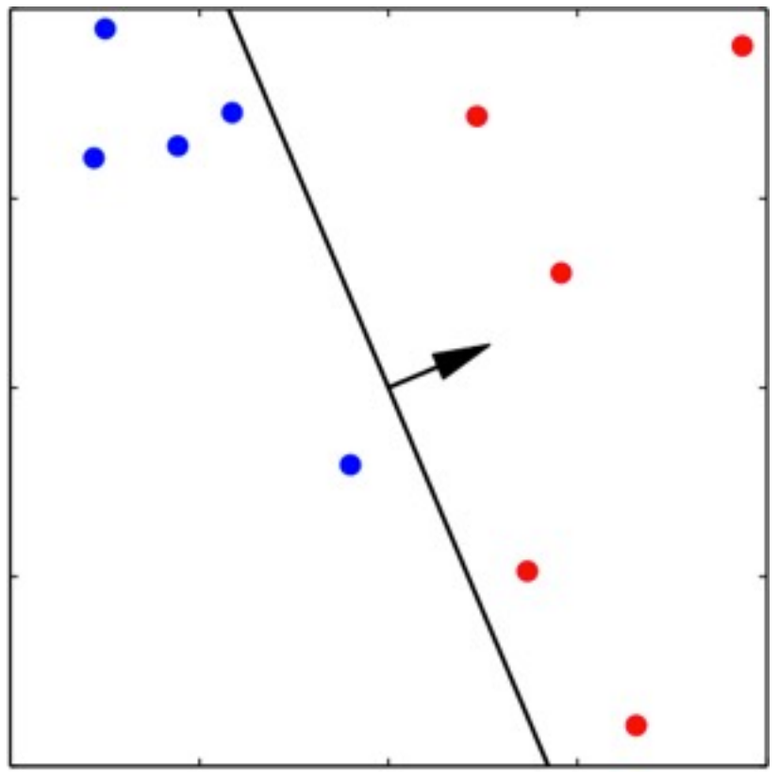
Machine Learning

A Geometric Approach



Linear Classification: Perceptron

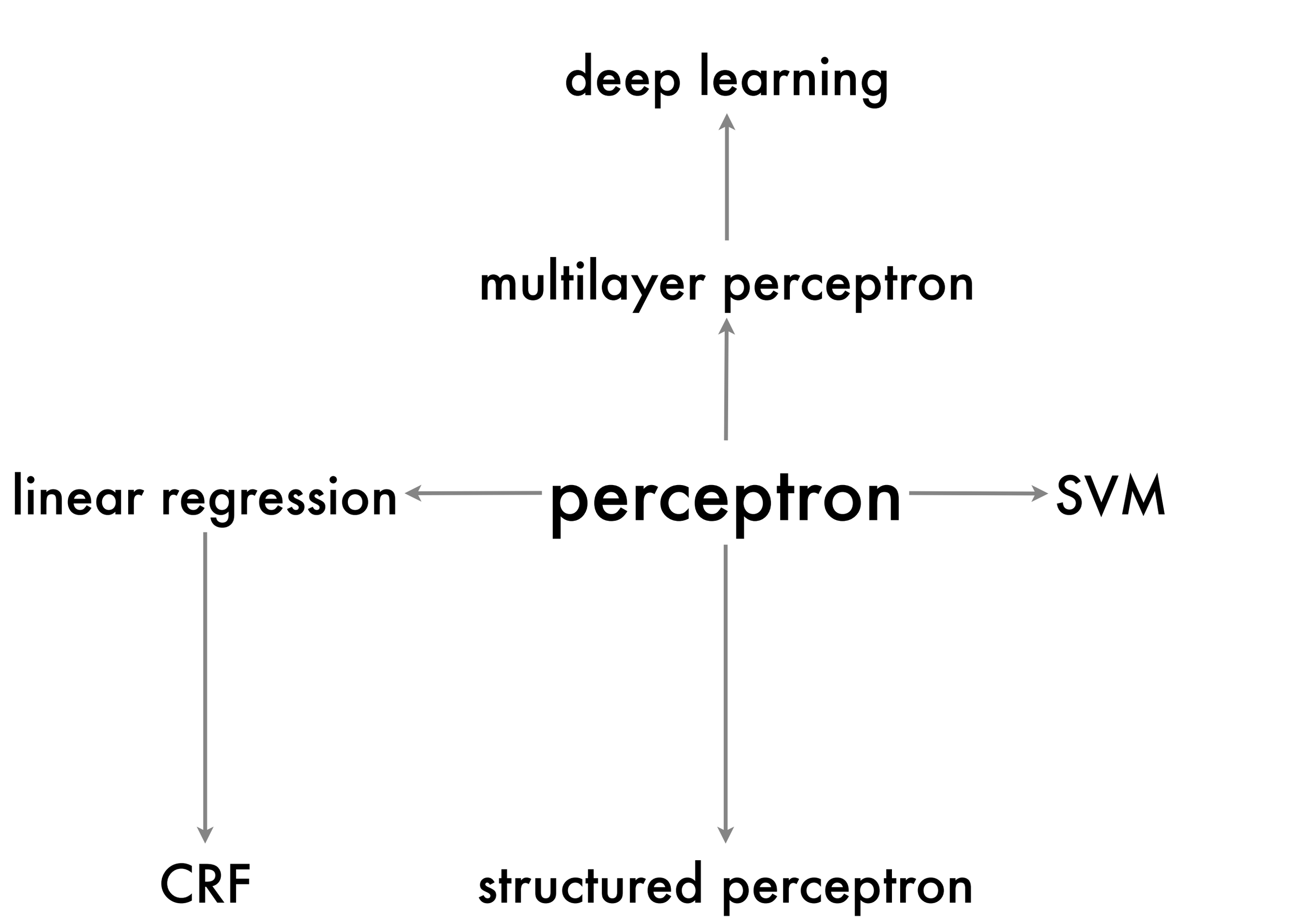
Professor Liang Huang

some slides from Alex Smola (CMU)

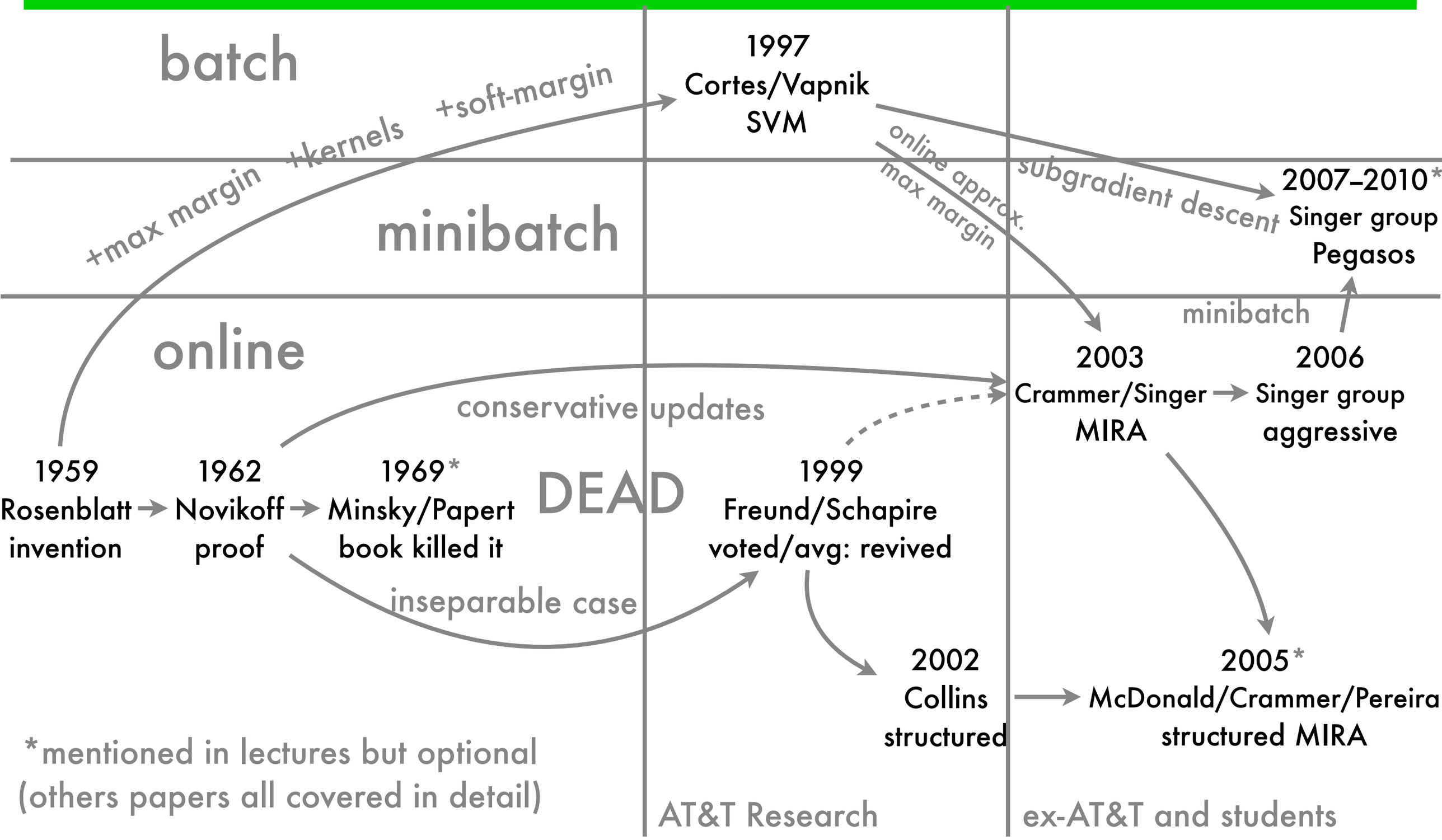
MAGIC Etch A Sketch® SCREEN

Perceptron

Frank Rosenblatt

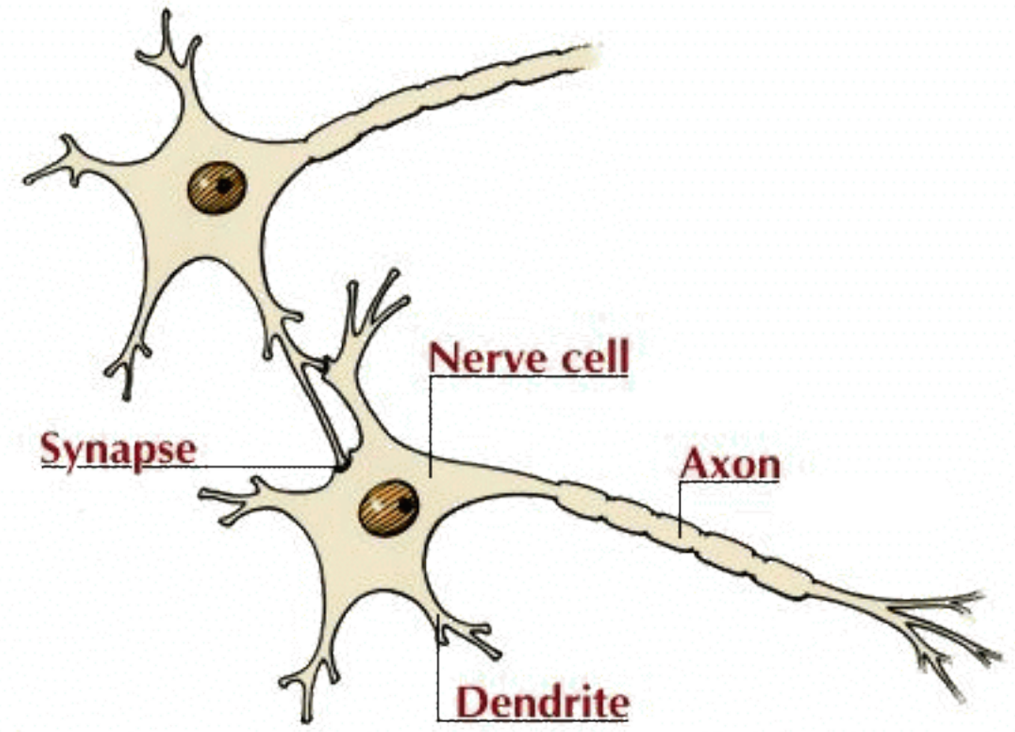

deep learning
multilayer perceptron
linear regression
perceptron
SVM
CRF
structured perceptron

Brief History of Perceptron

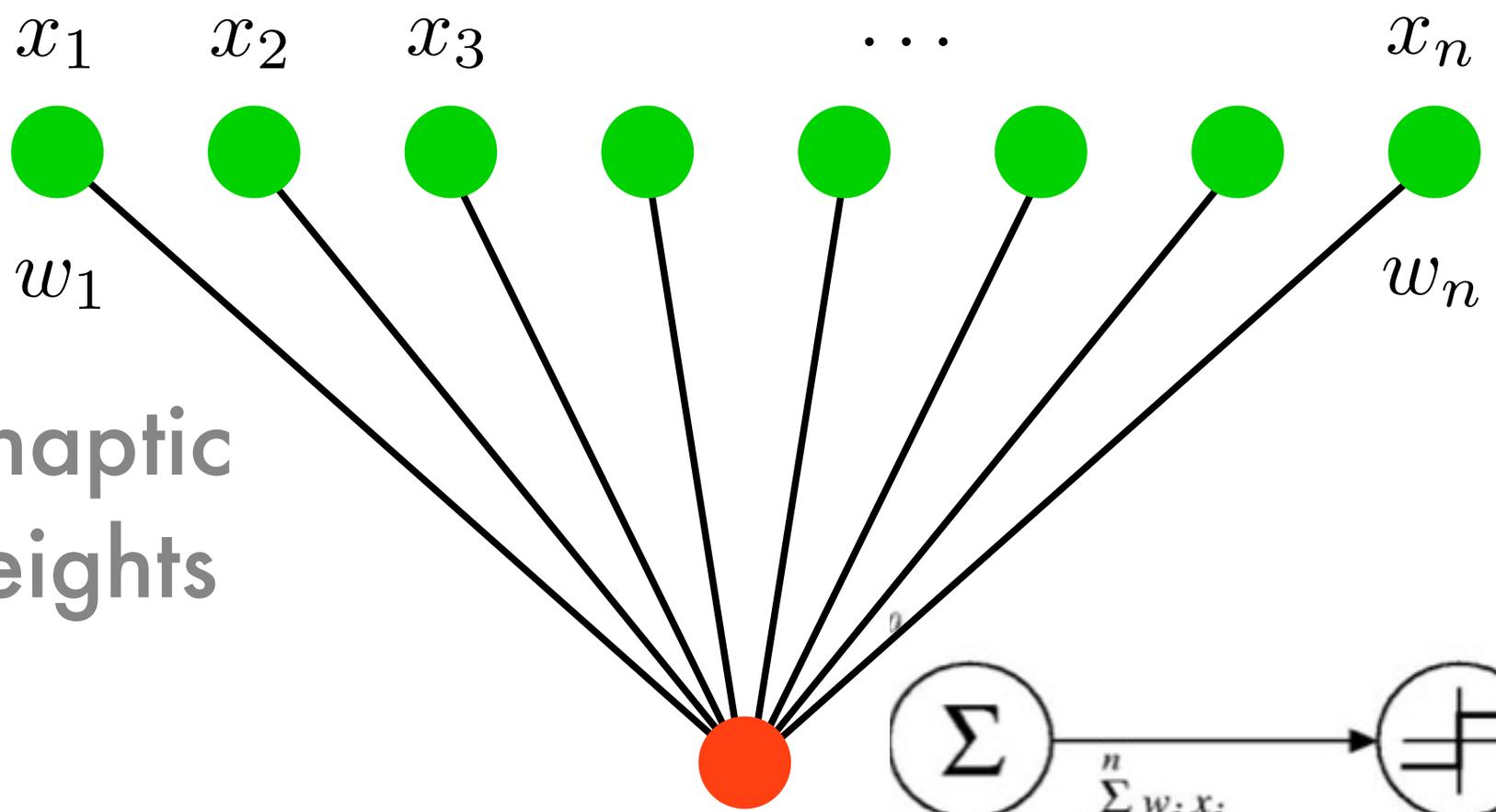


Neurons

- Soma (CPU)
  Cell body - combines signals
- Dendrite (input bus)
  Combines the inputs from several other nerve cells
- Synapse (interface)
  Interface and parameter store between neurons
- Axon (output cable)
  May be up to 1m long and will transport the activation signal to neurons at different locations

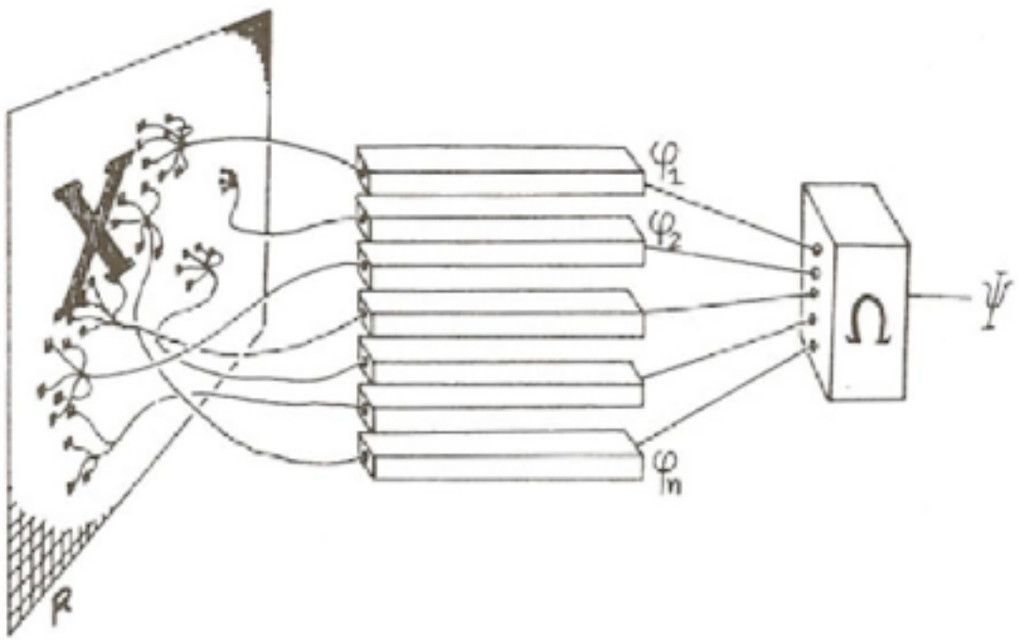


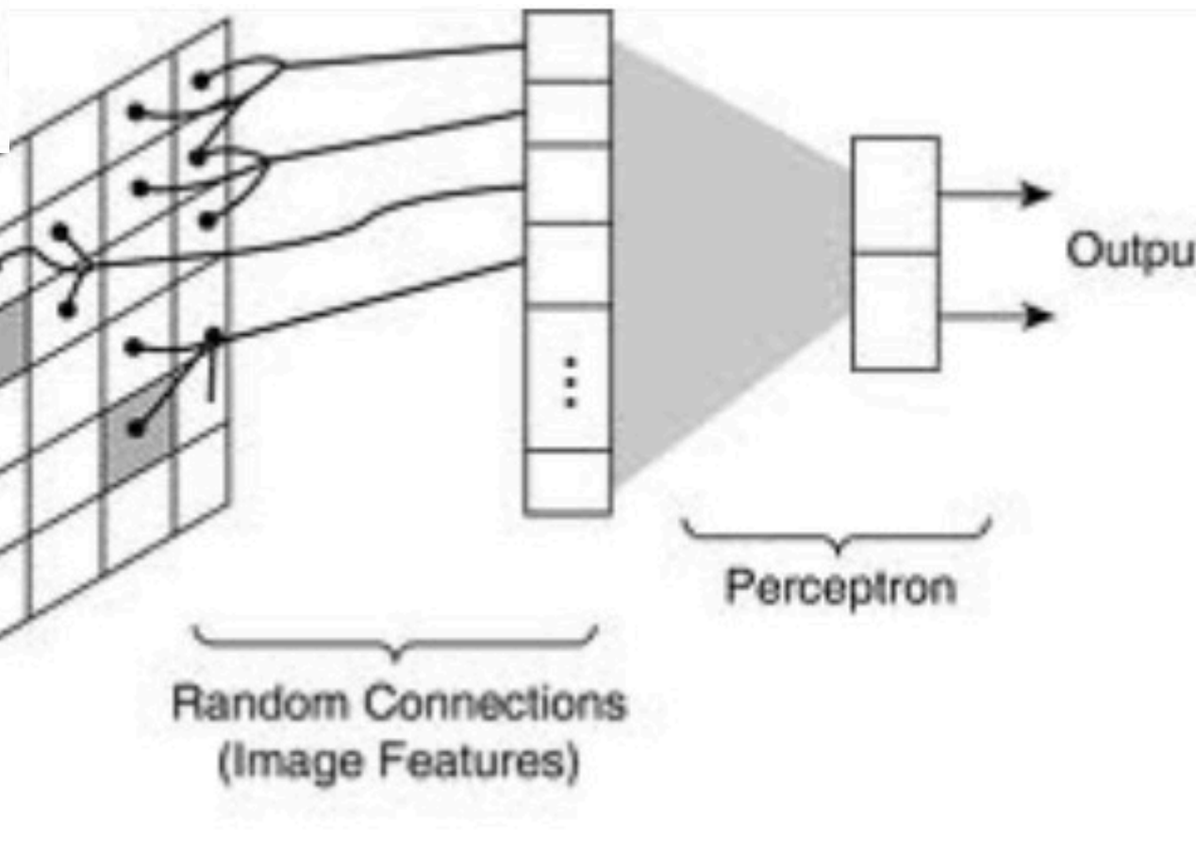
Neurons



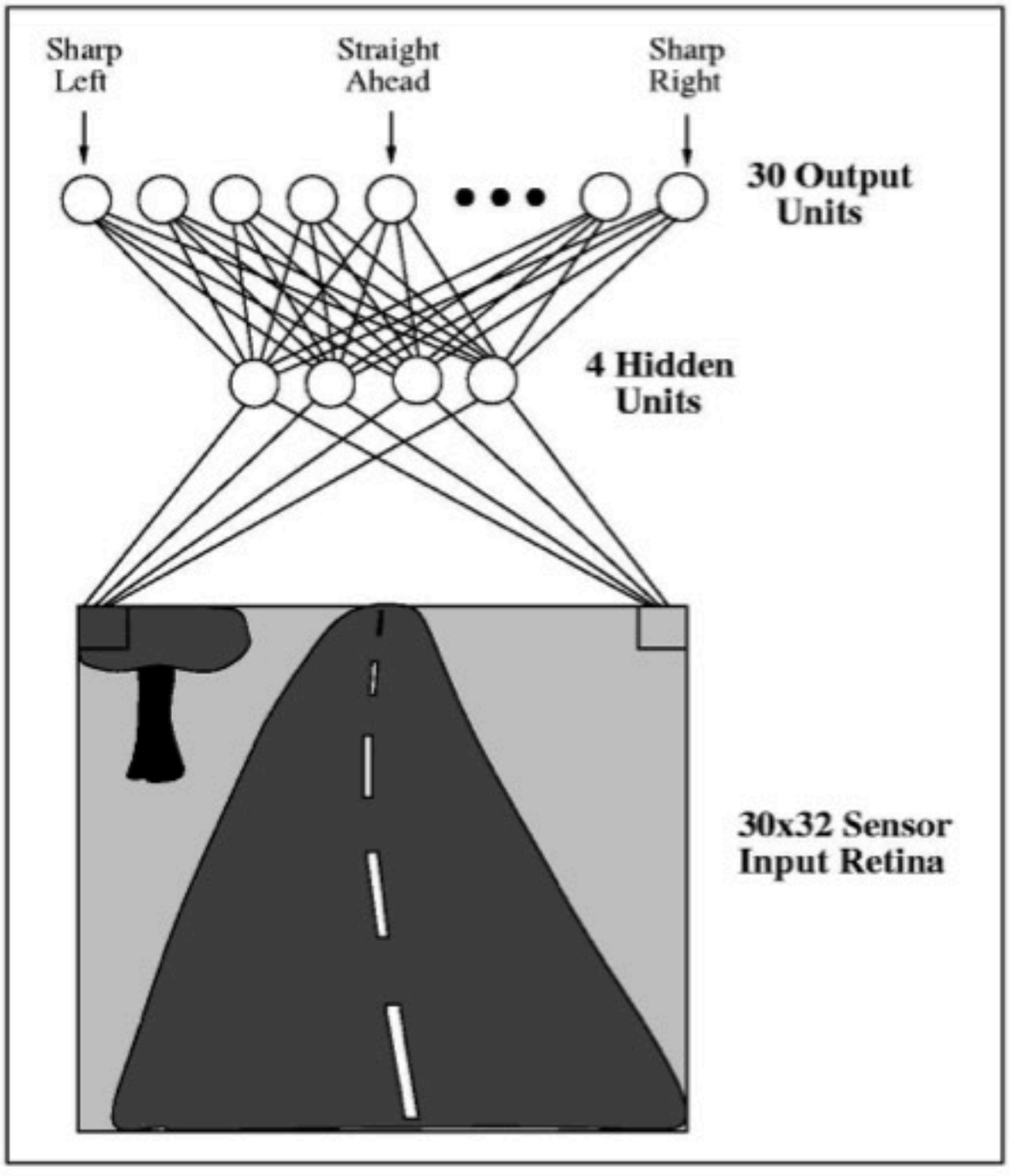
$$f(x) = \sigma\left(\sum_i w_i x_i = \langle w, x \rangle\right)$$

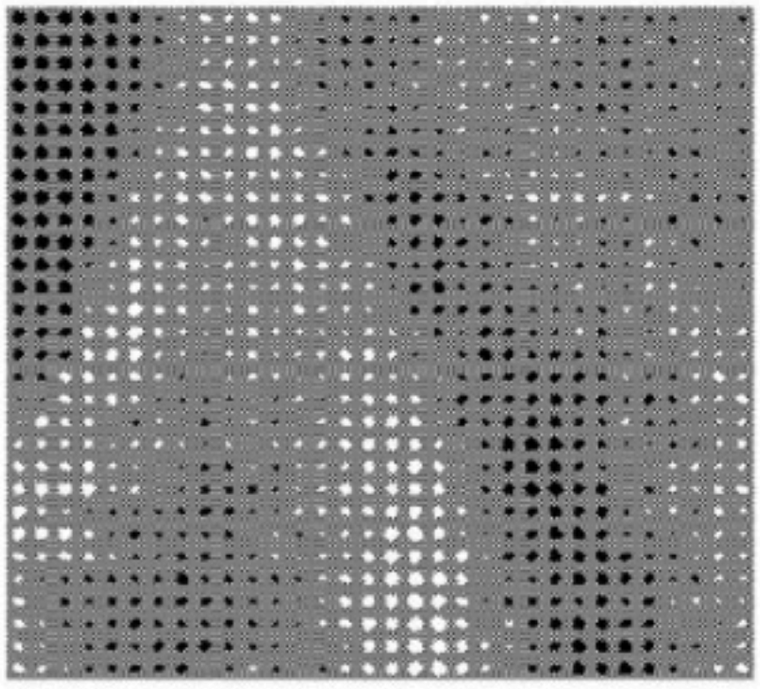
Frank Rosenblatt's Perceptron





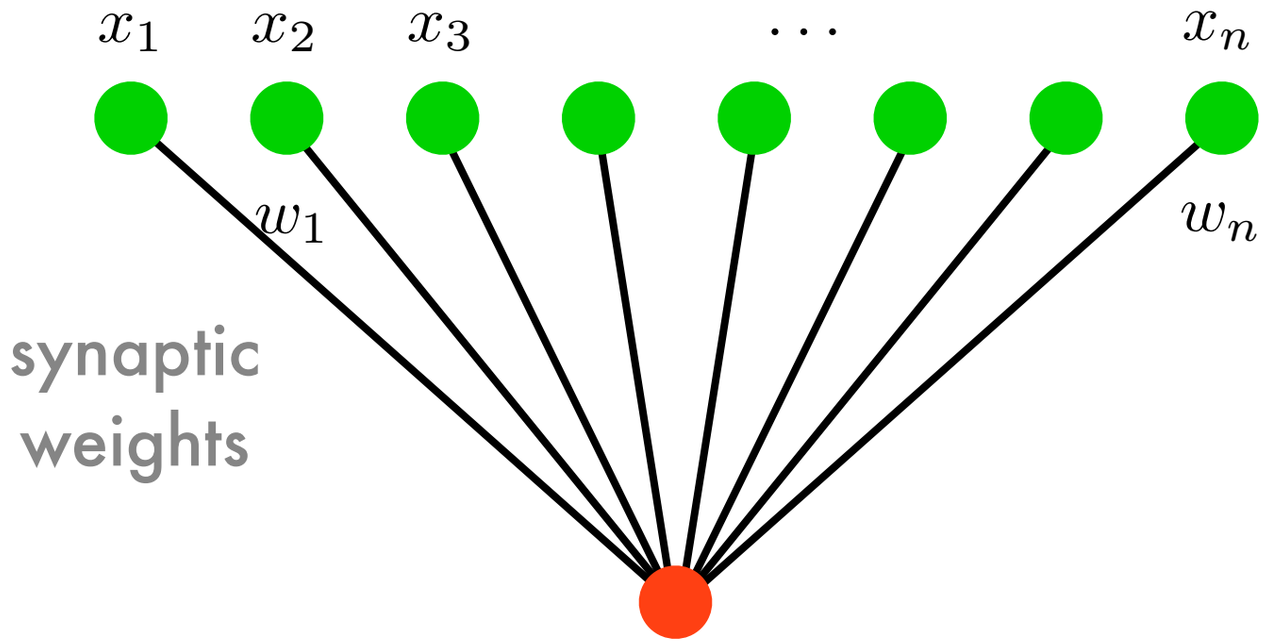
Multilayer Perceptron (Neural Net)





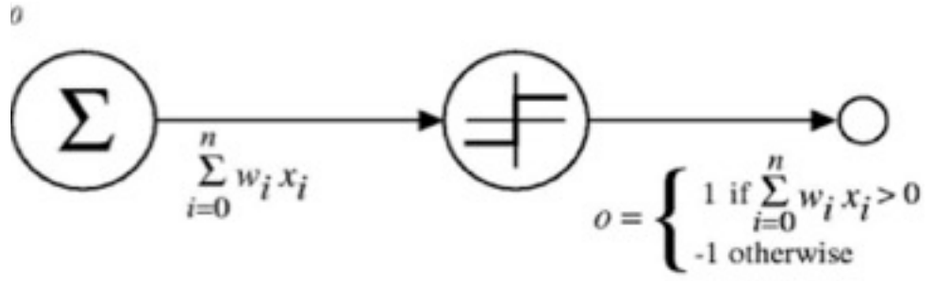
Perceptron w/ bias

- Weighted linear combination
- Nonlinear decision function
- Linear offset (bias)



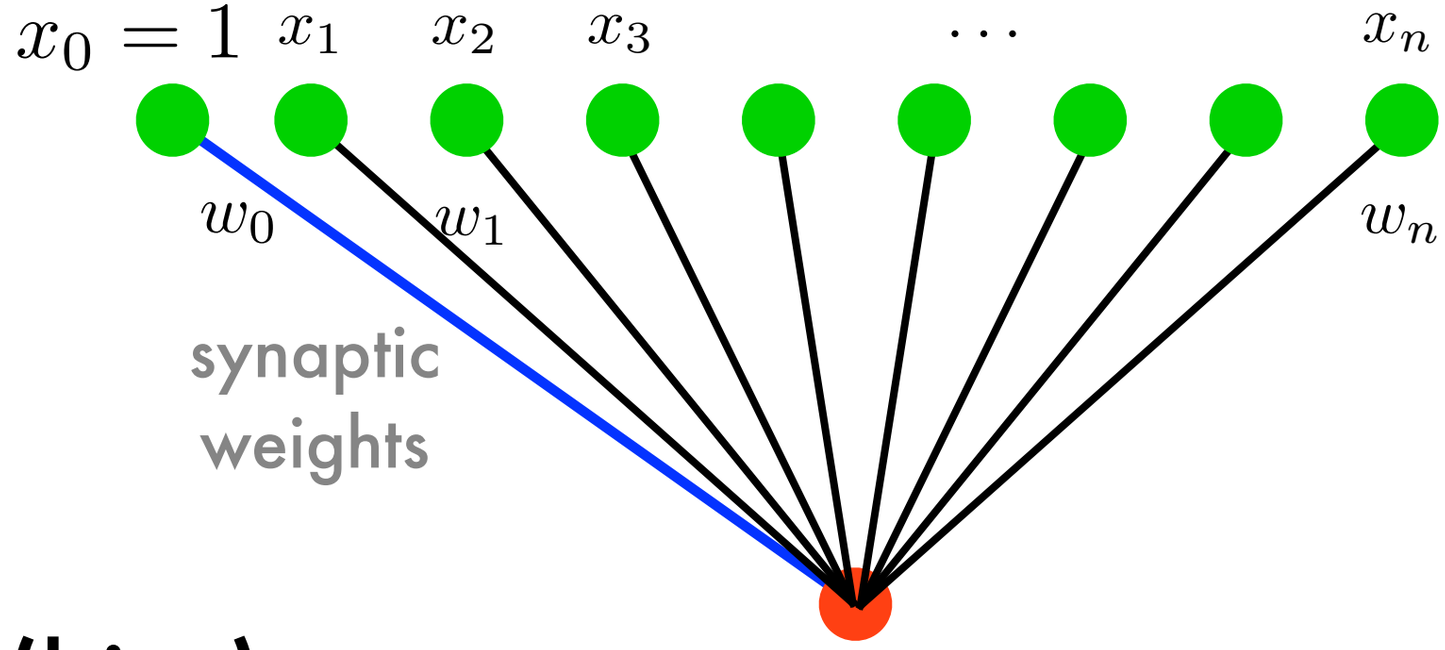
$$f(x) = \sigma\left(\langle w, x \rangle + b\right)$$

- Linear separating hyperplanes
- Learning: w and b

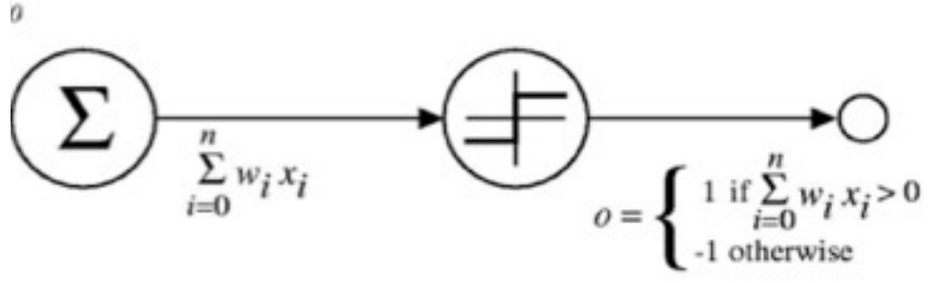


Perceptron w/o bias

- Weighted linear combination
- Nonlinear decision function
- No Linear offset (bias): hyperplane through the origin
- Linear separating hyperplanes
- Learning: w

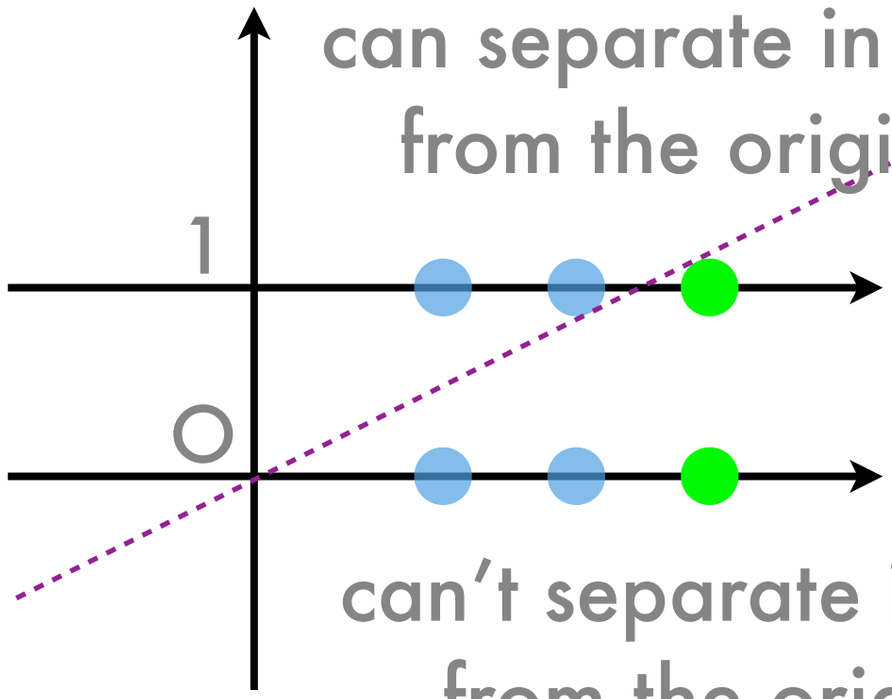


$$f(x) = \sigma\left(\langle w, x\rangle + b\right)$$



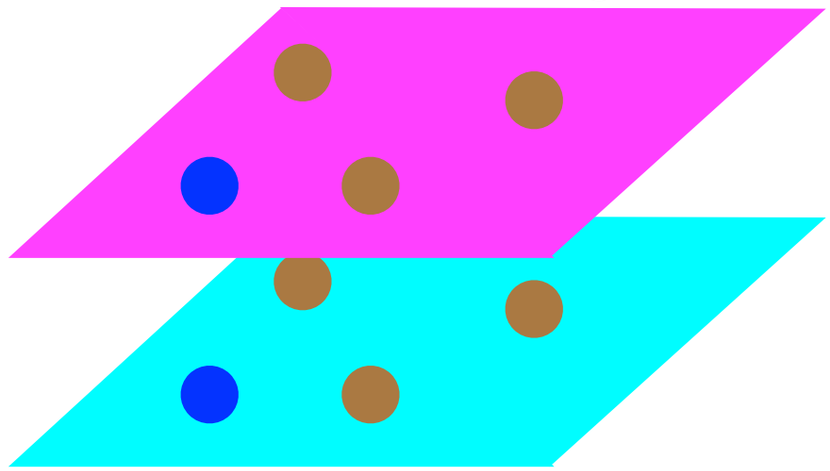
Augmented Space

can separate in 2D
from the origin



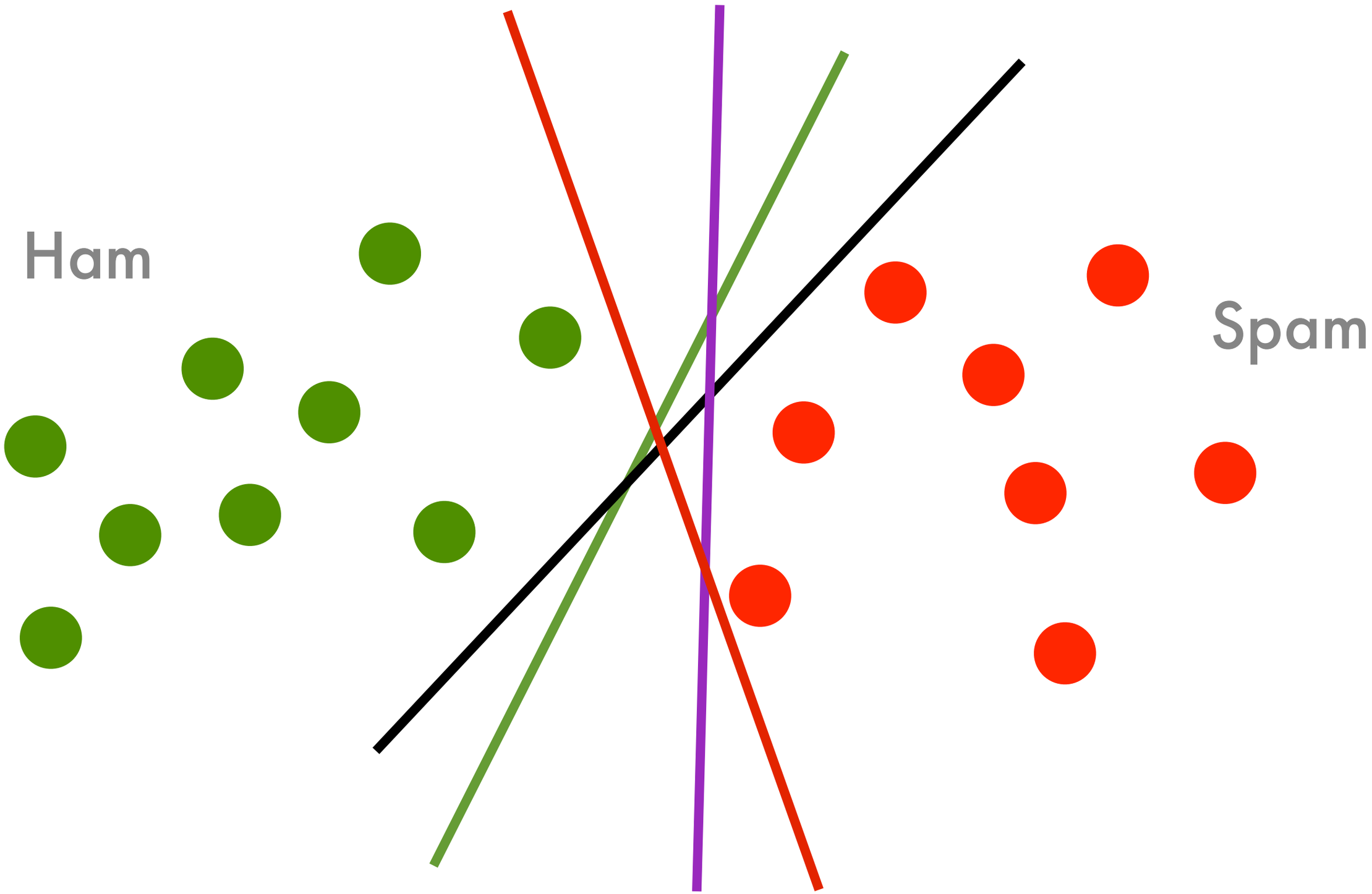
can’t separate in 1D
from the origin

can separate in 3D
from the origin



can’t separate in 2D
from the origin

Perceptron



The Perceptron w/o bias

initialize $w = 0$

repeat

  if $y_i \left[\langle w, x_i \rangle \right] \leq 0$ **then**

    $w \leftarrow w + y_i x_i$

  end if

until all classified correctly

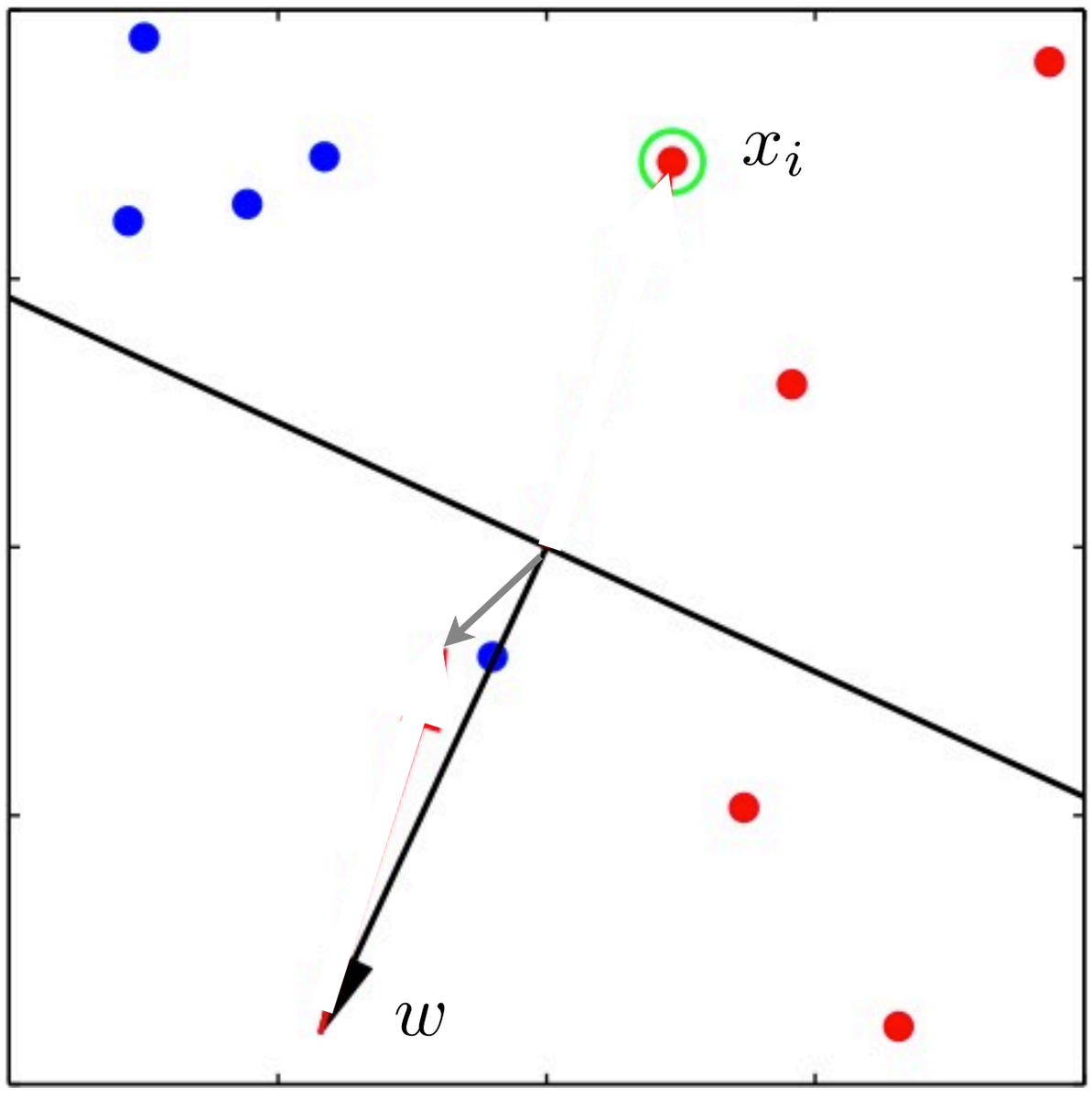
- Nothing happens if classified correctly
- Weight vector is linear combination $w = \sum_{i \in I} y_i x_i$
- Classifier is linear combination of inner products $f(x) = \left(\sum_{i \in I} y_i \langle x_i, x \rangle \right)$

The Perceptron w/ bias

initialize $w = 0$ and $b = 0$
repeat
  if $y_i \left[\langle w, x_i \rangle + b\right] \leq 0$ **then**
    $w \leftarrow w + y_i x_i$ and $b \leftarrow b + y_i$
  end if
until all classified correctly

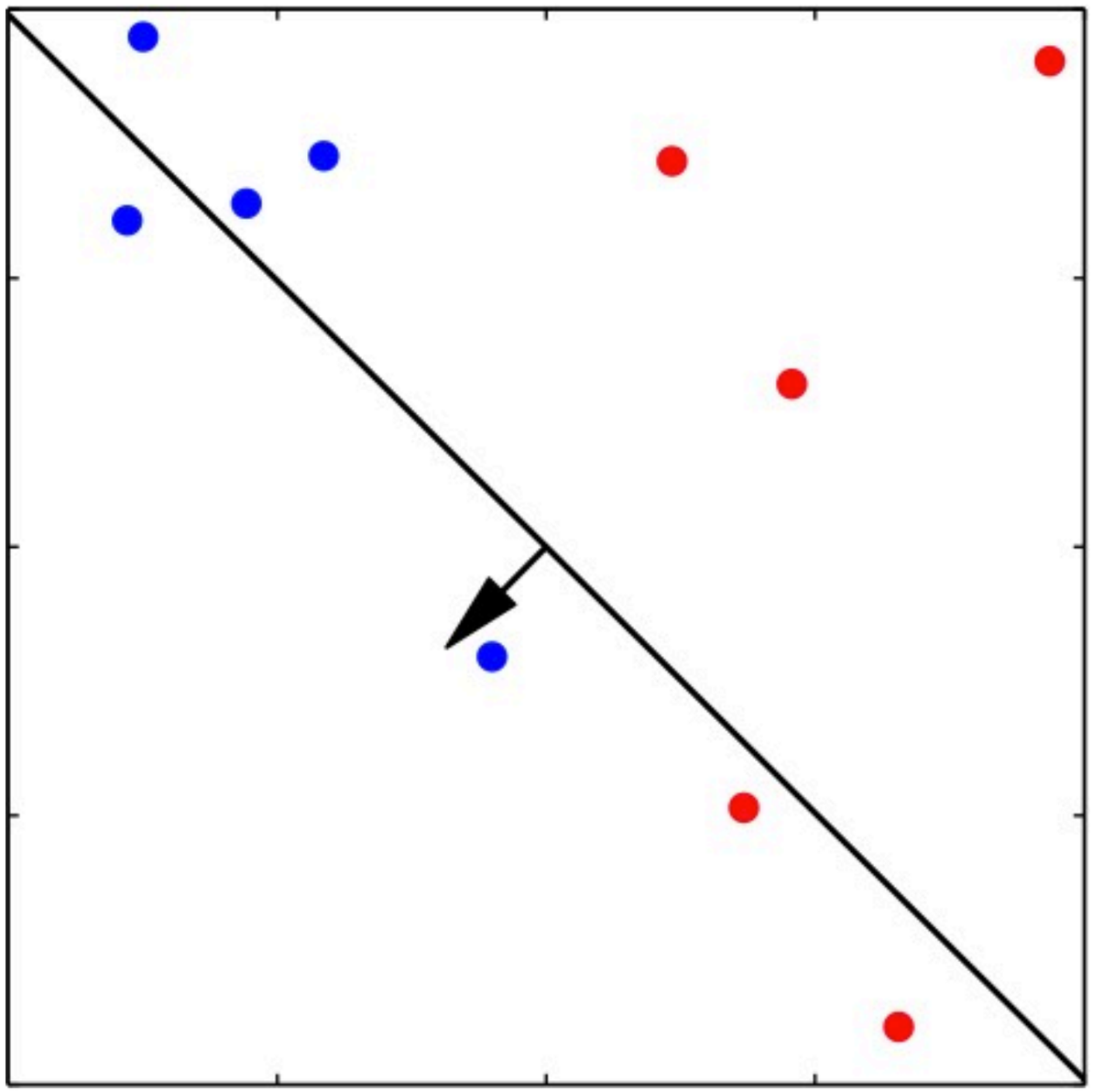
- Nothing happens if classified correctly
- Weight vector is linear combination $w = \sum_{i \in I} y_i x_i$
- Classifier is linear combination of inner products $f(x) = \sigma\left(\sum_{i \in I} y_i \langle x_i, x \rangle + b\right)$

Demo

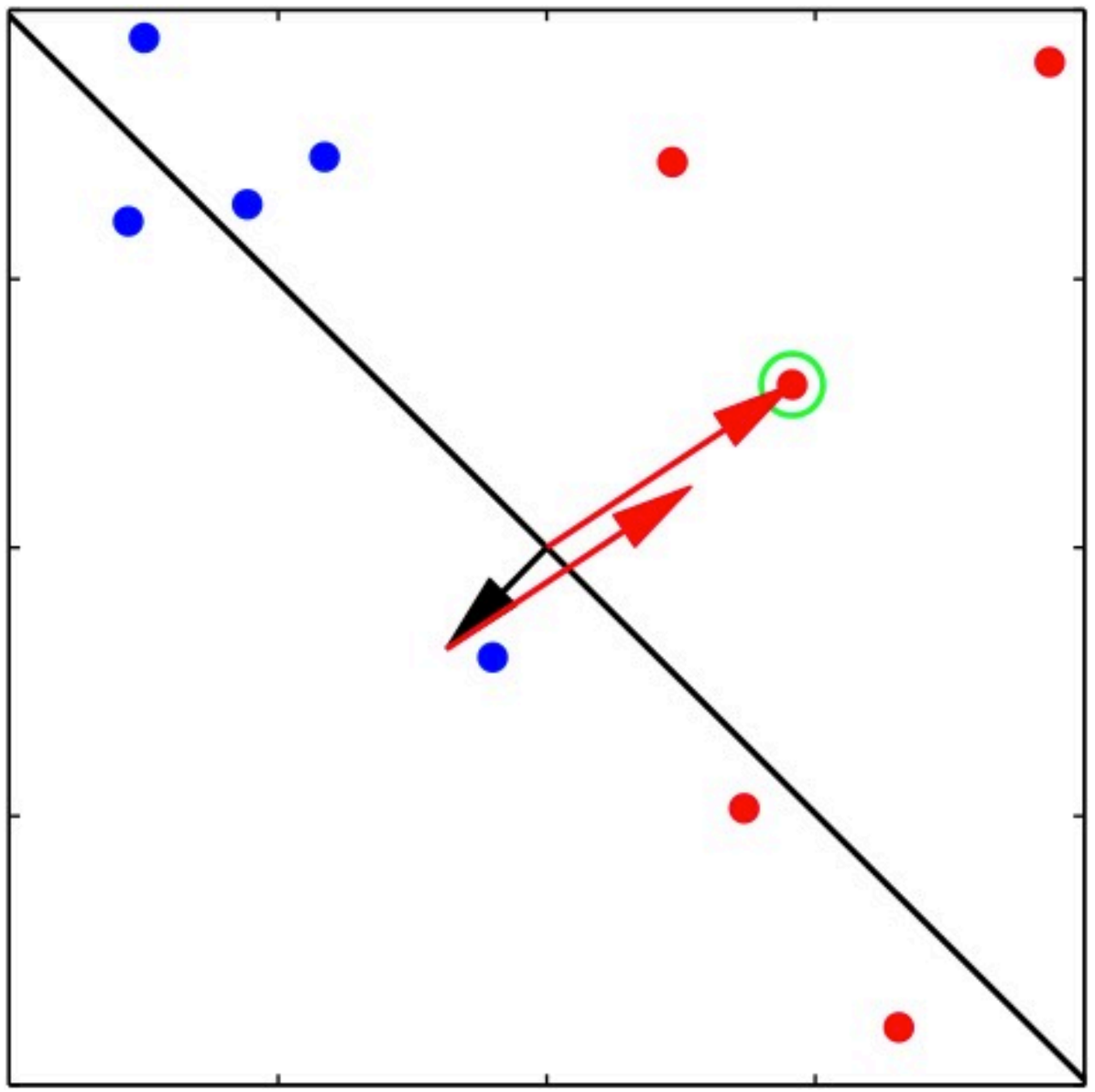


(bias=0)

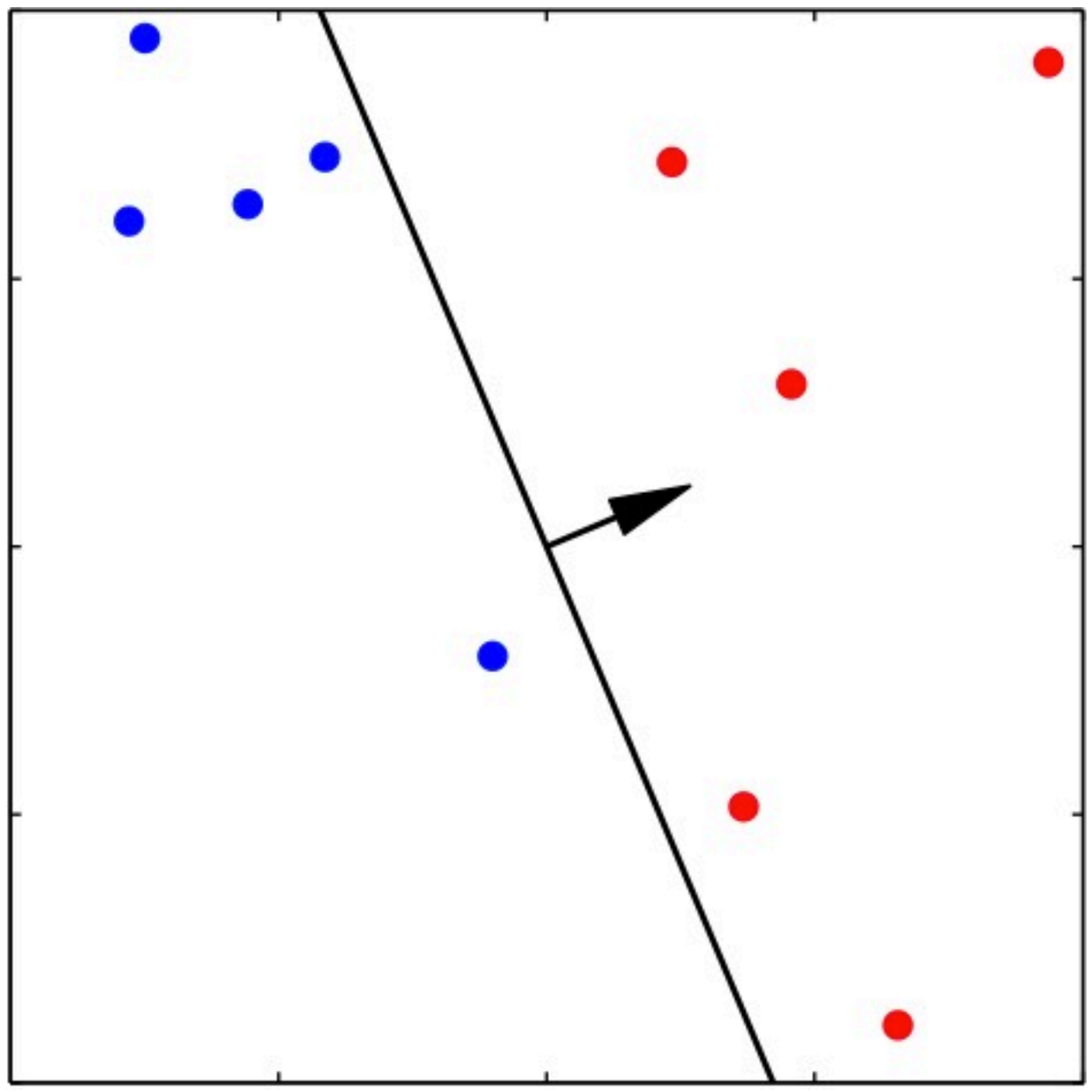
Demo

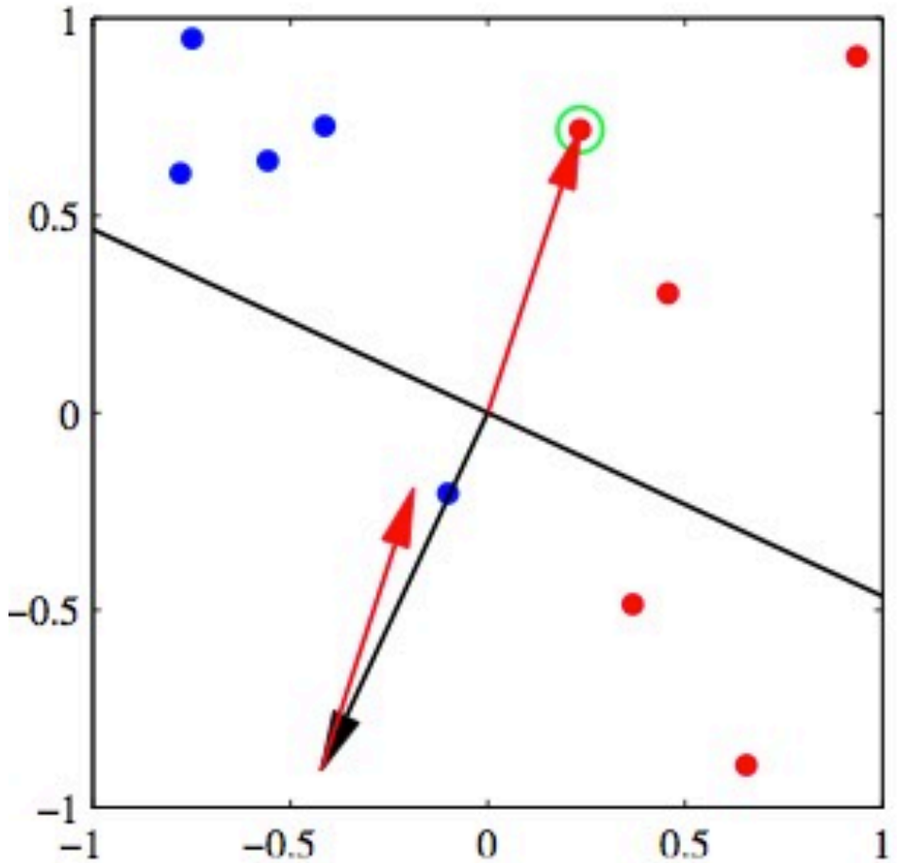


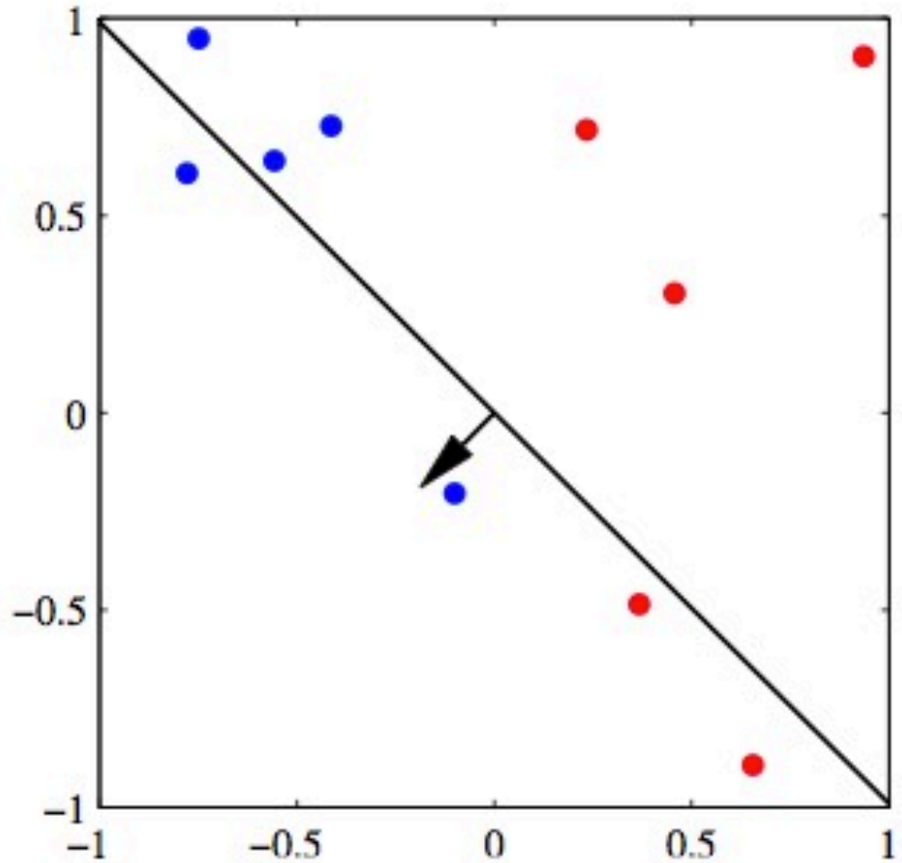
Demo

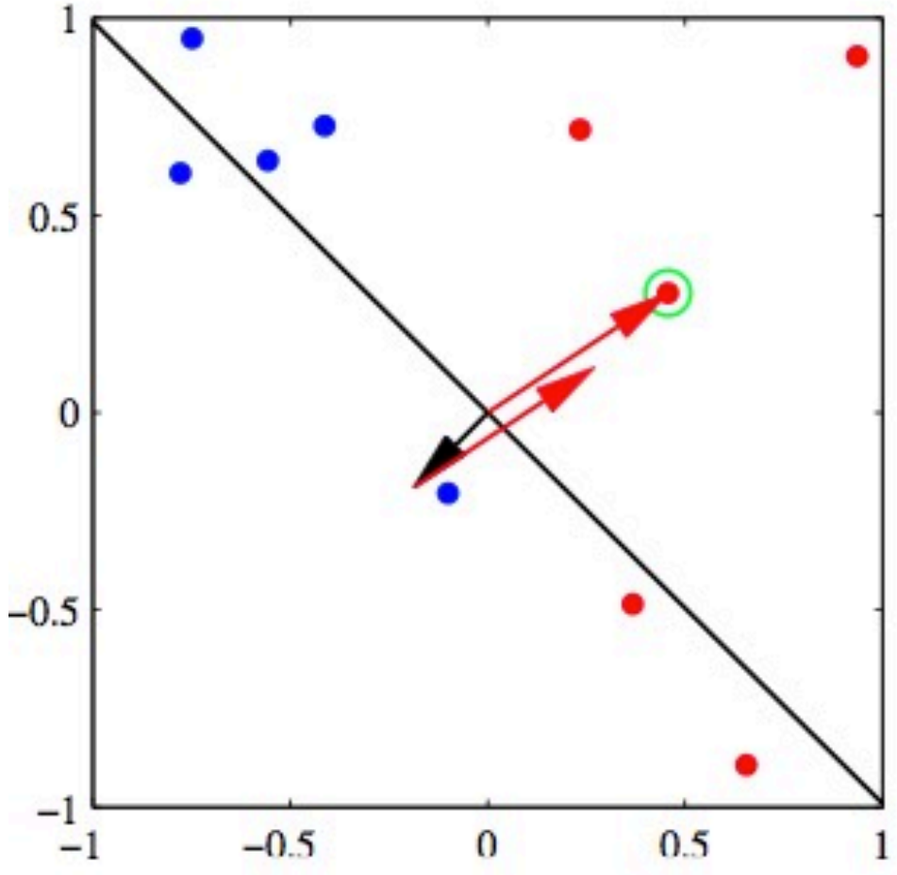


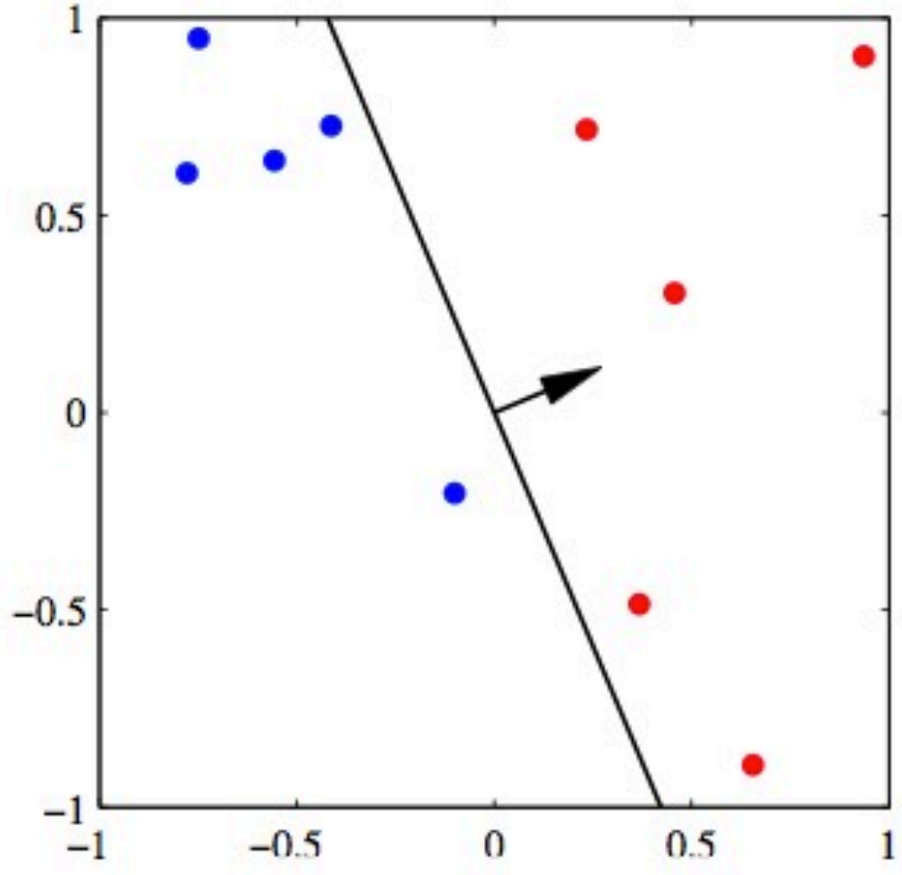
Demo




1
0.5
0
−0.5
−1
−1
−0.5
0
0.5
1


1
0.5
0
−0.5
−1
−1
−0.5
0
0.5
1


1
0.5
0
−0.5
−1
−1
−0.5
0
0.5
1


1
0.5
0
−0.5
−1
−1
−0.5
0
0.5
1

Convergence Theorem

- If there exists some oracle unit vector $u : \|u\| = 1$

$$y_i(u \cdot x_i) \geq \delta \text{ for all } i$$

then the perceptron converges to a linear separator after a number of *updates* bounded by

$$R^2/\delta^2 \text{ where } \quad R = \max_i \|x_i\|$$

- Dimensionality independent
- Order independent (but order matters in output)
- Dataset size independent
- Scales with 'difficulty' of problem

Geometry of the Proof

- **part 1: progress (alignment) on oracle projection**

assume w_i is the weight vector **before** the ith update (on $\langle x_i, y_i \rangle$) and assume initial $w_0 = 0$

$$w_{i+1} = w_i + y_i x_i$$

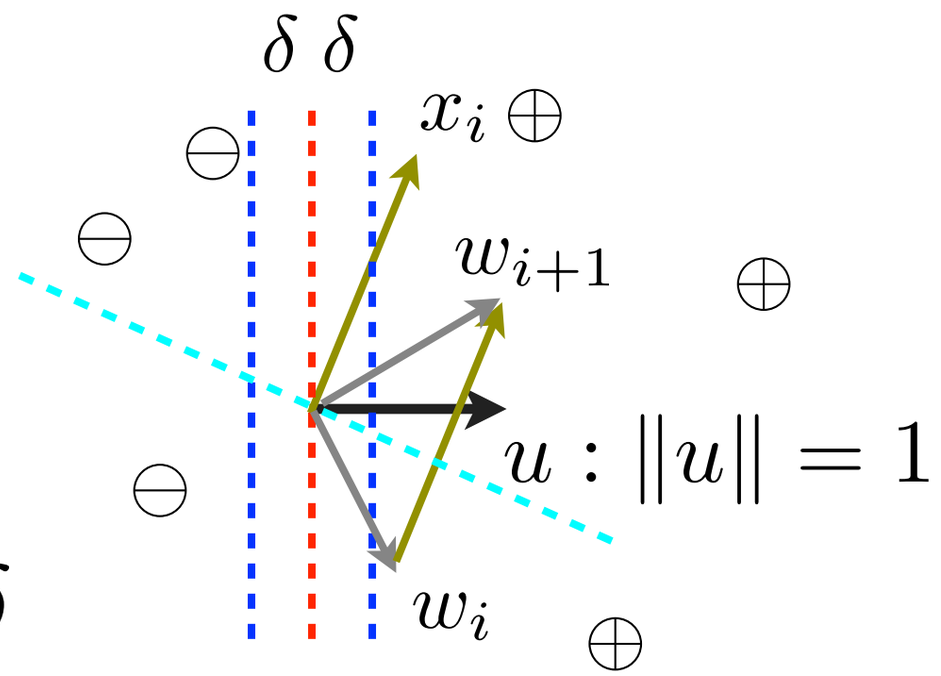
$$u \cdot w_{i+1} = u \cdot w_i + y_i (u \cdot x_i)$$

$$u \cdot w_{i+1} \geq u \cdot w_i + \delta$$

$$u \cdot w_{i+1} \geq i\delta$$

$y_i (u \cdot x_i) \geq \delta$ for all

projection on *u* increases!
(more agreement w/ oracle)



$$\|w_{i+1}\| = \|u\| \|w_{i+1}\| \geq u \cdot w_{i+1} \geq i\delta$$

Geometry of the Proof

- part 2: bound the norm of the weight vector

$$w_{i+1} = w_i + y_i x_i$$

$$\|w_{i+1}\|^2 = \|w_i + y_i x_i\|^2$$

$$= \|w_i\|^2 + \|x_i\|^2 + 2y_i(w_i x_i)$$

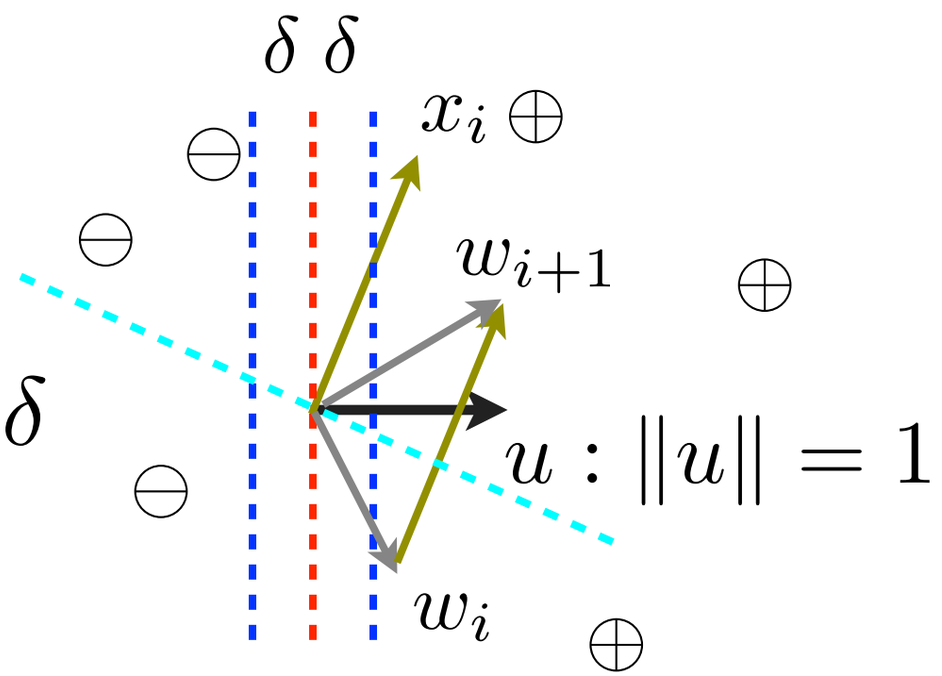
$$\leq \|w_i\|^2 + R^2 \quad \text{"mistake on x_i"}$$

$$\leq iR^2 \quad \text{(radius)}$$

Combine with part 1

$$\|w_{i+1}\| = \|u\|\|w_{i+1}\| \geq u \cdot w_{i+1} \geq i\delta$$

$$i \leq R^2/\delta^2$$

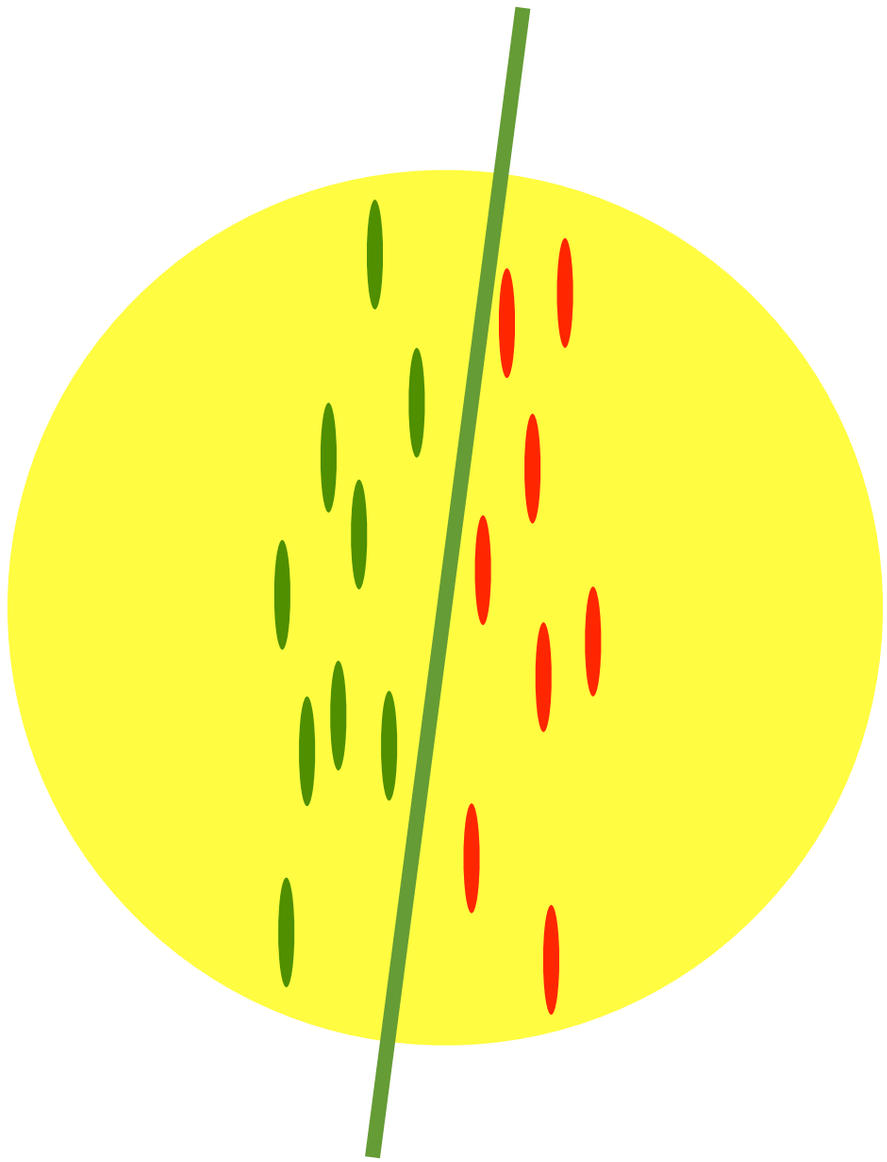


Convergence Bound R^2/δ^2

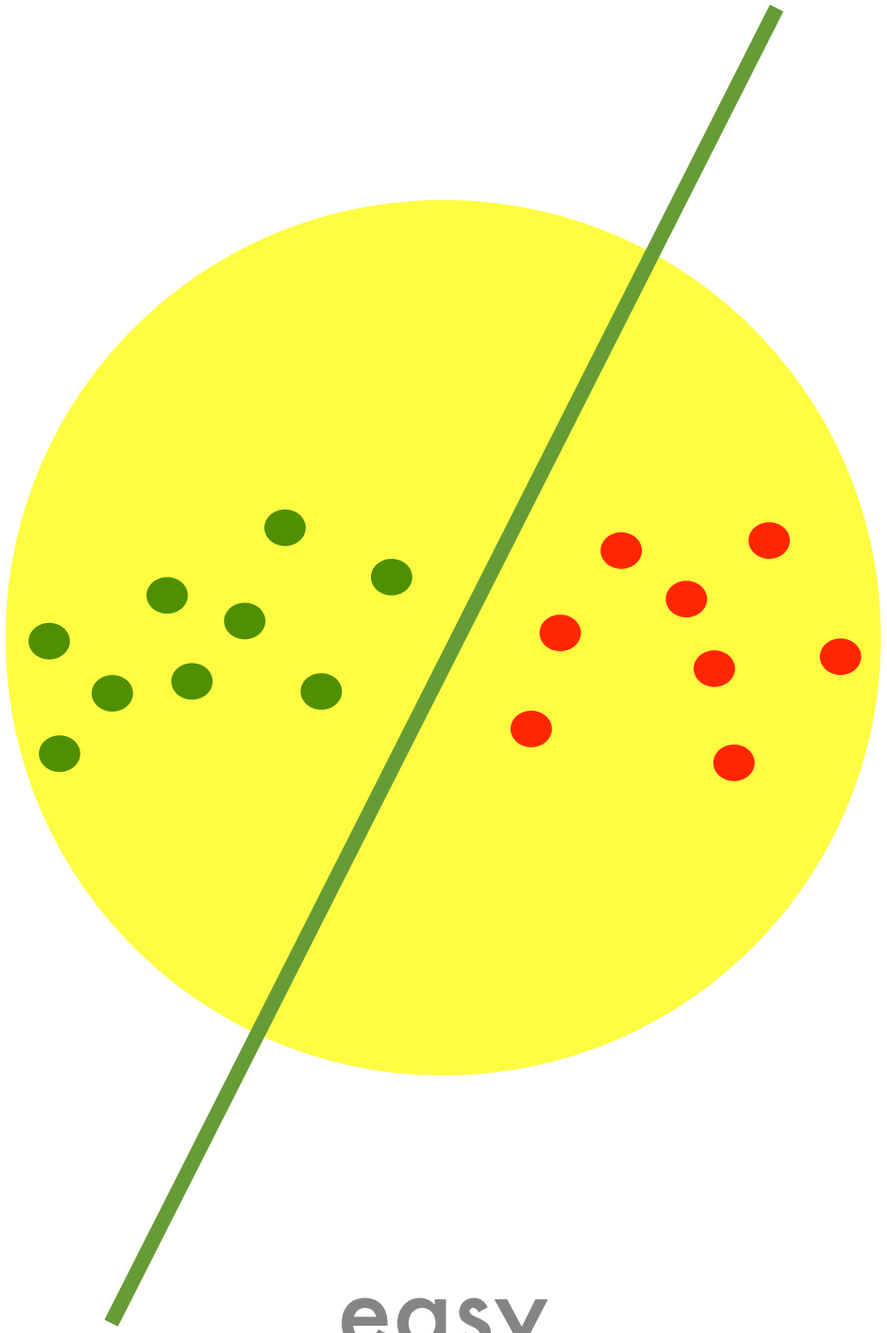
- is independent of:
  - dimensionality
  - number of examples
  - starting weight vector
  - order of examples
  - constant learning rate
- and is dependent of:
  - separation difficulty
  - feature scale

- but test accuracy is dependent of:
  - order of examples (shuffling helps)
  - variable learning rate (1/total#error helps)
    - can you still prove convergence?

Hardness
margin vs. size

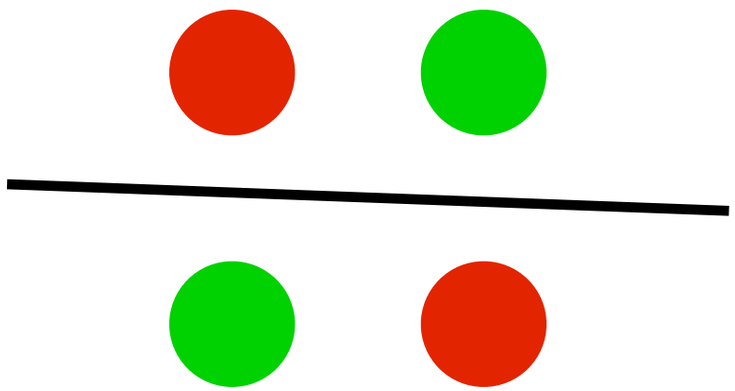


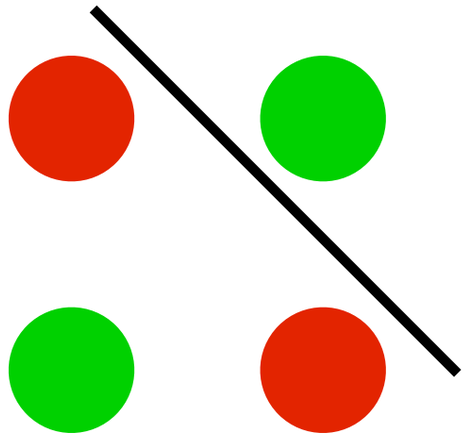
hard

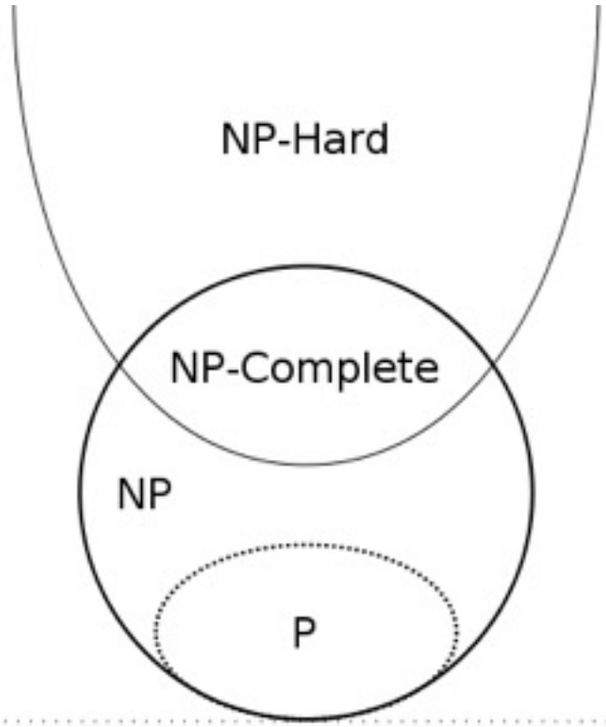


easy

XOR



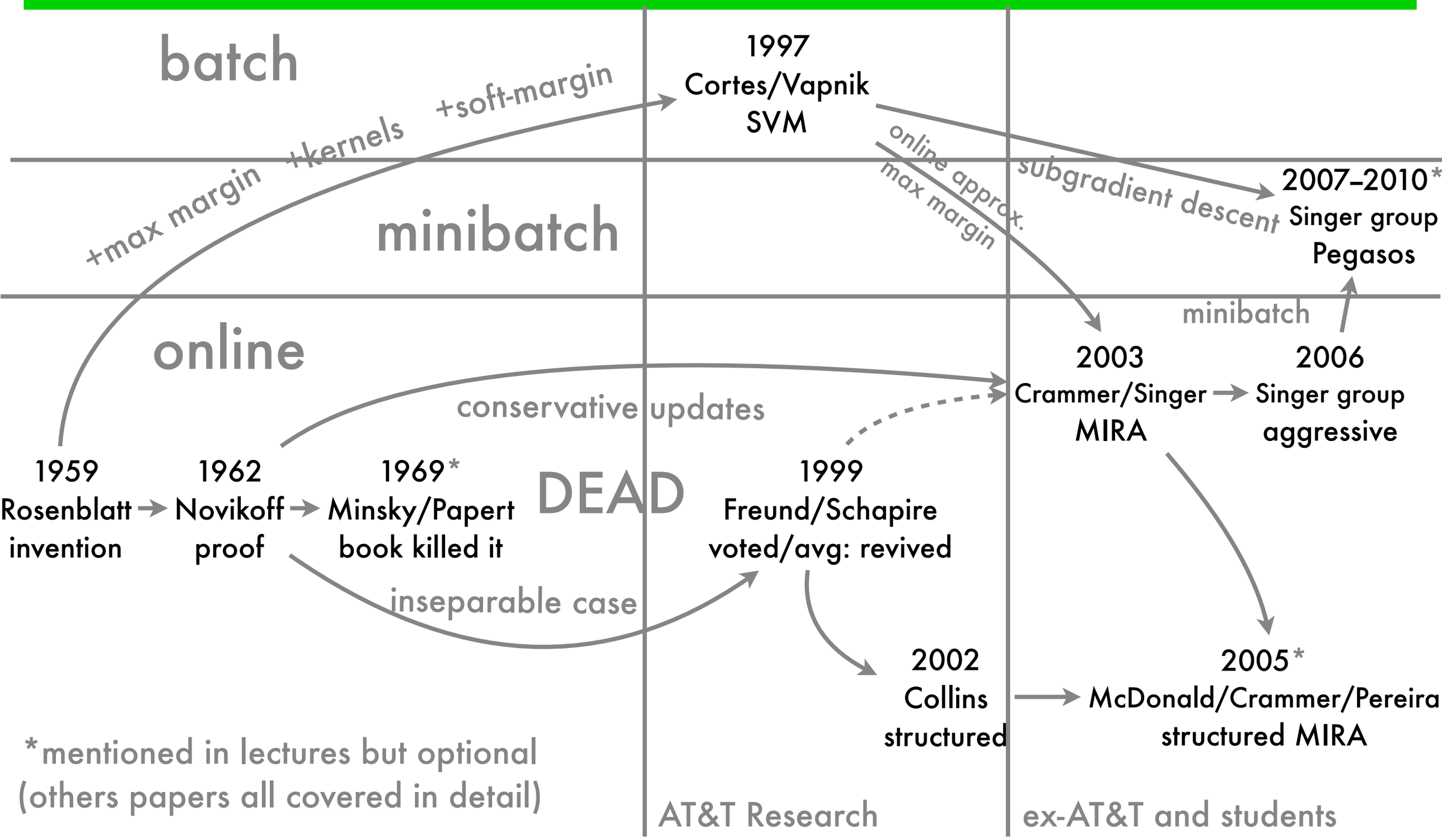




$P \neq NP$

- XOR - not linearly separable
- Nonlinear separation is trivial
- Caveat from "Perceptrons" (Minsky & Papert, 1969)
  Finding the minimum error linear separator
  is NP hard (this killed Neural Networks in the 70s).

Brief History of Perceptron



Extensions of Perceptron

- Problems with Perceptron
  - doesn't converge with inseparable data
  - update might often be too "bold"
  - doesn't optimize margin
  - is sensitive to the order of examples
- Ways to alleviate these problems
  - voted perceptron and average perceptron
  - MIRA (margin-infused relaxation algorithm)

Voted/Avged Perceptron

- motivation: updates on later examples taking over!
- voted perceptron (Freund and Schapire, 1999)
  - record the weight vector after each example in D
    - (not just after each update)
  - and vote on a new example using |D| models
  - shown to have better generalization power
- averaged perceptron (from the same paper)
  - an approximation of voted perceptron
  - just use the average of all weight vectors
  - can be implemented efficiently

Voted Perceptron

Input: a labeled training set $\langle(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\rangle$
number of epochs T
Output: a list of weighted perceptrons $\langle(\mathbf{v}_1, c_1), \ldots, (\mathbf{v}_k, c_k)\rangle$

- Initialize: $k := 0$, $\mathbf{v}_1 := \mathbf{0}$, $c_1 := 0$.
- Repeat T times:
  - For $i = 1, \ldots, m$:
    - Compute prediction: $\hat{y} := \text{sign}(\mathbf{v}_k \cdot \mathbf{x}_i)$
    - If $\hat{y} = y$ then $c_k := c_k + 1$.
      else $\mathbf{v}_{k+1} := \mathbf{v}_k + y_i \mathbf{x}_i$;
      $c_{k+1} := 1$;
      $k := k + 1$.

Prediction

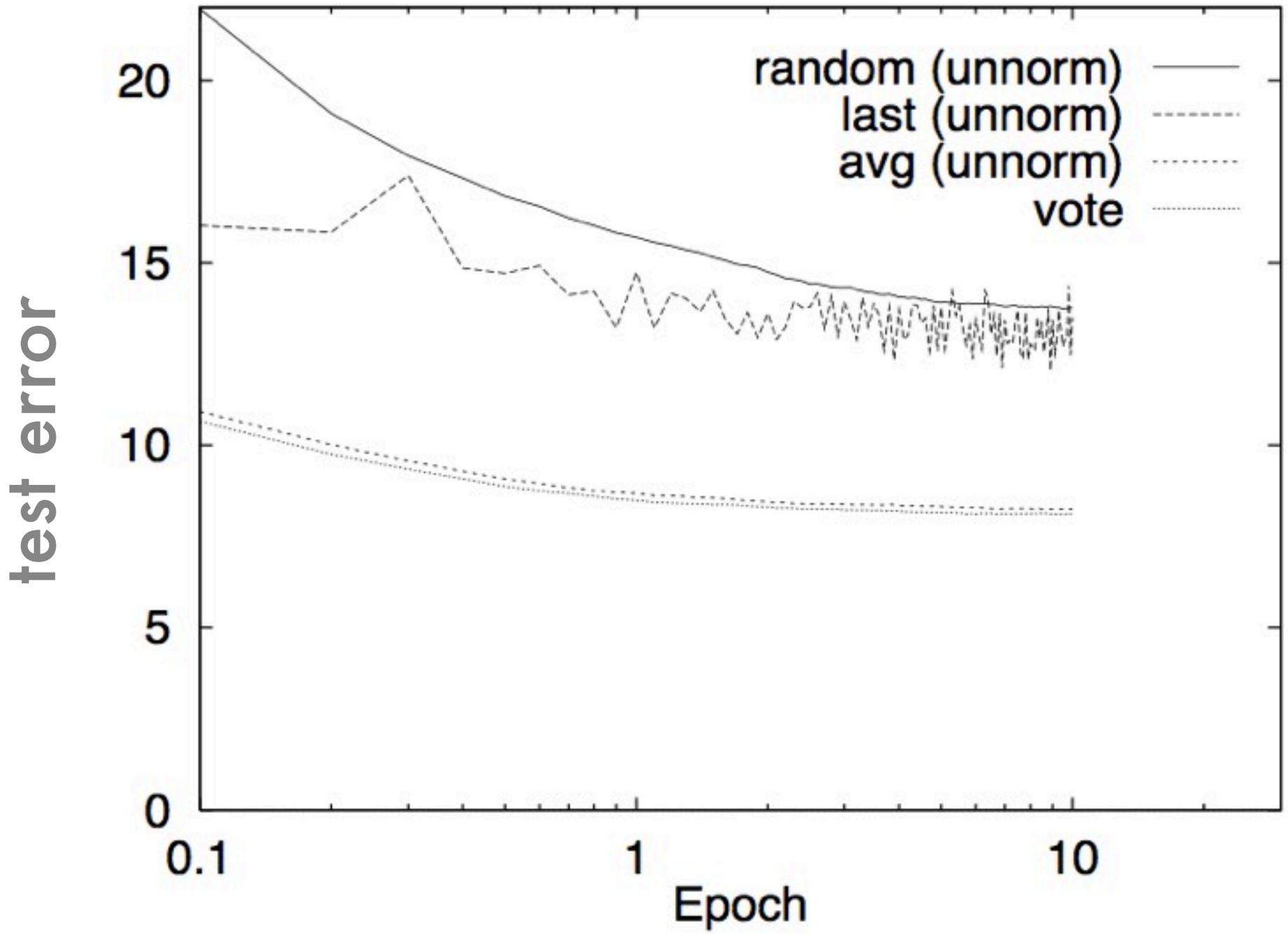
Given: the list of weighted perceptrons: $\langle(\mathbf{v}_1, c_1), \ldots, (\mathbf{v}_k, c_k)\rangle$
an unlabeled instance: $\mathbf{x}$

compute a predicted label $\hat{y}$ as follows:

$$s = \sum_{i=1}^{k} c_i \text{ sign}(\mathbf{v}_i \cdot \mathbf{x}); \qquad \hat{y} = \text{sign}(s) \ .$$

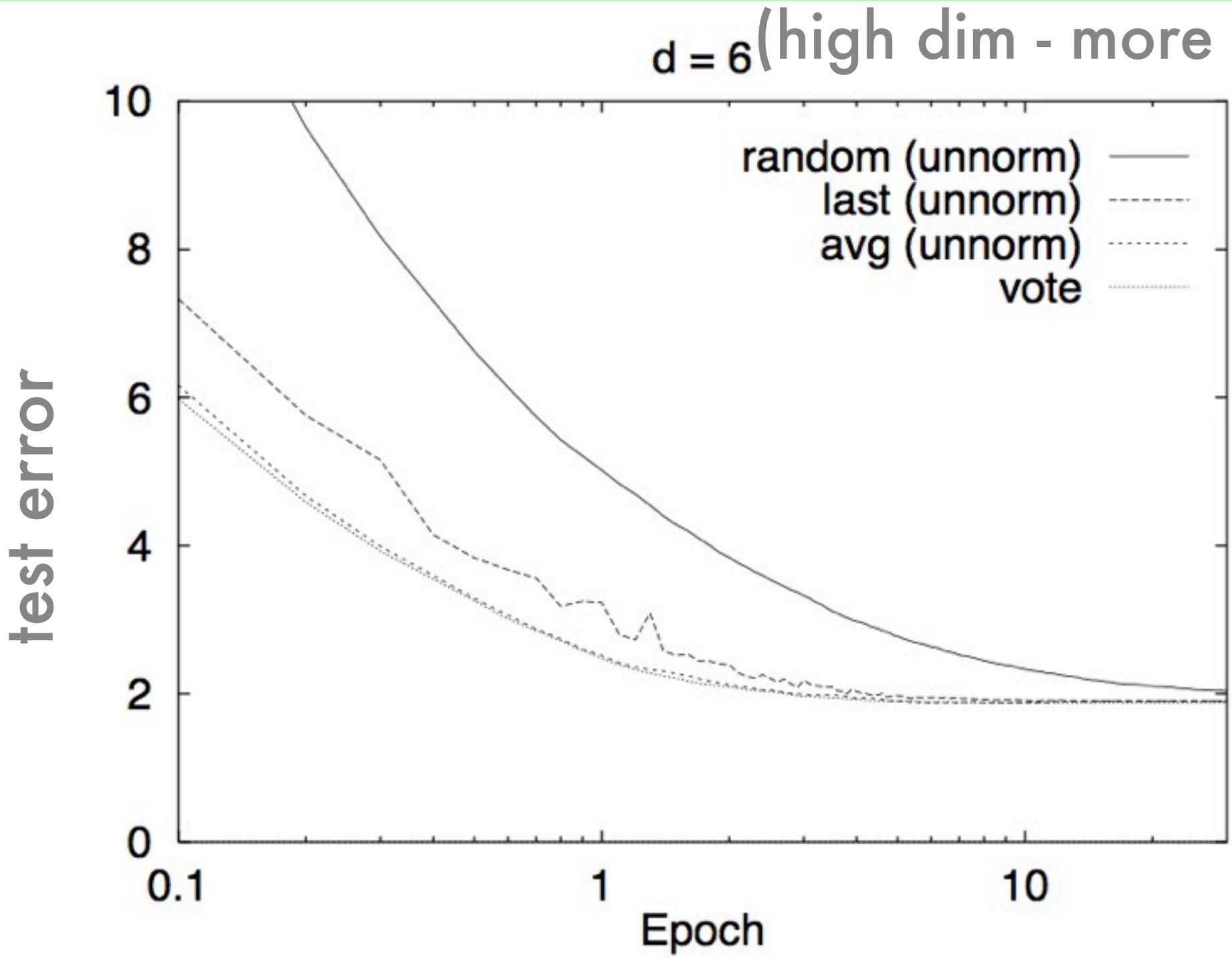
Voted/Avged Perceptron

d = 1 (low dim - less separable)



Voted/Avged Perceptron

(high dim - more separable)



Averaged Perceptron

- voted perceptron is not scalable
  - and does not output a single model
- avg perceptron is an approximation of voted perceptron

initialize $w = 0$ and $w' = 0$
repeat $c \leftarrow c + 1$
  if $y_i \left[\langle w, x_i \rangle + b\right] \leq 0$ **then**
    $w \leftarrow w + y_i x_i$ and $b \leftarrow b + y_i$
  $w' \leftarrow w' + w$
until all classified correctly
output w' / c

after each example, not each update

Efficient Implementation of Averaging

- naive implementation (running sum) doesn't scale
- very clever trick from Daume (2006, PhD thesis)

initialize $w = 0$ and $w_a = 0$

repeat $\boxed{c \leftarrow c + 1}$

  if $y_i \left[\langle w, x_i \rangle + b\right] \leq 0$ **then**

    $w \leftarrow w + y_i x_i$

    $w_a \leftarrow w_a + c y_i x_i$

until all classified correctly

output $w - w_a / c$

$\mathbf{w}^t$ $\Delta\mathbf{w}^t$

$\mathbf{w}^{(0)} =$				
$\mathbf{w}^{(1)} =$	$\Delta\mathbf{w}^{(1)}$			
$\mathbf{w}^{(2)} =$	$\Delta\mathbf{w}^{(1)}$	$\Delta\mathbf{w}^{(2)}$		
$\mathbf{w}^{(3)} =$	$\Delta\mathbf{w}^{(1)}$	$\Delta\mathbf{w}^{(2)}$	$\Delta\mathbf{w}^{(3)}$	
$\mathbf{w}^{(4)} =$	$\Delta\mathbf{w}^{(1)}$	$\Delta\mathbf{w}^{(2)}$	$\Delta\mathbf{w}^{(3)}$	$\Delta\mathbf{w}^{(4)}$

MIRA

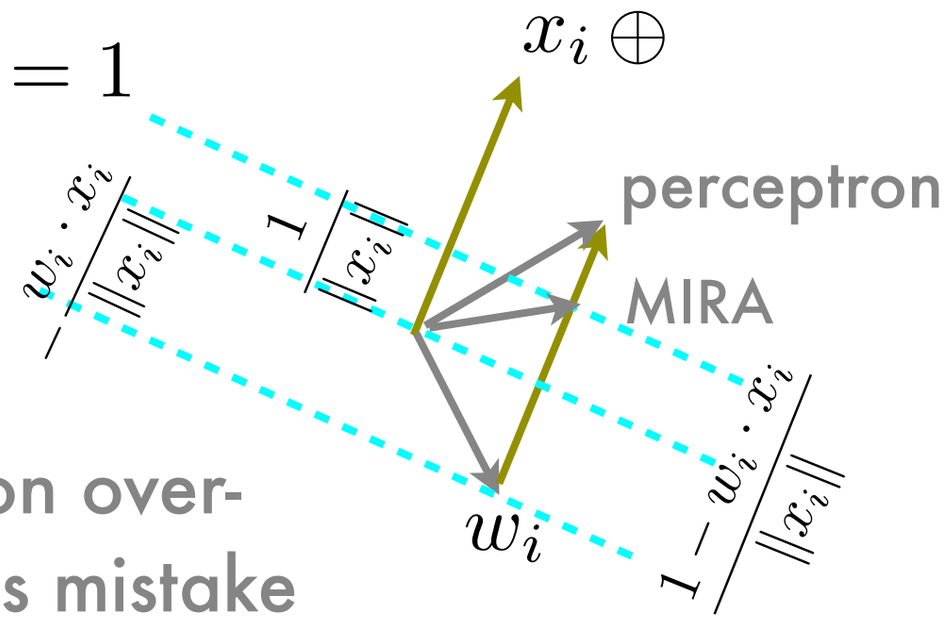
- perceptron often makes too bold updates
  - but hard to tune learning rate
- the smallest update to correct the mistake?

$$w_{i+1} = w_i + \frac{y_i - w_i \cdot x_i}{\|x_i\|^2} x_i$$

easy to show:

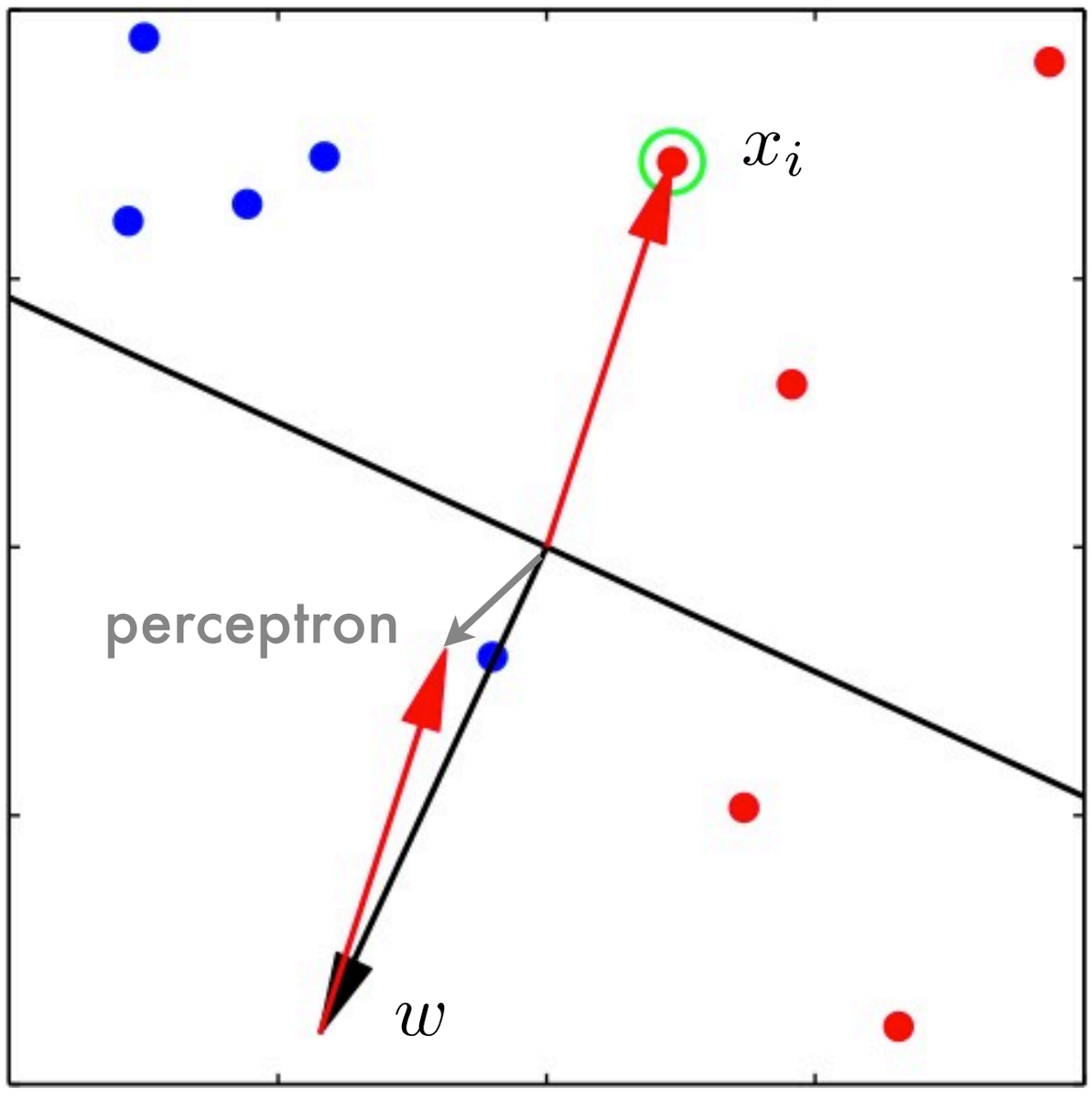
$$y_i(w_{i+1} \cdot x_i) = y_i \left(w_i + \frac{y_i - w_i \cdot x_i}{\|x_i\|^2} x_i\right) \cdot x_i = 1$$

margin-infused relaxation algorithm (MIRA)



perceptron over-corrects this mistake

Perceptron



perceptron under-corrects this mistake

(bias=0)

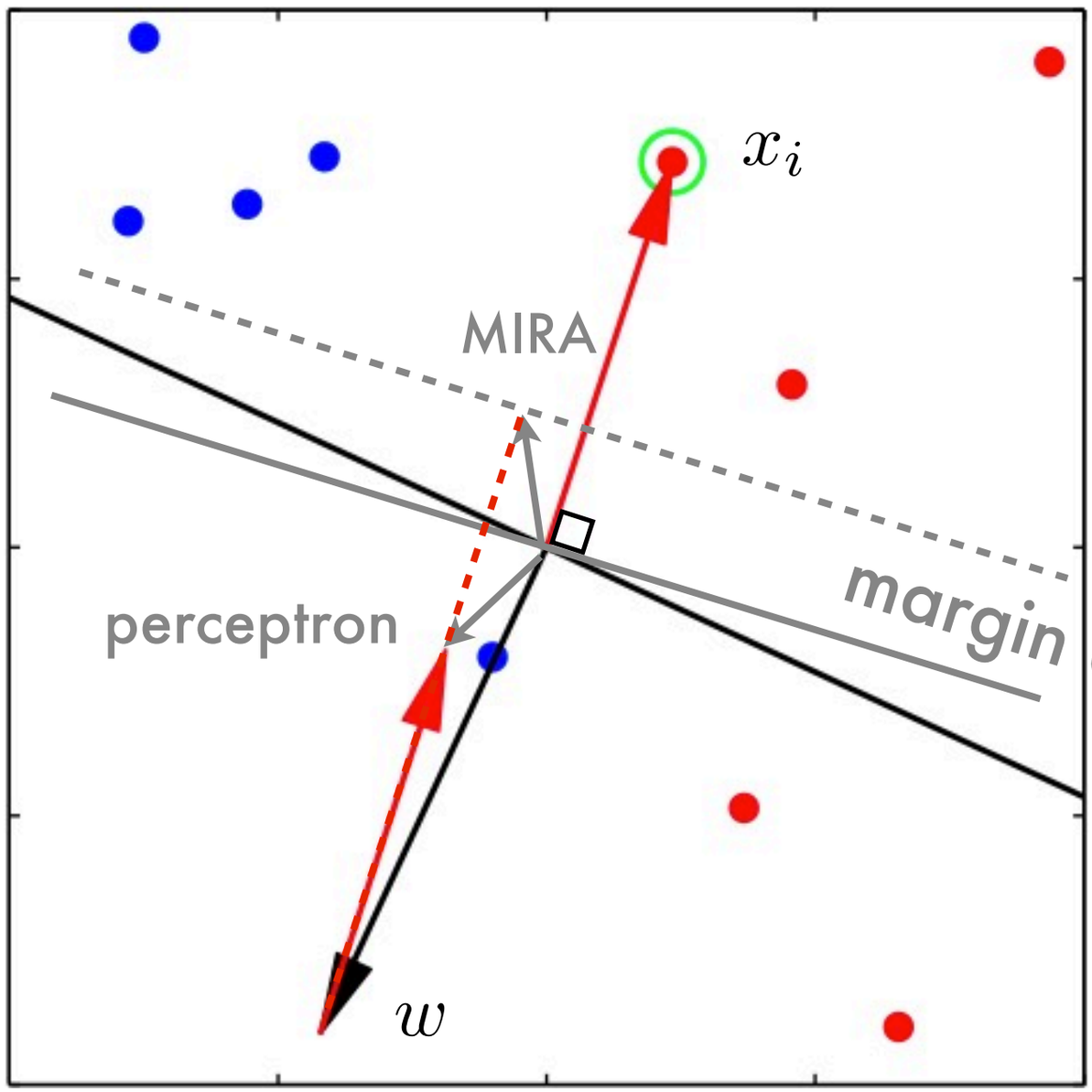
MIRA

$$\min_{\mathbf{w}'} \|\mathbf{w}' - \mathbf{w}\|^2$$

$$\text{s.t. } \mathbf{w}' \cdot \mathbf{x} \geq 1$$

minimal change to ensure margin

MIRA ≈ 1-step SVM



MIRA makes sure after update, dot-product w · x_i = 1

of 1/|x_i|

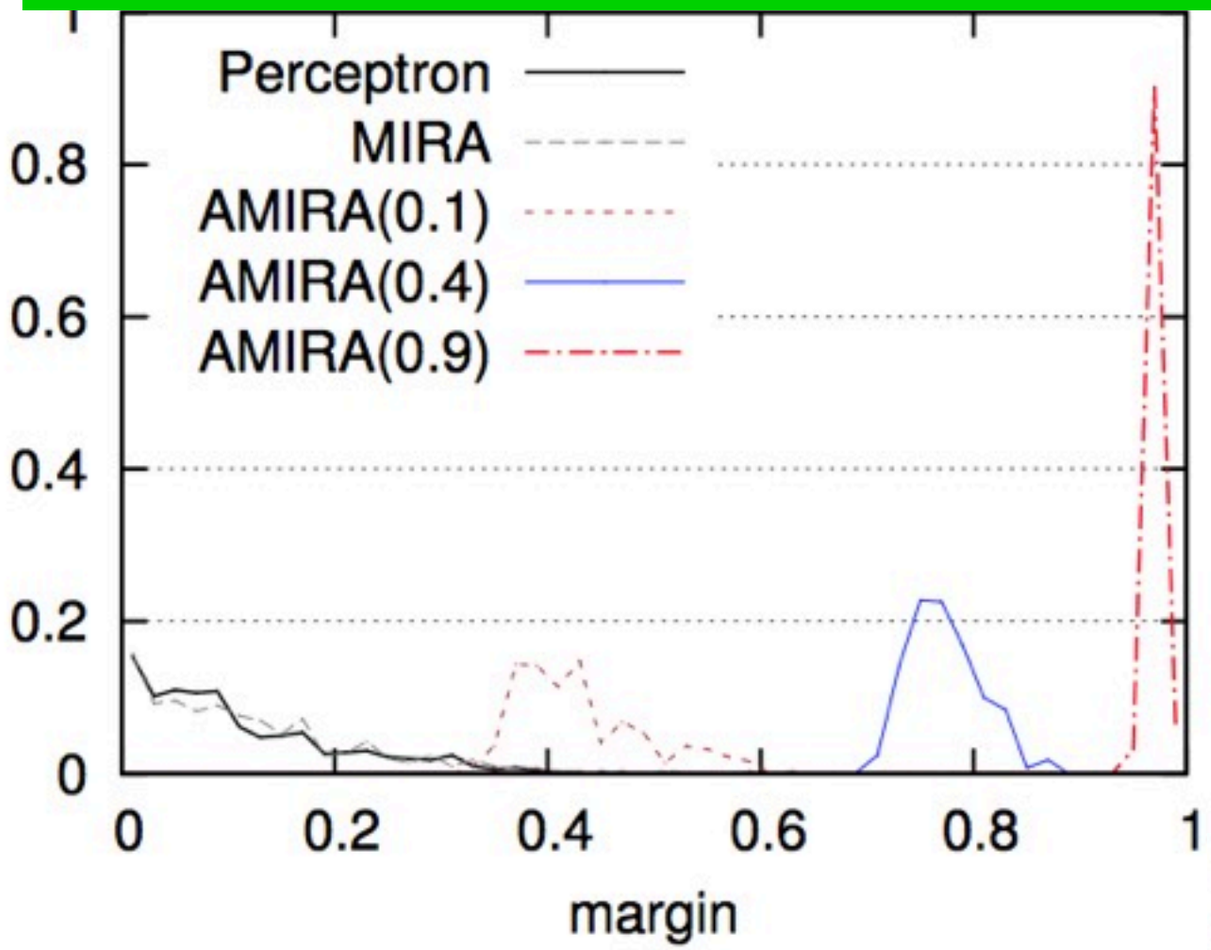
perceptron under-corrects this mistake

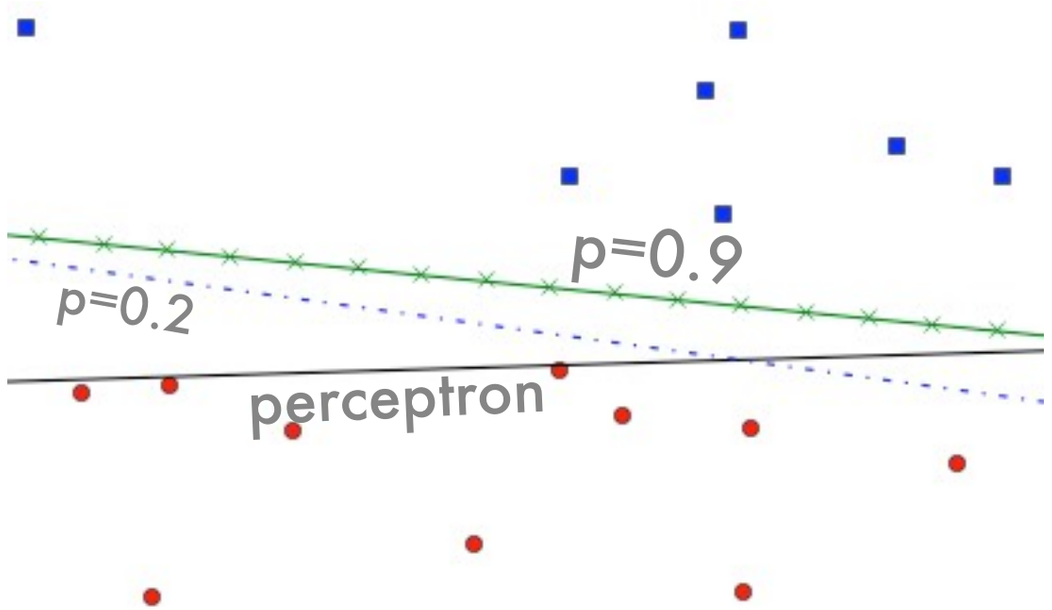
(bias=0)

Aggressive MIRA

- aggressive version of MIRA
  - also update if correct but margin isn't big enough
- functional margin: $y_i(\mathbf{w} \cdot x_i)$
- geometric margin: $\dfrac{y_i(\mathbf{w} \cdot x_i)}{\|\mathbf{w}\|}$
- update if functional margin is <=*p* (0<=*p*<1)
  - update rule is same as MIRA
  - called *p*-aggressive MIRA (MIRA: *p*=0)
- larger *p* leads to a larger geometric margin
  - but slower convergence

Aggressive MIRA





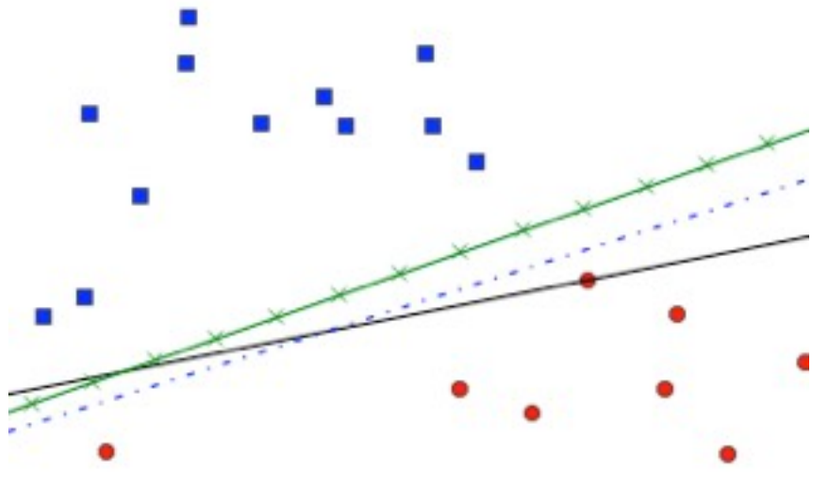
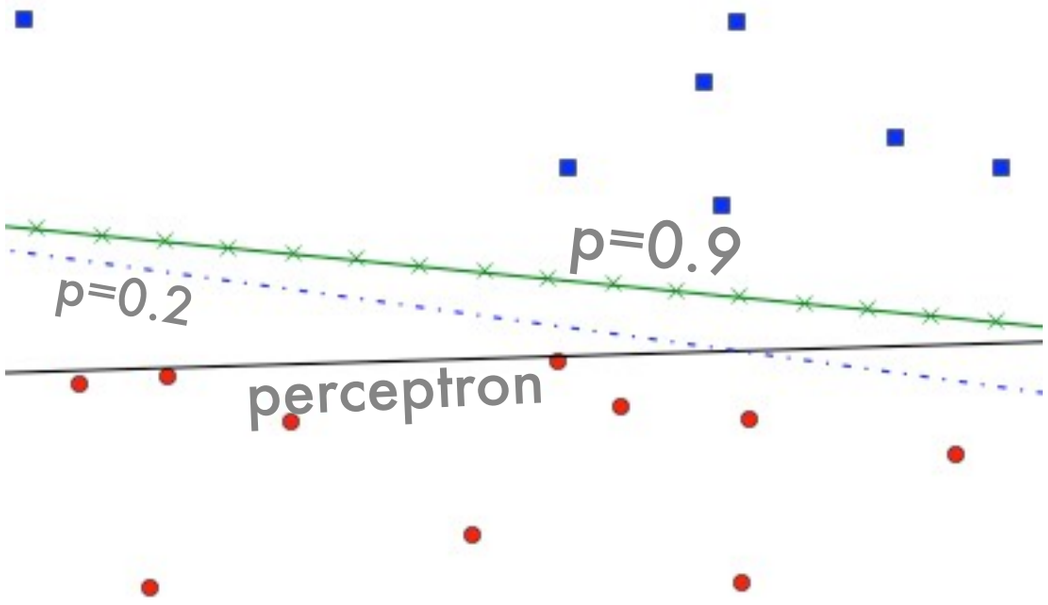


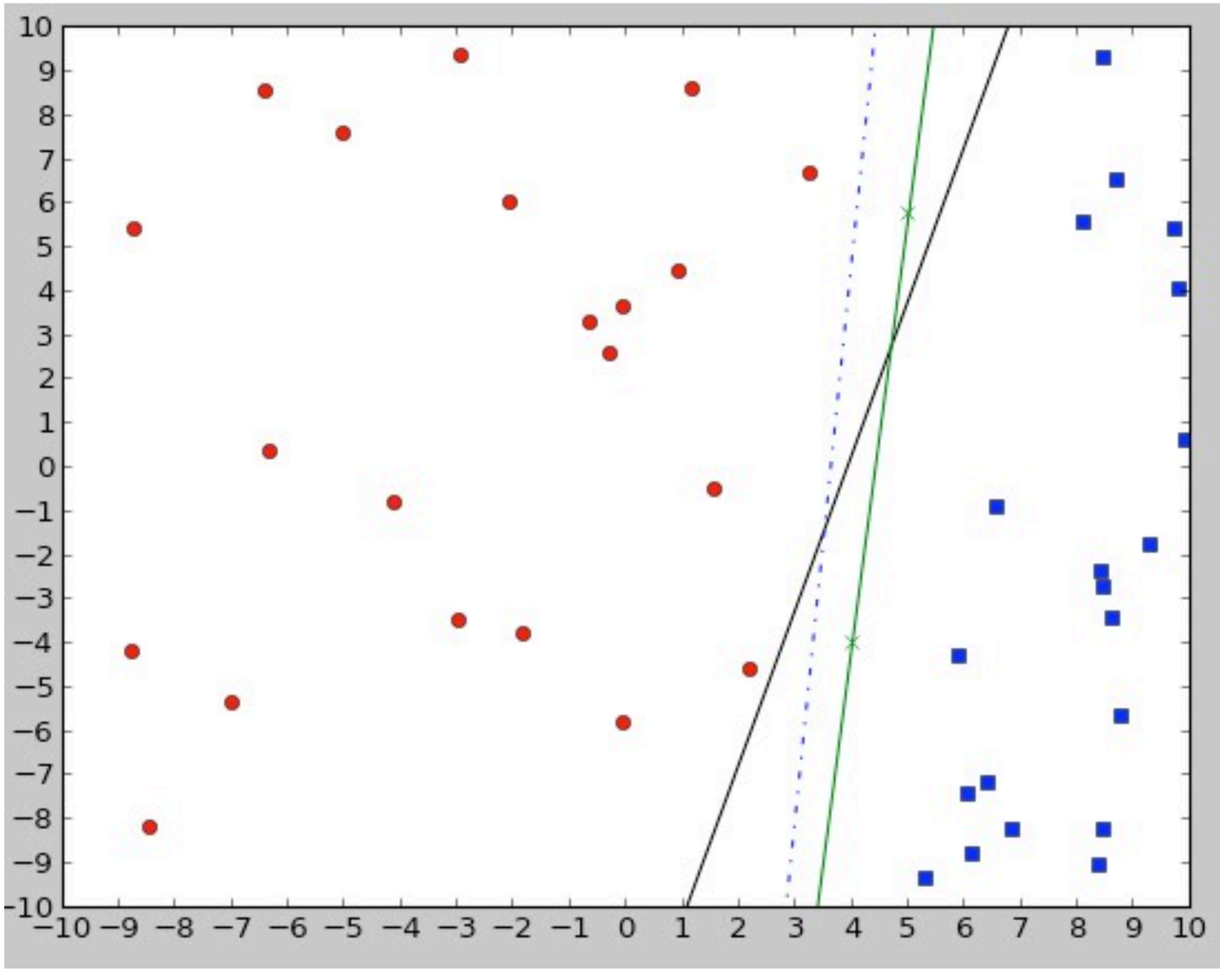
Table 3. Error rates on MNIST dataset. Both ROMMA and Aggressive ROMMA use a scale of 1100. The numbers in parentheses denote the aggressive parameters for AMIRA.

Epoch	1	2	3	4
Perceptron	2.98%	2.32%	1.94%	1.88%
Perceptron(avg.)	2.16%	1.85%	1.73%	1.69%
ROMMA	2.48%	1.96%	1.79%	1.77%
aggr-ROMMA	**2.14%**	1.82%	1.71%	1.67%
MIRA	2.56%	2.03%	1.74%	1.70%
bin AMIRA(0.1)	2.20%	**1.78%**	**1.67%**	**1.64%**

Demo

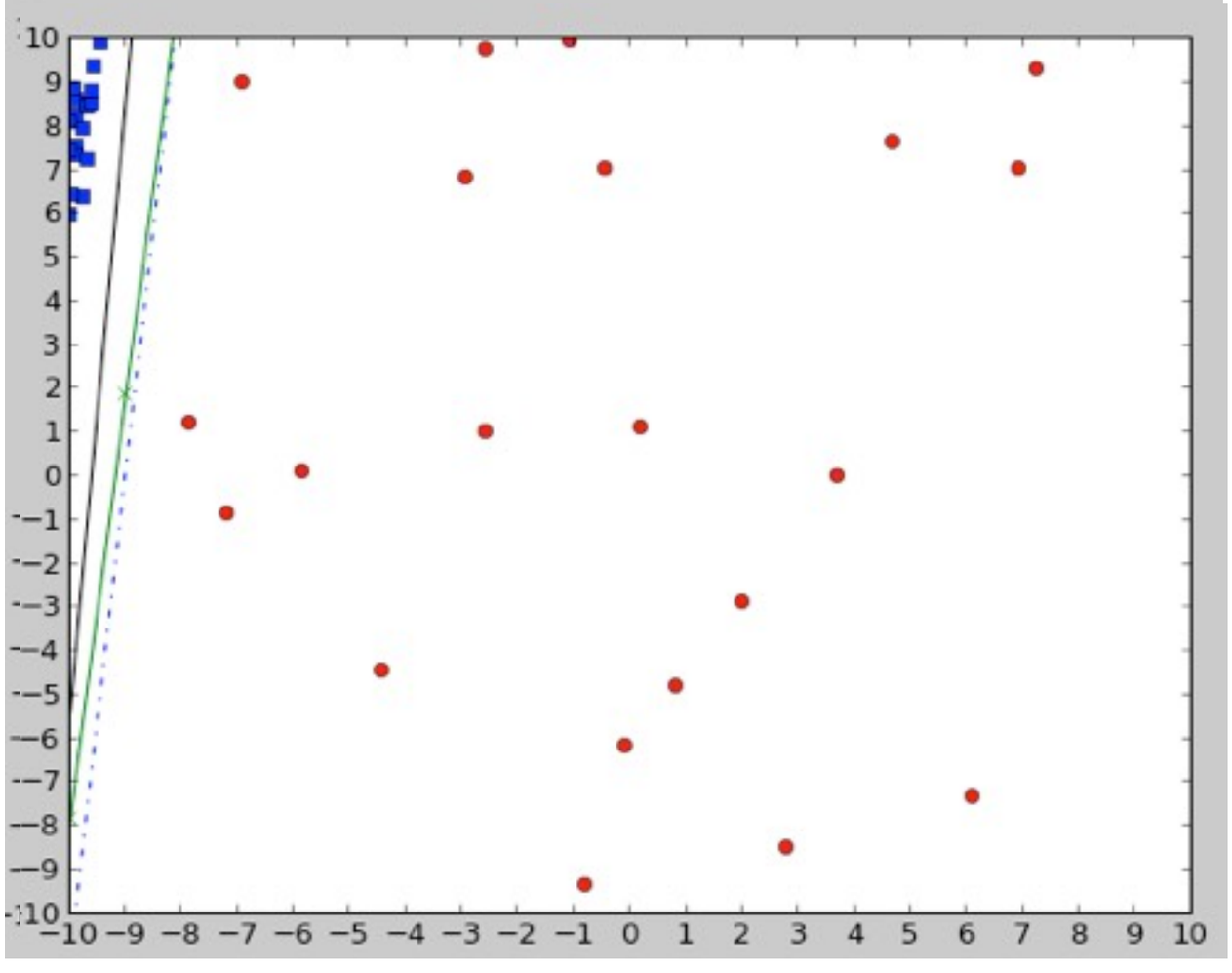
- perceptron vs. 0.2-aggressive vs. 0.9-aggressive



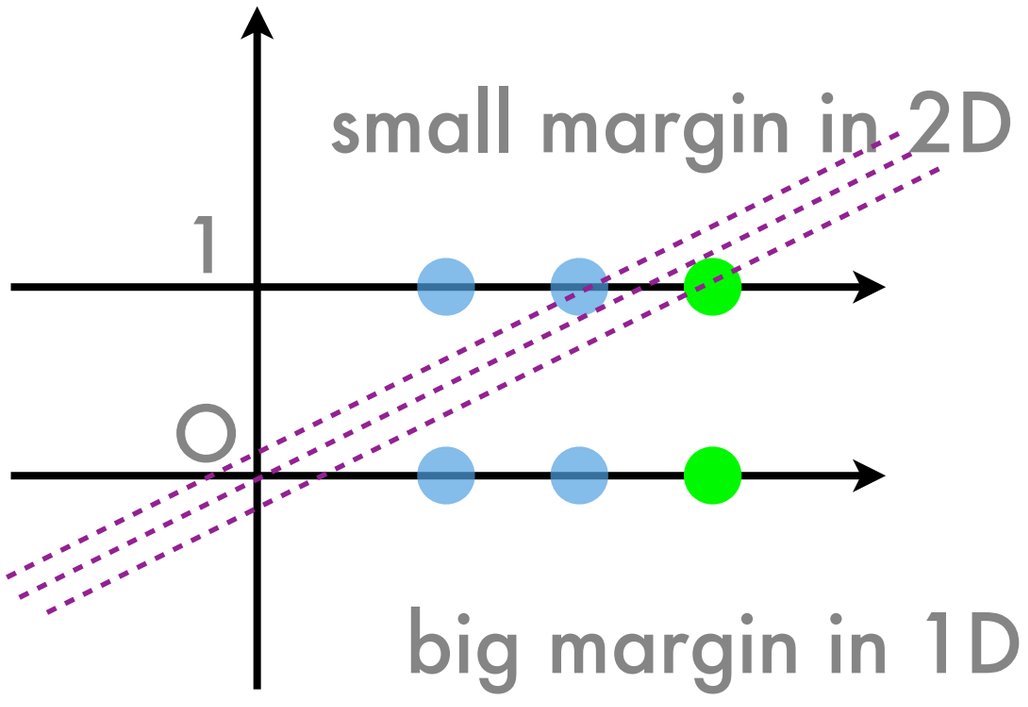


Demo

- perceptron vs. 0.2-aggressive vs. 0.9-aggressive
- why does this dataset so slow to converge?
  - perceptron: 22, p=0.2: 87, p=0.9: 2,518 epochs

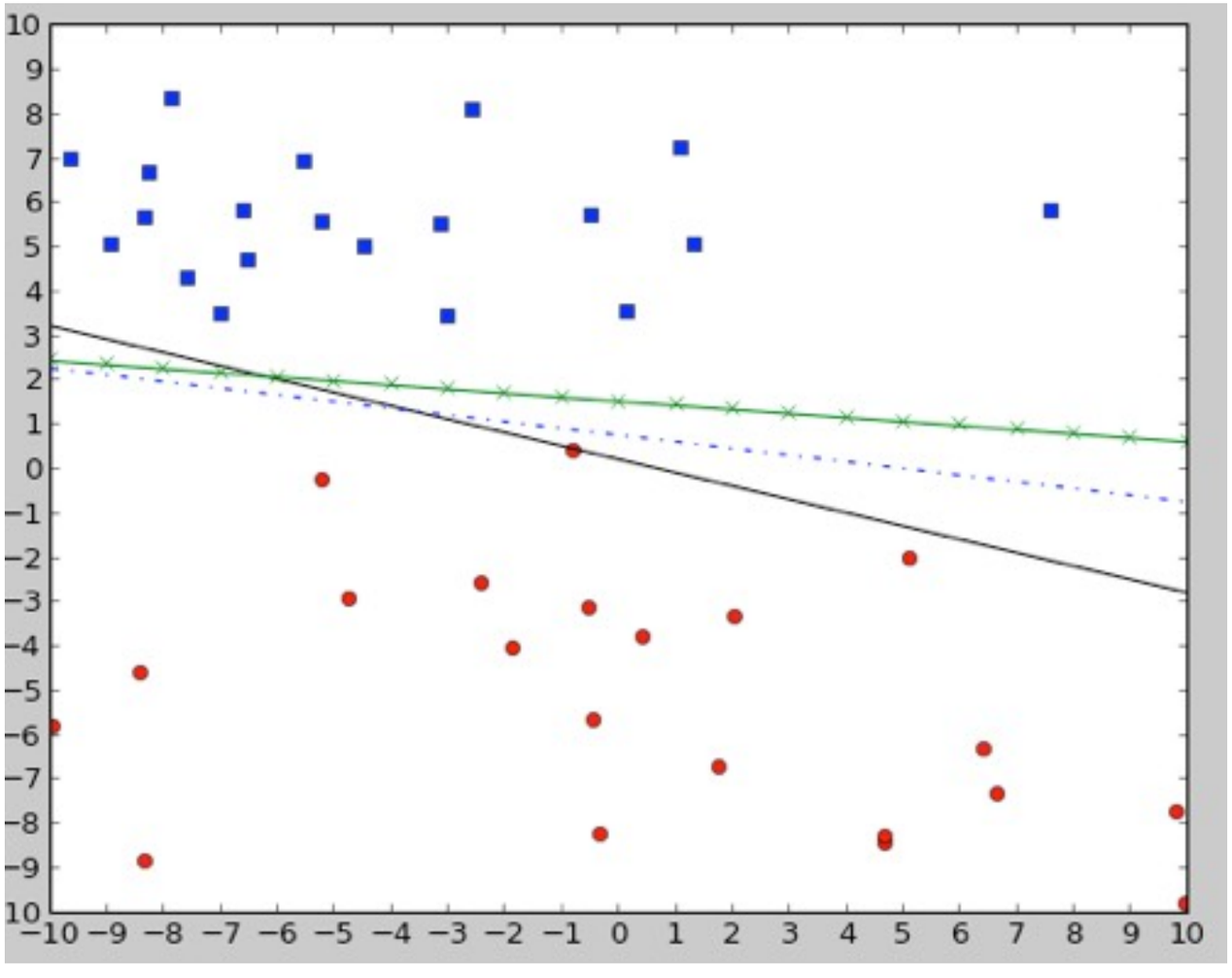


answer: margin shrinks in augmented space!

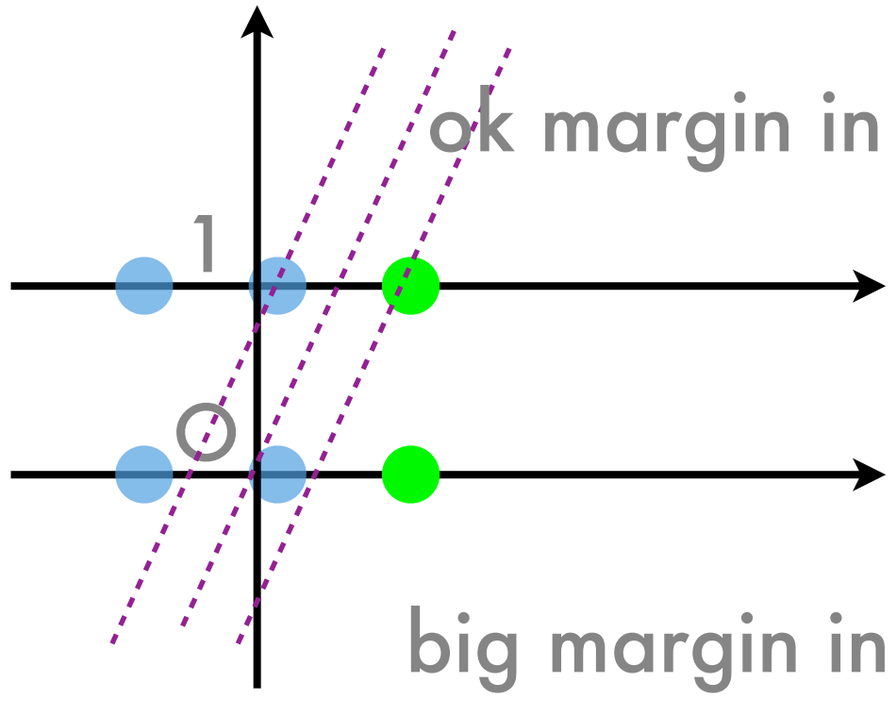


Demo

- perceptron vs. 0.2-aggressive vs. 0.9-aggressive
- why does this dataset so fast to converge?
  - perceptron: 3, p=0.2: 1, p=0.9: 5 epochs



answer: margin shrinks in augmented space!



What if data is not separable

- in practice, data is almost always inseparable
  - wait, what does that mean??
- perceptron cycling theorem (1970)
  - weights will remain bounded and not diverge
- use dev set for when to stop (prevents overfitting)
- higher-order features by combining atomic ones
- kernels => separable in higher dimensions

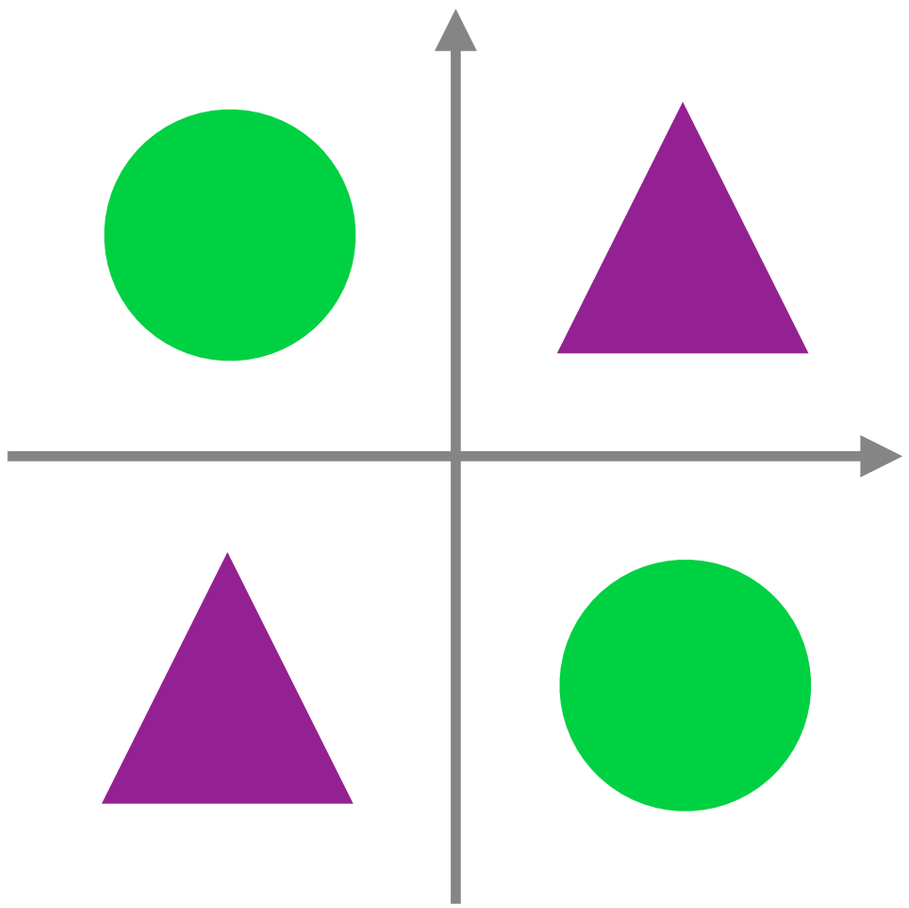
ON THE BOUNDEDNESS OF AN ITERATIVE PROCEDURE FOR SOLVING A SYSTEM OF LINEAR INEQUALITIES[1]

H. D. BLOCK AND S. A. LEVIN

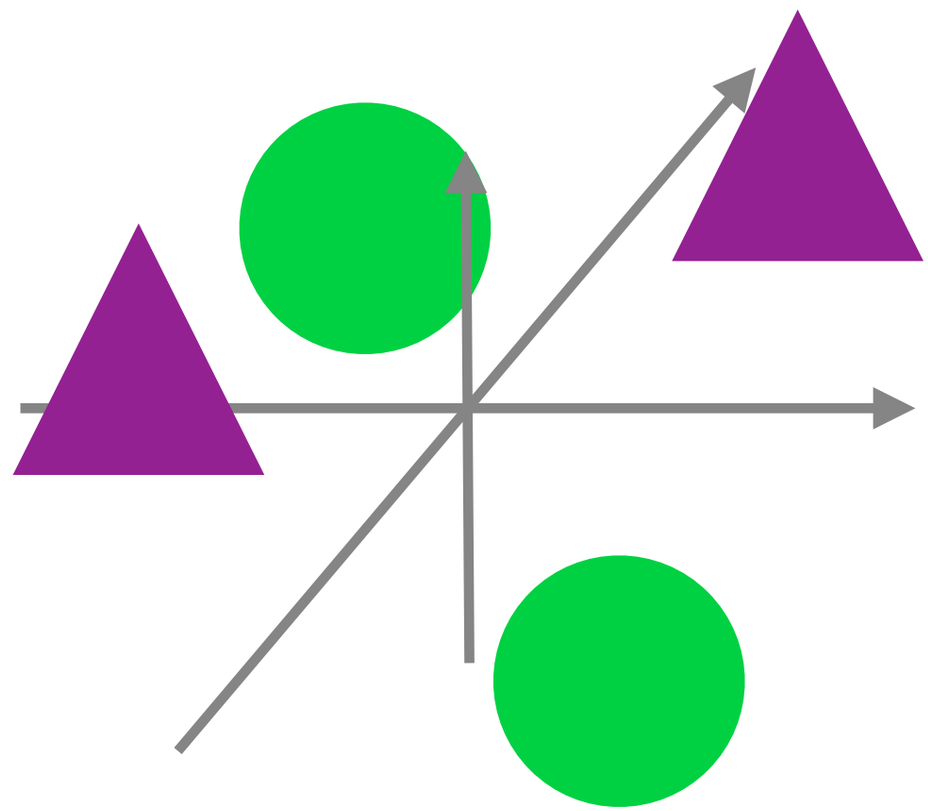
SVM with a polynomial Kernel visualization

Created by:
Udi Aharoni

Solving XOR



(x_1, x_2)



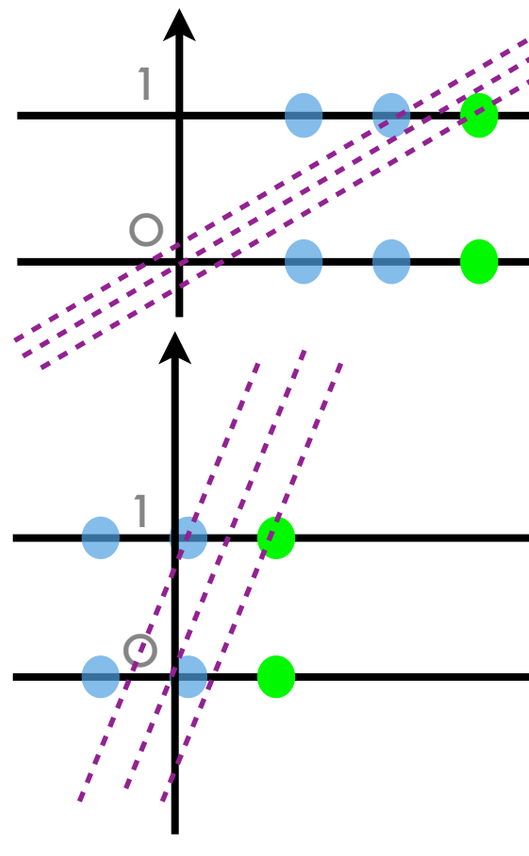
$(x_1, x_2, x_1 x_2)$

- XOR not linearly separable
- Mapping into 3 dimensions makes it easily solvable

Useful Engineering Tips:

averaging, shuffling, variable learning rate, fixing feature scale

- averaging helps significantly; MIRA helps a tiny little bit
  - perceptron < MIRA < avg. perceptron ≈ avg. MIRA
- shuffling the data helps hugely if classes were ordered
- shuffling before each epoch helps a little
- variable learning rate often helps a little
  - 1/(total#updates) or 1/(total#examples) helps
  - any requirement in order to converge?
    - how to prove convergence now?
- centering of each feature dim helps
  - why? => R smaller, margin bigger
- unit variance also helps (why?)
  - 0-mean, 1-var => each feature ≈ a unit Gaussian



Useful Engineering Tips:

categorical=>binary, feature bucketing (binning/quantization)

```
Age, Workclass, Education, Marital_Status,     Occupation,     Race,  Sex,  Hours, Country,      Target
40,  Private,   Doctorate, Married-civ-spouse, Prof-specialty, White, Male, 60,    United-States, >50K
```

- HW1 Adult income dataset: <=50K, or >50K?
  - 2 numerical features
    - age and hours-per-week
    - option 1: treat them as numerical features
      - but is older and more hours always better?
    - option 2: treat them as <u>binary</u> features
      - e.g., age=22, hours=38, …
    - option 3: bin them (e.g., age=0-25, hours=41-60…)
  - *7* categorical features: convert to <u>binary</u> features
    - country, race, occupation, etc.
    - e.g., country:United_States, education:Doctorate,…
    - *optional*: you can probably add a numerical feature "edu_level"
  - perceptron: ~20% error, avg. perceptron: ~16% error

Brief History of Perceptron