \forall t. y^t (w^* \cdot x^t) \geq \gamma$$

- Also assume there is a number  $R$  such that:

$$\forall t. \|x^t\|_2 \leq R$$

- **Theorem:** The number of mistakes (parameter updates) made by the perceptron is bounded:

$$mistakes \leq \frac{R^2}{\gamma^2}$$

# Perceptron Convergence (by Induction)

- Let  $w^k$  be the weights after the  $k$ -th update (mistake), we will show that:

$$k^2 \gamma^2 \leq \|w^k\|_2^2 \leq k R^2$$

- Therefore:

$$k \leq \frac{R^2}{\gamma^2}$$

- Because  $R$  and  $\gamma$  are fixed constants that do not change as you learn, there are a finite number of updates!
- Proof does each bound separately (next two slides)

# Lower bound

Perceptron update:

$$w = w + y^t x^t$$

- Remember our margin assumption:

$$\exists w^* \text{ s.t. } \|w^*\|_2 = 1 \text{ and } \forall t. y^t (w^* \cdot x^t) \geq \gamma$$

- Now, by the definition of the perceptron update, for k-th mistake on t-th training example:

$$\begin{aligned} w^{k+1} \cdot w^* &= (w^k + y^t x^t) \cdot w^* \\ &= w^k \cdot w^* + y^t (w^* \cdot x^t) \\ &\geq w^k \cdot w^* + \gamma \end{aligned}$$

- So, by induction with  $w^0=0$ , for all k:

$$\begin{aligned} k\gamma &\leq w^k \cdot w^* \\ &\leq \|w^k\|_2 \\ k^2\gamma^2 &\leq \|w^k\|_2^2 \end{aligned}$$

Because:

$$\begin{aligned} w^k \cdot w^* &\leq \|w^k\|_2 \times \|w^*\|_2 \\ \text{and } \|w^*\|_2 &= 1 \end{aligned}$$

# Upper Bound

Perceptron update:

$$w = w + y^t x^t$$

Data Assumption:

$$\forall t. \|x^t\|_2 \leq R$$

- By the definition of the Perceptron update, for k-th mistake on t-th training example:

$$\|w^{k+1}\|_2^2 = \|w^k + y^t x^t\|_2^2$$

$$= \|w^k\|_2^2 + (y^t)^2 \|x^t\|_2^2 + 2y^t x^t \cdot w^k$$

$$\leq \|w^k\|_2^2 + R^2$$

$\leq R^2$  because  
 $(y^t)^2 = 1$  and  $\|x^t\|_2 \leq R$

- So, by induction with  $w_0=0$  have, for all k:

$$\|w_k\|_2^2 \leq kR^2$$

$< 0$  because Perceptron made error ( $y^t$  has different sign than  $x^t \cdot w^t$ )

# Perceptron Convergence (by Induction)

- Let  $w^k$  be the weights after the  $k$ -th update (mistake), we will show that:

$$k^2 \gamma^2 \leq \|w^k\|_2^2 \leq k R^2$$

- Therefore:

$$k \leq \frac{R^2}{\gamma^2}$$

- Because  $R$  and  $\gamma$  are fixed constants that do not change as you learn, there are a finite number of updates!
- If there is a linear separator, Perceptron will find it!!!
- Proof does each bound separately (last two slides)

# From Logistic Regression to the Perceptron: 2 easy steps!

Perceptron update when  $y$  is  $\{-1, 1\}$ :

$$w = w + y^j x^j$$

- **Logistic Regression:** (in vector notation):  $y$  is  $\{0, 1\}$

$$w = w + \eta \sum_j [y^j - P(y^j | x^j, w)] x^j$$

- **Perceptron:** when  $y$  is  $\{0, 1\}$ :

$$w = w + [y^j - \text{sign}^0(w \cdot x^j)] x^j$$

- $\text{sign}^0(x) = +1$  if  $x > 0$  and 0 otherwise

## Differences?

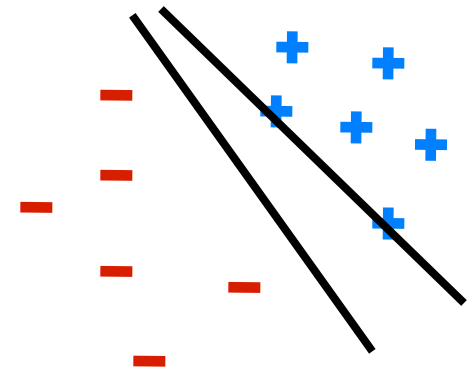
- Drop the  $\Sigma_j$  over training examples: **online vs. batch learning**
- Drop the dist'n: **probabilistic vs. error driven learning**

# Properties of Perceptrons

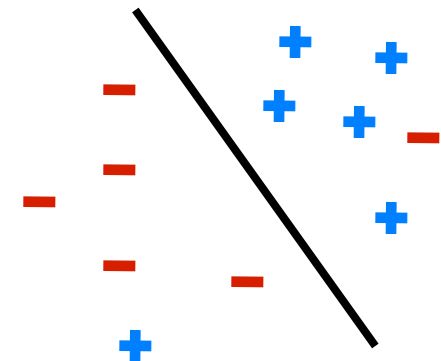
- **Separability:** some parameters get the training set perfectly correct
- **Convergence:** if the training is separable, perceptron will eventually converge (binary case)
- **Mistake Bound:** the maximum number of mistakes (binary case) related to the margin or degree of separability

$$\text{mistakes} \leq \frac{R^2}{\gamma^2}$$

Separable



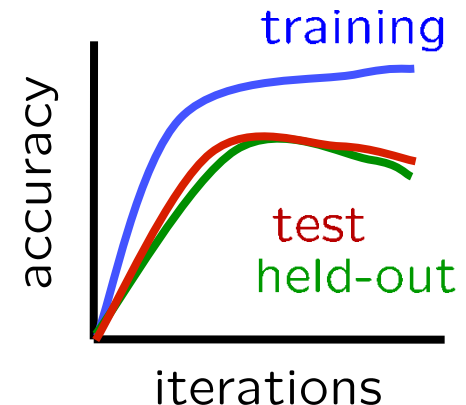
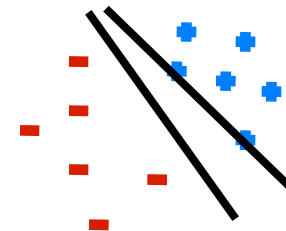
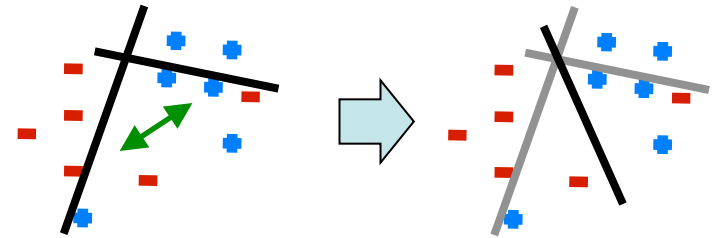
Non-Separable





# Problems with the Perceptron

- Noise: if the data isn't separable, weights might thrash
  - Averaging weight vectors over time can help (averaged perceptron)
- Mediocre generalization: finds a “barely” separating solution
- Overtraining: test / held-out accuracy usually rises, then falls
  - Overtraining is a kind of overfitting



# Voted Perceptron

## » Problem:

- » Later training examples are counted more than earlier ones

## » Example:

- » 10,000 training examples
- » After first 100, no no more updates until the 10,000th example
- » (could ruin the weight vector that did well on most of the data)

# Voted Perceptron

- » Would like weight vectors that “survive” to get more say
- » **Voting!**
- » As the perceptron runs, remember how long each hyperplane survives
- » Prediction:

$$\hat{y} = \text{sign} \left( \sum_{k=1}^K c^{(k)} \text{sign}(w^{(k)} \cdot \hat{x} + b^{(k)}) \right)$$

# Voted Perceptron

- » Would like weight vectors that “survive” to get more say
- » **Voting!**
- » As the perceptron runs, remember how long each hyperplane survives
- » Prediction:

Too slow /  
too much  
memory!

$$\hat{y} = \text{sign} \left( \sum_{k=1}^K c^{(k)} \text{sign}(w^{(k)} \cdot \hat{x} + b^{(k)}) \right)$$

# Averaged Perceptron (much more practical)

$$\hat{y} = \text{sign} \left( \sum_{k=1}^K c^{(k)} (w^{(k)} \cdot \hat{x} + b^{(k)}) \right)$$

$$\hat{y} = \text{sign} \left( \left( \sum_{k=1}^K c^{(k)} w^{(k)} \right) \cdot \hat{x} + \sum_{k=1}^K c^{(k)} b^{(k)} \right)$$

# Averaged Perceptron (much more practical)

---

**Algorithm 7** AVERAGEDPERCEPTRONTRAIN( $\mathbf{D}$ ,  $MaxIter$ )

---

```
1:  $\mathbf{w} \leftarrow \langle 0, 0, \dots, 0 \rangle$  ,  $b \leftarrow 0$  // initialize weights and bias
2:  $\mathbf{u} \leftarrow \langle 0, 0, \dots, 0 \rangle$  ,  $\beta \leftarrow 0$  // initialize cached weights and bias
3:  $c \leftarrow 1$  // initialize example counter to one
4: for  $iter = 1 \dots MaxIter$  do
5:   for all  $(x, y) \in \mathbf{D}$  do
6:     if  $y(\mathbf{w} \cdot \mathbf{x} + b) \leq 0$  then
7:        $\mathbf{w} \leftarrow \mathbf{w} + y \mathbf{x}$  // update weights
8:        $b \leftarrow b + y$  // update bias
9:        $\mathbf{u} \leftarrow \mathbf{u} + y c \mathbf{x}$  // update cached weights
10:       $\beta \leftarrow \beta + y c$  // update cached bias
11:     end if
12:      $c \leftarrow c + 1$  // increment counter regardless of update
13:   end for
14: end for
15: return  $\mathbf{w} - \frac{1}{c} \mathbf{u}$ ,  $b - \frac{1}{c} \beta$  // return averaged weights and bias
```

---

# Averaged Perceptron (much more practical)

---

**Algorithm 7** AVERAGEDPERCEPTRONTRAIN( $\mathbf{D}$ ,  $MaxIter$ )

---

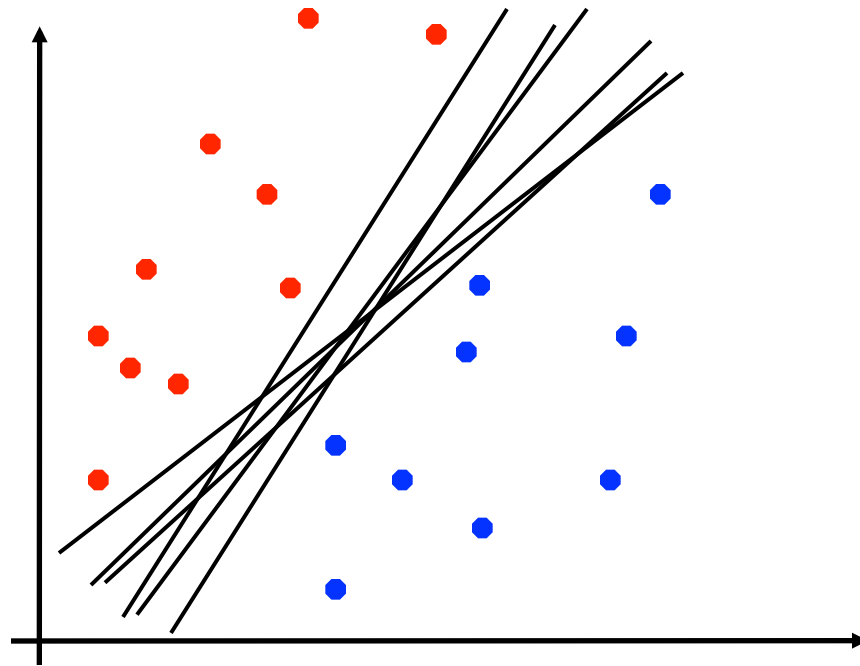
```
1:  $\mathbf{w} \leftarrow \langle 0, 0, \dots, 0 \rangle$  ,  $b \leftarrow 0$  // initialize weights and bias
2:  $\mathbf{u} \leftarrow \langle 0, 0, \dots, 0 \rangle$  ,  $\beta \leftarrow 0$  // initialize cached weights and bias
3:  $c \leftarrow 1$  // initialize example counter to one
4: for  $iter = 1 \dots MaxIter$  do
5:   for all  $(x, y) \in \mathbf{D}$  do
6:     if  $y(\mathbf{w} \cdot \mathbf{x} + b) \leq 0$  then
7:        $\mathbf{w} \leftarrow \mathbf{w} + y \mathbf{x}$  // update weights
8:        $b \leftarrow b + y$  // update bias
9:        $\mathbf{u} \leftarrow \mathbf{u} + y c \mathbf{x}$  // update cached weights
10:       $\beta \leftarrow \beta + y c$  // update cached bias
11:     end if
12:      $c \leftarrow c + 1$  // increment counter regardless of update
13:   end for
14: end for
15: return  $\mathbf{w} - \frac{1}{c} \mathbf{u}$ ,  $b - \frac{1}{c} \beta$  // return averaged weights and bias
```

---

Efficiency  
Trick

# Linear Separators

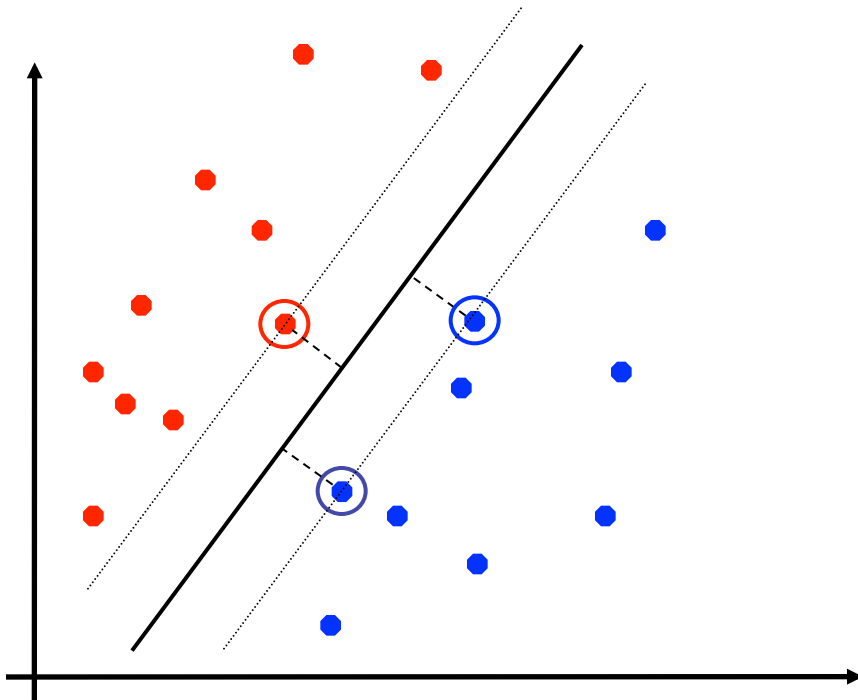
- Which of these linear separators is optimal?





# Support Vector Machines

- **Maximizing the margin:** good according to intuition, theory, practice
- Support vector machines (SVMs) find the separator with max margin

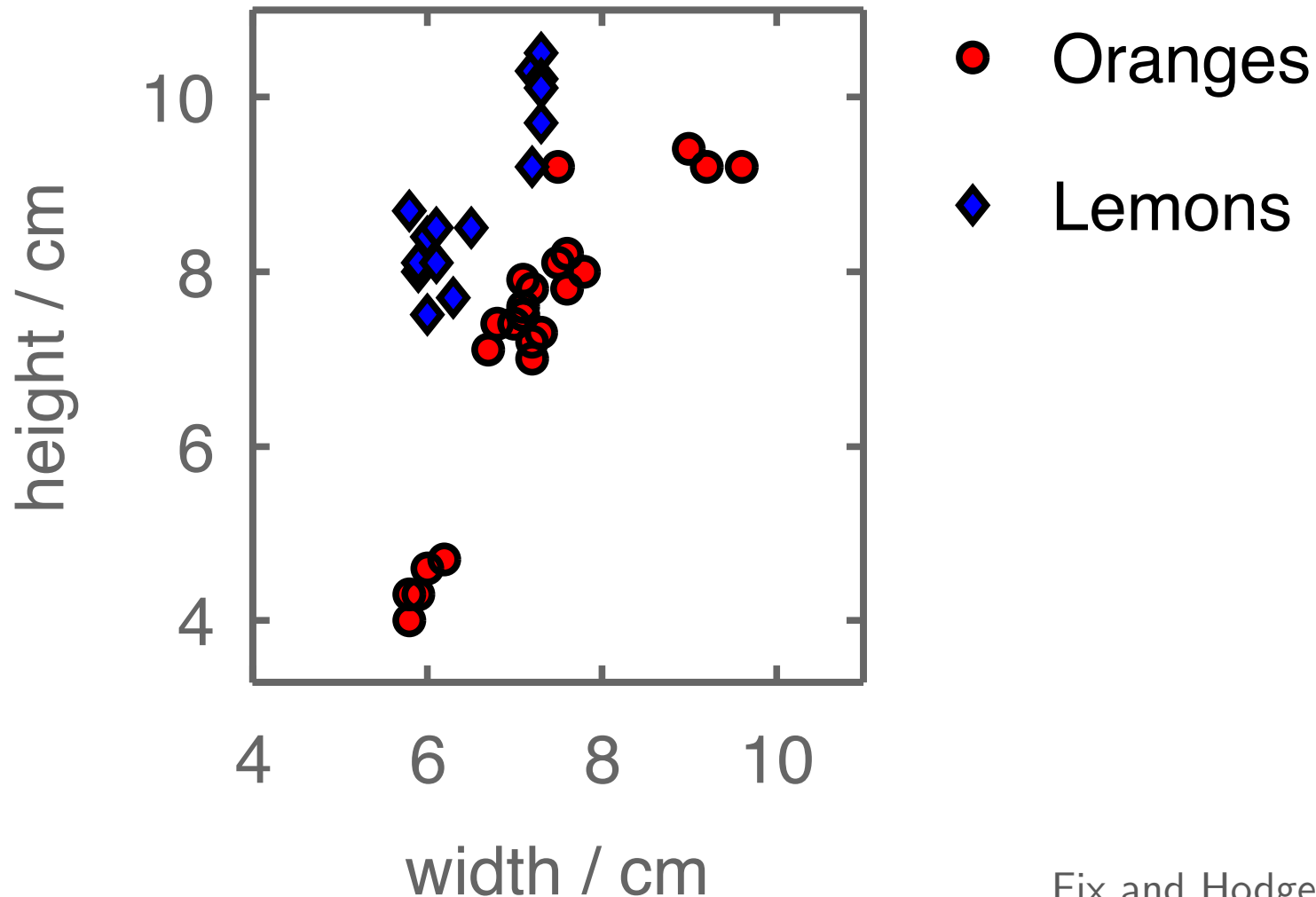


SVM

$$\min_w \frac{1}{2} \|w\|^2$$
$$\forall i, y \quad w_{y^*} \cdot x^i \geq w_y \cdot x^i + 1$$

# K-Nearest neighbours classifier

---



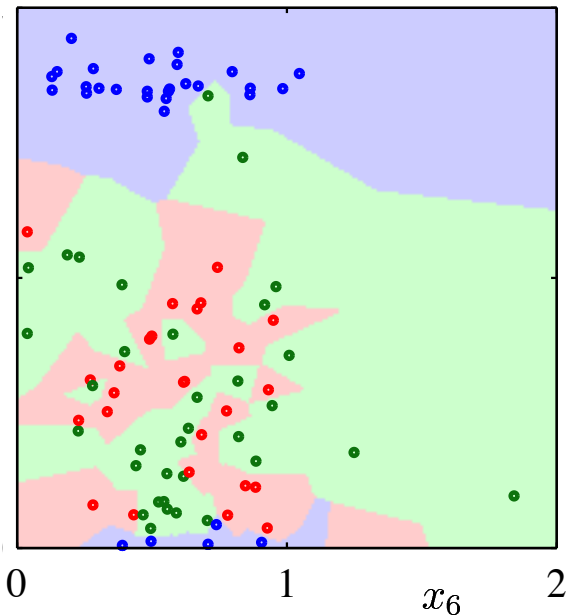
Fix and Hodges (1951)

# Decision boundaries

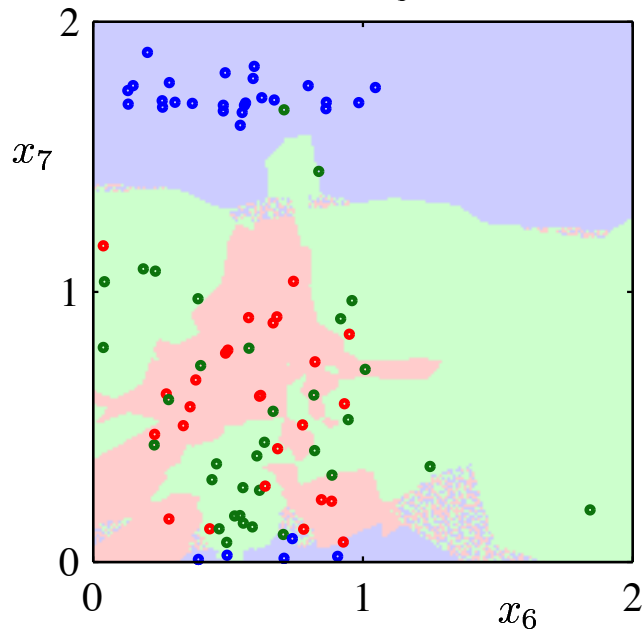
---

Classifying oil flow

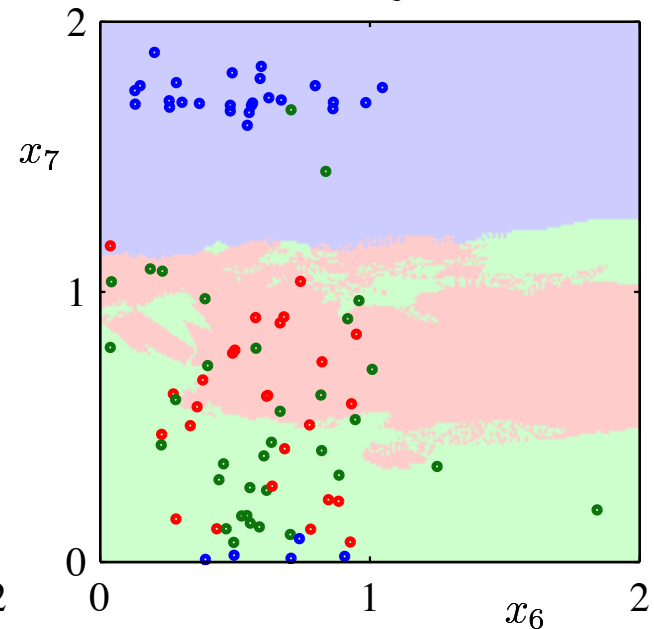
$K = 1$



$K = 3$



$K = 31$



# Parametric vs. non-parametric

Started assuming decision boundary is a plane

Non-parametric KNN has no fixed assumption:  
boundary gets more complicated with more data

Non-parametric methods may need more data  
and can be computationally intensive

# Nonlinear basis functions

Create new feature vector:  $\mathbf{x}' = \begin{bmatrix} \phi_1(\mathbf{x}) \\ \phi_2(\mathbf{x}) \\ 1 \end{bmatrix}$

