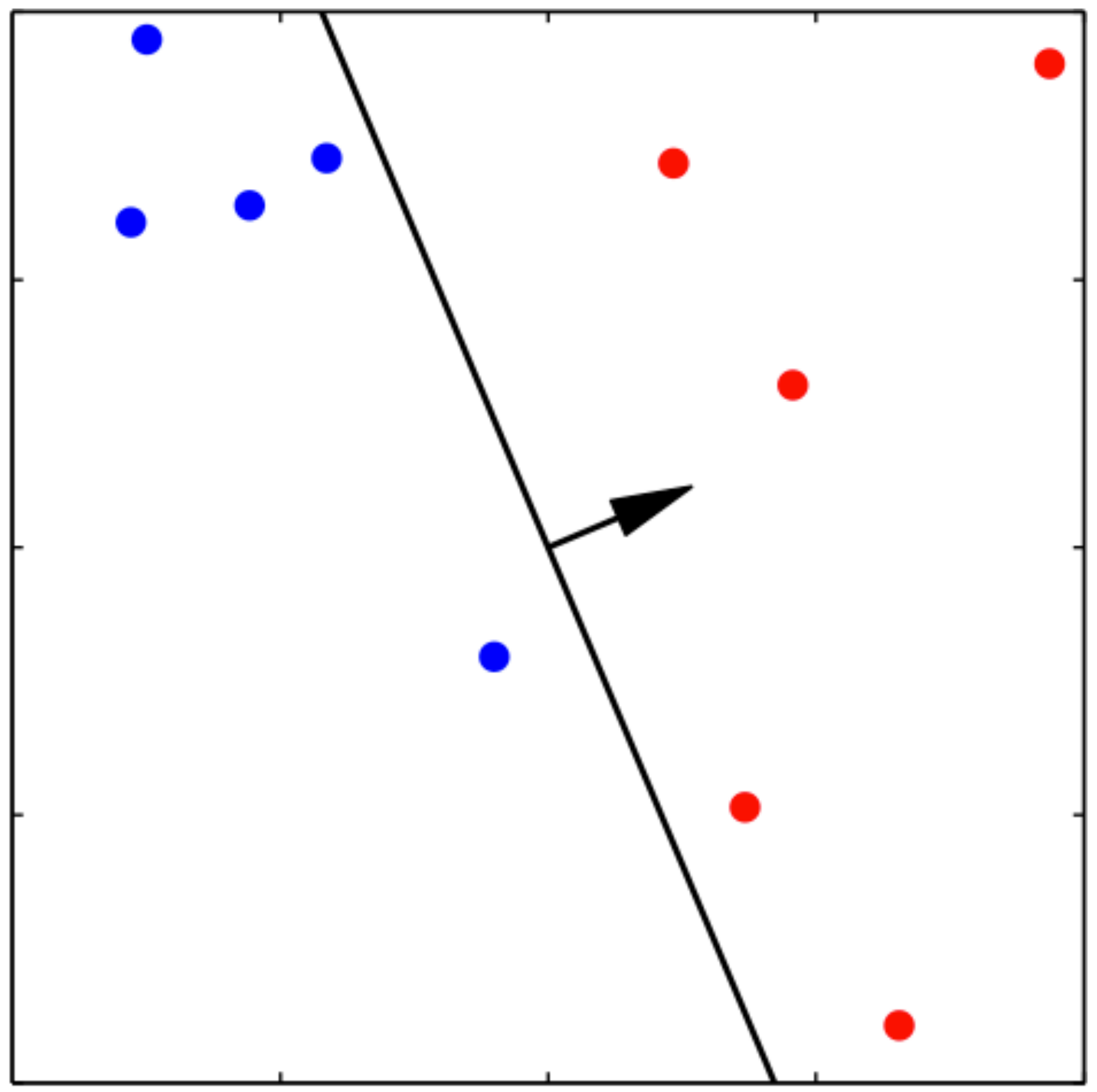
Applied Machine Learning

CIML Chap 4 (A Geometric Approach)



"Equations are just the boring part of mathematics. I attempt to see things in terms of geometry."
—Stephen Hawking

Week 4: Linear Classification: Perceptron

Professor Liang Huang

some slides from Alex Smola (CMU/Amazon)

Roadmap for Unit 2 (Weeks 4-5)

- Week 4: Linear Classifier and Perceptron
  - Part I: Brief History of the Perceptron
  - Part II: Linear Classifier and Geometry (testing time)
  - Part III: Perceptron Learning Algorithm (training time)
  - Part IV: Convergence Theorem and Geometric Proof
  - Part V: Limitations of Linear Classifiers, Non-Linearity, and Feature Maps
- Week 5: Extensions of Perceptron and Practical Issues
  - Part I: My Perceptron Demo in Python
  - Part II: Voted and Averaged Perceptrons
  - Part III: MIRA and Aggressive MIRA
  - Part IV: Practical Issues
  - Part V: Perceptron vs. Logistic Regression (hard vs. soft); Gradient Descent

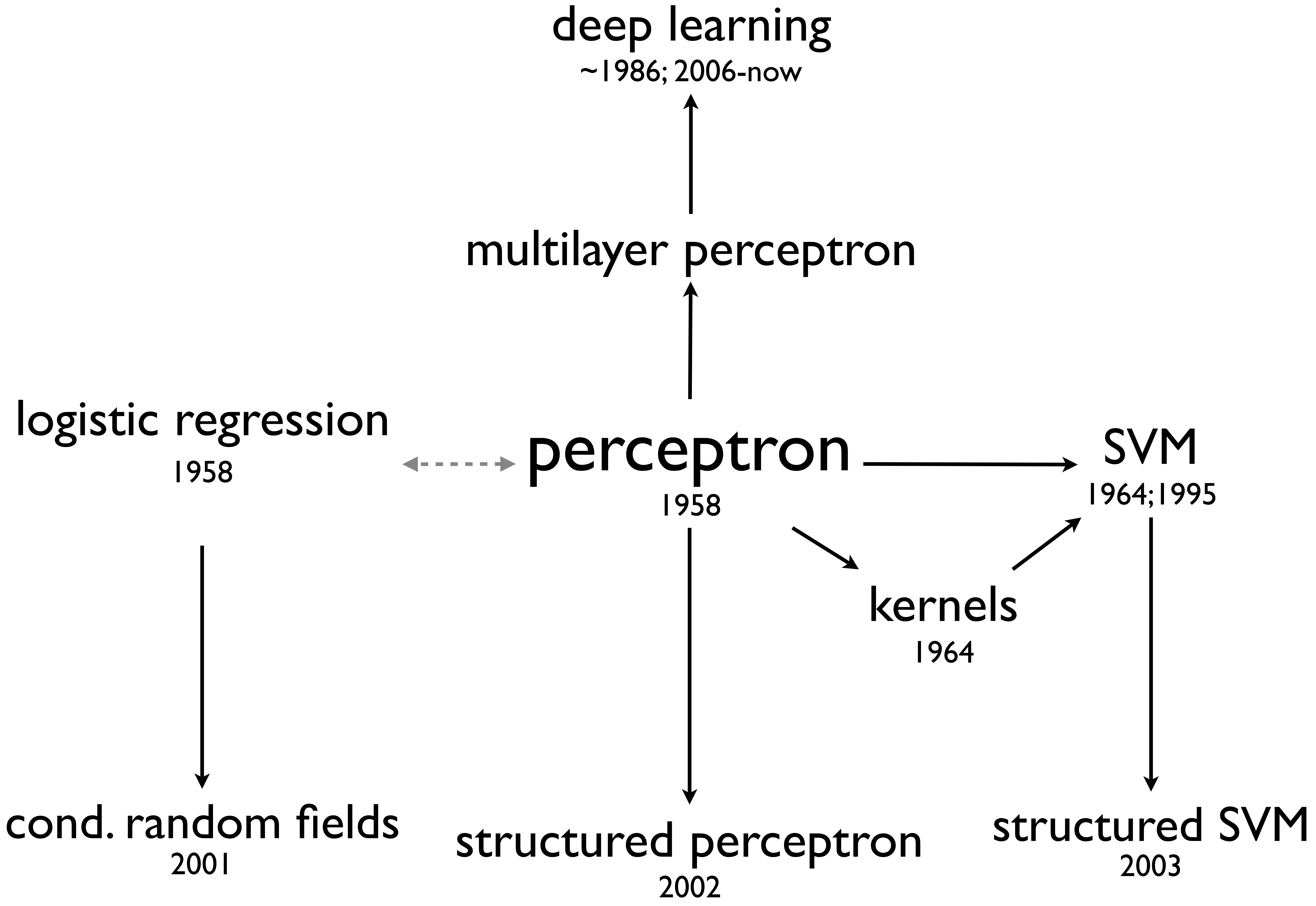
Part I

- Brief History of the Perceptron

Perceptron

(1959-now)

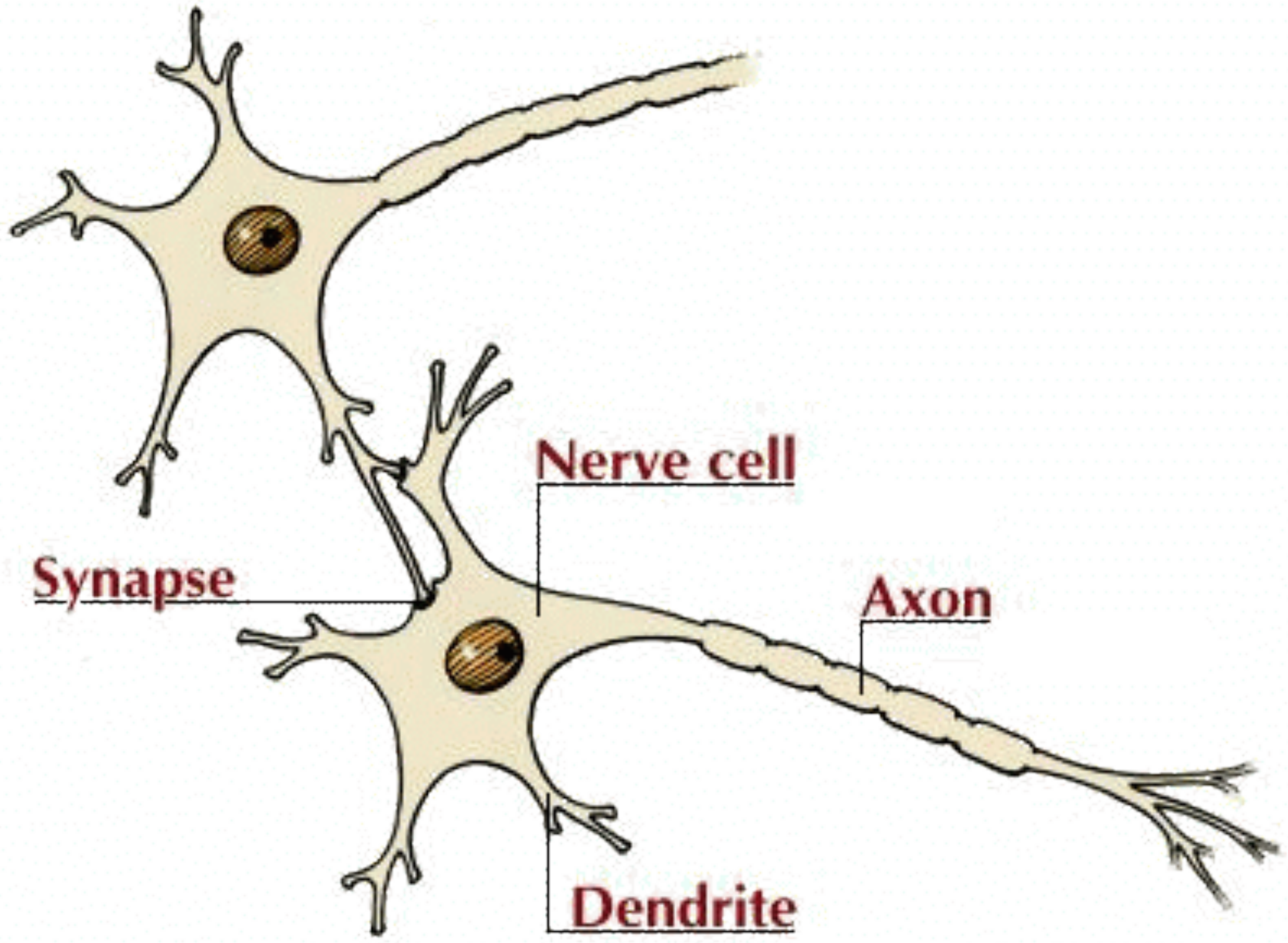
Frank Rosenblatt


deep learning
~1986; 2006-now
multilayer perceptron
logistic regression
1958
perceptron
1958
SVM
1964;1995
kernels
1964
cond. random fields
2001
structured perceptron
2002
structured SVM
2003

Neurons

- Soma (CPU)
  Cell body - combines signals

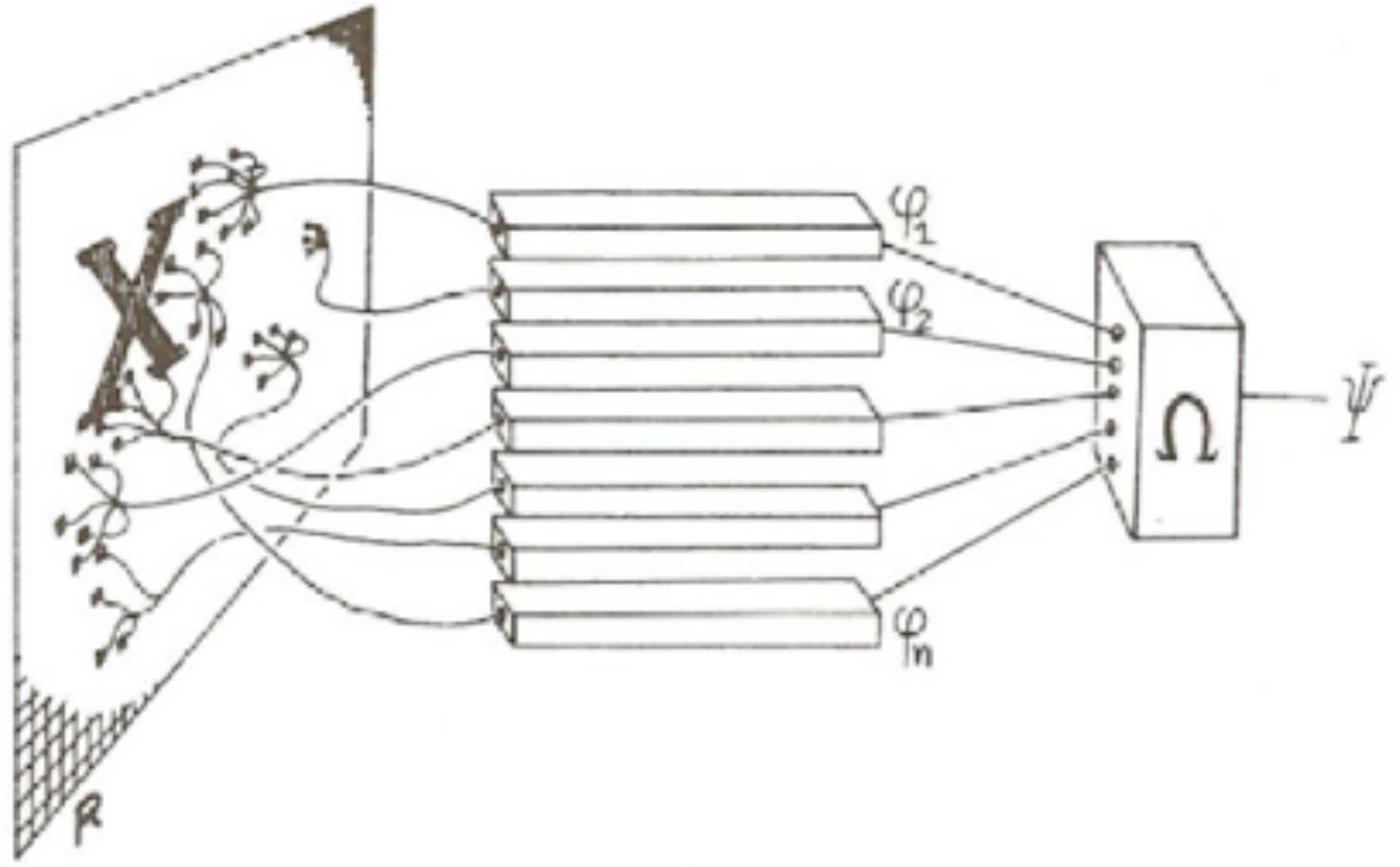
- Dendrite (input bus)
  Combines the inputs from
  several other nerve cells

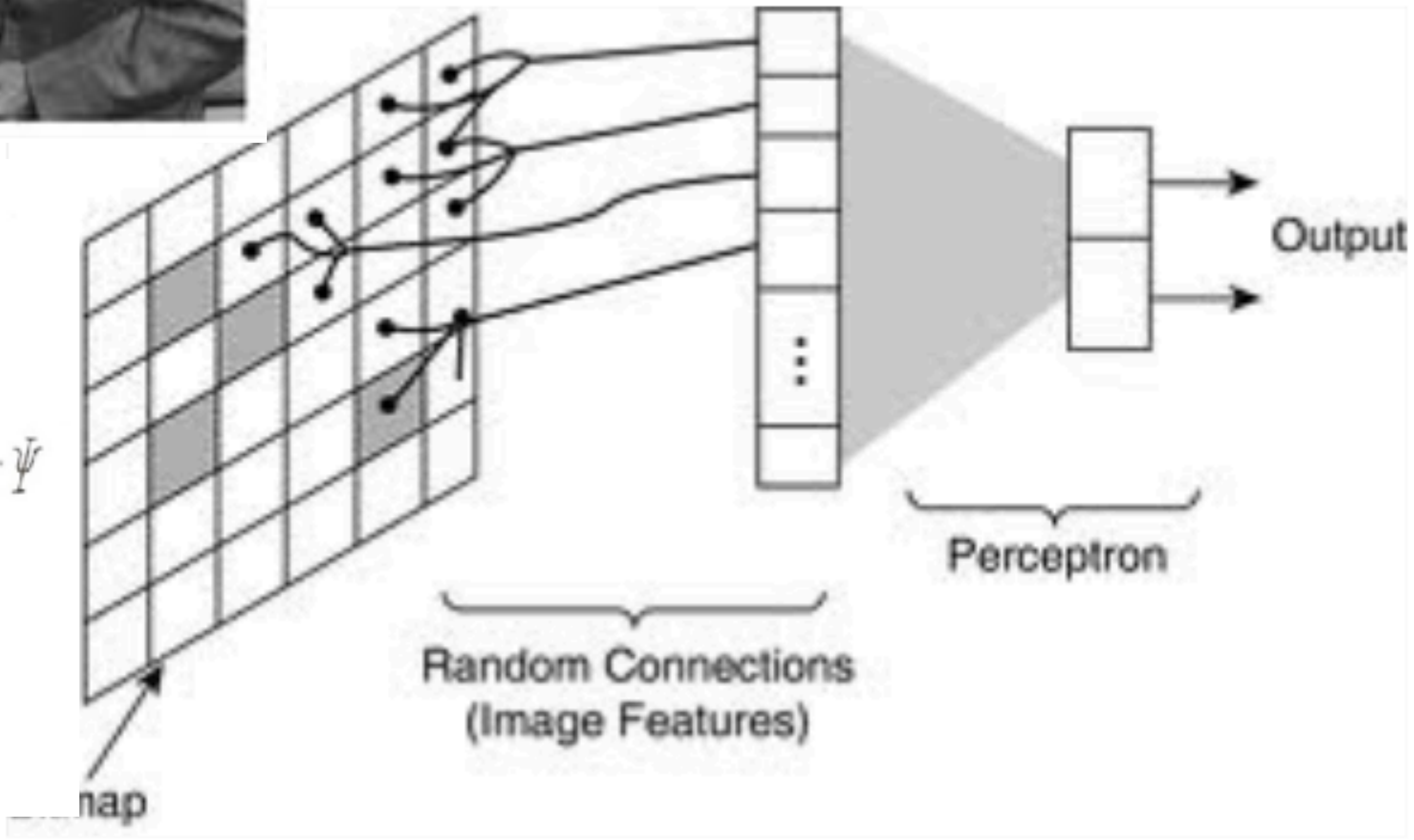


- Synapse (interface)
  Interface and parameter store between neurons

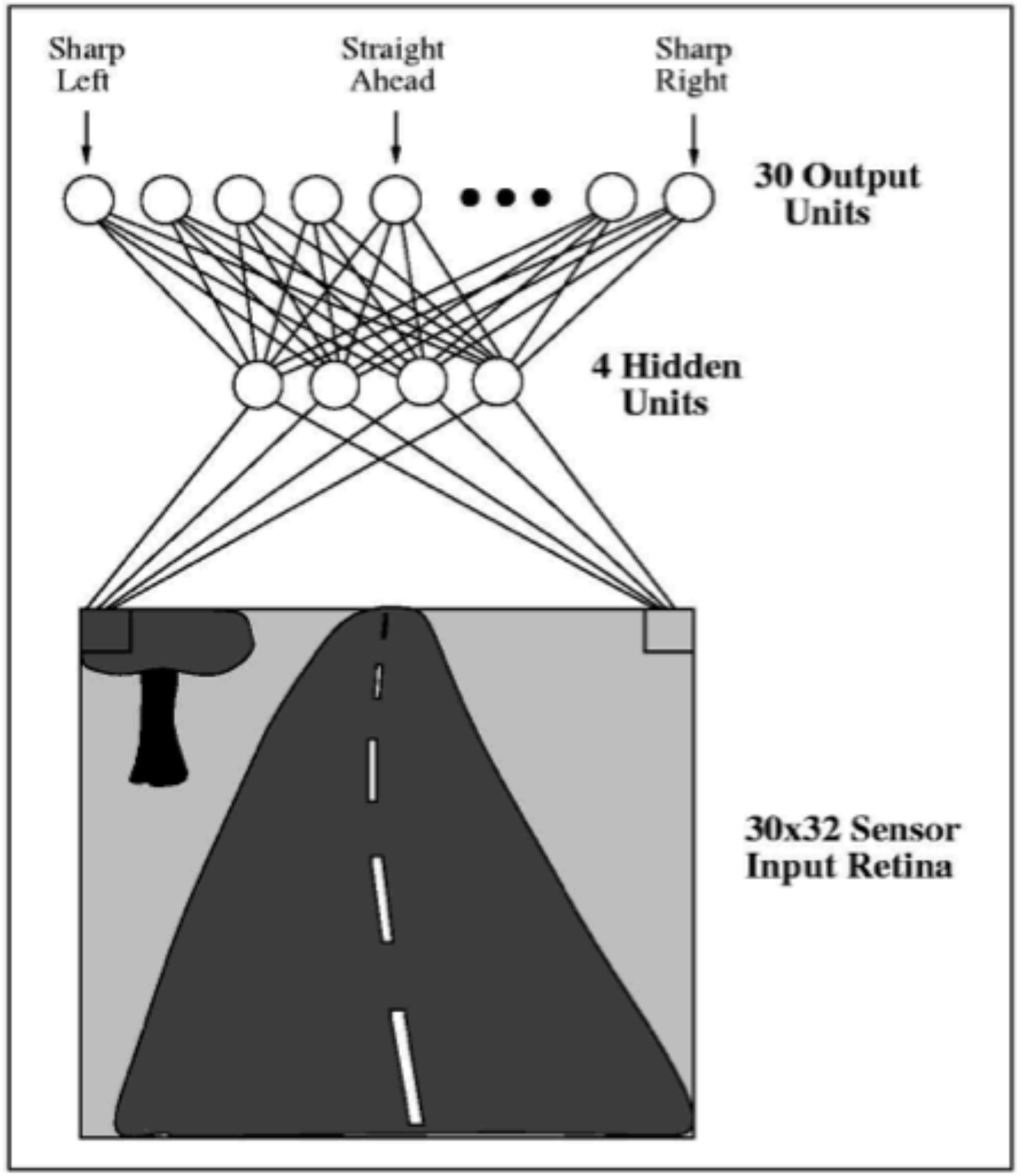
- Axon (output cable)
  May be up to 1m long and will transport the activation signal to neurons at different locations

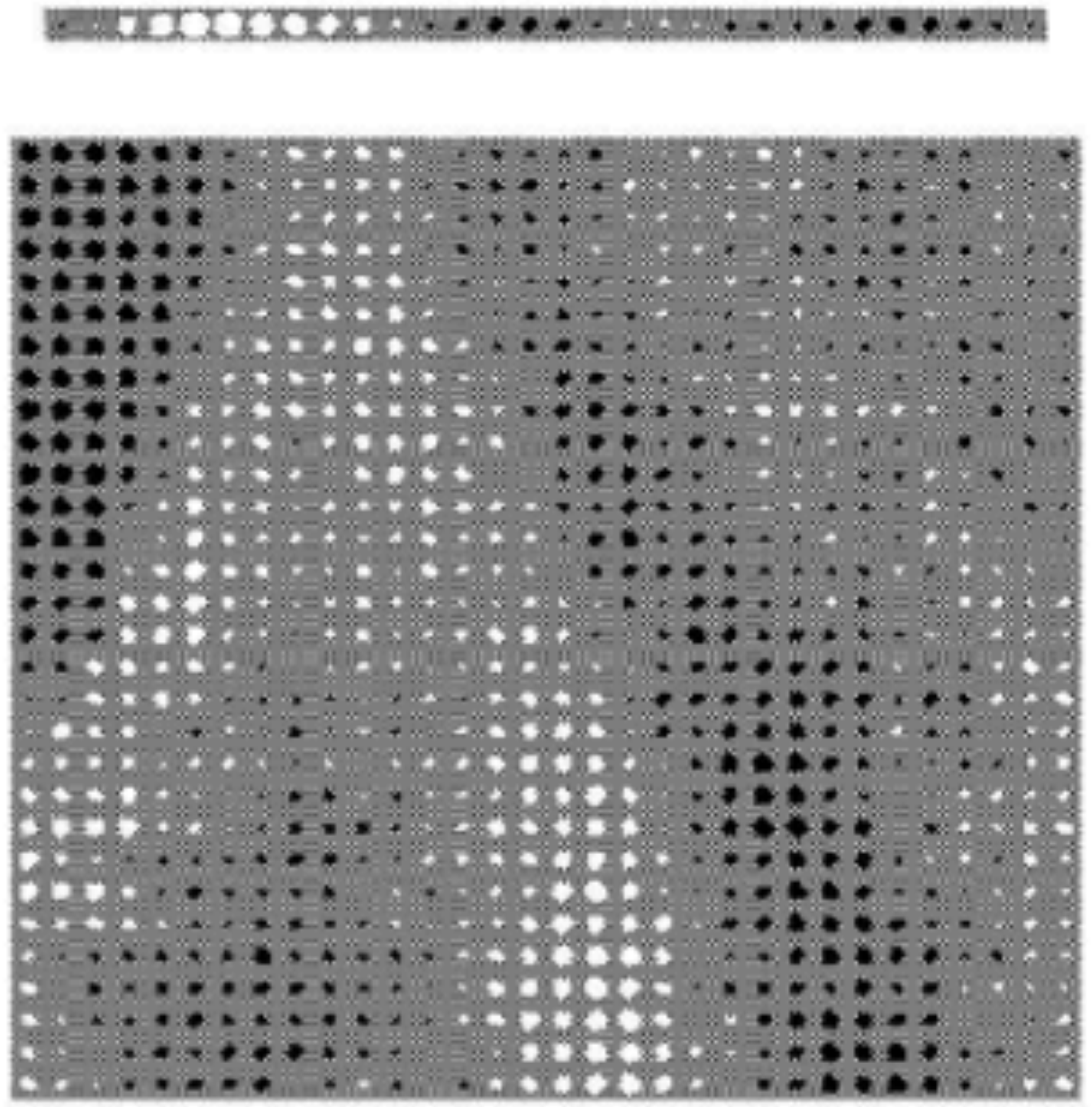
Frank Rosenblatt's Perceptron



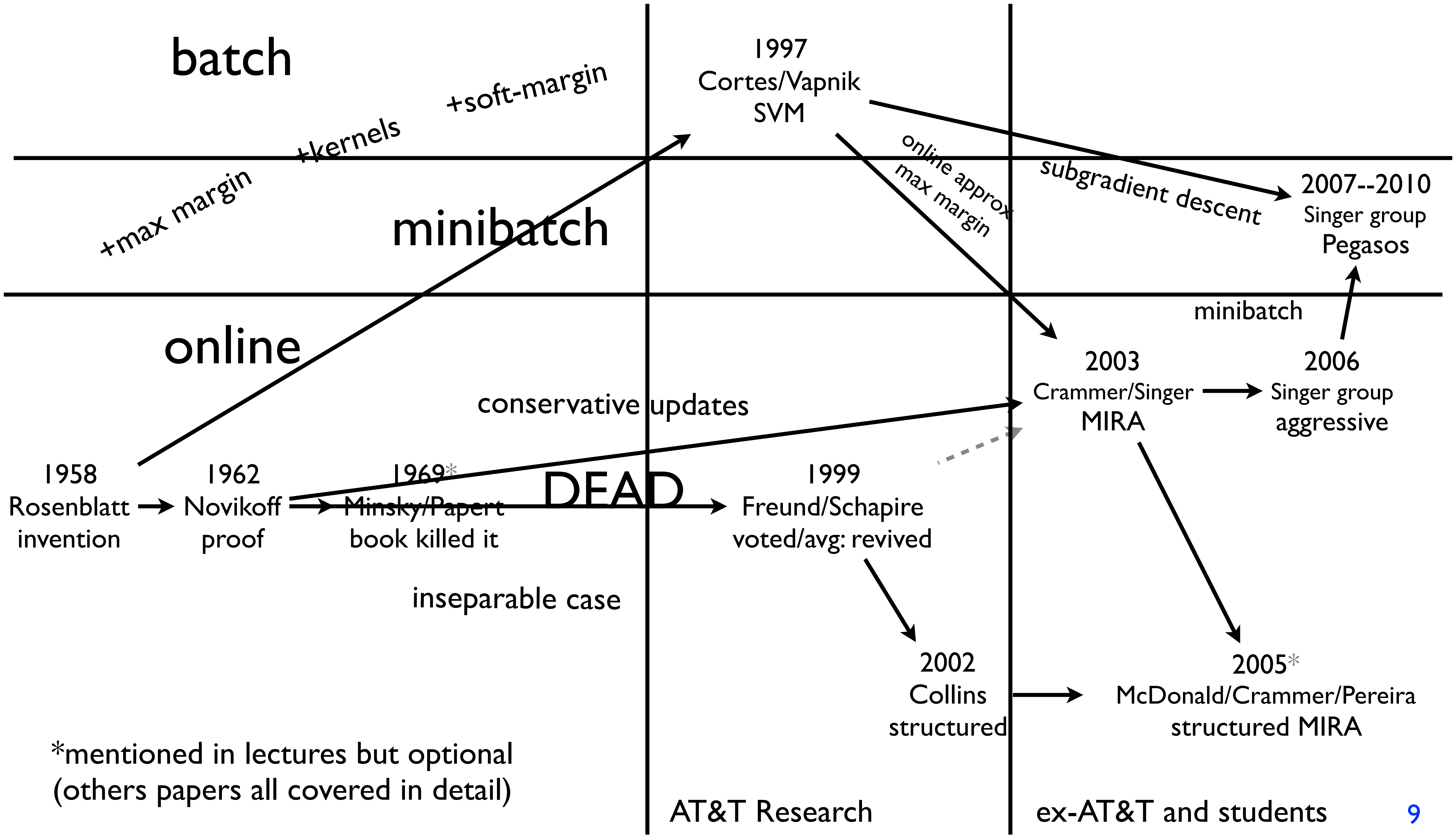


Multilayer Perceptron (Neural Net)





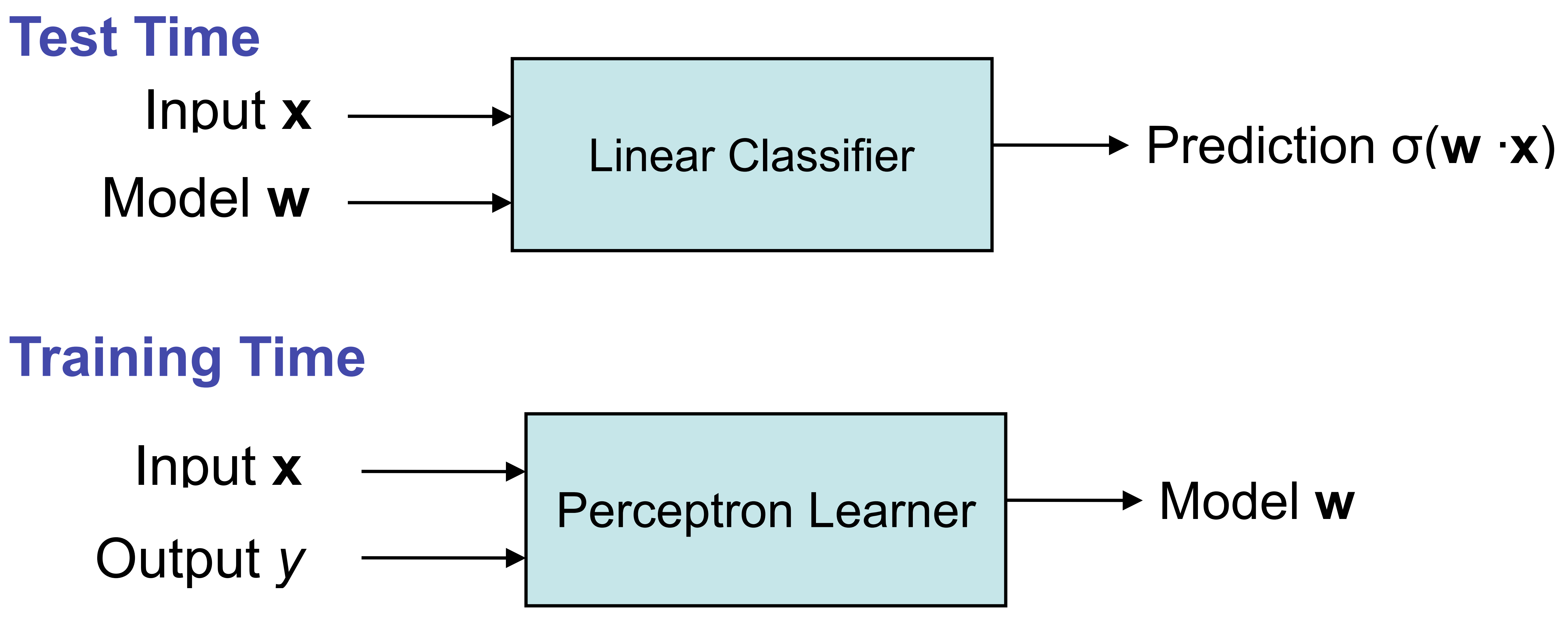
Brief History of Perceptron



*mentioned in lectures but optional
(others papers all covered in detail)

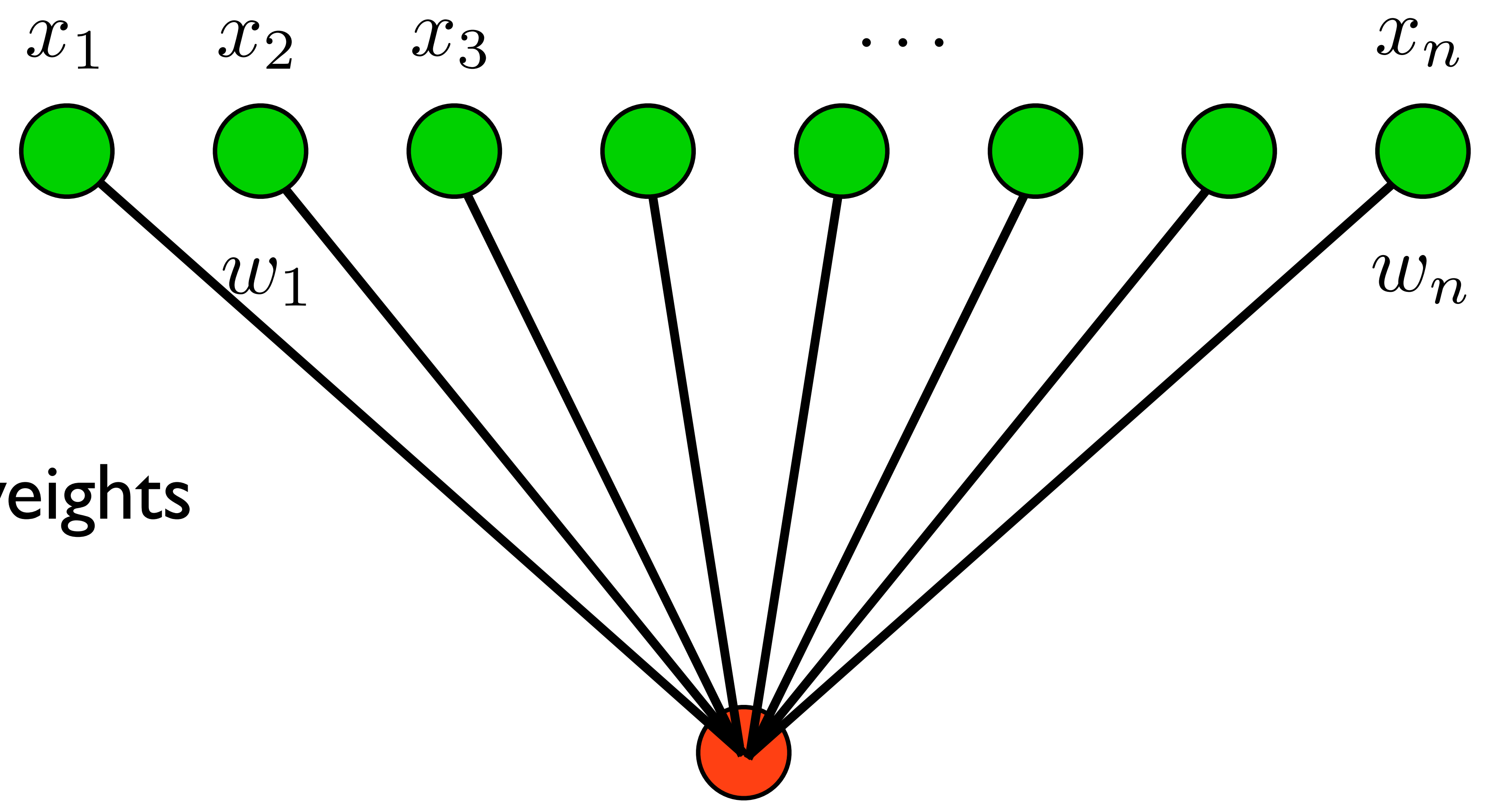
Part II

- Linear Classifier and Geometry (testing time)
  - decision boundary and normal vector **w**
  - not separable through the origin: add bias *b*
  - geometric review of linear algebra
  - augmented space (no explicit bias; implicit as $w_0=b$)



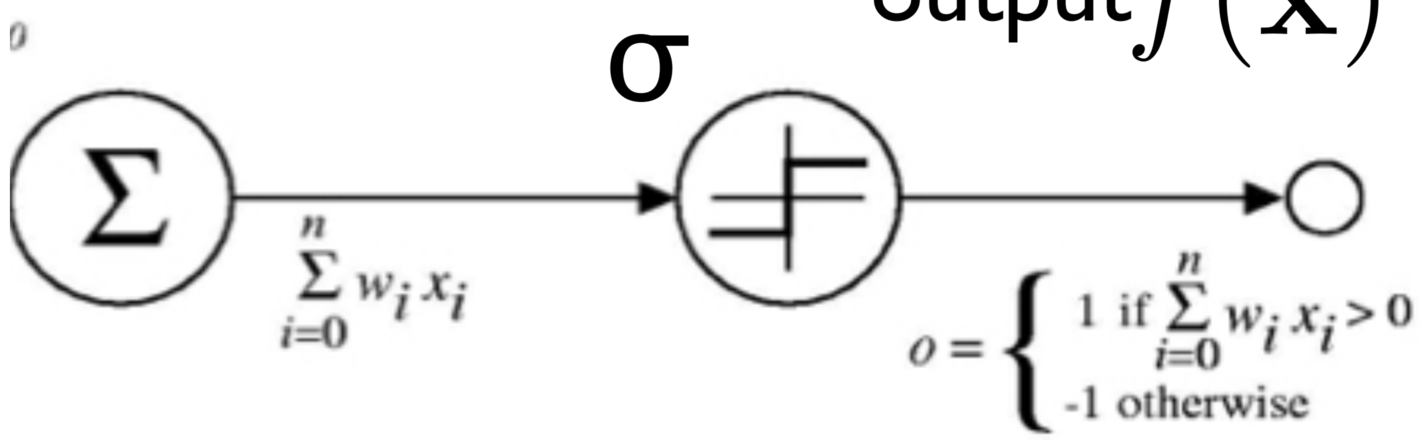
Linear Classifier and Geometry

linear classifiers: perceptron, logistic regression, (linear) SVMs, etc.

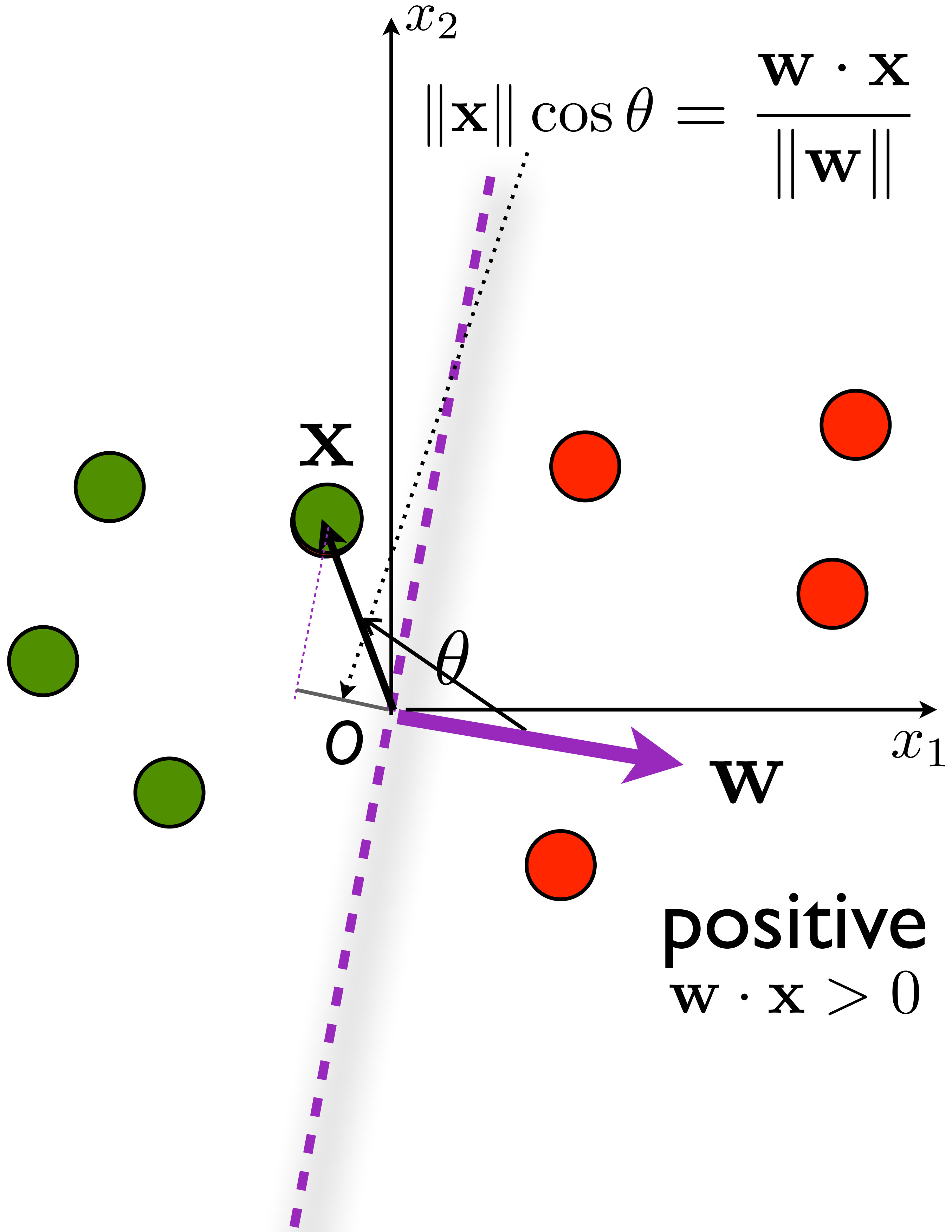


weights

output $f(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x})$



negative
$\mathbf{w} \cdot \mathbf{x} < 0$



weight vector **w** is a "prototype" of positive examples
it's also the normal vector of the decision boundary
meaning of **w**·**x**: agreement with positive direction

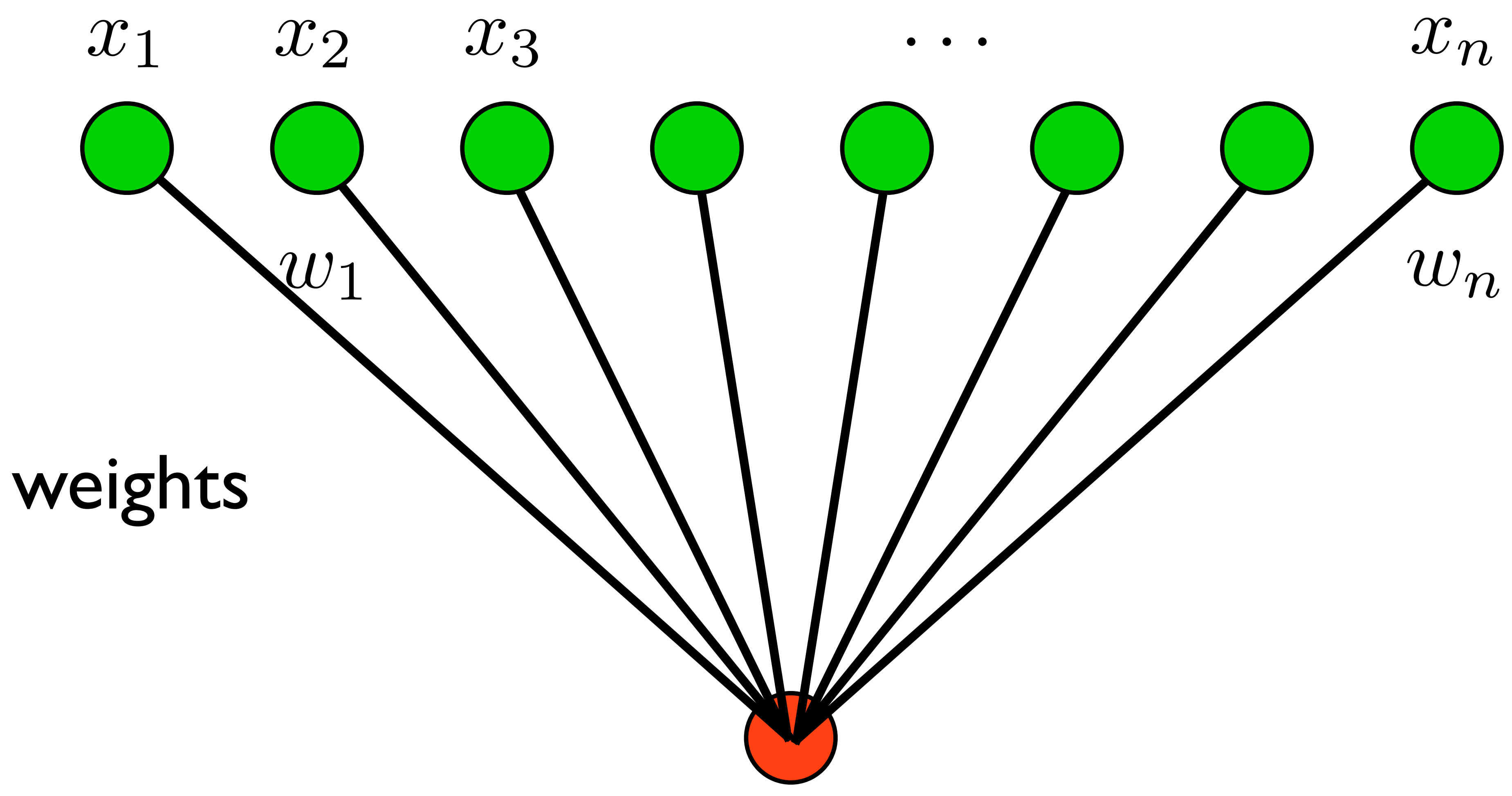
test: input: **x**, **w**; output: 1 if **w**·**x** >0 else -1

training: input: (**x**, *y*) pairs; output: **w**

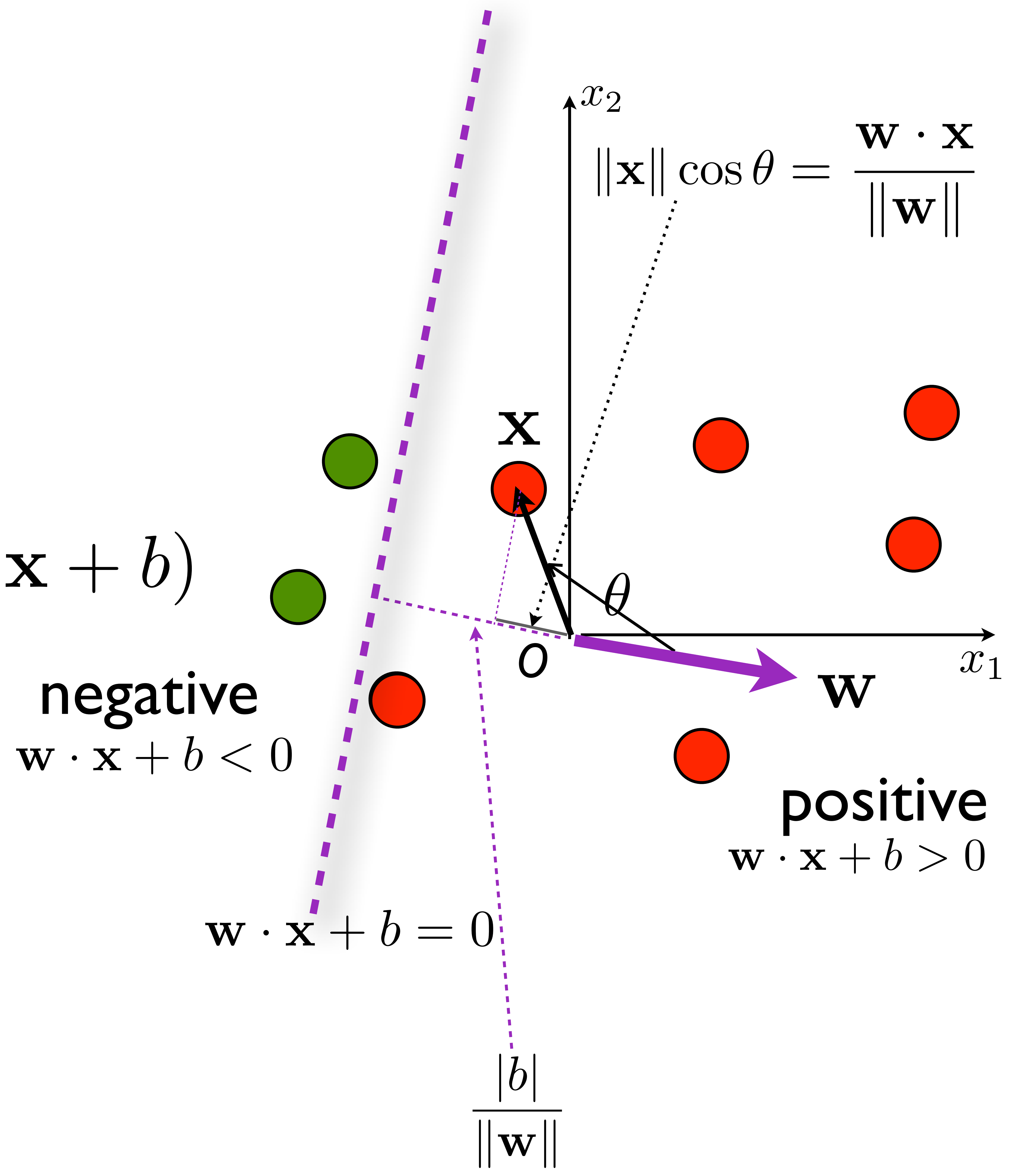
separating hyperplane
(decision boundary)
$\mathbf{w} \cdot \mathbf{x} = 0$

What if not separable through origin?

solution: add bias *b*

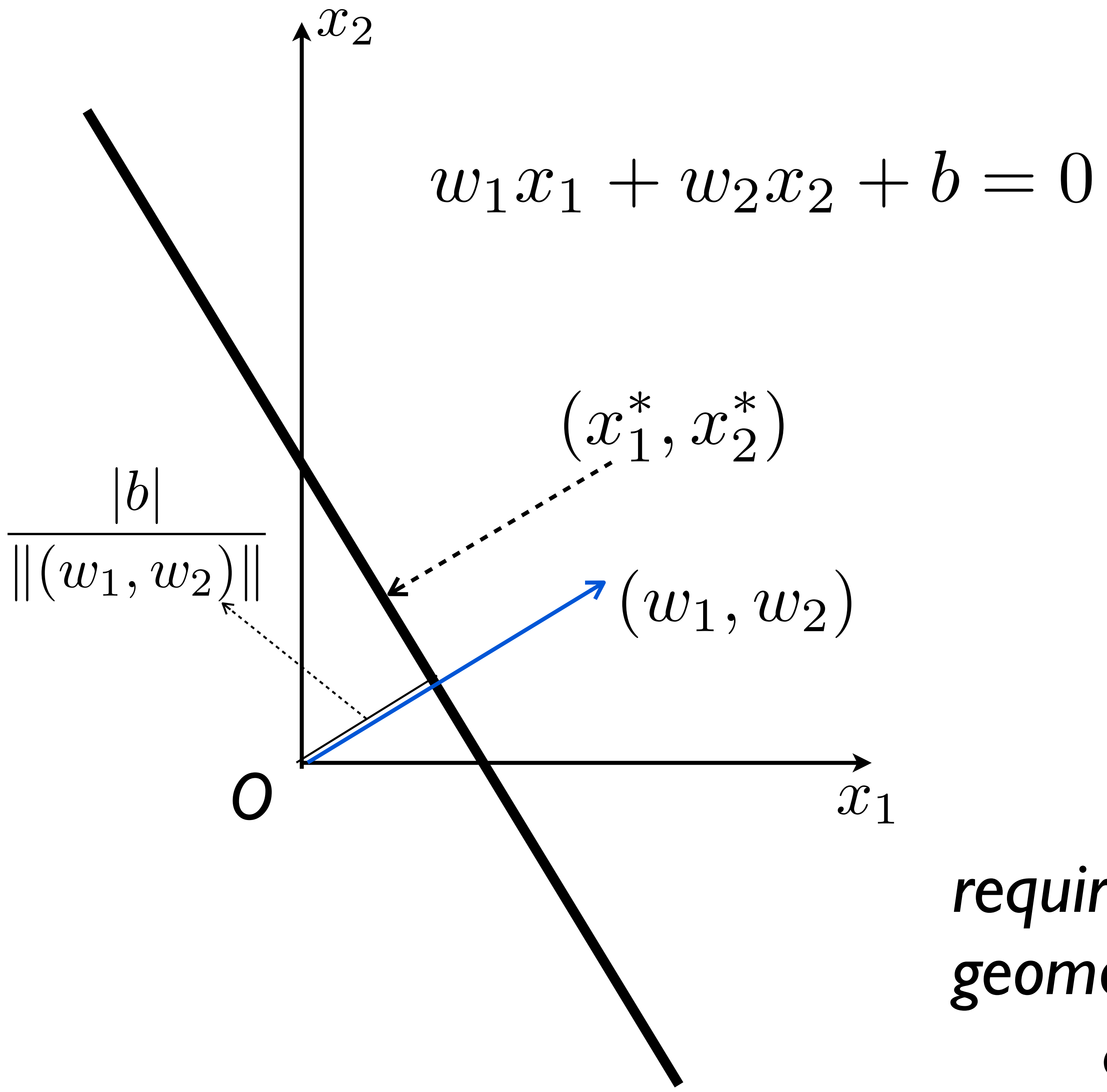


$f(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$

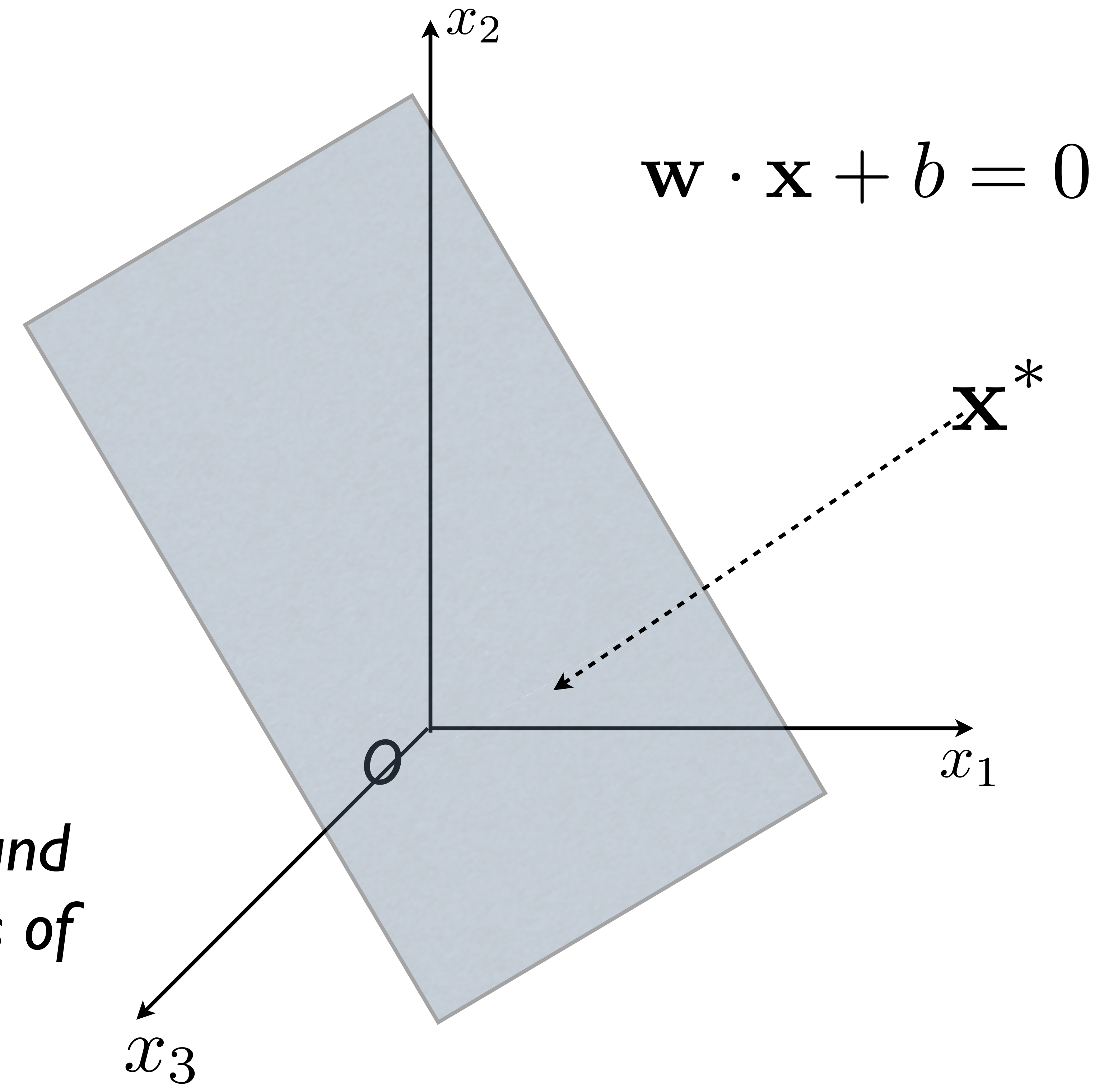


Geometric Review of Linear Algebra

line in 2D



(n-1)-dim hyperplane in n-dim



required: algebraic and geometric meanings of dot product

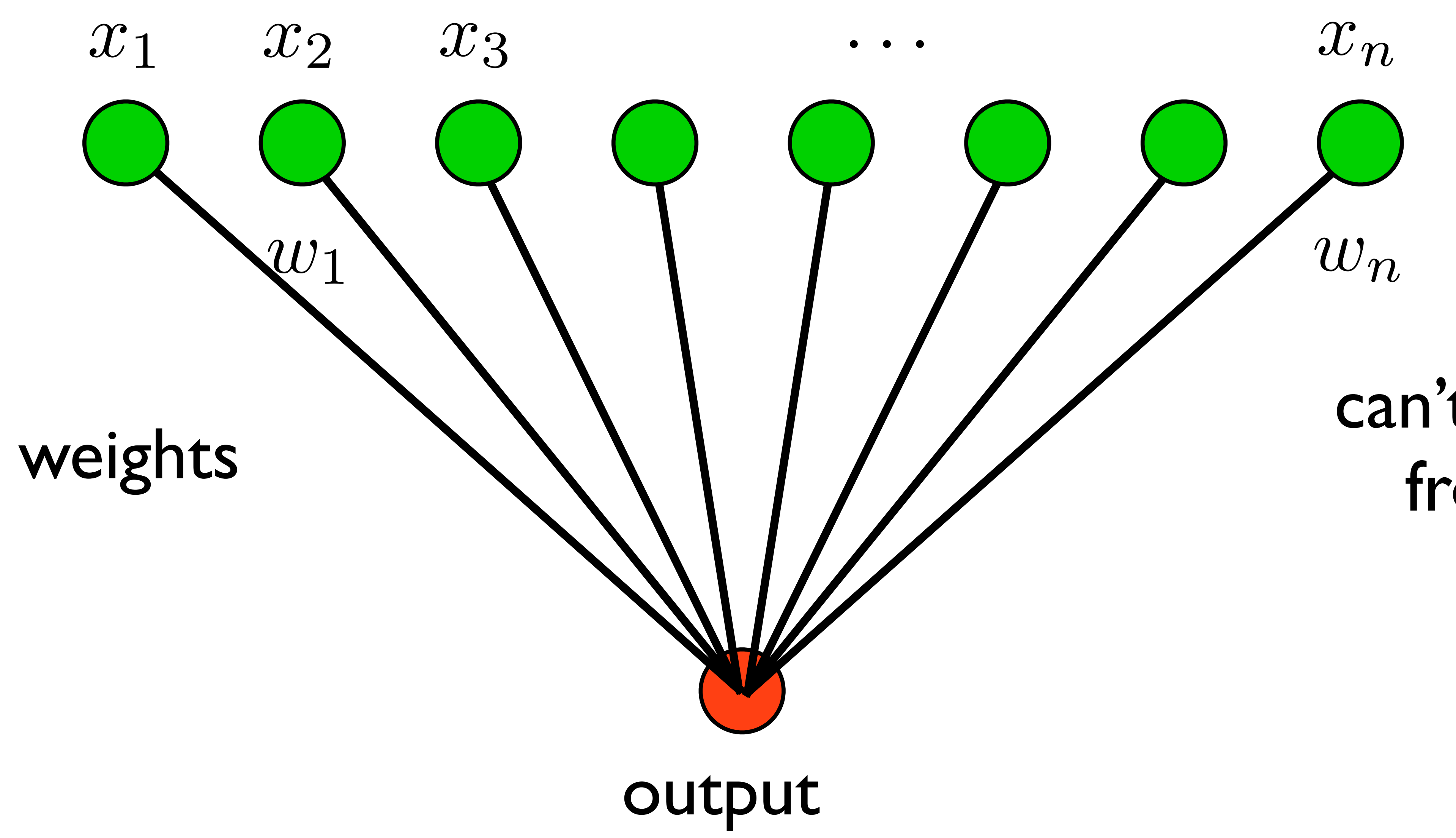
$$\frac{|w_1 x_1^* + w_2 x_2^* + b|}{\sqrt{w_1^2 + w_2^2}} = \frac{|(w_1, w_2) \cdot (x_1, x_2) + b|}{\|(w_1, w_2)\|}$$

point-to-line distance

$$\frac{|\mathbf{w} \cdot \mathbf{x} + b|}{\|\mathbf{w}\|}$$

point-to-hyperplane distance

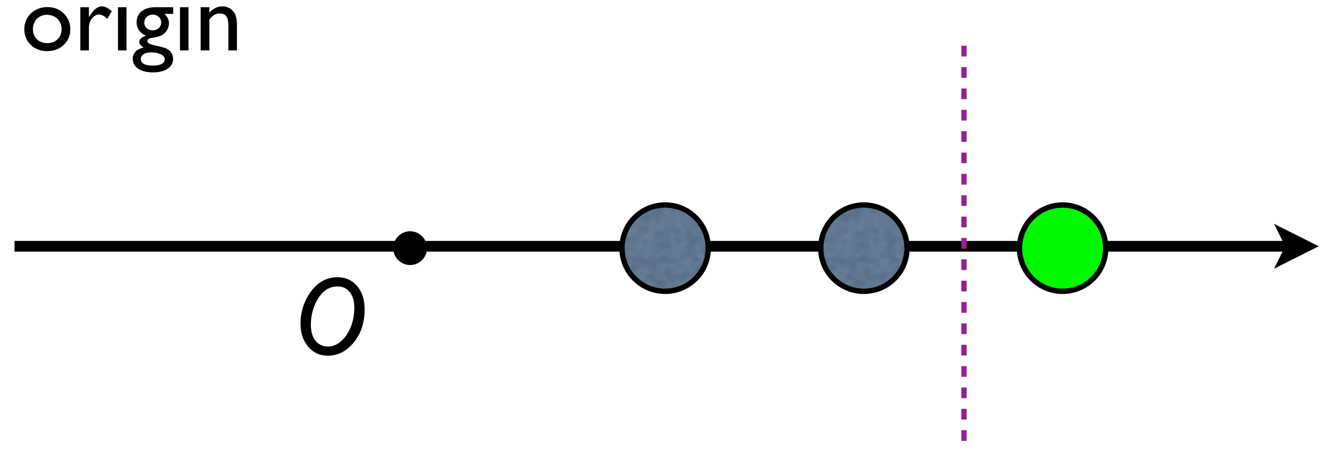
Augmented Space: dimensionality+1

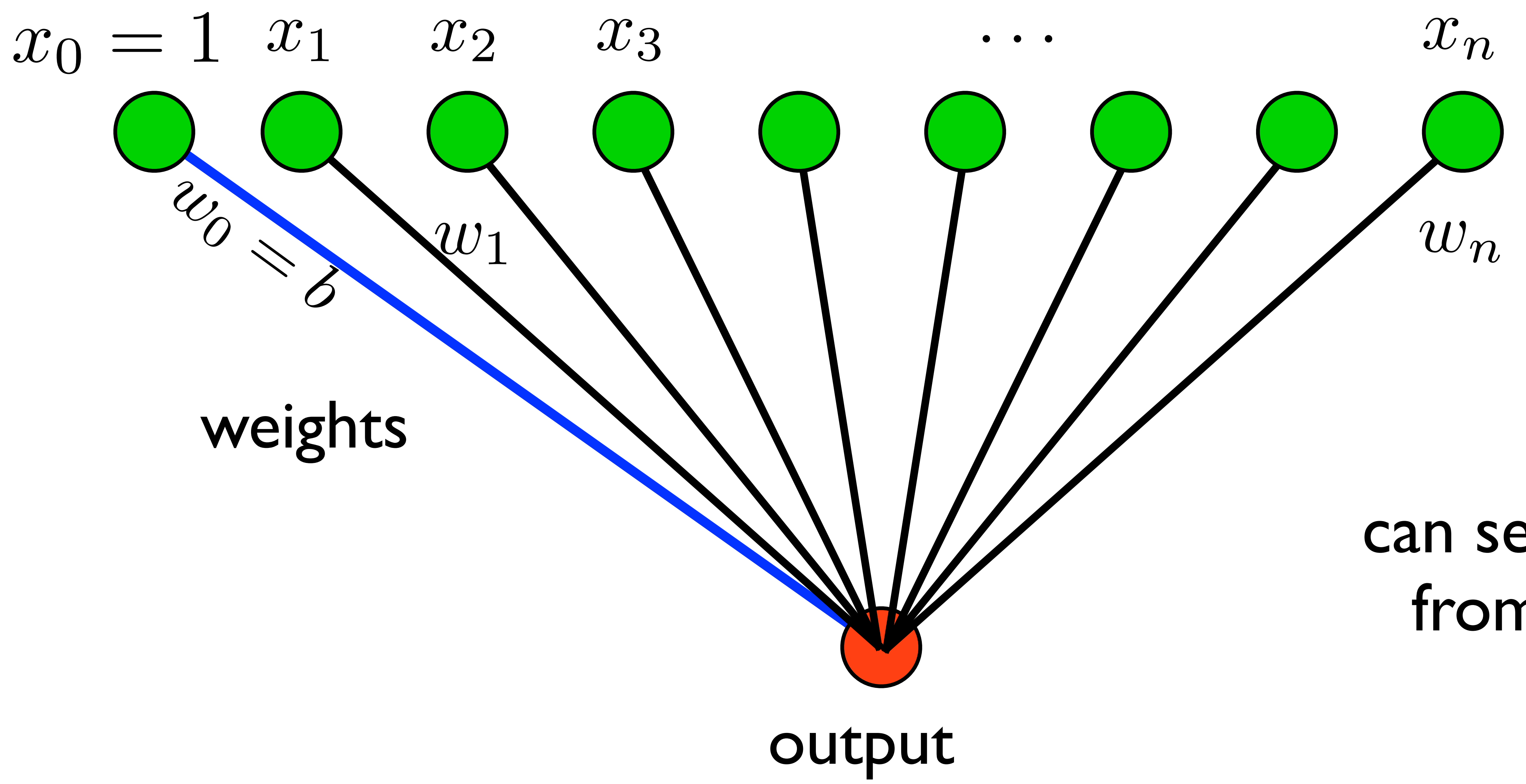


explicit bias

$$f(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$$

can't separate in 1D
from the origin

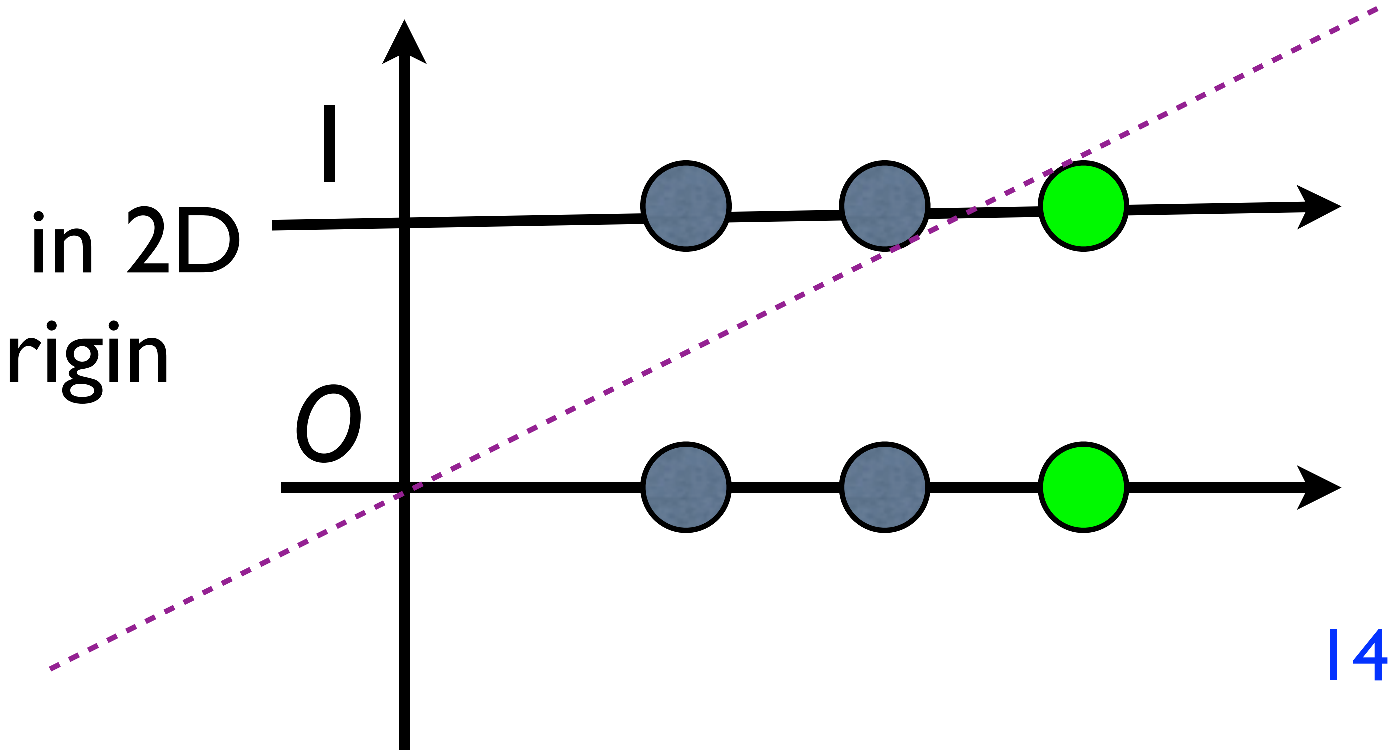




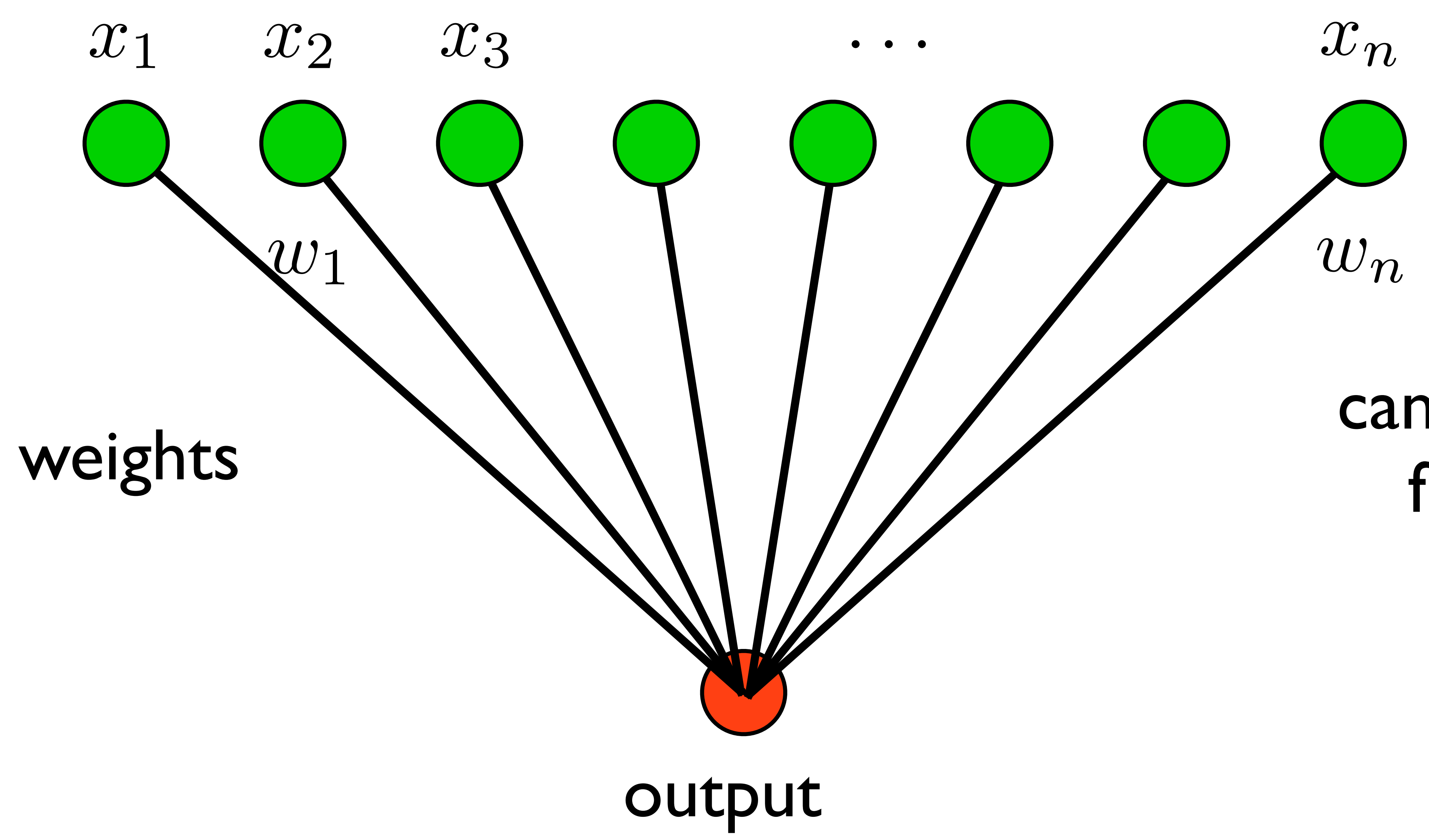
augmented space

$$f(\mathbf{x}) = \sigma((b; \mathbf{w}) \cdot (1; \mathbf{x}))$$

can separate in 2D
from the origin



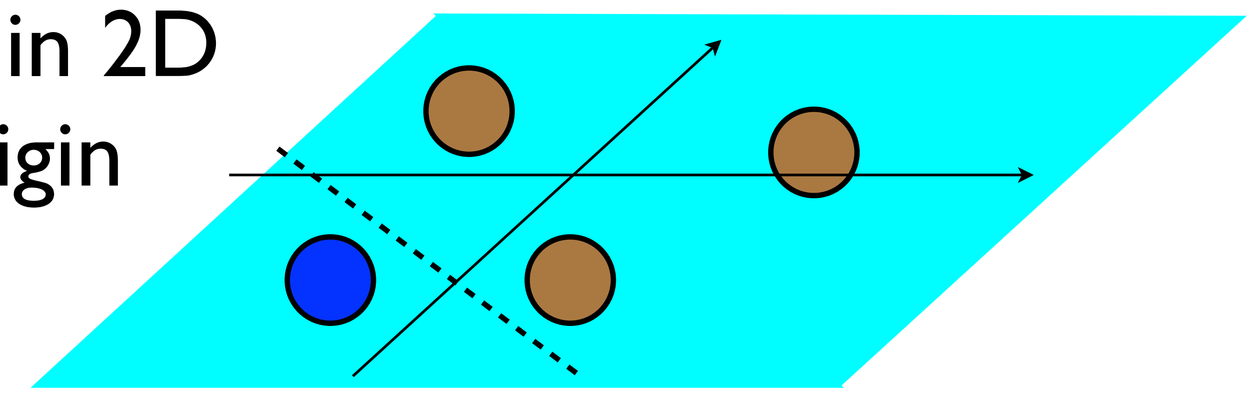
Augmented Space: dimensionality+1

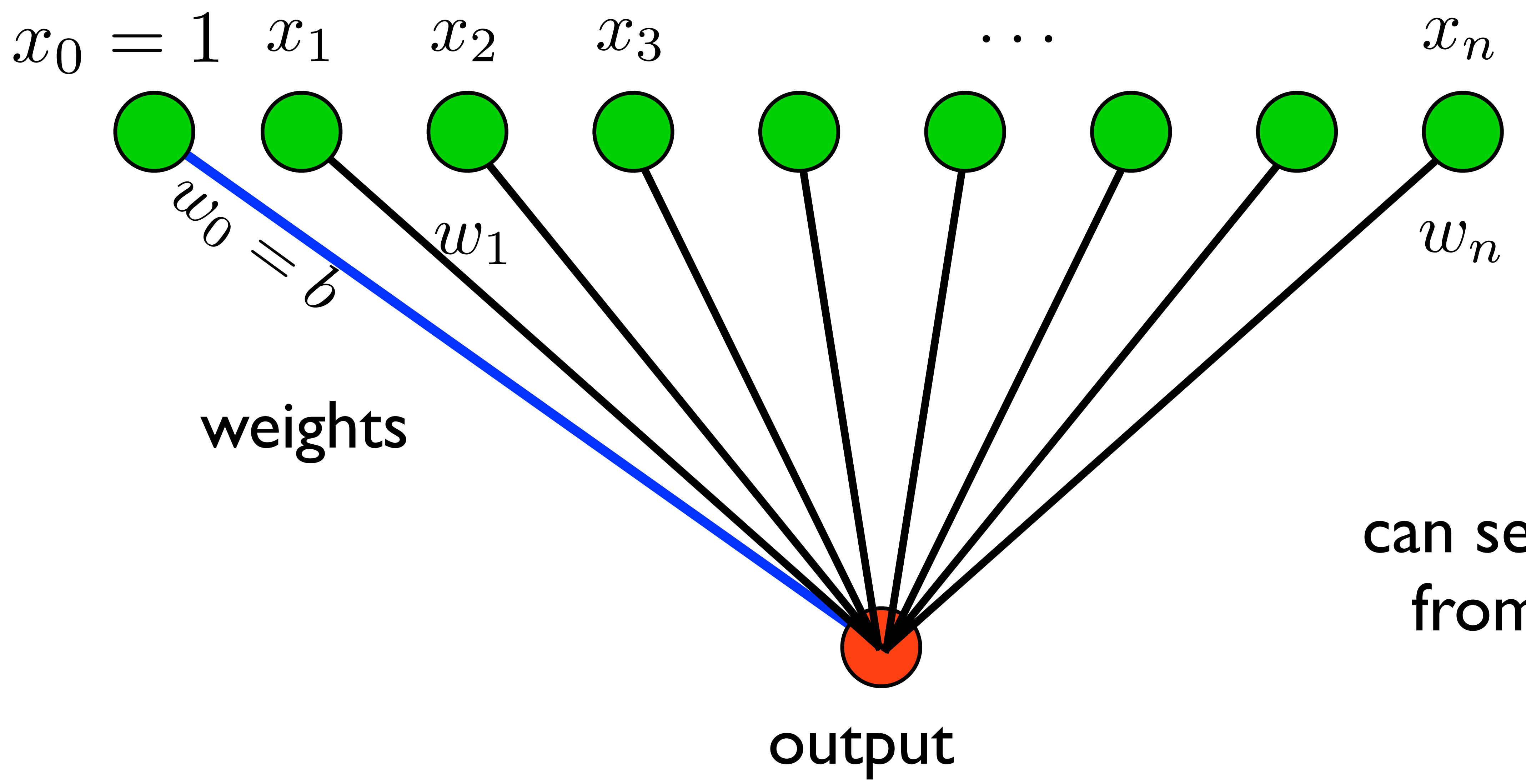


explicit bias

$$f(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$$

can't separate in 2D from the origin

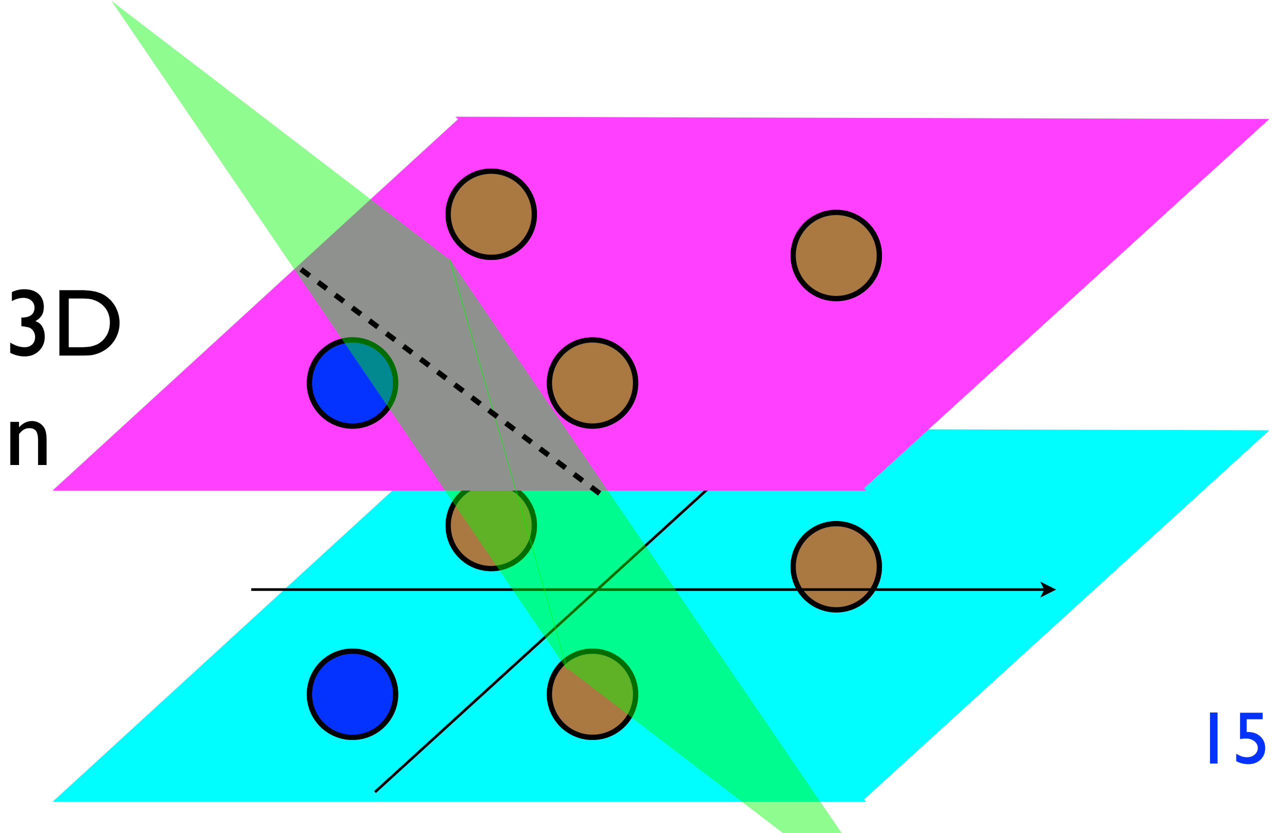




augmented space

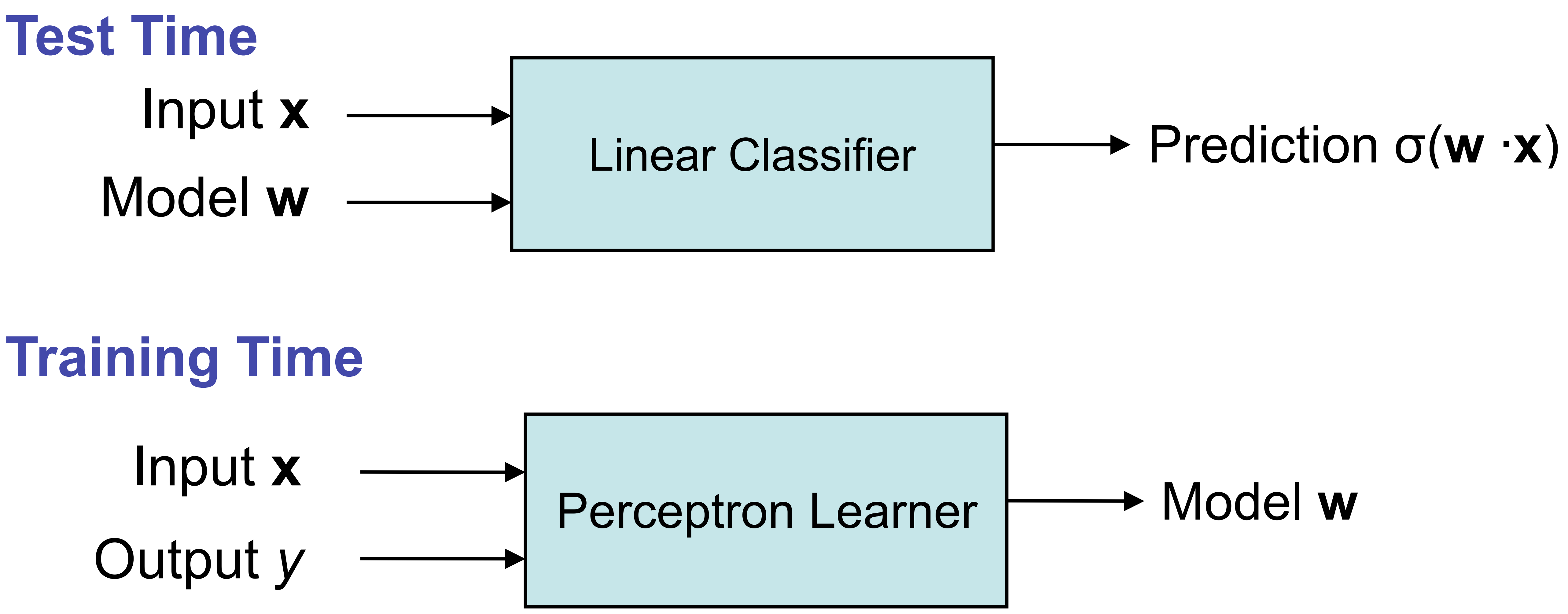
$$f(\mathbf{x}) = \sigma((b; \mathbf{w}) \cdot (1; \mathbf{x}))$$

can separate in 3D from the origin

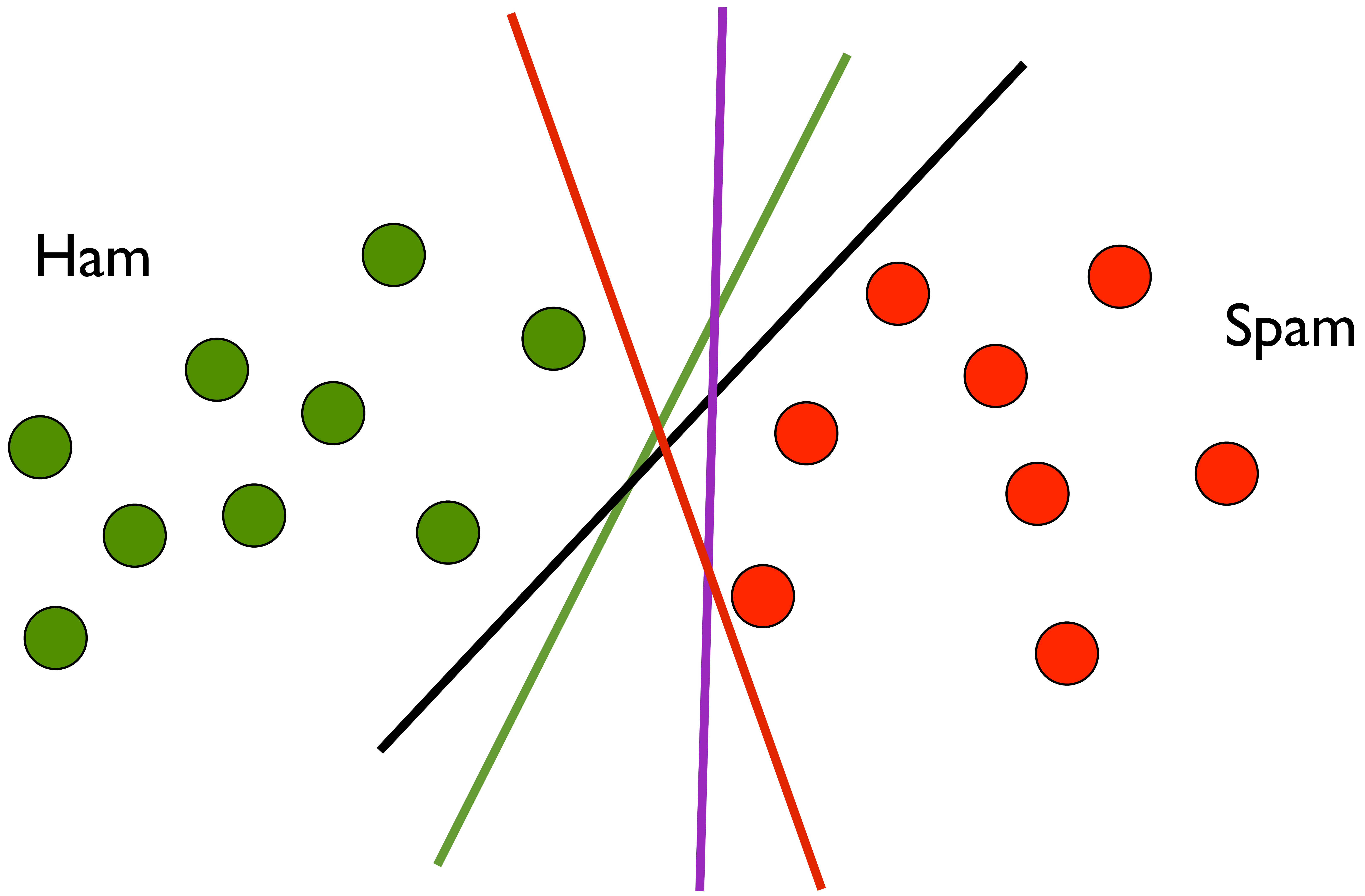


Part III

- The Perceptron Learning Algorithm (training time)
  - the version without bias (augmented space)
  - side note on mathematical notations
  - mini-demo

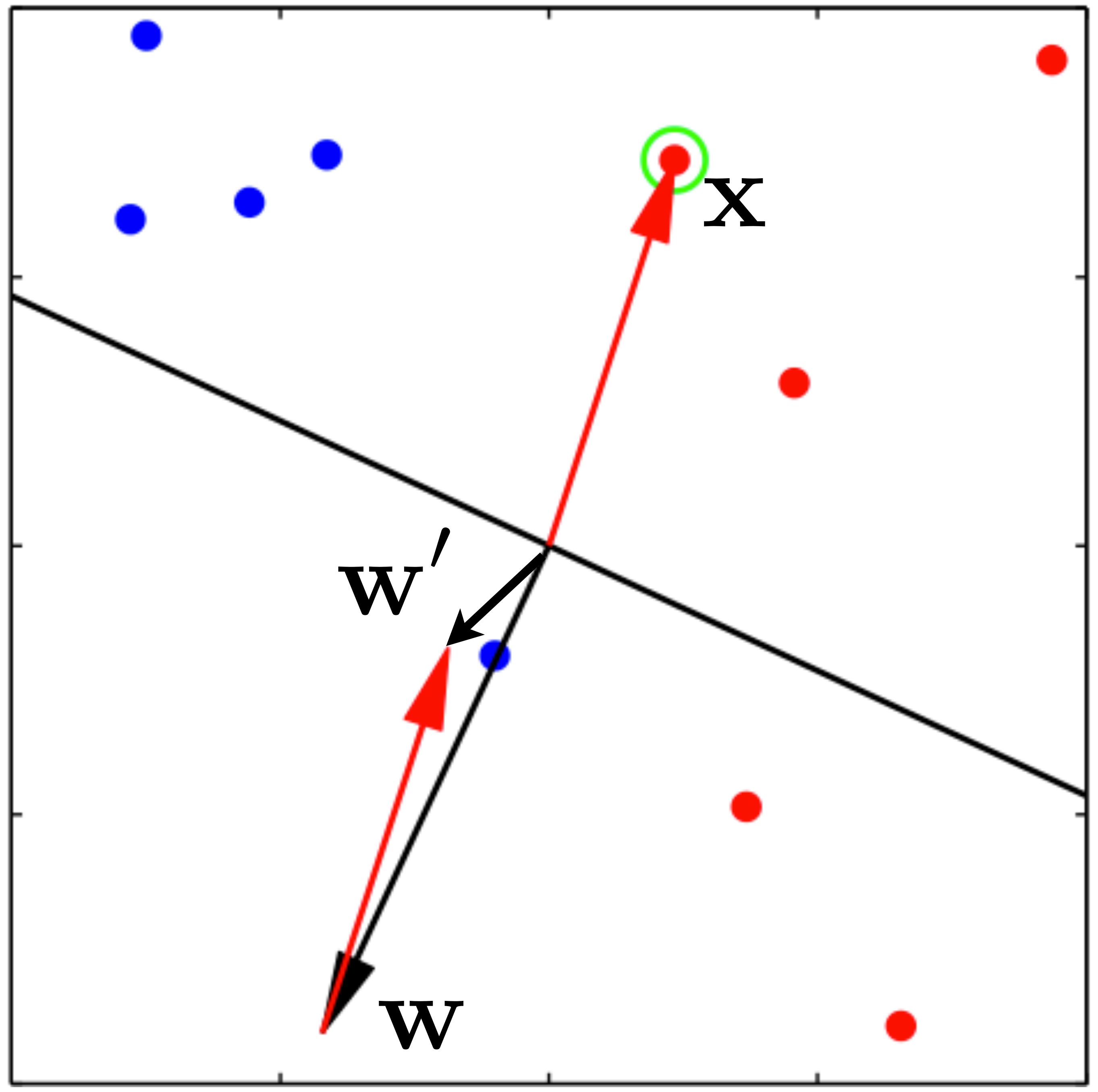


Perceptron



The Perceptron Algorithm

input: training data D
output: weights $\mathbf{w}$
initialize $\mathbf{w} \leftarrow \mathbf{0}$
while not converged
  for $(\mathbf{x}, y) \in D$
    if $y(\mathbf{w} \cdot \mathbf{x}) \leq 0$
      $\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$



- the simplest machine learning algorithm
- keep cycling through the training data
  - update **w** if there is a mistake on example (**x**, *y*)
- until all examples are classified correctly

Side Note on Mathematical Notations

- I'll try my best to be consistent in notations
  - e.g., bold-face for vectors, italic for scalars, etc.
- avoid unnecessary superscripts and subscripts by using a "Pythonic" rather than a "C" notational style
  - most textbooks have consistent but bad notations

initialize $\mathbf{w} \leftarrow \mathbf{0}$
while not converged
  for $(\mathbf{x}, y) \in D$
    if $y(\mathbf{w} \cdot \mathbf{x}) \leq 0$
      $\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$

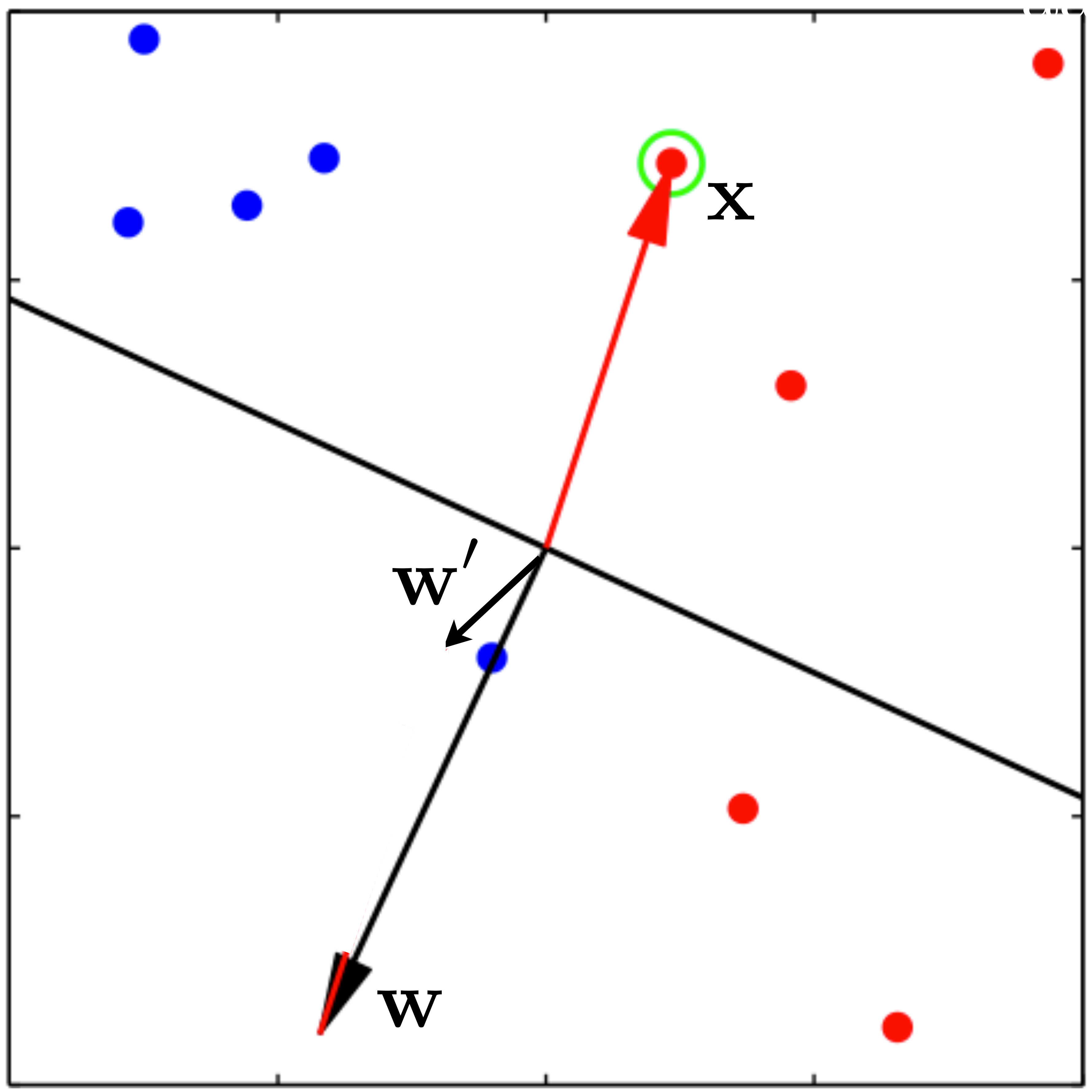
good notations:
consistent, Pythonic style

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

bad notations:
inconsistent, unnecessary *i* and *b*

Demo

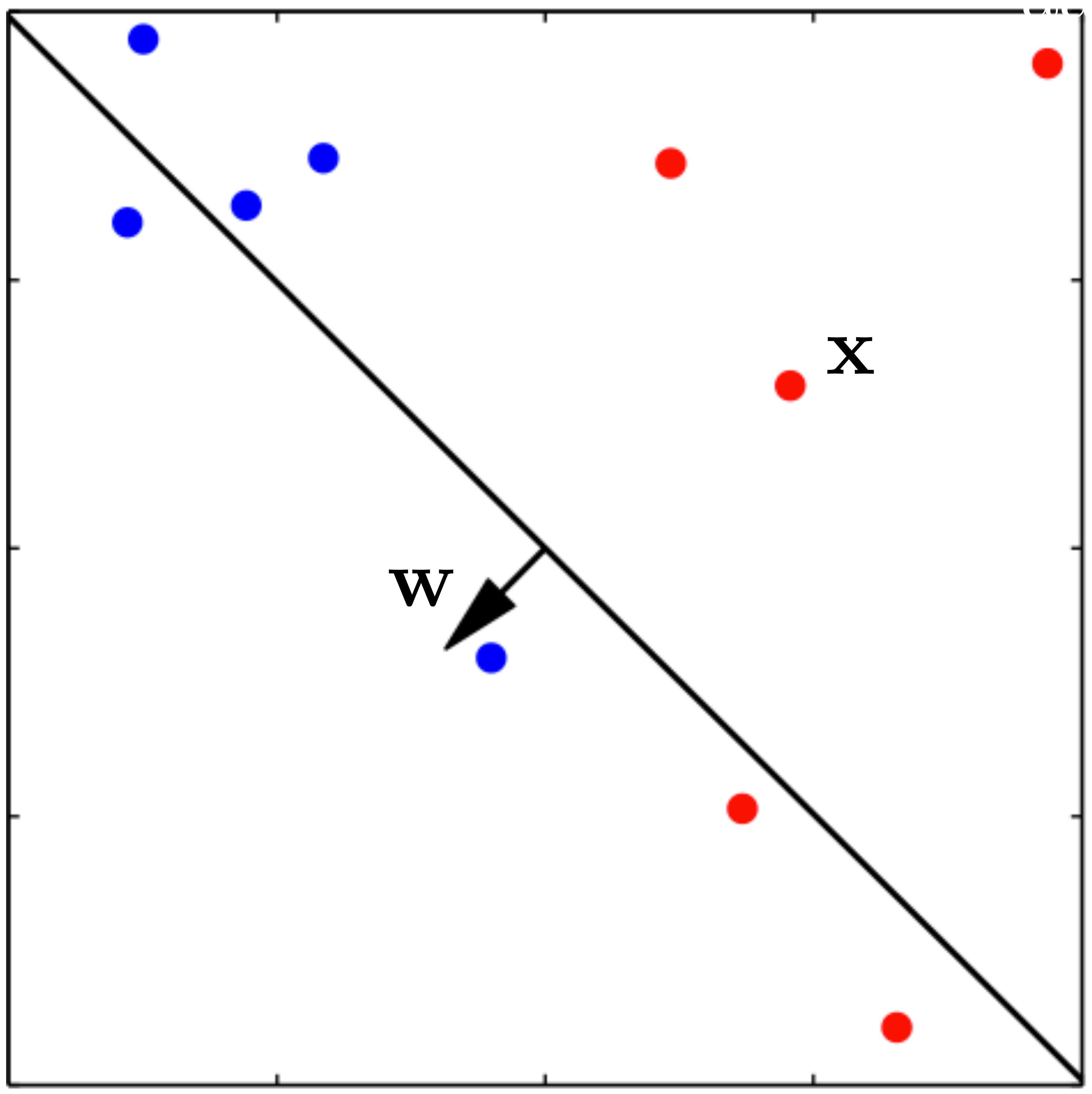
while not converged
  for $(\mathbf{x}, y) \in D$
    if $y(\mathbf{w} \cdot \mathbf{x}) \leq 0$
      $\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$



(bias=0)

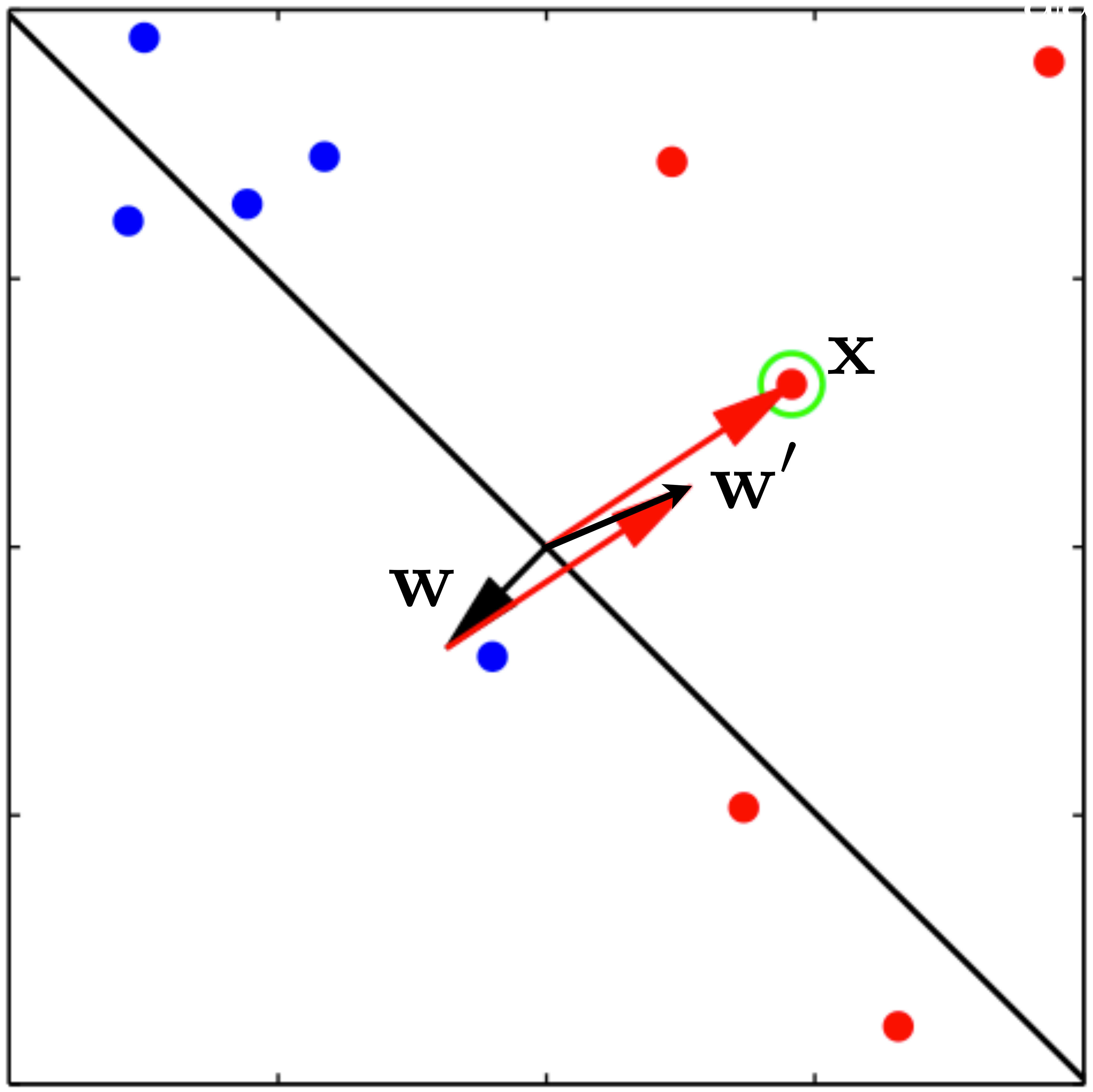
Demo

while not converged
  for $(\mathbf{x}, y) \in D$
    if $y(\mathbf{w} \cdot \mathbf{x}) \leq 0$
      $\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$



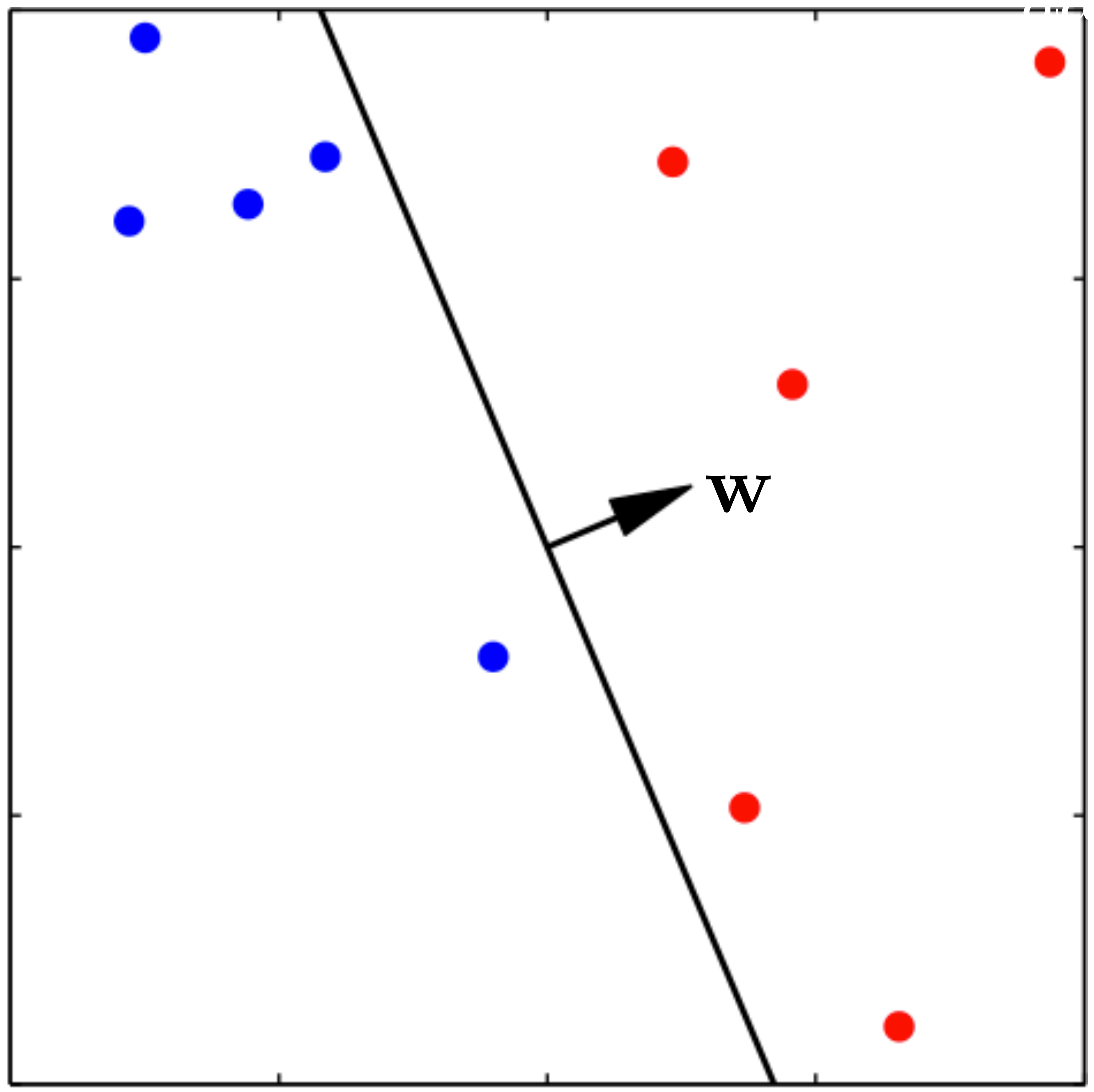
Demo

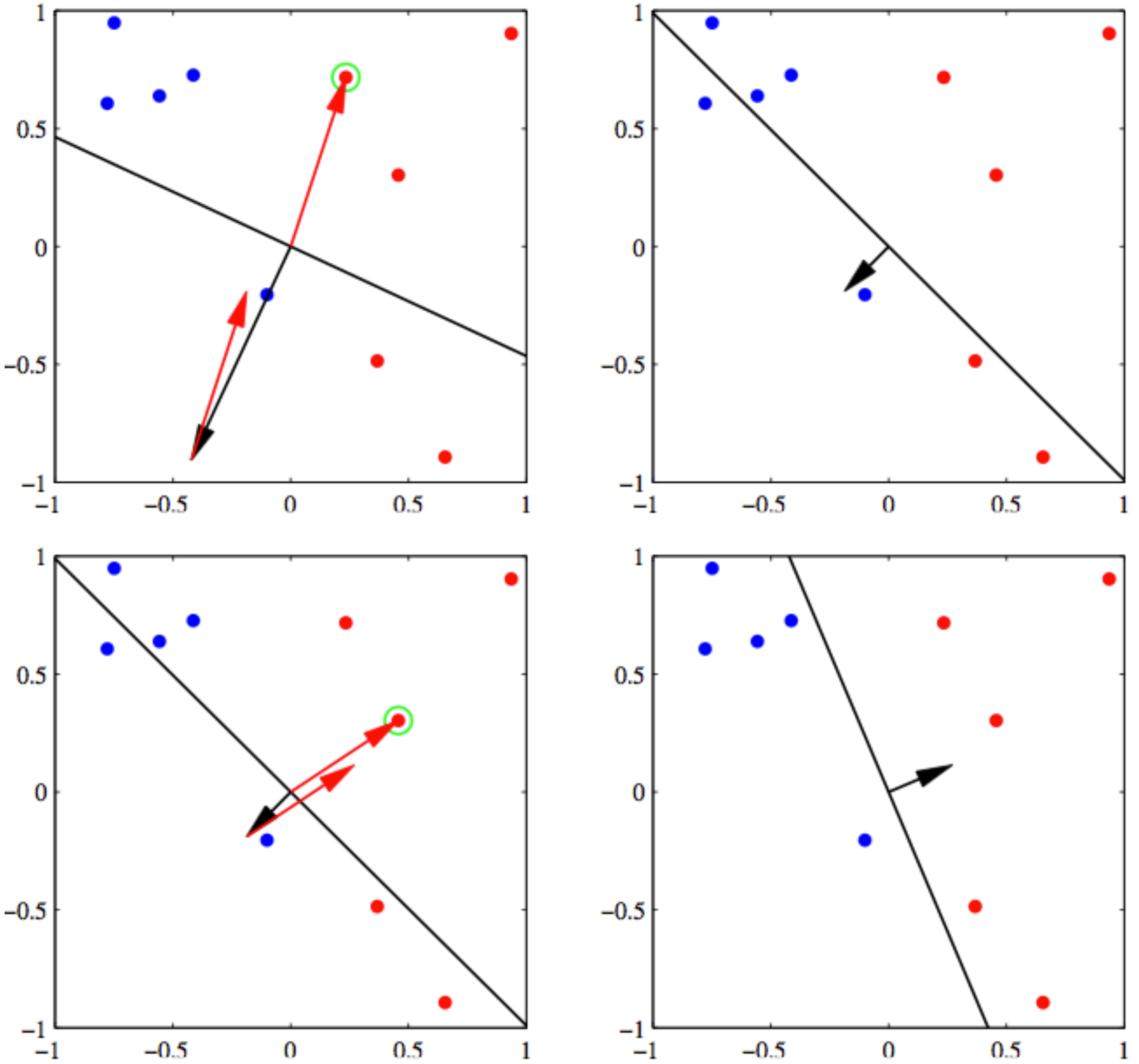
$\textbf{while}$ not converged
$\quad \textbf{for } (\mathbf{x}, y) \in D$
$\quad\quad \textbf{if } y(\mathbf{w} \cdot \mathbf{x}) \leq 0$
$\quad\quad\quad \mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$



Demo

while not converged
  for $(\mathbf{x}, y) \in D$
    if $y(\mathbf{w} \cdot \mathbf{x}) \leq 0$
      $\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$




1
0.5
0
−0.5
−1
−1
−0.5
0
0.5
1

Part IV

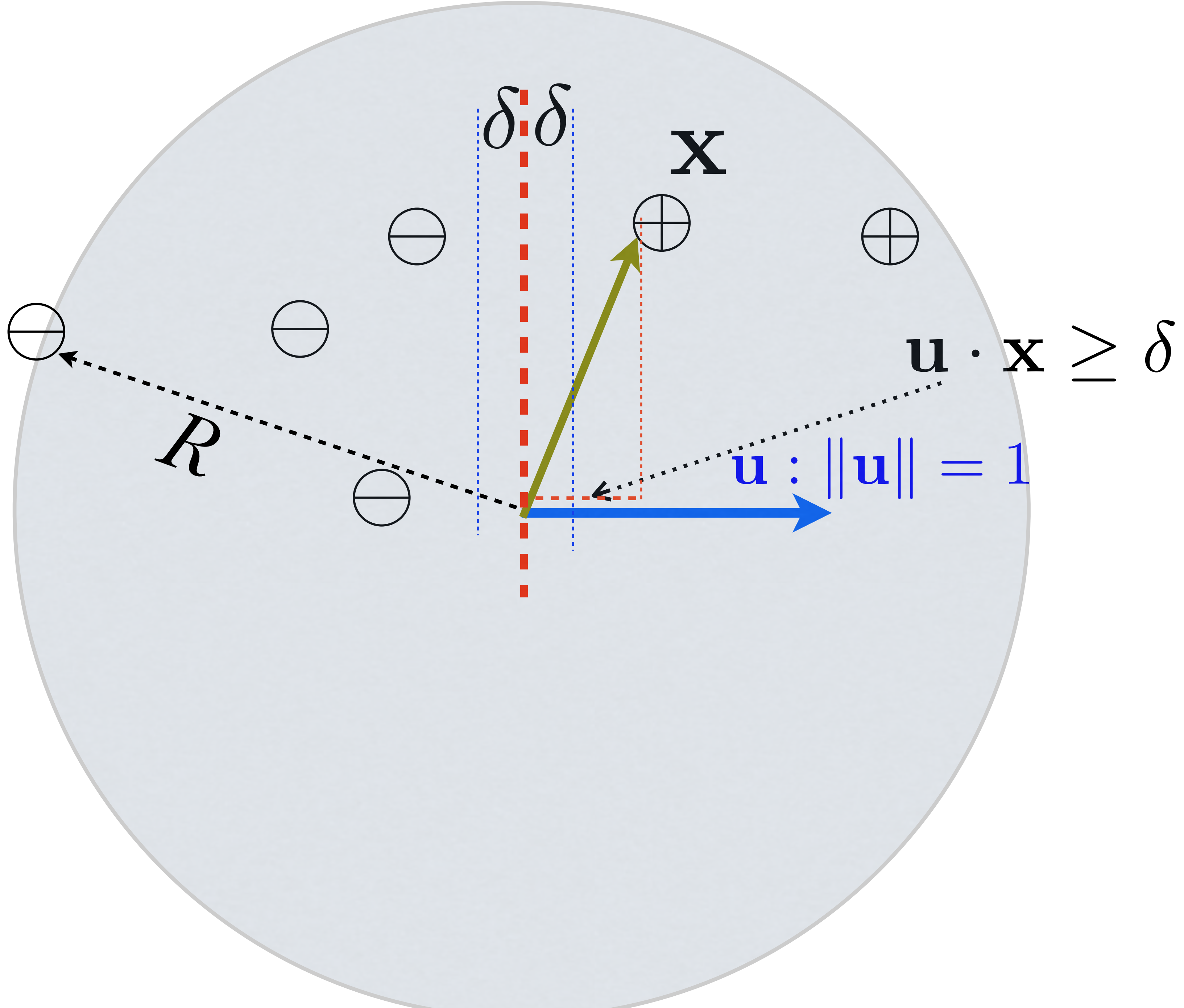
- Linear Separation, Convergence Theorem and Proof
  - formal definition of linear separation
  - perceptron convergence theorem
  - geometric proof
  - what variables affect convergence bound?

Linear Separation; Convergence Theorem

- dataset D is said to be "linearly separable" if there exists some unit oracle vector $\mathbf{u}$: $||\mathbf{u}|| = 1$ which correctly classifies every example $(\mathbf{x}, y)$ with a margin at least δ:

$$y(\mathbf{u} \cdot \mathbf{x}) \geq \delta \text{ for all } (\mathbf{x}, y) \in D$$

- then the perceptron must converge to a linear separator after at most R^2/δ^2 mistakes (updates) where $R = \max_{(\mathbf{x},y)\in D} \|\mathbf{x}\|$
- convergence rate R^2/δ^2
  - dimensionality independent
  - dataset size independent
  - order independent (but order matters in output)
  - scales with 'difficulty' of problem



Geometric Proof, part 1

- part 1: progress (alignment) on oracle projection

assume $\mathbf{w}^{(0)} = \mathbf{0}$, and $\mathbf{w}^{(i)}$ is the weight **before** the ith update (on $(\mathbf{x}, y)$)

$$\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} + y\mathbf{x}$$

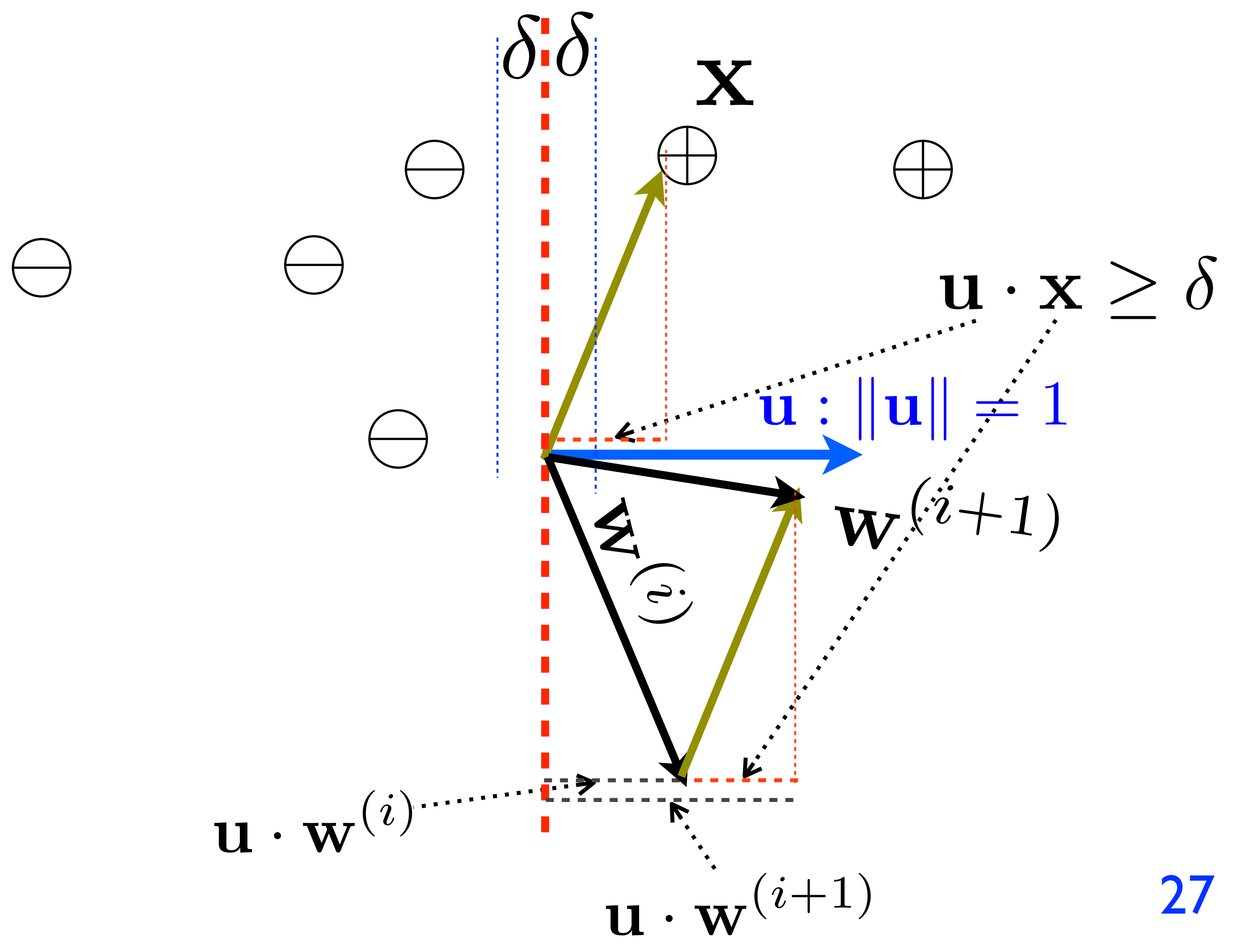
$$\mathbf{u} \cdot \mathbf{w}^{(i+1)} = \mathbf{u} \cdot \mathbf{w}^{(i)} + y(\mathbf{u} \cdot \mathbf{x})$$

$$\mathbf{u} \cdot \mathbf{w}^{(i+1)} \geq \mathbf{u} \cdot \mathbf{w}^{(i)} + \delta \qquad y(\mathbf{u} \cdot \mathbf{x}) \geq \delta \text{ for all } (\mathbf{x}, y) \in D$$

$$\mathbf{u} \cdot \mathbf{w}^{(i+1)} \geq i\delta$$

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

$$\left\|\mathbf{w}^{(i+1)}\right\| = \|\mathbf{u}\| \left\|\mathbf{w}^{(i+1)}\right\| \geq \mathbf{u} \cdot \mathbf{w}^{(i+1)} \geq i\delta$$



Geometric Proof, part 2

- part 2: upperbound of the norm of the weight vector

$$\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} + y\mathbf{x}$$

$$\left\|\mathbf{w}^{(i+1)}\right\|^2 = \left\|\mathbf{w}^{(i)} + y\mathbf{x}\right\|^2$$

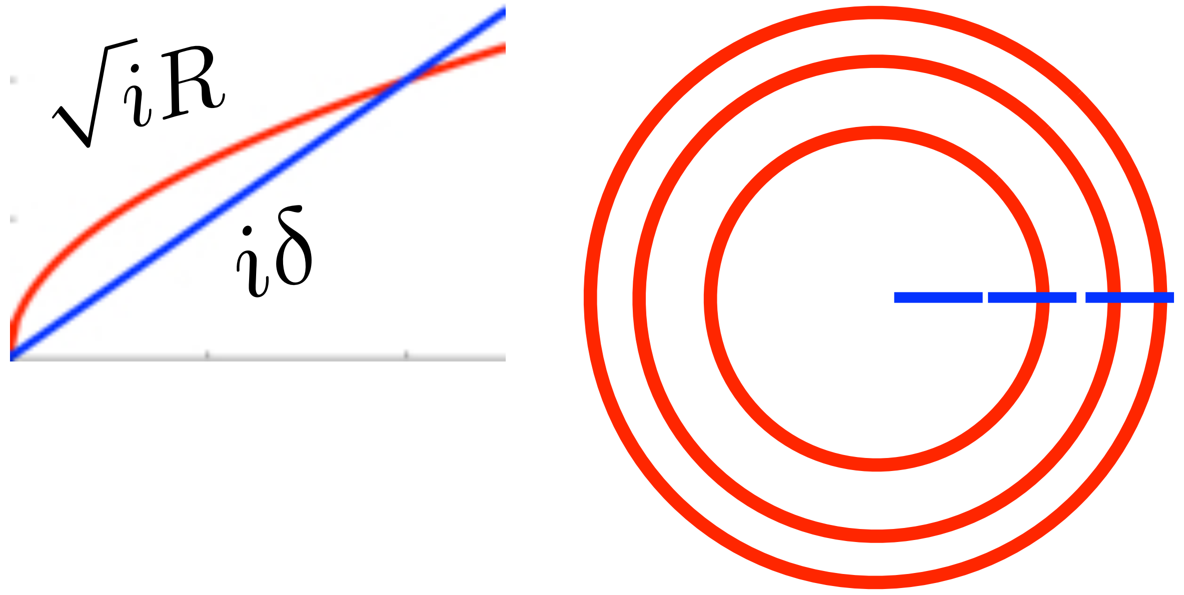
$$= \left\|\mathbf{w}^{(i)}\right\|^2 + \|\mathbf{x}\|^2 + 2y(\mathbf{w}^{(i)} \cdot \mathbf{x})$$

mistake on $\mathbf{x}$

$$\leq \left\|\mathbf{w}^{(i)}\right\|^2 + R^2$$

$$\leq iR^2$$

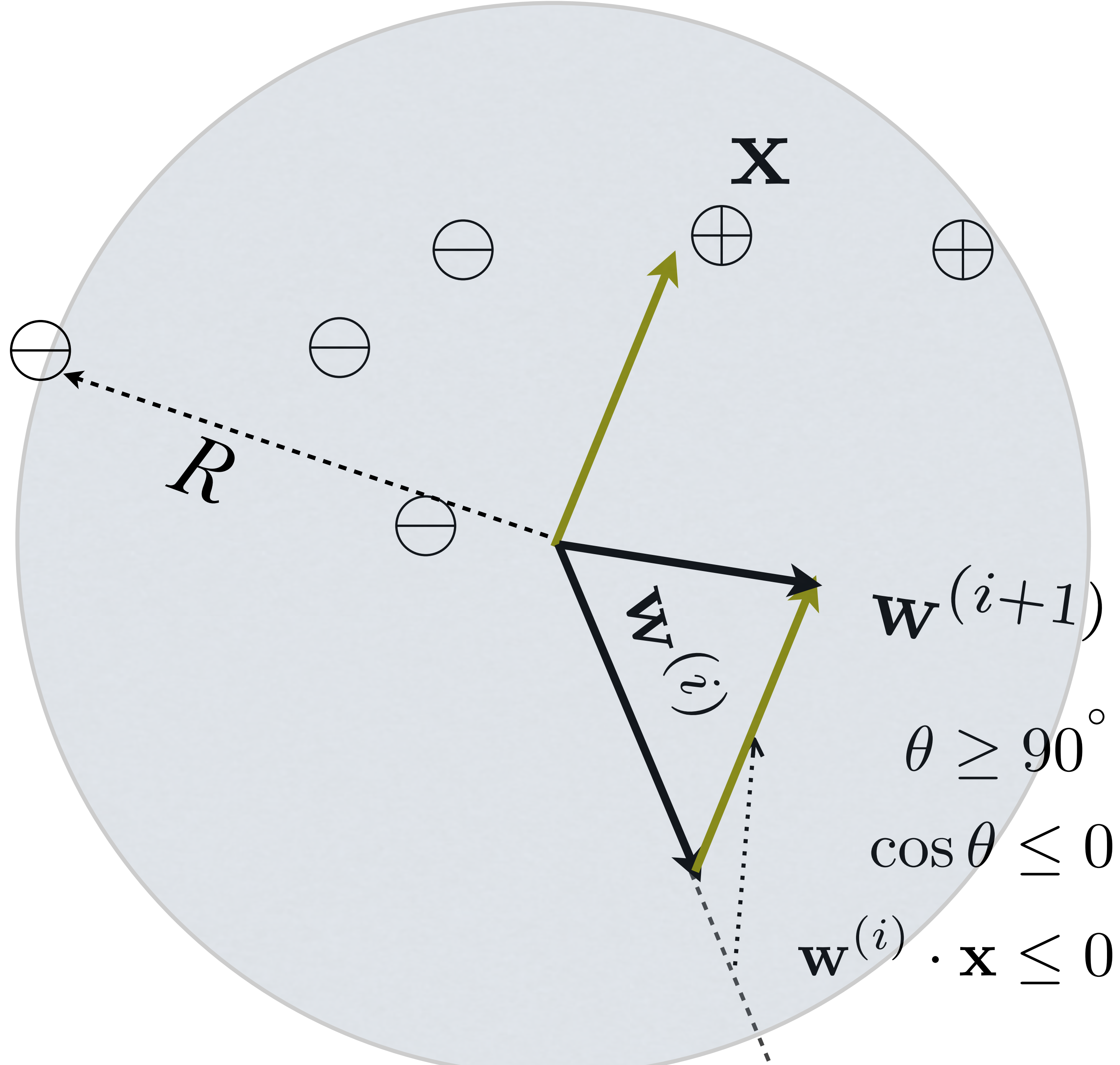
$$R = \max_{(\mathbf{x},y)\in D} \|\mathbf{x}\|$$



Combine with part 1:

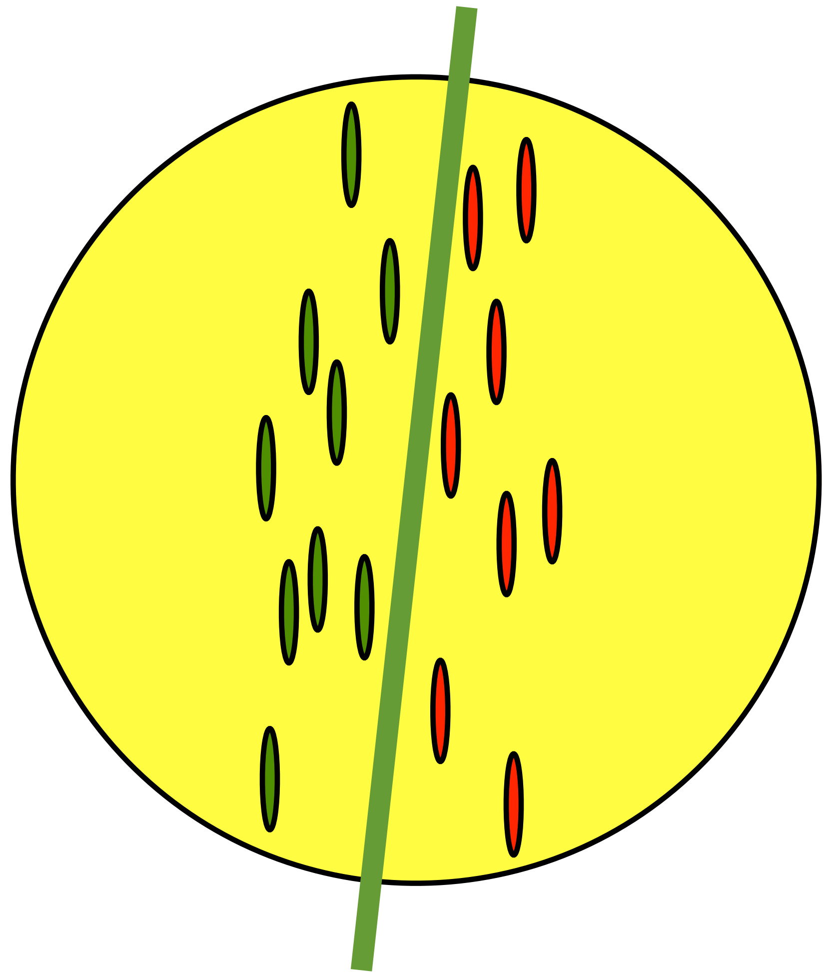
$$\left\|\mathbf{w}^{(i+1)}\right\| = \|\mathbf{u}\| \left\|\mathbf{w}^{(i+1)}\right\| \geq \mathbf{u} \cdot \mathbf{w}^{(i+1)} \geq i\delta$$

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

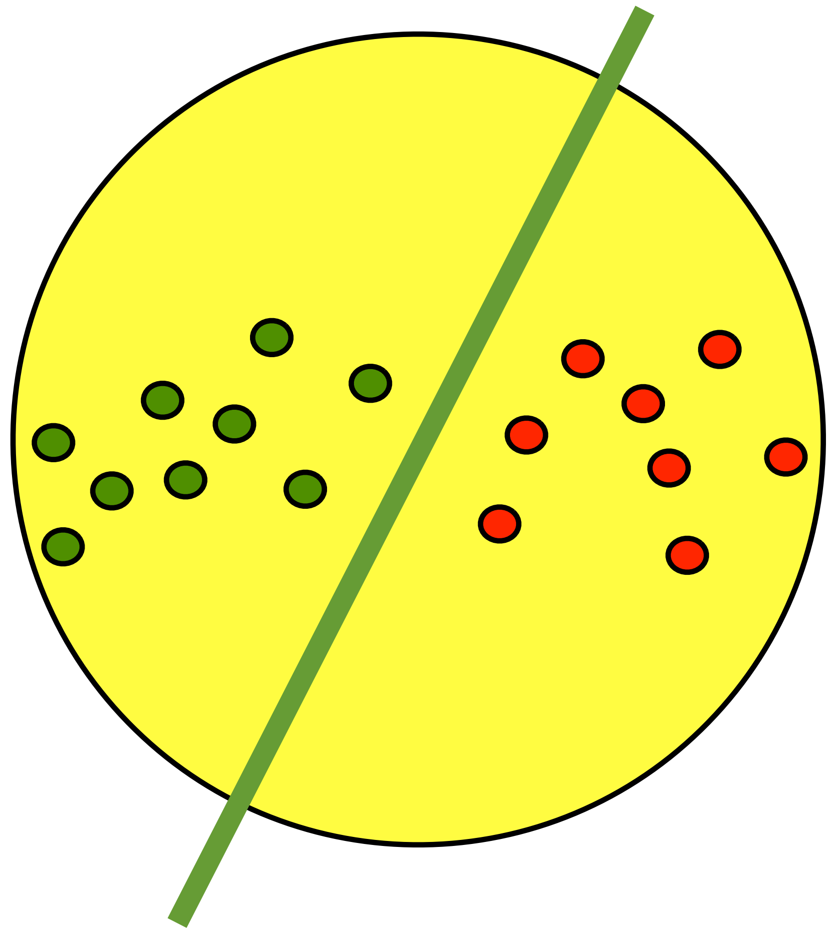


Convergence Bound R^2/δ^2

- is independent of:
  - dimensionality
  - number of examples
  - order of examples
  - constant learning rate
- and is dependent of:
  - separation difficulty (margin δ)
  - feature scale (radius R)
  - initial weight $\mathbf{w}^{(0)}$
    - changes how fast it converges, but not <u>whether</u> it'll converge



narrow margin:
hard to separate

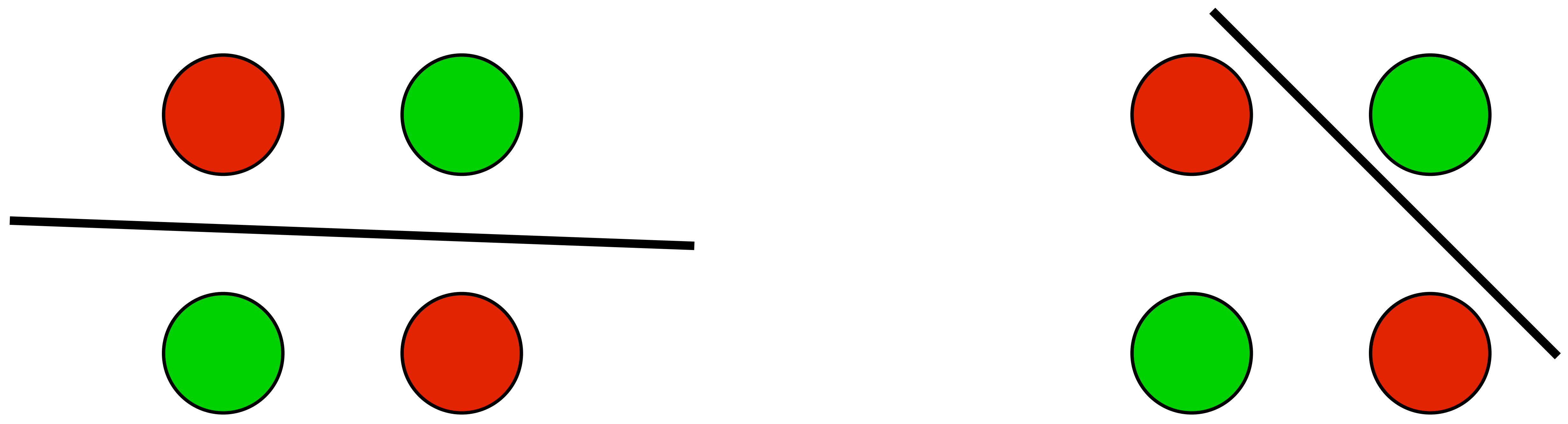


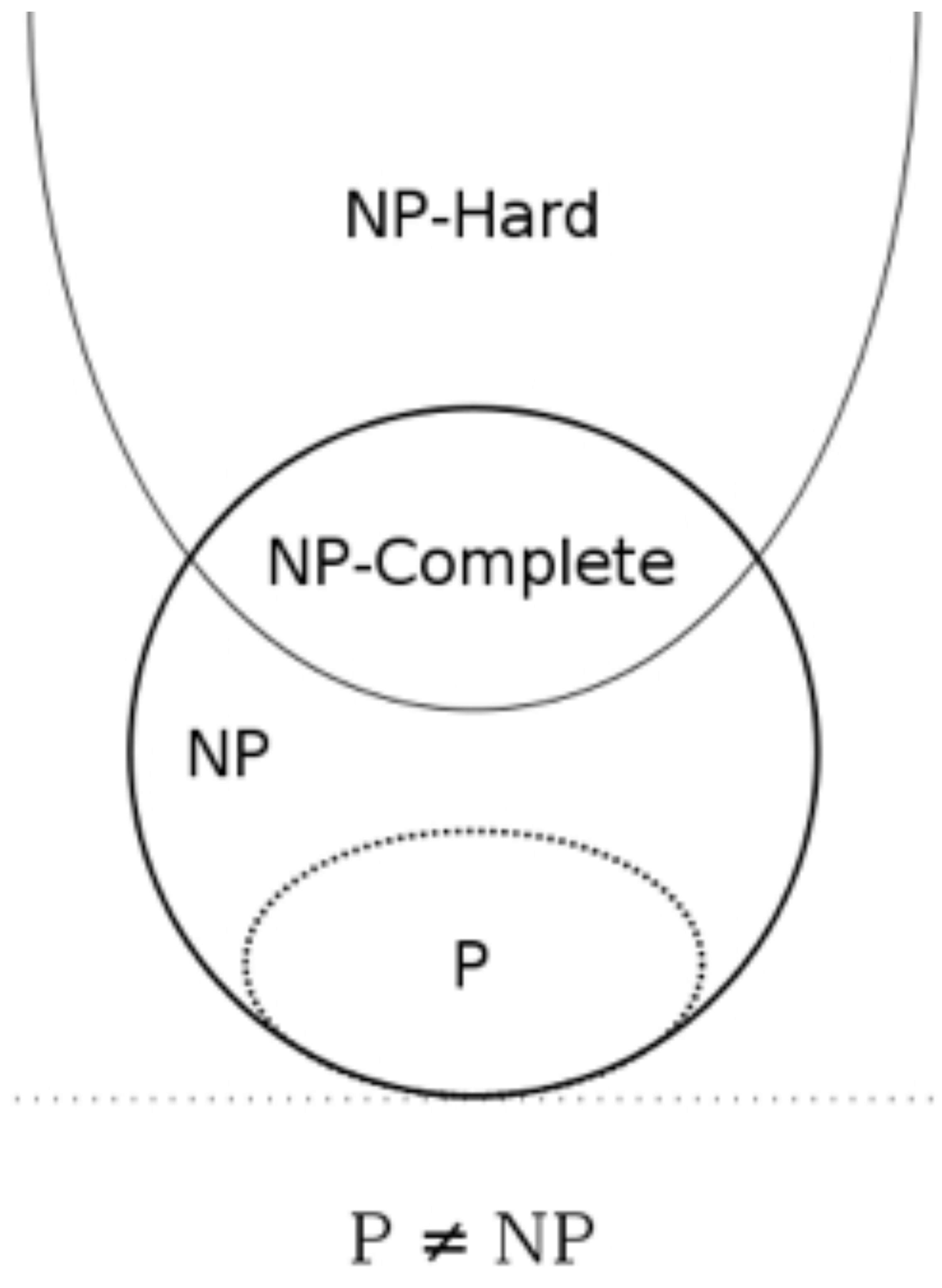
wide margin:
easy to separate

Part V

- Limitations of Linear Classifiers and Feature Maps
  - XOR: not linearly separable
  - perceptron cycling theorem
  - solving XOR: non-linear feature map
  - “preview demo”: SVM with non-linear kernel
  - redefining “linear” separation under feature map

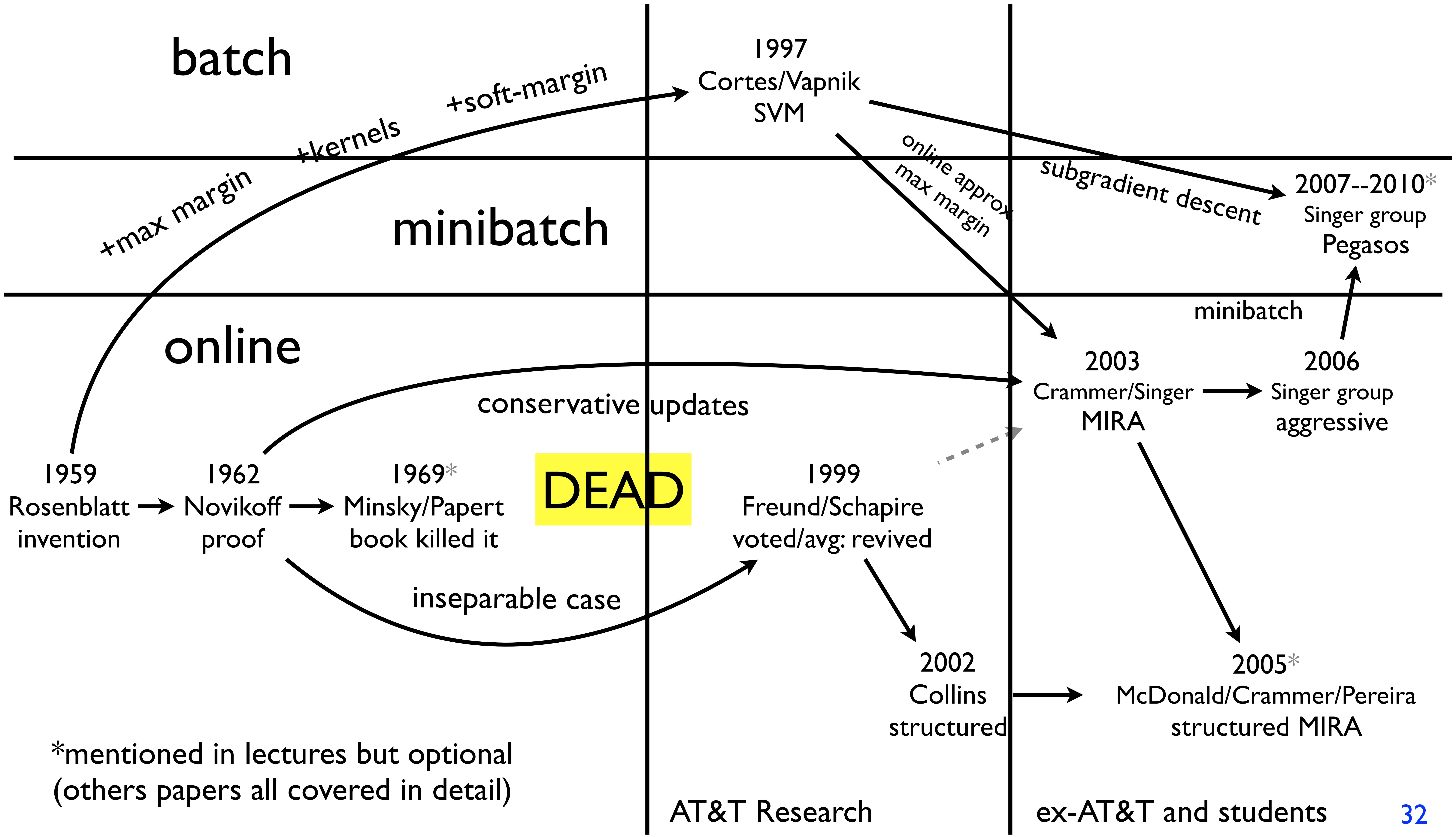
XOR





- 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



*mentioned in lectures but optional
(others papers all covered in detail)

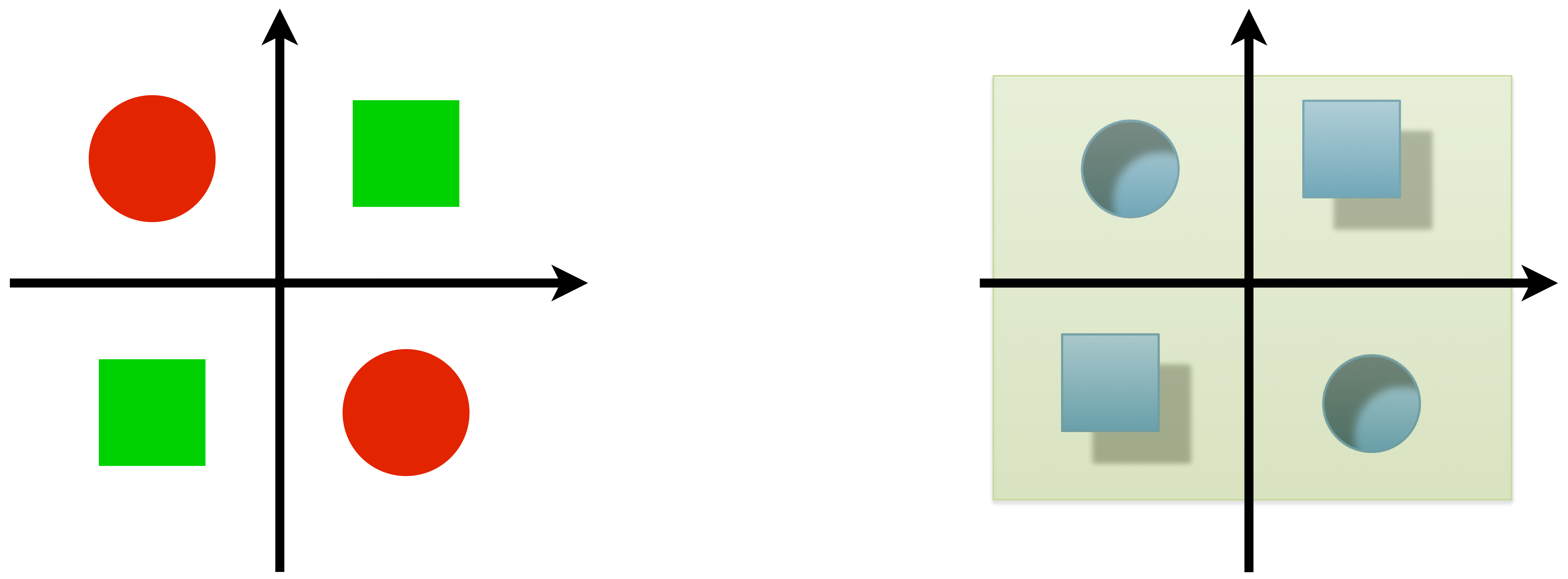
What if data is not separable

- in practice, data is almost always inseparable
  - wait, what exactly does that mean?
- perceptron cycling theorem (1970)
  - weights will remain bounded and will not diverge
- use dev set for early stopping (prevents overfitting)
- non-linearity (inseparable in low-dim => separable in high-dim)
  - higher-order features by combining atomic ones (cf. XOR)
  - a more systematic way: kernels (more details in week 5)

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

H. D. BLOCK AND S. A. LEVIN

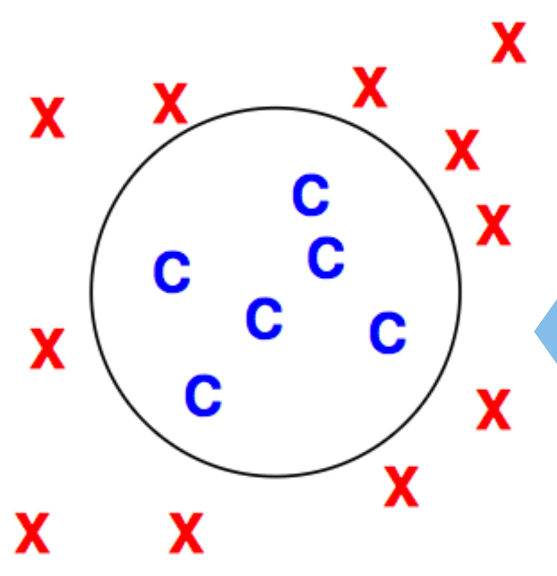
Solving XOR: Non-Linear Feature Map

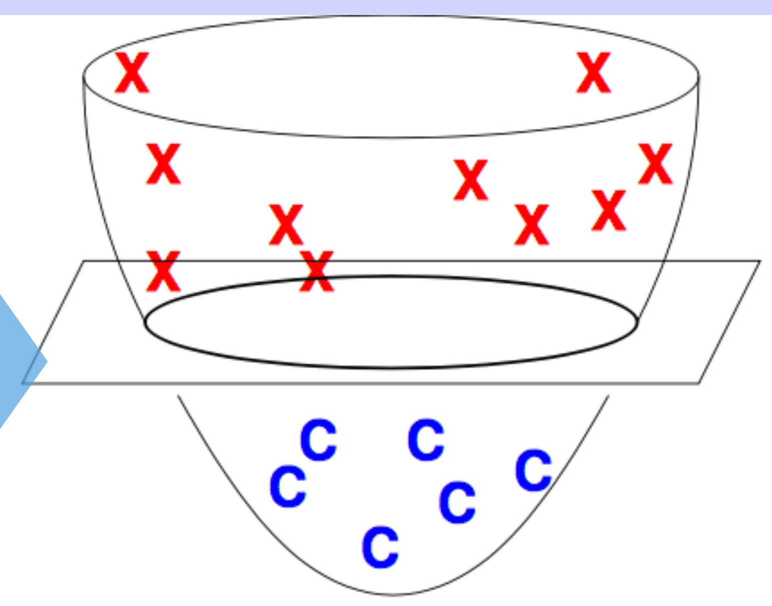


$$(x_1, x_2) \mapsto (x_1, x_2, x_1 x_2)$$

- XOR not linearly separable
- Mapping into 3D makes it easily linearly separable
  - this mapping is actually non-linear (quadratic feature $x_1 x_2$)
  - a special case of "polynomial kernels" (week 5)
  - linear decision boundary in 3D => non-linear boundaries in 2D

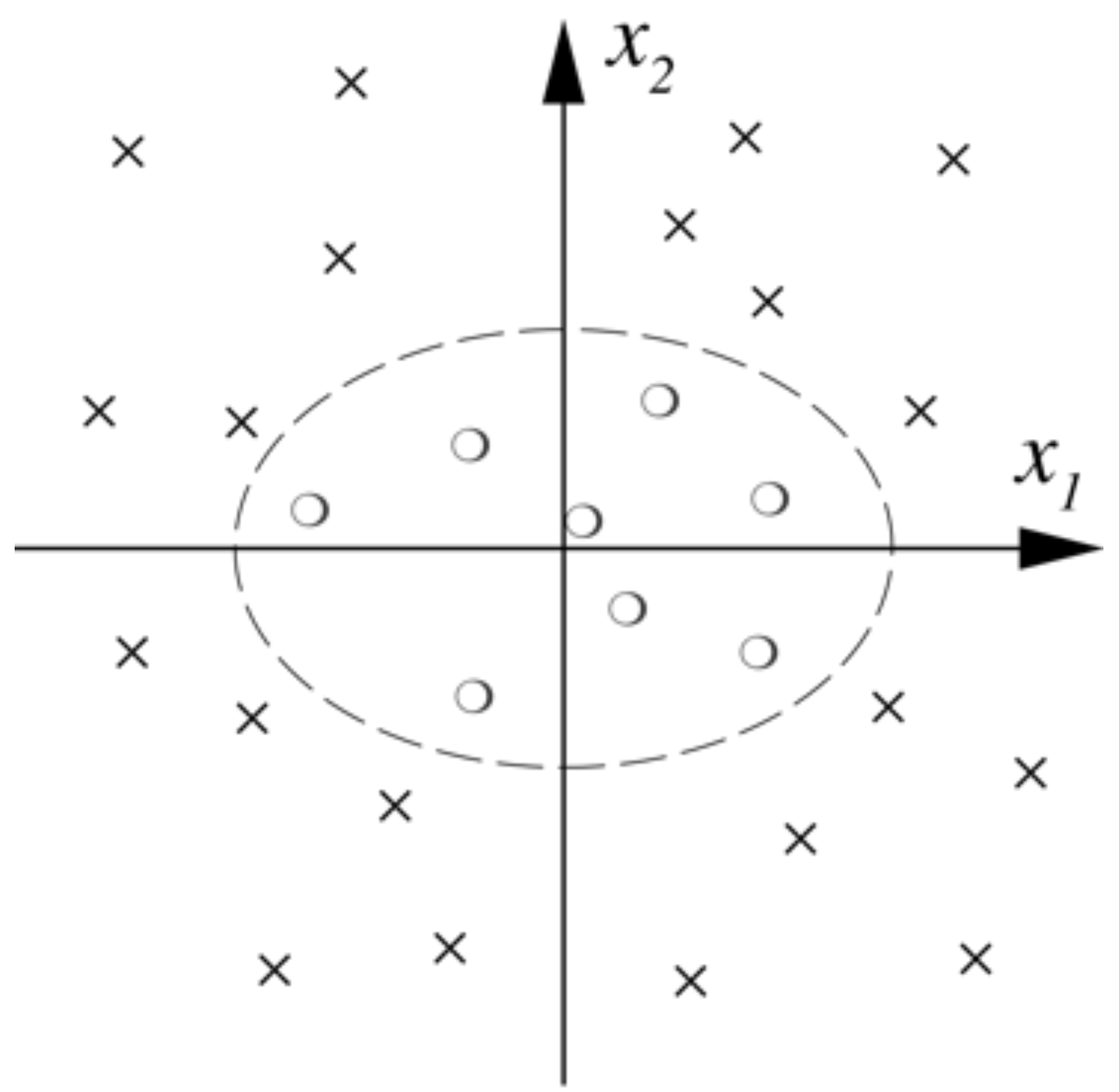
Low-dimension <=> High-dimension

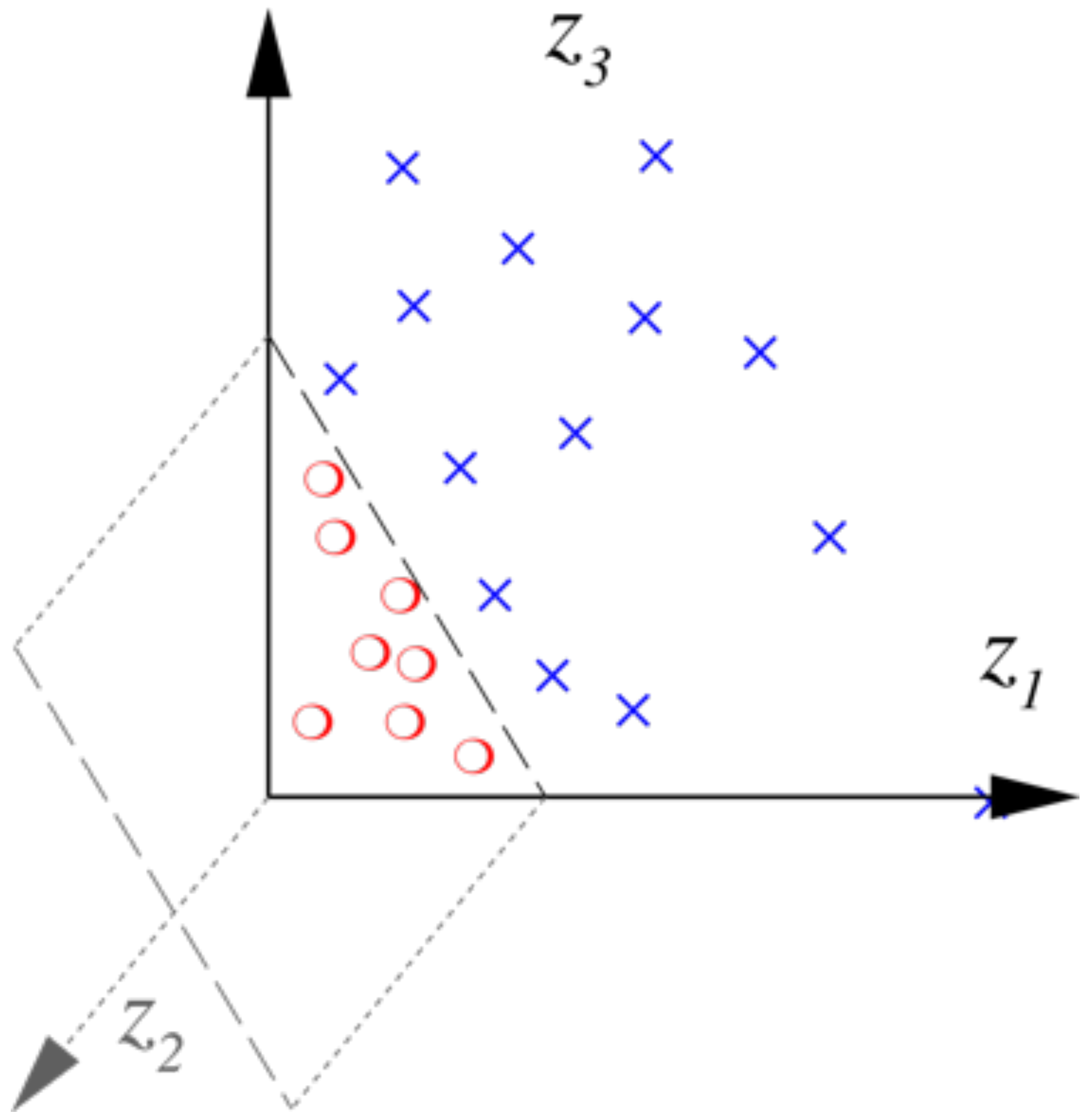




not linearly separable in 2D → linearly separable in 3D

non-linear boundaries in 2D ← linear decision boundary in 3D





SVM with a polynomial Kernel visualization

Created by:
Udi Aharoni

Linear Separation under Feature Map

- we have to redefine separation and convergence theorem
- dataset D is said to be linearly separable under feature map ϕ if there exists some unit oracle vector $\mathbf{u}$: $||\mathbf{u}|| = 1$ which correctly classifies every example $(\mathbf{x}, y)$ with a margin at least δ:

$$y(\mathbf{u} \cdot \boldsymbol{\Phi}(\mathbf{x})) \geq \delta \text{ for all } (\mathbf{x}, y) \in D$$

- then the perceptron must converge to a linear separator after at most R^2/δ^2 mistakes (updates) where

$$R = \max_{(\mathbf{x},y)\in D} \|\boldsymbol{\Phi}(\mathbf{x})\|$$

- in practice, the choice of feature map ("feature engineering") is often more important than the choice of learning algorithms
  - the first step of any ML project is data preprocessing: transform each $(\mathbf{x}, y)$ to $(\phi(\mathbf{x}), y)$
  - at testing time, also transform each $\mathbf{x}$ to $\phi(\mathbf{x})$
  - deep learning aims to automate feature engineering