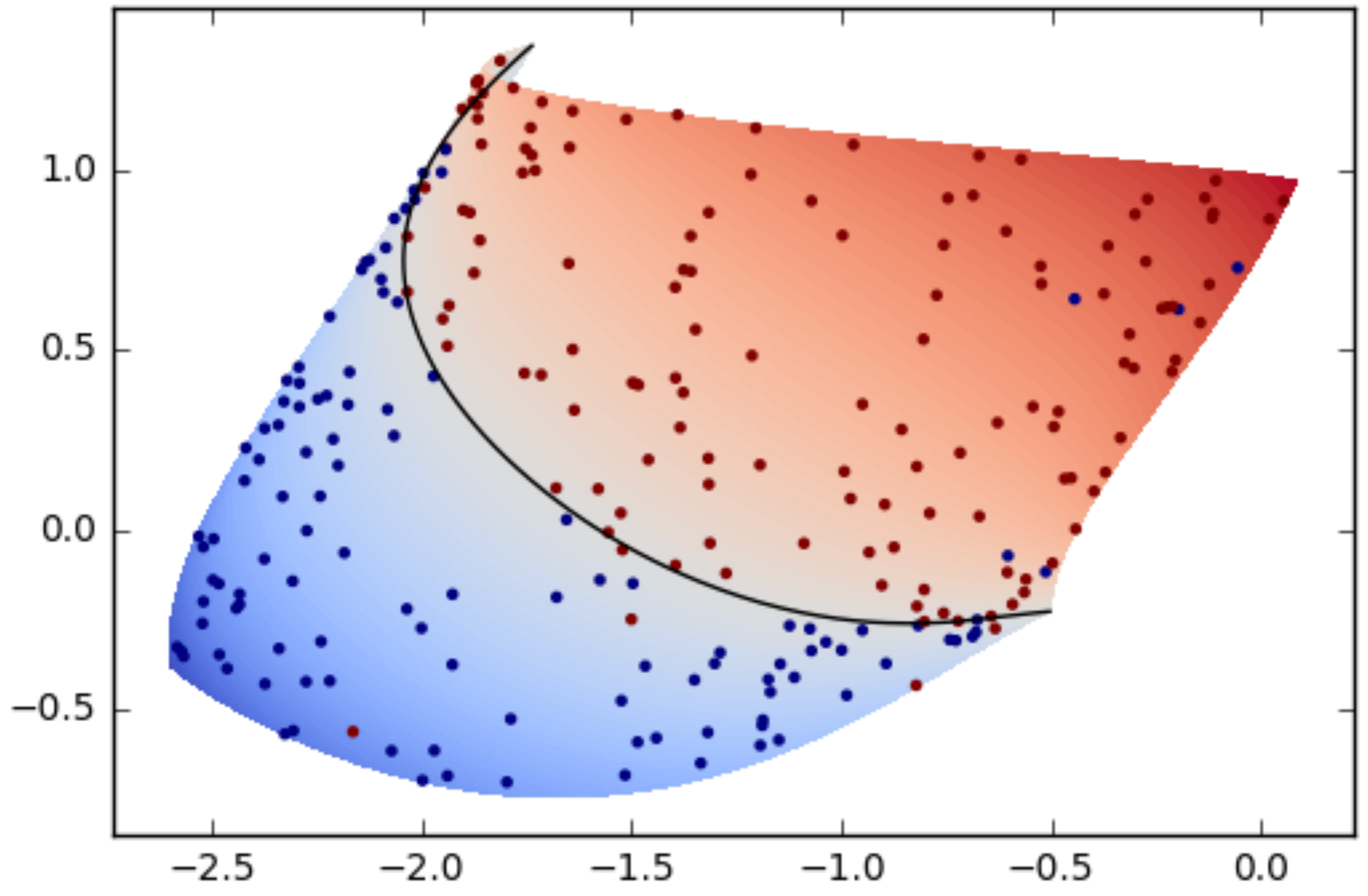
CS 273A: Machine Learning

Fall 2021

Lecture 12: Support Vector Machines

Roy Fox
Department of Computer Science
Bren School of Information and Computer Sciences
University of California, Irvine

All slides in this course adapted from Alex Ihler & Sameer Singh



Logistics

assignments

- Assignment 4 will be published soon, due Fri, Nov 12

project

- Project abstract due Tue, Nov 16

midterm

- Midterm exam on Thu, Nov 4, 11am–12:20 in **SH 128**
- If you're eligible to be remote — let us know immediately

Today's lecture

Multi-Layer Perceptrons

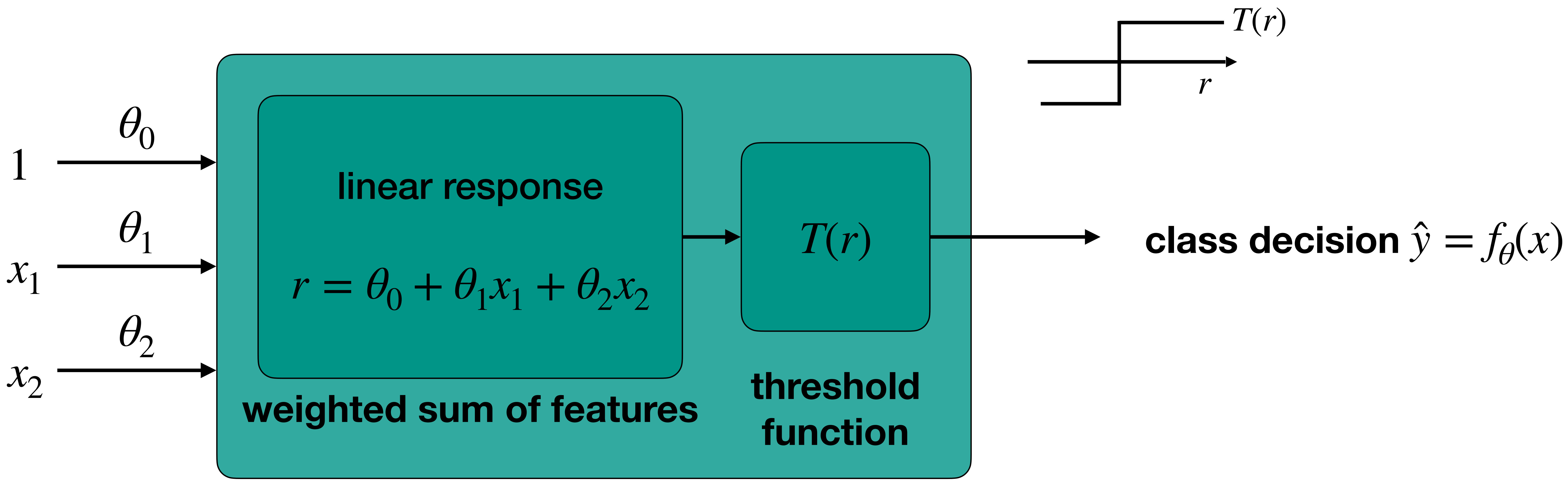
Support Vector Machines

Lagrangian and duality

Kernel Machines

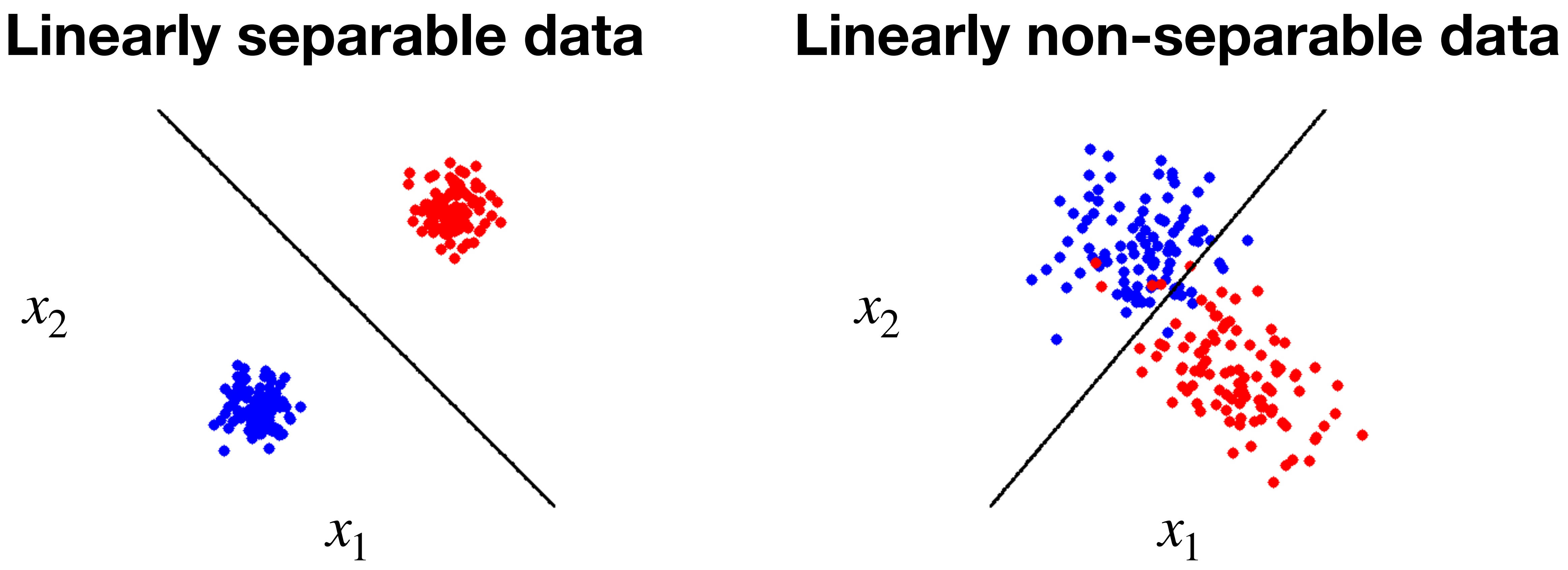
Linear classifiers

- Perceptron = use hyperplane to partition feature space → classes
  - Soft classifiers (logistic) = sensitive to margin from decision boundary



Adding features

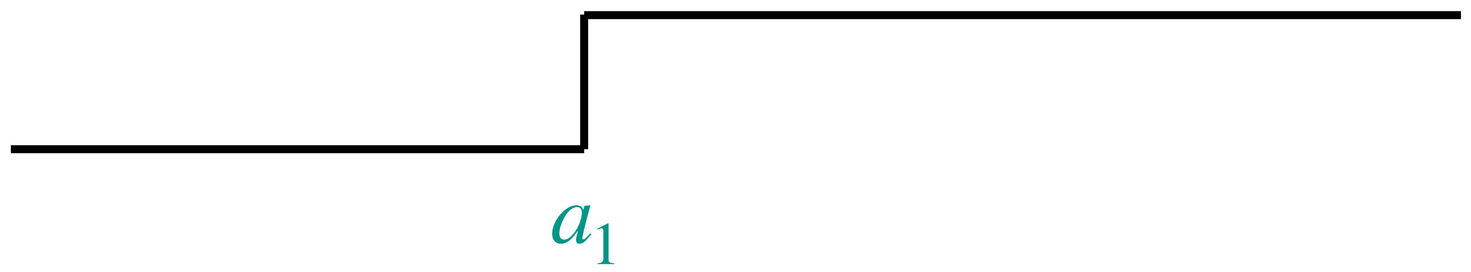
- If data is non-separable in current feature space
  - Perhaps it will be separable in higher dimension $\Longrightarrow$ add more features
  - E.g., polynomial features: linear classifier → polynomial classifier
- Which features to add?
  - Perhaps outputs of simpler perceptrons?



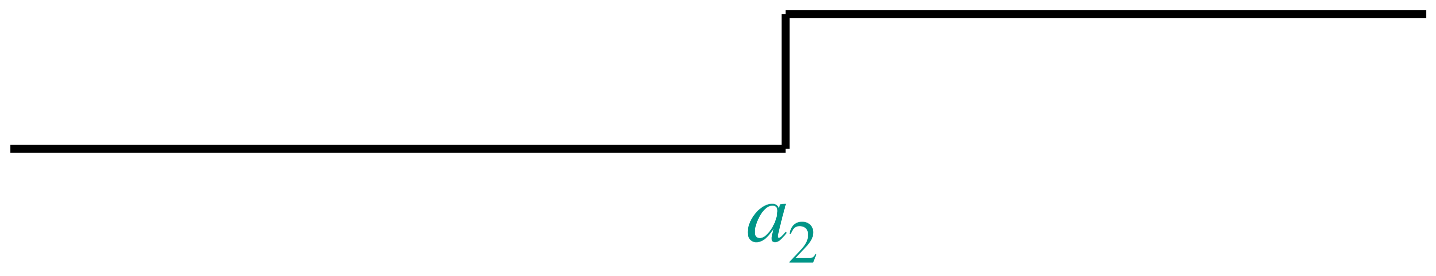
Combining step functions

- Combinations of step functions allow more complex decision boundaries

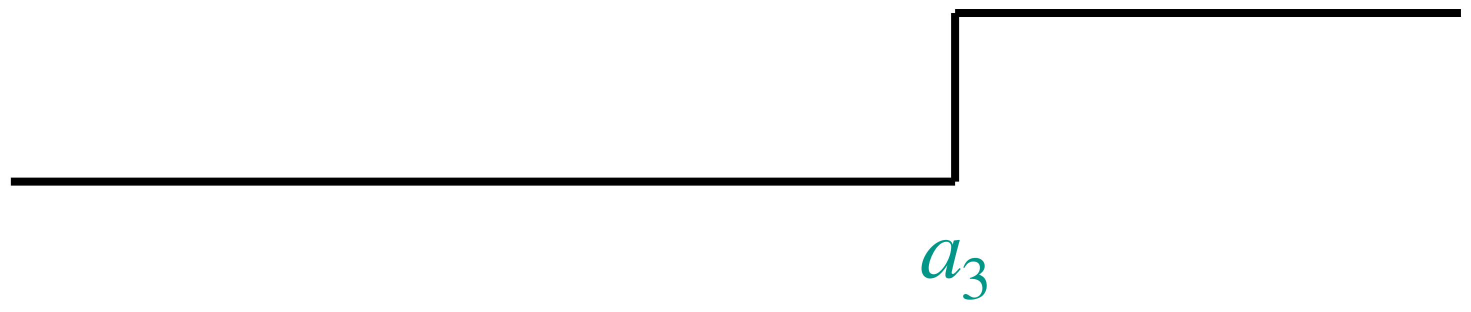
$F_1(x) = T(x - a_1)$



$F_2(x) = T(x - a_2)$



$F_3(x) = T(x - a_3)$



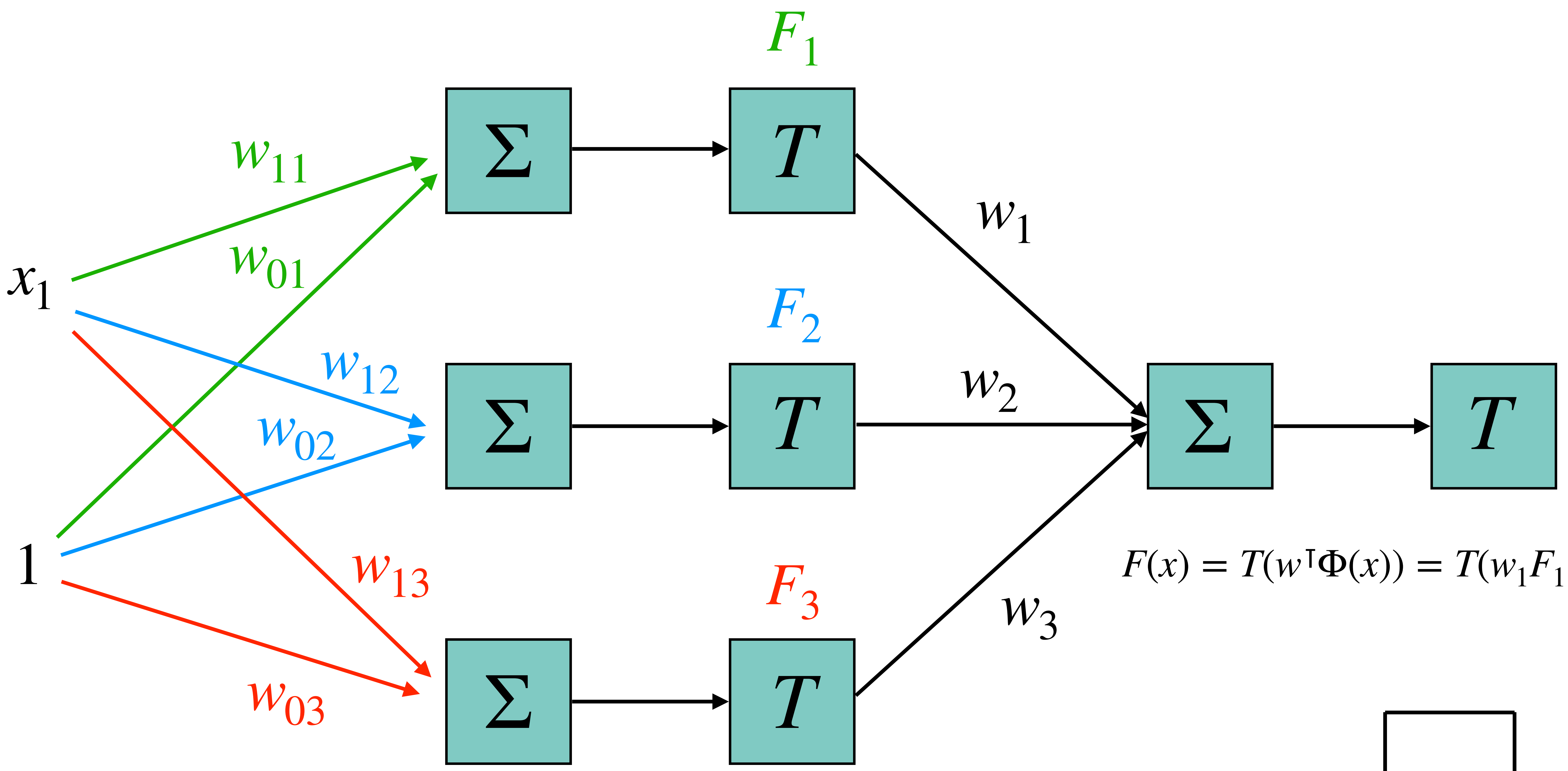
$$\Phi(x) = \begin{bmatrix} F_1(x) & F_2(x) & F_3(x) \end{bmatrix}$$

is piecewise constant

$$F(x) = T(w^\intercal \Phi(x)) = T(w_1 F_1(x) + w_2 F_2(x) + w_3 F_3(x) + w_4)$$

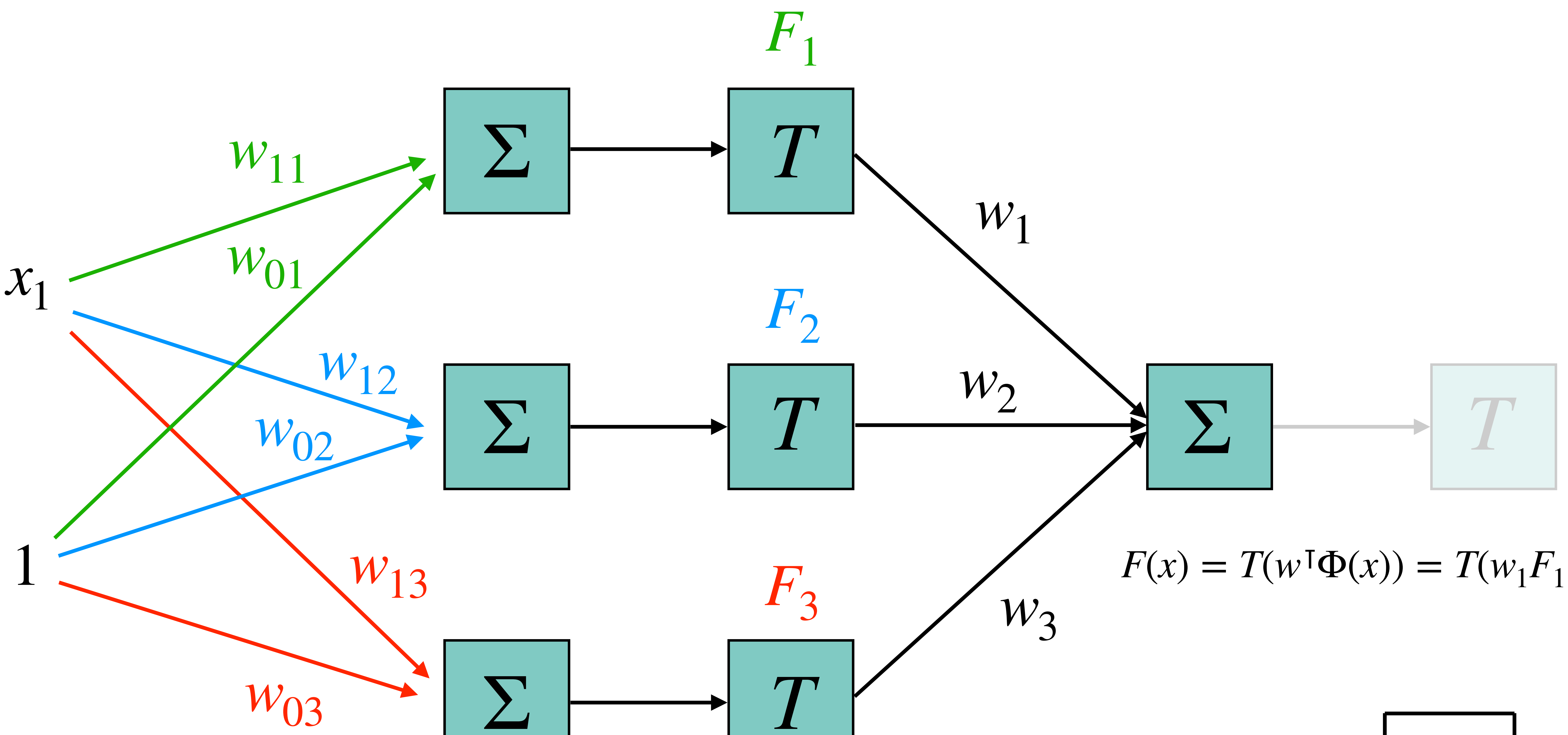
- Need to learn:
  - Thresholds a_1, a_2, a_3
  - Weights w_1, w_2, w_3, w_4

Multi-Layer Perceptron (MLP)



$$F(x) = T(w^\intercal \Phi(x)) = T(w_1 F_1(x) + w_2 F_2(x) + w_3 F_3(x) + w_4)$$

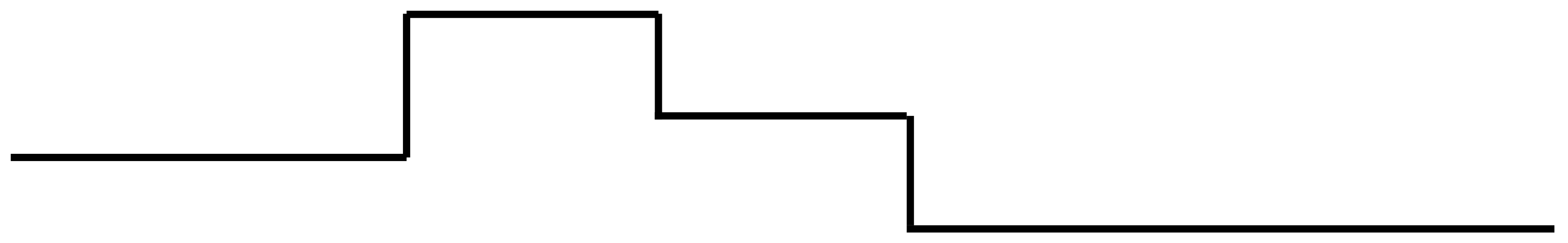
Multi-Layer Perceptron (MLP)



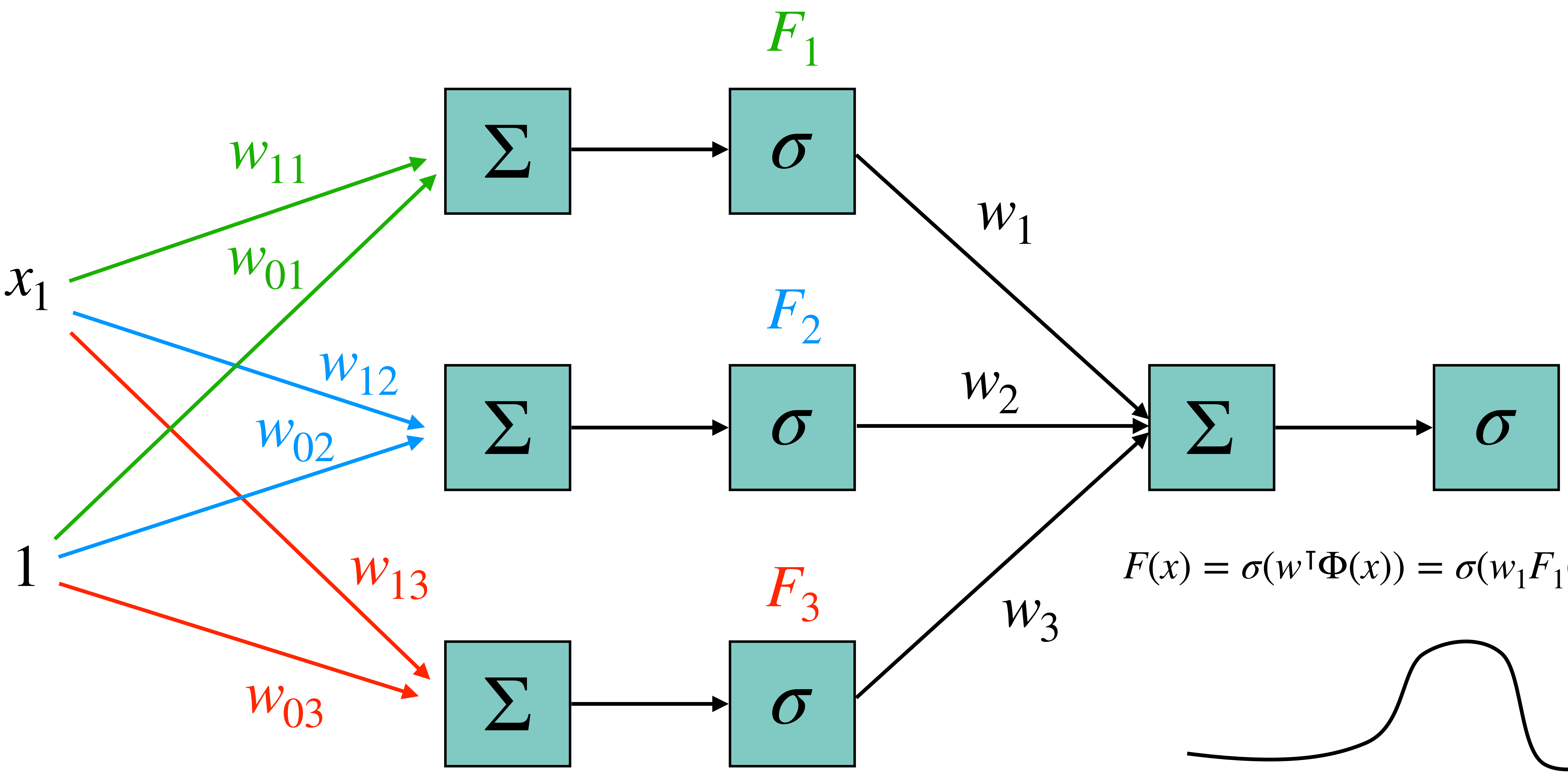
$F(x) = T(w^\intercal \Phi(x)) = T(w_1 F_1(x) + w_2 F_2(x) + w_3 F_3(x) + w_4)$

regression

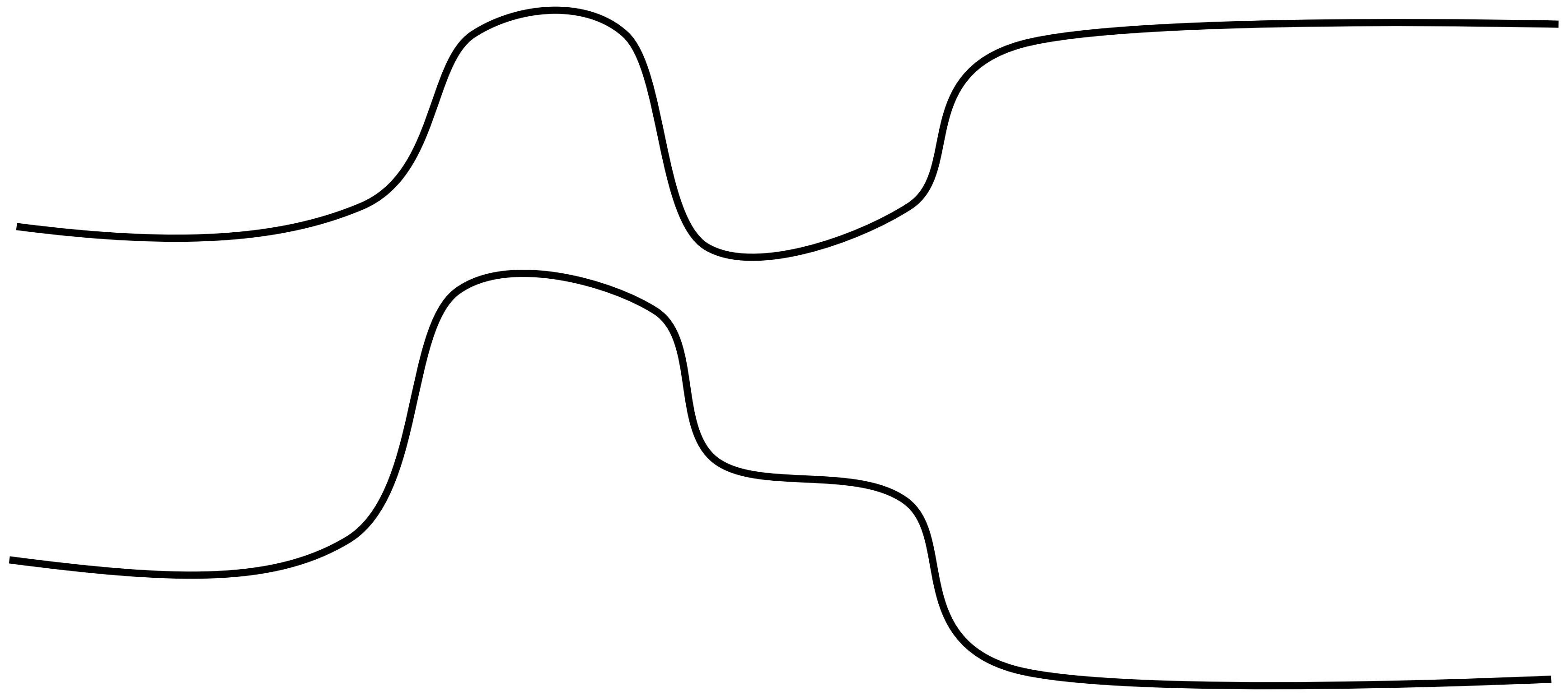
$F(x) = w^\intercal \Phi(x)$



Multi-Layer Perceptron (MLP)



$F(x) = \sigma(w^\intercal \Phi(x)) = \sigma(w_1 F_1(x) + w_2 F_2(x) + w_3 F_3(x) + w_4)$

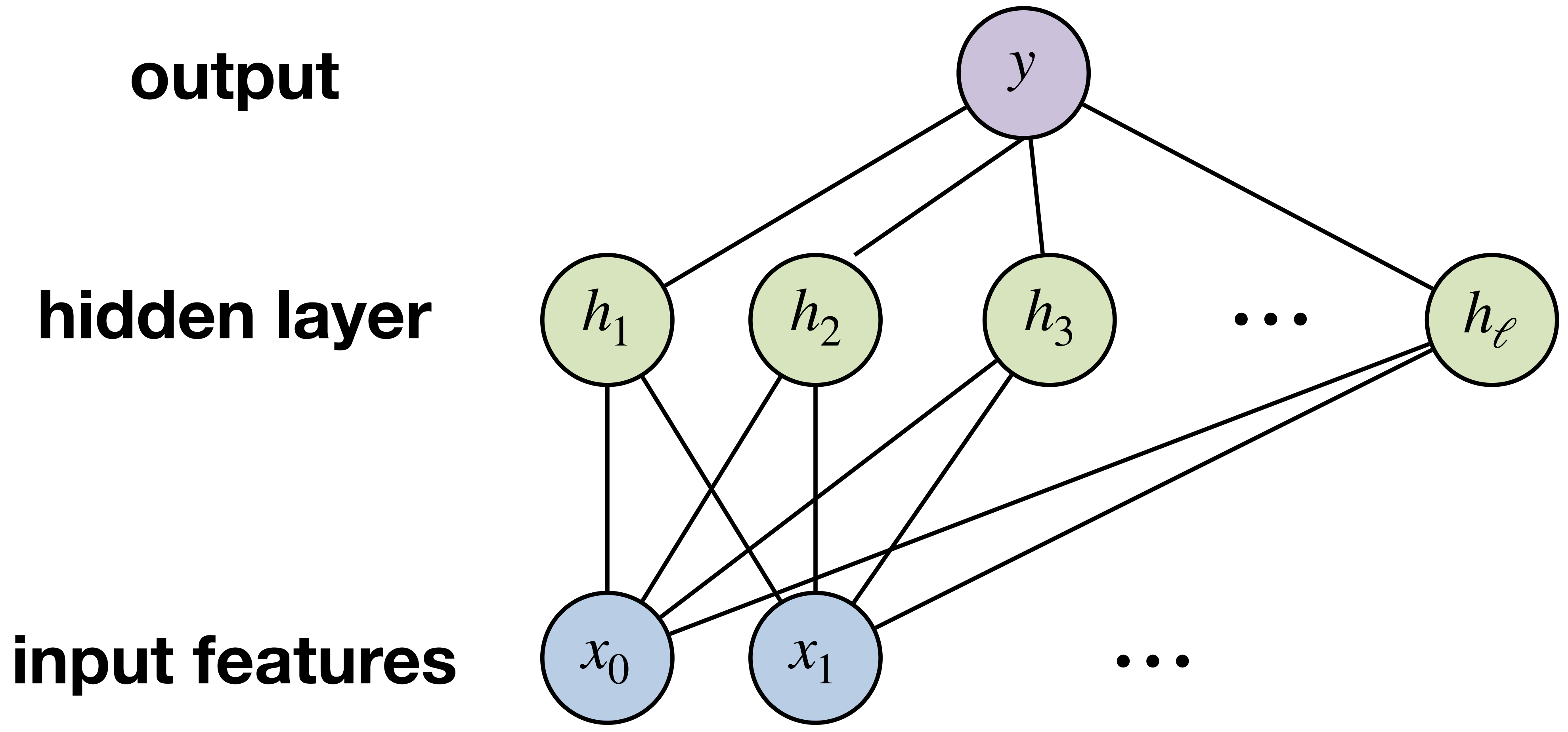


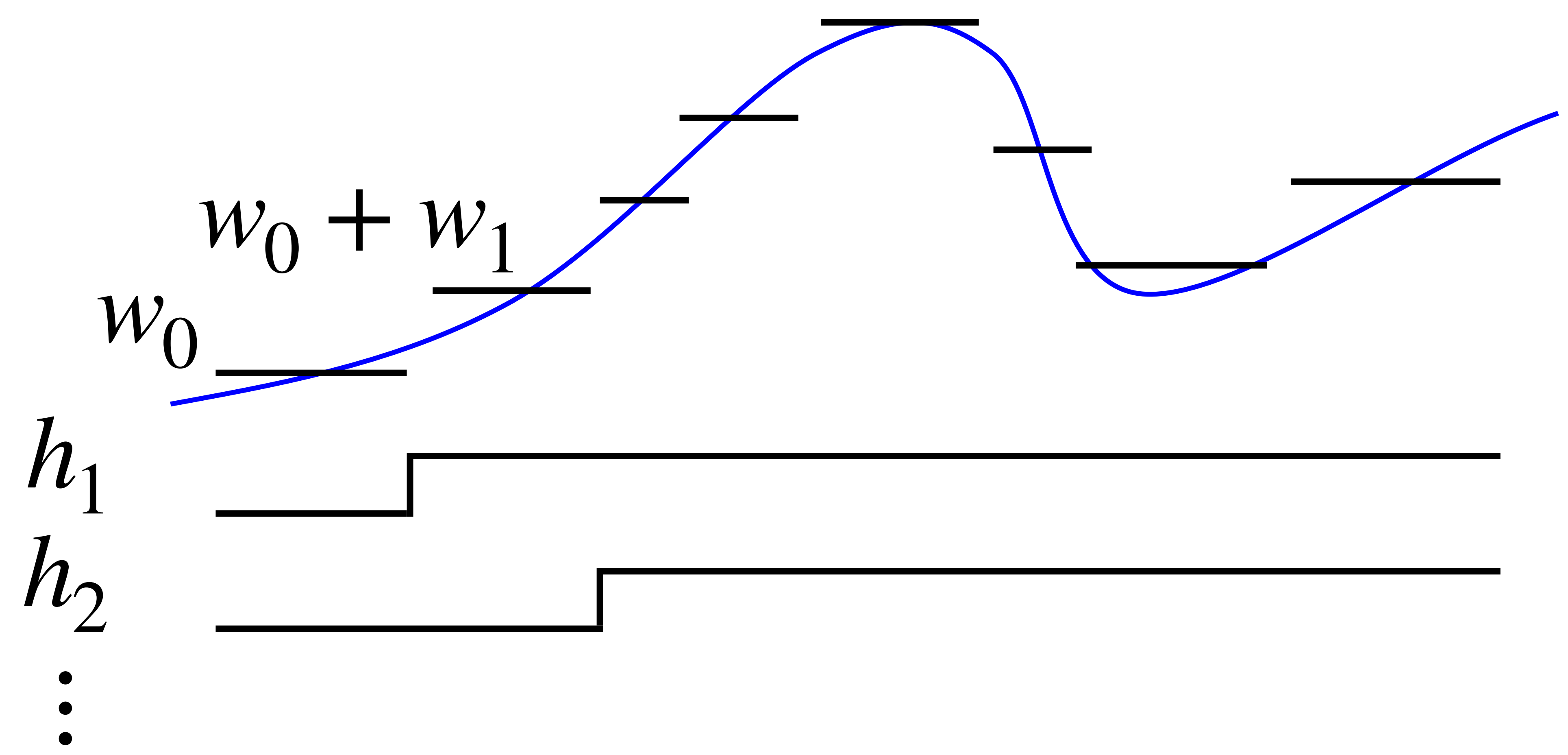
regression

$F(x) = w^\intercal \Phi(x)$

MLPs: properties

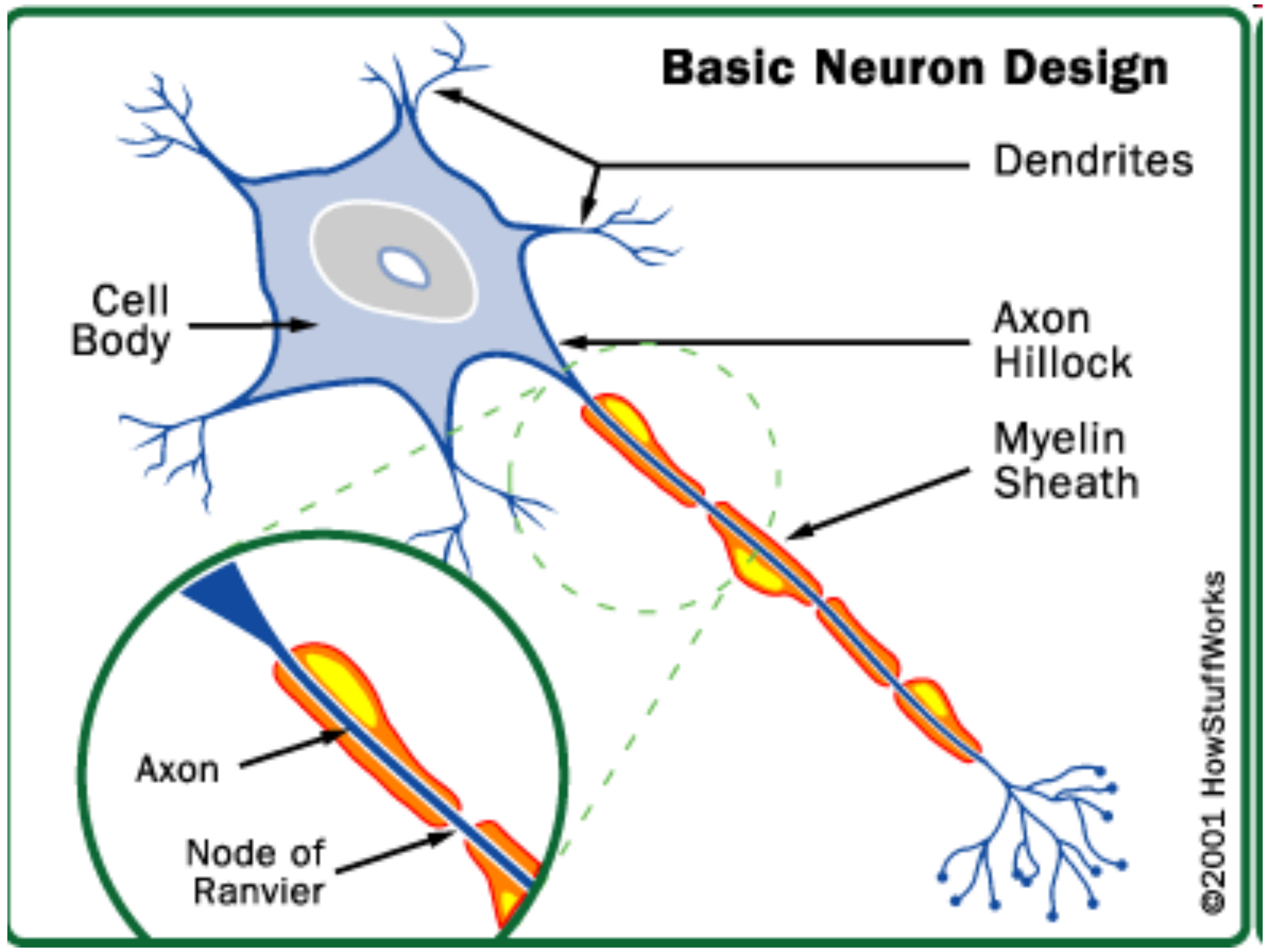
- Simple building blocks
  - Each unit is a perceptron: linear response → non-linear activation
- MLPs are universal approximators:
  - Can approximate any function arbitrarily well, with enough units





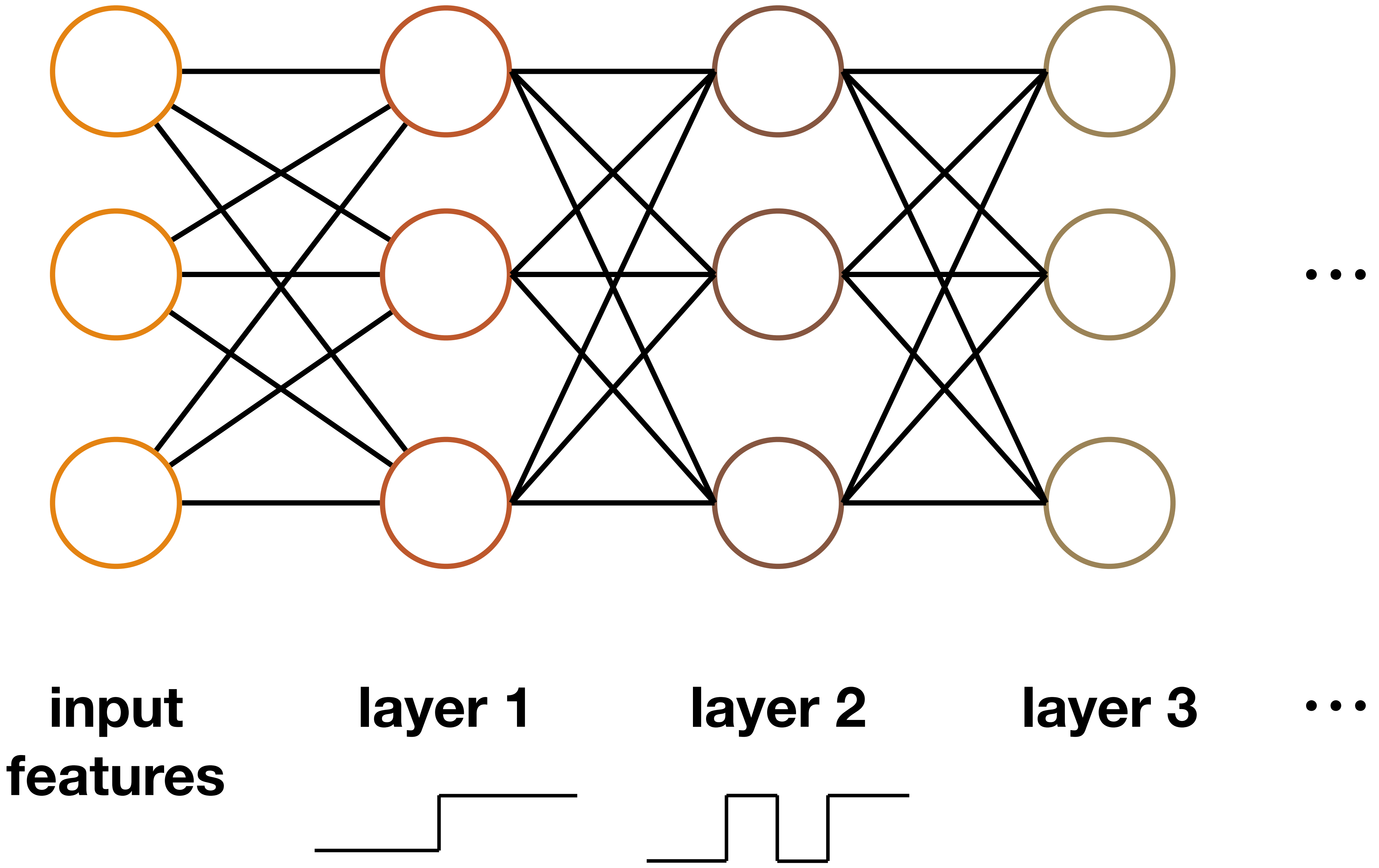
“Neural” Networks

- Biologically inspired
- Neurons:
  - “Simple” cells
  - Dendrites take input voltage
  - Cell body “weights” inputs
  - Axons “fire” voltage
  - Synapses connect to other cells



Deep Neural Networks (DNNs)

- Layers of perceptrons can be stacked deeply
  - Deep architectures are subject of much current research



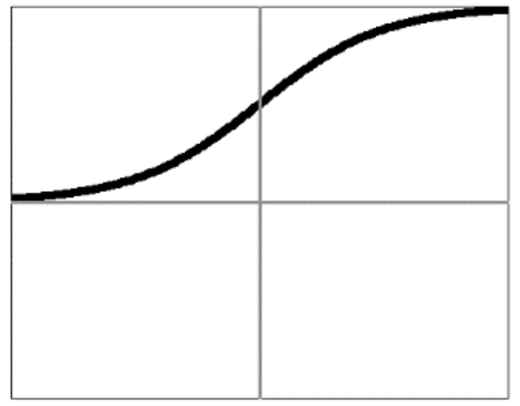

```
r1 = w[0].T @ x + b[0]    # linear response
h1 = sig(r1)              # activation function

r2 = w[1].T @ h1 + b[1]   # linear response
h2 = sig(r2)              # activation function

                          # ...
```

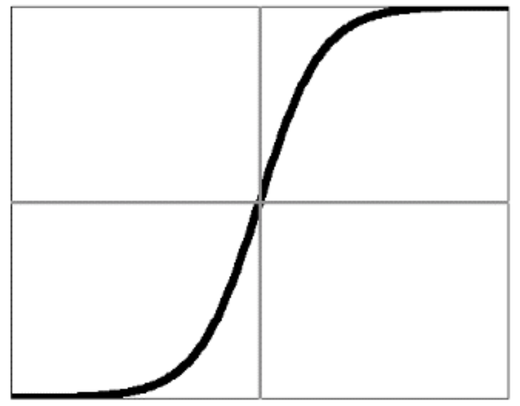
Activation functions

- Logistic $\sigma(z) = \dfrac{1}{1 + \exp(-z)}$



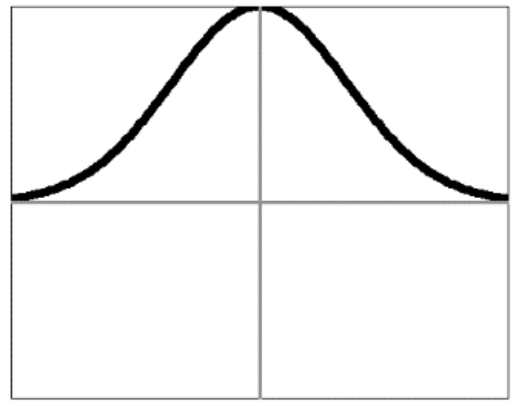
$\sigma'(z) = \sigma(z)(1 - \sigma(z))$

- Hyperbolic tangent $\sigma(z) = \dfrac{1 - \exp(-2z)}{1 + \exp(-2z)}$



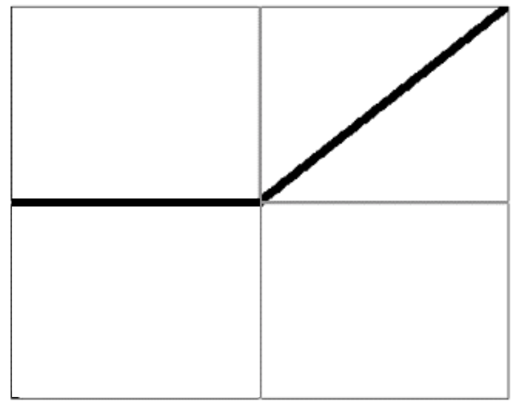
$\sigma'(z) = 1 - \sigma^2(z)$

- Gaussian $\sigma(z) = \exp(-\frac{1}{2}z^2)$



$\sigma'(z) = - z\sigma(z)$

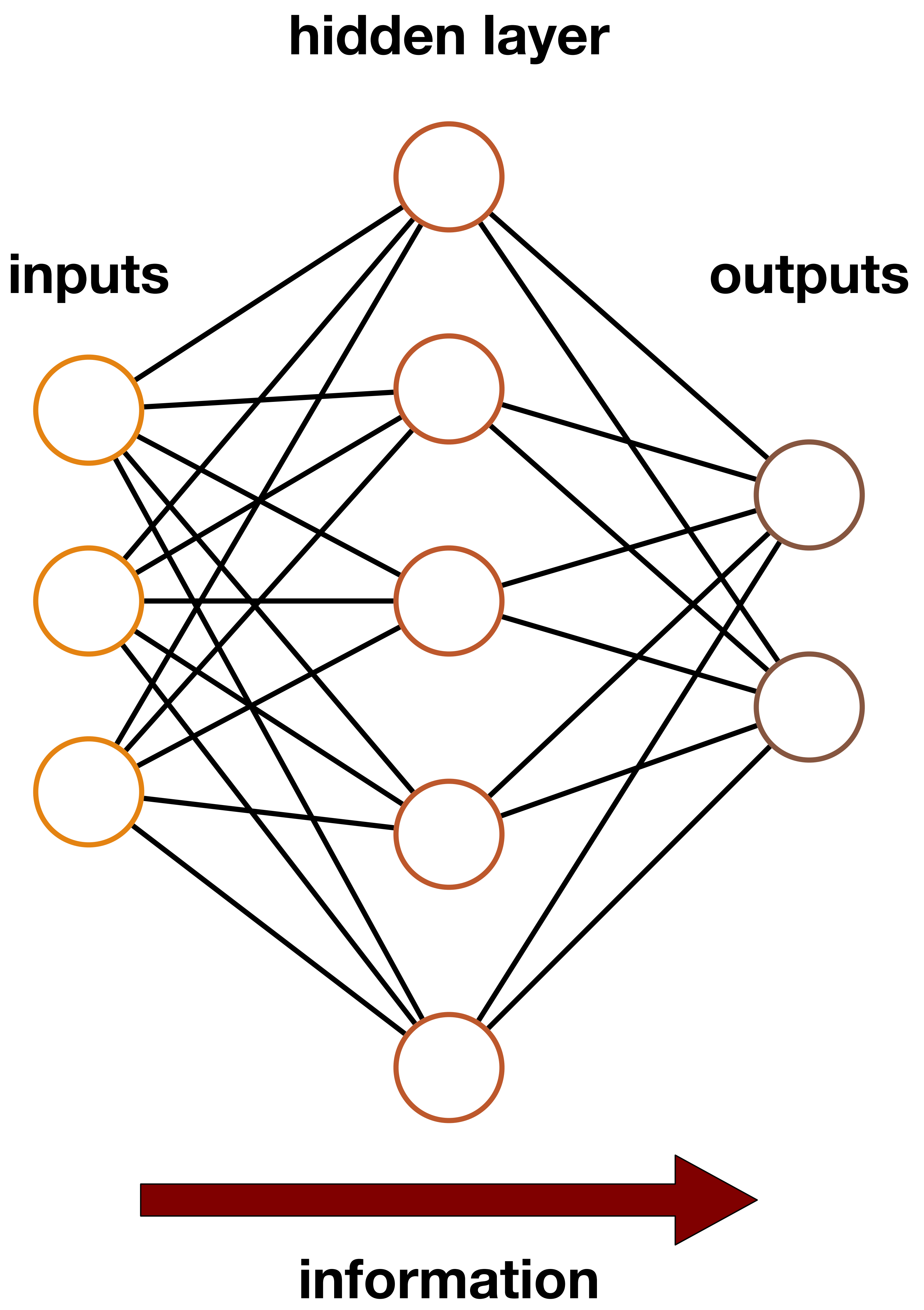
- Rectified linear (ReLU) $\sigma(z) = \max(0,z)$



$\sigma'(z) = \delta[z > 0]$

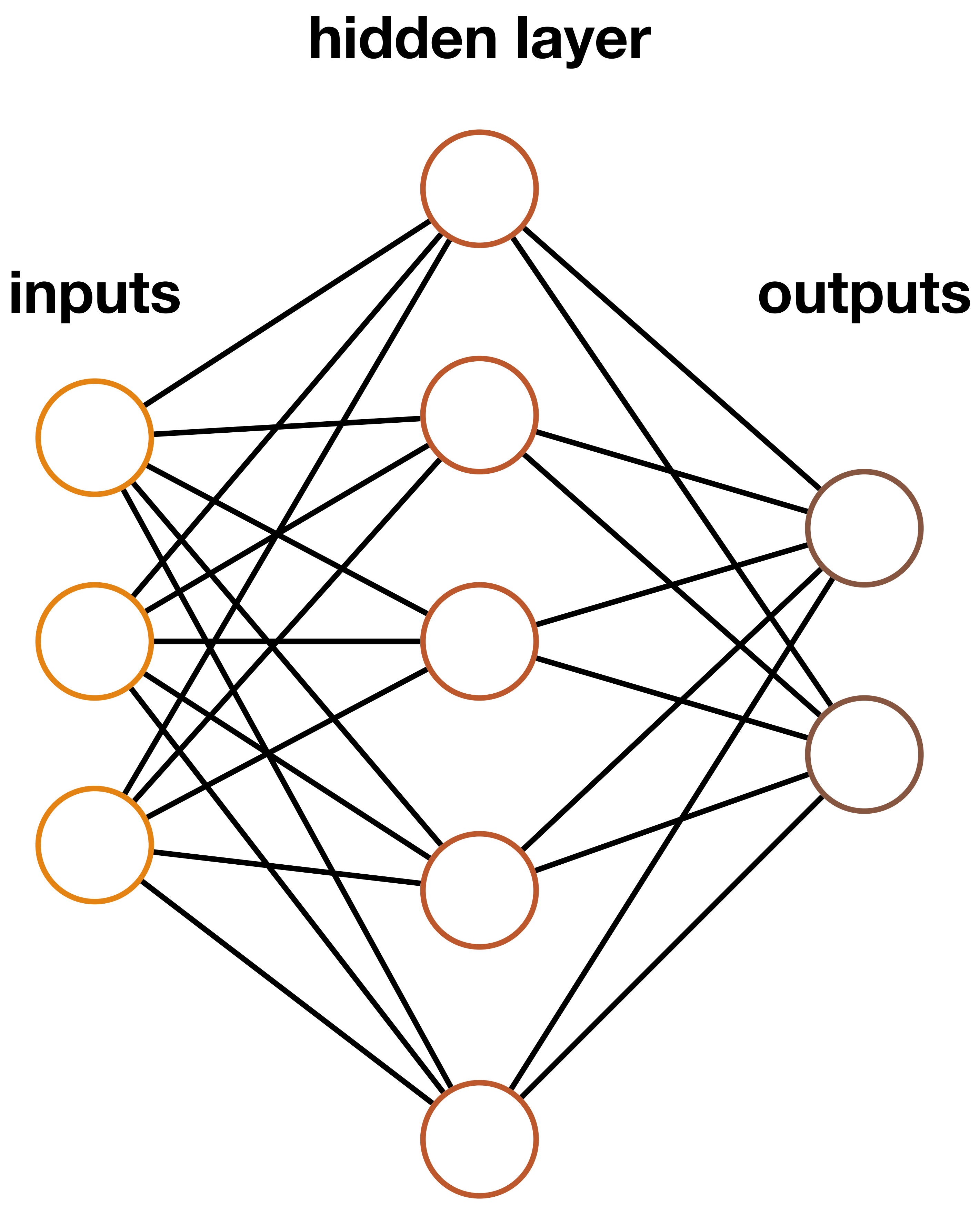
Feed-forward (FF) networks

- Information flow in feed-forward (FF) networks:
  - Inputs → shallow layers → deeper layers → outputs
  - Alternative: recurrent NNs (information loops back)
- Multiple outputs $\Longrightarrow$ efficiency:
  - Shared parameters, less data, less computation
- Multi-class classification:
  - One-hot labels $y = [0 \quad 0 \quad 1 \quad 0 \quad \cdots]$
  - Multilogistic regression (softmax): $\hat{y}_c = \dfrac{\exp(h_c)}{\sum_{\bar{c}} \exp(h_{\bar{c}})}$



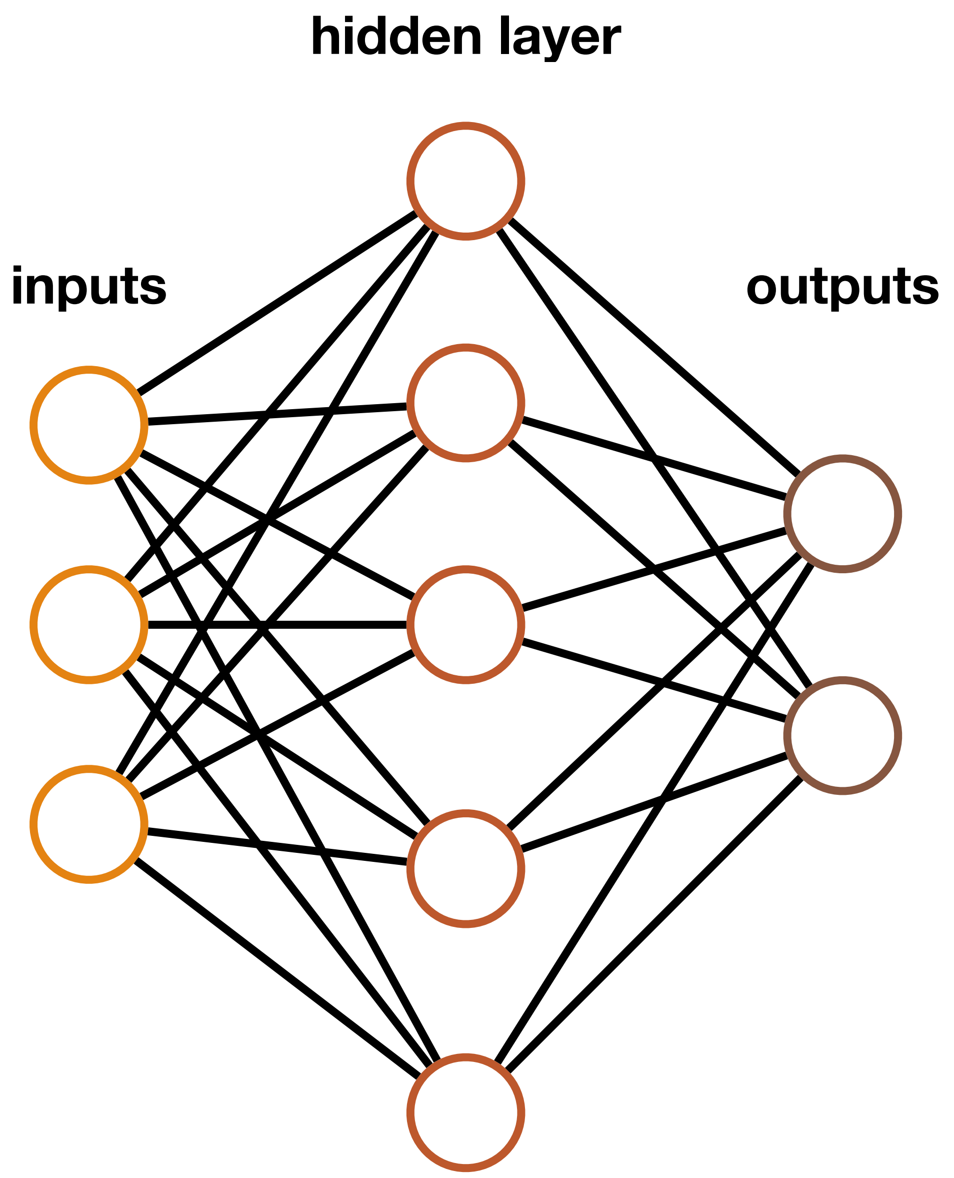
Training MLPs

- Observe instance x, target y
- Feed x forward through NN = prediction $\hat{y}$
- Loss = $\ell_w(y, \hat{y}) = (y - \hat{y})^2$ (or another loss function)
- How should we update the weights?
- Single layer:
  - Use differentiable activation function, e.g. logistic
  - (Stochastic) Gradient Descent = logistic regression

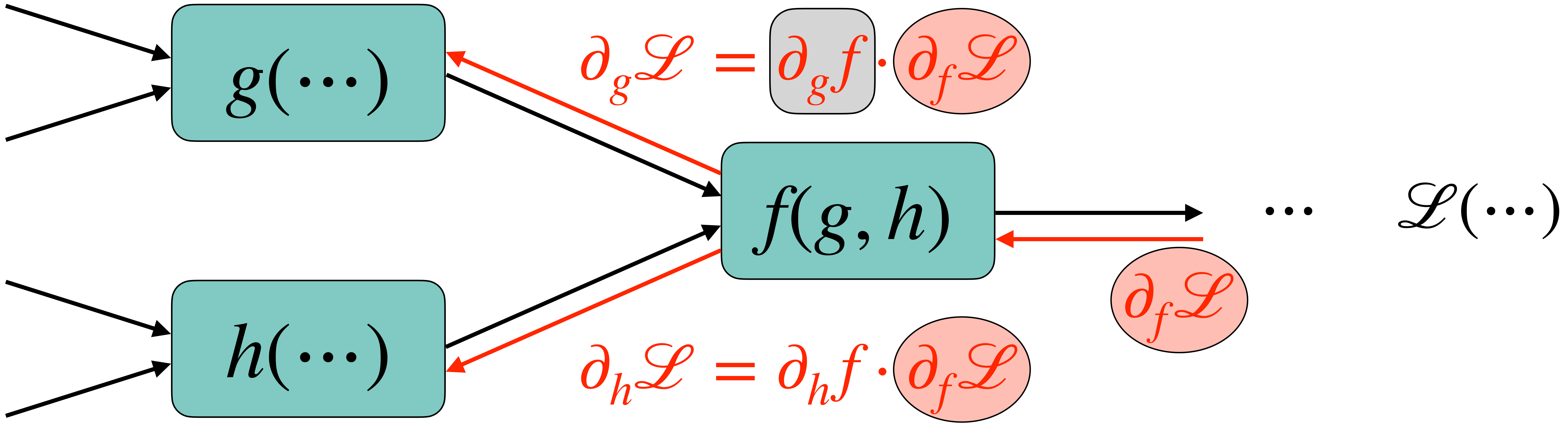


Gradient computation

- MLPs are function compositions of single layers



  - Apply chain rule:



example: $f(g,h) = \sigma(g+h) \implies \partial_g f = f(1-f)$

$\implies$ **reuse f from the forward pass**

- Backpropagation = chain rule + dynamic programming to avoid repetitions

Today's lecture

Multi-Layer Perceptrons

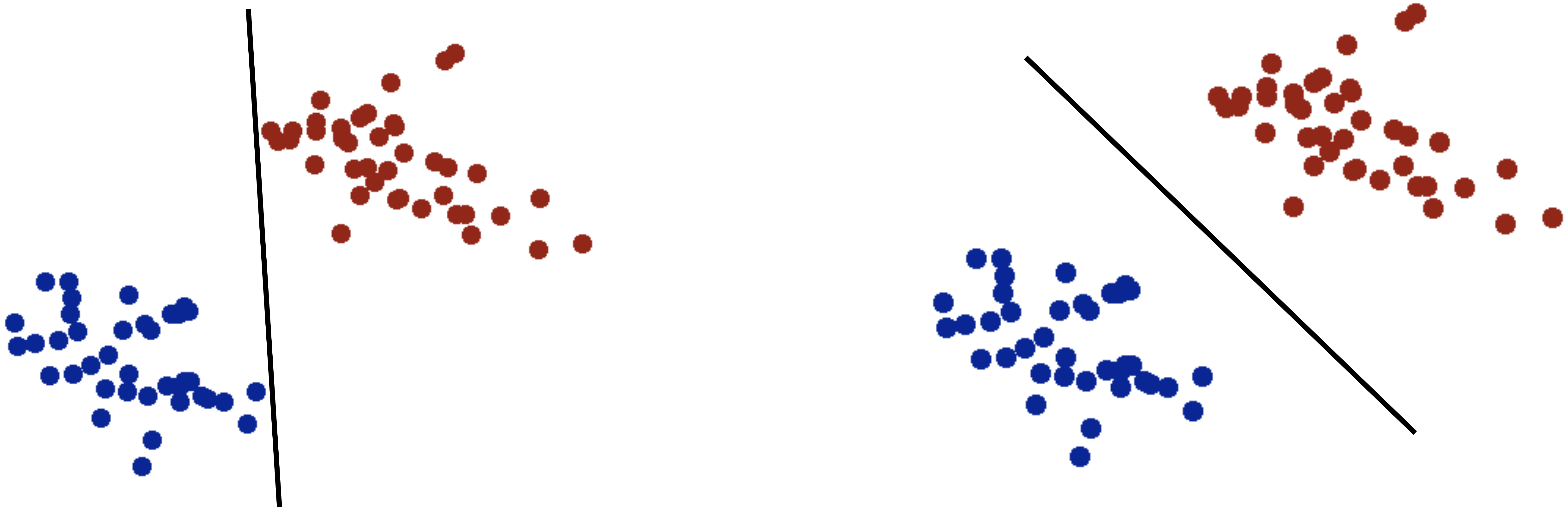
Support Vector Machines

Lagrangian and duality

Kernel Machines

Linear classifiers

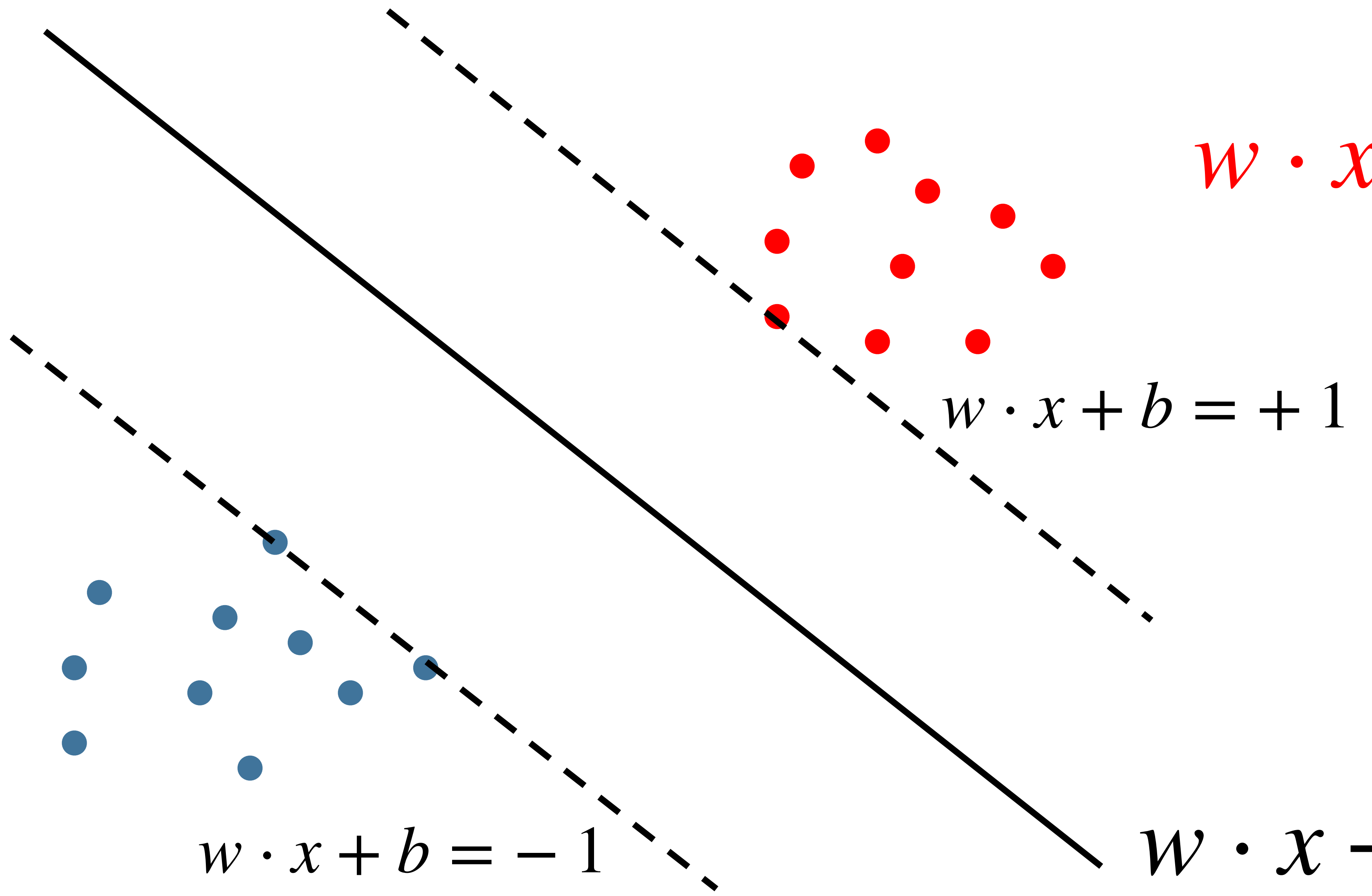
- Assume separable training data
- Which decision boundary is "better"?
  - Both have 0 training error, but one seems to generalize better
- Let's quantify this intuition



Decision margin

- Let's try to maximize the margin = distance of data from boundary
- Logistic regression: $\mathscr{L}_{w,b}(x, y) = y \log \sigma(w \cdot x + b) + (1 - y)\log(1 - \sigma(w \cdot x + b))$
  - What if we scale $w \cdot x + b \to 10w \cdot x + 10b$? $\implies$ loss gets better as $\sigma \to \pm 1$
  - Optimum at infinity! but the decision boundary $w \cdot x + b = 0$ is unchanged...

$w \cdot x + b > 0 \implies f(x) = +1$



scale invariance:
let's choose w such that

$w \cdot x + b < 0 \implies f(x) = -1$

$w \cdot x + b = 0$

Computing the margin

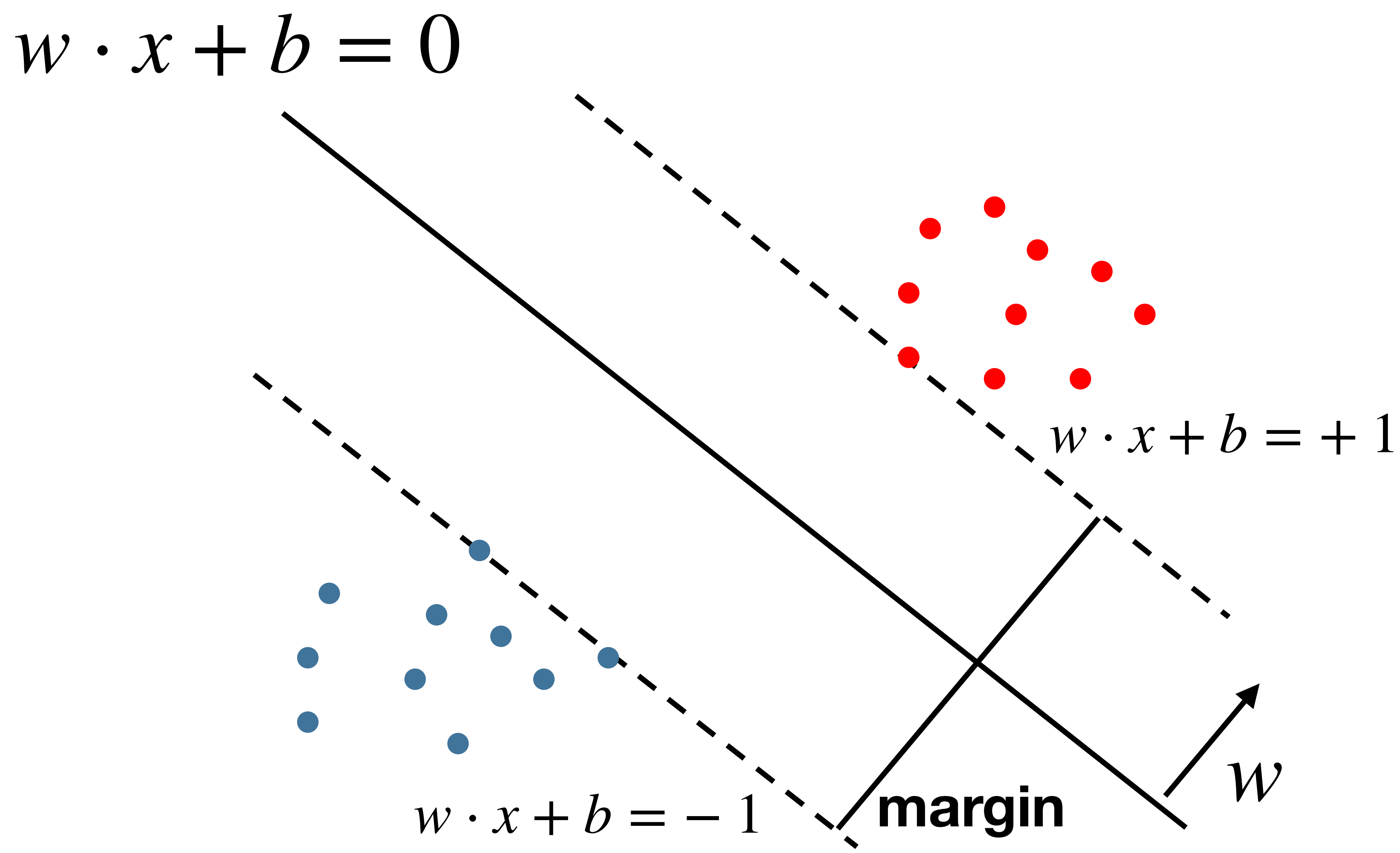
- Basic linear algebra: $x = rw + z = \frac{w \cdot x}{\|w\|^2} w + z$, with z orthogonal to w

- Support vectors = x^+ and x^- that are closest points to the boundary

$$w \cdot x^+ + b = +1$$
$$w \cdot x^- + b = -1$$
$$w \cdot (r^+ w + z^+ + b - r^- w - bz^- - b) = 2$$
$$(r^+ - r^-)\|w\|^2 = 2$$

- Margin $= \|(r^+ - r^-)w\| = \frac{2}{\|w\|}$

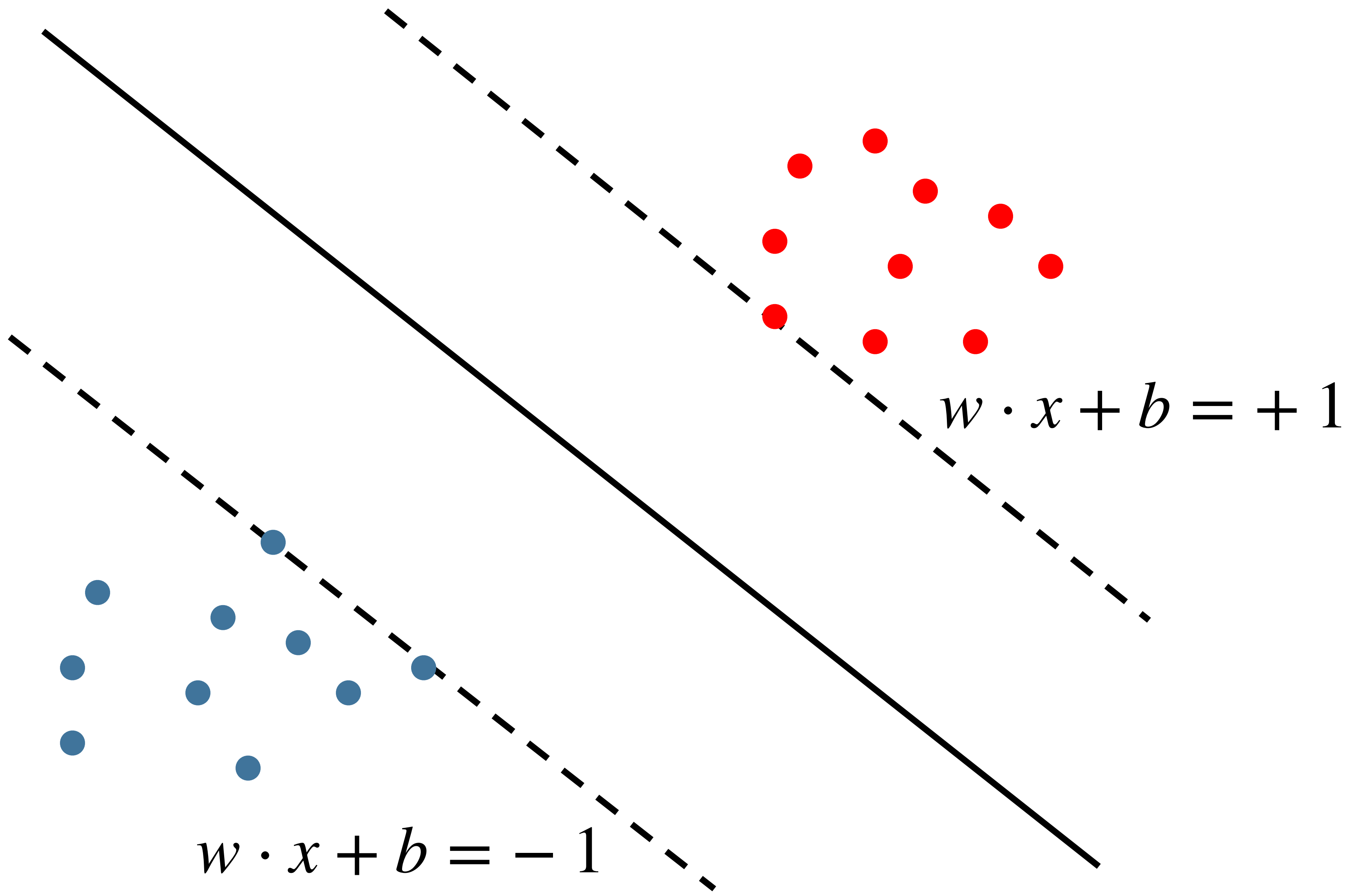
- Maximizing the margin = minimizing $\|w\|^2$



Maximizing the margin

- Constrained optimization: get all data points correctly + maximize the margin

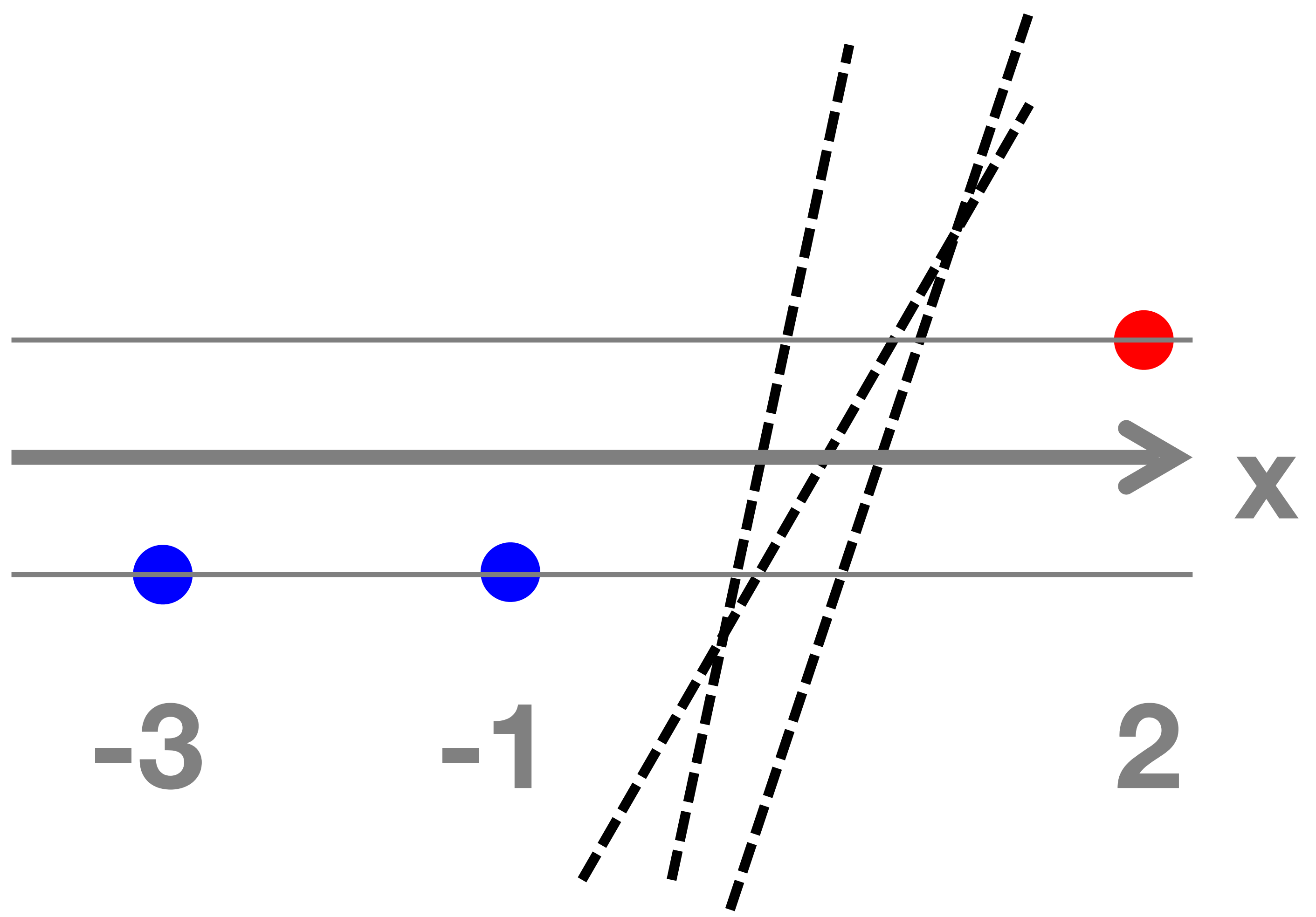
- $w^* = \arg\max_w \frac{2}{\|w\|} = \arg\min_w \|w\|$
  - such that all data points predicted with enough margin: $\begin{cases} w \cdot x^{(j)} + b \geq +1 & \text{if } y^{(j)} = +1 \\ w \cdot x^{(j)} + b \leq -1 & \text{if } y^{(j)} = -1 \end{cases}$
  - $\implies$ s.t. $y^{(j)}(w \cdot x^{(j)} + b) \geq 1$ (m constraints)



- Example of Quadratic Program (QP)
  - Quadratic objective, linear constraints

Example: one feature

- Suppose we have three data points
    - $x = -3, y = -1$
    - $x = -1, y = -1$
    - $x = 2, y = +1$



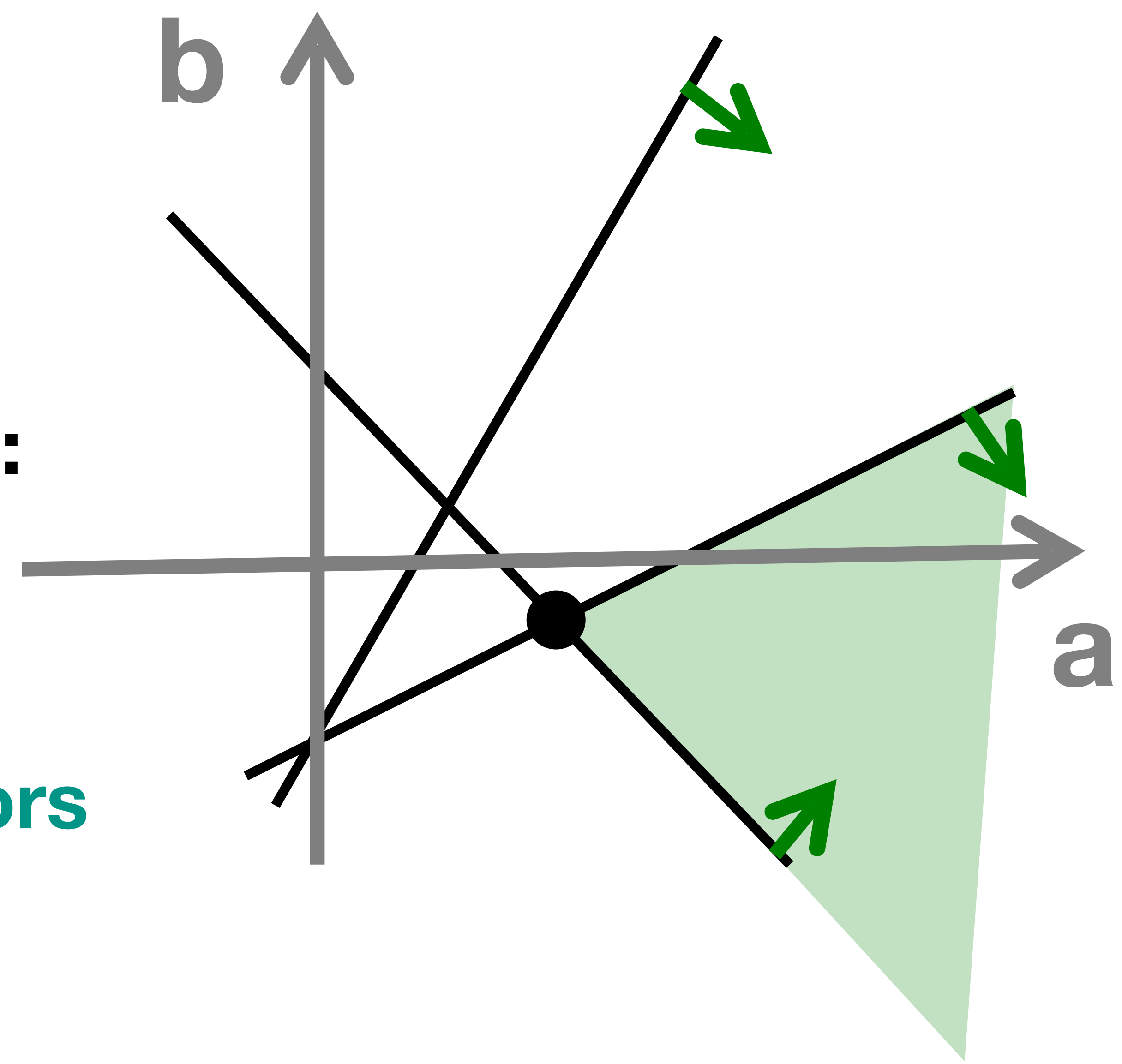
- Many separating perceptrons $T(ax + b)$
    - Separating if $a > 0$ and $-\frac{b}{a} \in (-1,2)$
- Margin constraints:
    - $-3a + b \leq -1 \implies b \leq 3a - 1$
    - $-1a + b \leq -1 \implies b \leq a - 1$
    - $+2a + b \geq +1 \implies b \geq -2a + 1$

minimize $|a|$ and set b to match:

$a = \frac{2}{3} \qquad b = -\frac{1}{3}$

2 constraints are active

$\implies$ these are the support vectors



Today's lecture

Multi-Layer Perceptrons

Support Vector Machines

Lagrangian and duality

Kernel Machines

Lagrange method

- Constrained optimization: $w^*, b^* = \arg\min_{w,b} \underbrace{\frac{1}{2}\|w\|^2}_{f(\theta)}$ s.t. $\underbrace{1 - y^{(j)}(w \cdot x^{(j)} + b)}_{g(\theta)} \leq 0$

- Lagrange method: introduce Lagrange multipliers λ_j (one per constraint)

$$\theta^* = \arg\min_{\theta} \max_{\lambda \geq 0} f(\theta) + \sum_j \lambda_j g_j(\theta)$$

  - If $g_j(\theta) < 0 \implies$ optimally, $\lambda_j = 0$
  - If $g_j(\theta) > 0 \implies$ optimally, $\lambda_j \to \infty \implies$ this θ cannot achieve the minimum
  - If $g_j(\theta) = 0 \implies$ doesn't matter; generally, $\lambda_j > 0$
  - Complementary slackness: for optimal θ, λ, if $\lambda_j > 0 \implies g_j(\theta) = 0$

Margin optimization

- Original problem: $w^*, b^* = \arg\min_{w,b} \frac{1}{2}\|w\|^2 \quad$ s.t. $1 - y^{(j)}(w \cdot x^{(j)} + b) \leq 0$

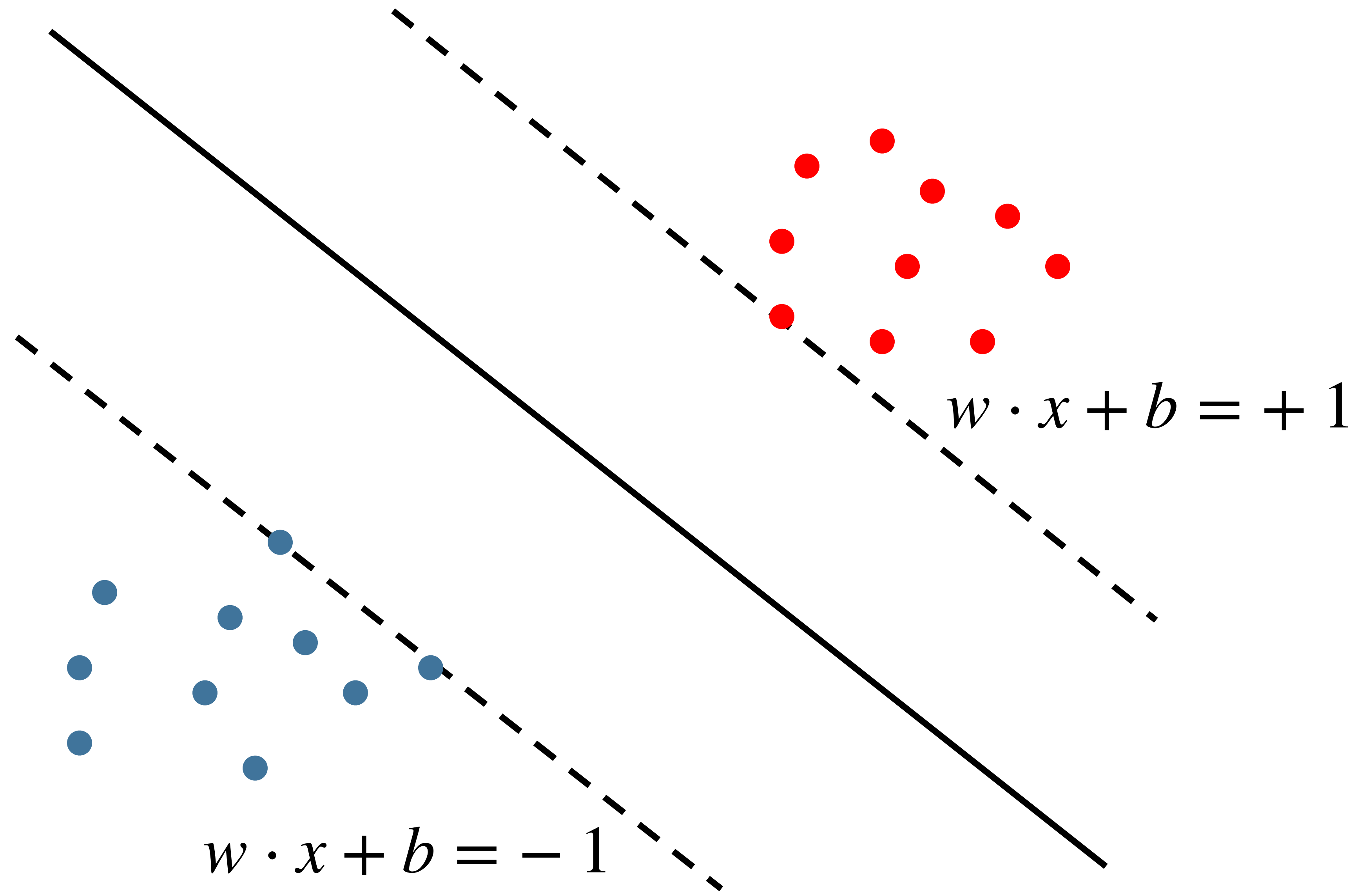
- Lagrangian: $w^*, b^* = \arg\min_{w,b} \max_{\lambda \geq 0} \frac{1}{2}\|w\|^2 + \sum_j \lambda_j (1 - y^{(j)}(w \cdot x^{(j)} + b))$

- Optimally: $w^* = \sum_j \lambda_j y^{(j)} x^{(j)}$

  - For support vector $j \in \mathrm{SV}$: $b^* = y^{(j)} - w^* \cdot x^{(j)}$

  - Lagrangian linear in b

$$\implies \sum_j \lambda_j y^{(j)} = 0 \text{ for } b^* \text{ to be finite}$$



Primal–dual optimization

- Primal problem: $w^*, b^* = \arg\min_{w,b} \frac{1}{2}\|w\|^2 \quad$ s.t. $1 - y^{(j)}(w \cdot x^{(j)} + b) \leq 0$

- Lagrangian: $w^*, b^* = \arg\min_{w,b} \max_{\lambda \geq 0} \frac{1}{2}\|w\|^2 + \sum_j \lambda_j (1 - y^{(j)}(w \cdot x^{(j)} + b))$

- Plug in the solution: $w = \sum_j \lambda_j y^{(j)} x^{(j)}$; constraint: $\sum_j \lambda_j y^{(j)} = 0$

  - Dual problem: $\max_{\lambda \geq 0} \sum_j \left(\lambda_j - \frac{1}{2} \sum_k \lambda_j \lambda_k y^{(j)} y^{(k)} x^{(j)} \cdot x^{(k)} \right) \quad$ s.t. $\sum_j \lambda_j y^{(j)} = 0$

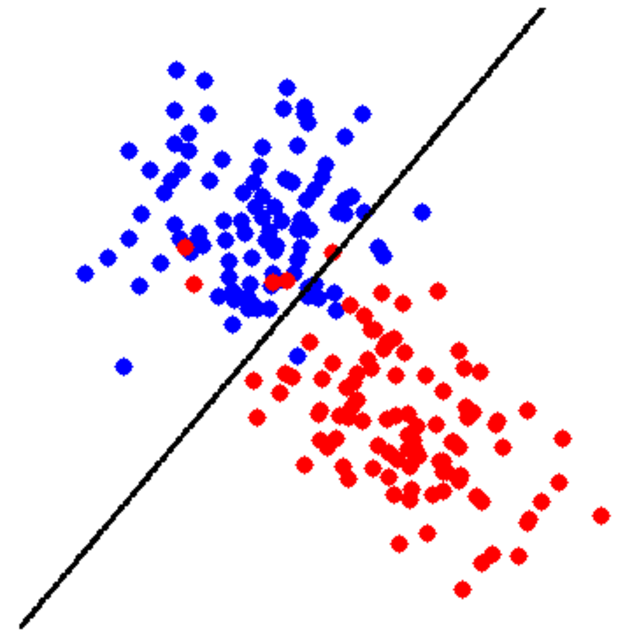
- Another Quadratic Program (QP):

  - Complicated objective in m variables; $m + 1$ simple constraints (instead of v.v.)

Non-separable problems

- SVM: $w^*, b^* = \arg\min_{w,b} \max_{\lambda \geq 0} \frac{1}{2}\|w\|^2 + \sum_j \lambda_j (1 - y^{(j)}(w \cdot x^{(j)} + b))$

- Can't work with non-separable data: constraints violated $\implies \lambda_j \to \infty$



- What if we fix $\lambda_j = R$?

$$w^*, b^* = \arg\min_{w,b} \frac{1}{2}\|w\|^2 - R \sum_j y^{(j)}(w \cdot x^{(j)} + b)$$

$$= \arg\min_{w,b} \sum_j |y^{(j)}M - (w \cdot x^{(j)} + b)| + \frac{1}{2R}\|w\|^2$$

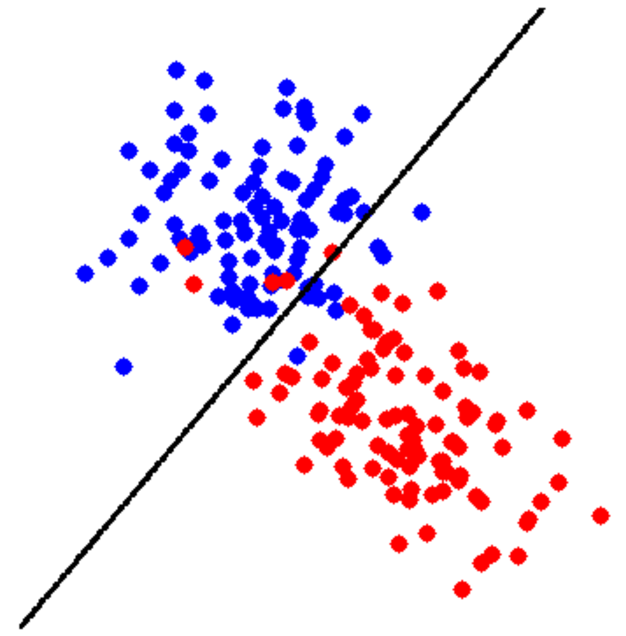
$M > |w \cdot x^{(j)} + b|$

- Similar to MAE + L_2 regularizer $\implies$ considers <u>all</u> data points (not just margin)

Soft margin

- Only consider points that violate the margin constraint:

$$\ell_{\text{hinge}}(y, \hat{y}) = \max\{0, 1 - y\hat{y}\}$$



$$w^*, b^* = \arg\min_{w,b} \frac{1}{2}\|w\|^2 + R\sum_j \ell_{\text{hinge}}(y^{(j)}, w \cdot x^{(j)} + b)$$

  - $\epsilon^{(j)} = \max\{0, 1 - y^{(j)}(w \cdot x^{(j)} + b)\}$ = how much is margin constraint violated

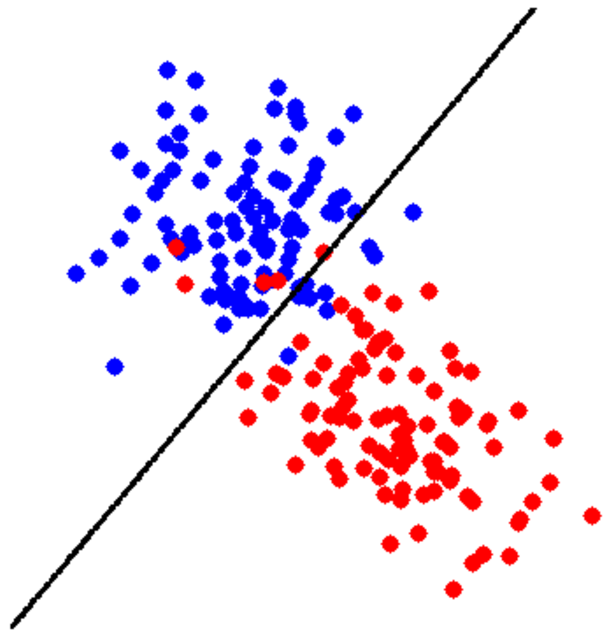
- Primal problem: $w^*, b^* = \arg\min_{w,b} \min_{\epsilon} \frac{1}{2}\|w\|^2 + R\sum_j \epsilon^{(j)}$

  - s.t. $y^{(j)}(w \cdot x^{(j)} + b) \geq 1 - \epsilon^{(j)}$ (relaxed constraints satisfied)

  - $\epsilon^{(j)} \geq 0$ (only “snug fit” violating points)

Soft margin: dual form

- Primal problem: $w^*, b^* = \arg\min_{w,b} \min_{\epsilon} \frac{1}{2}\|w\|^2 + R\sum_j \epsilon^{(j)}$
  - s.t. $y^{(j)}(w \cdot x^{(j)} + b) \geq 1 - \epsilon^{(j)}; \quad \epsilon^{(j)} \geq 0$



- Dual problem: $\max_{0 \leq \lambda \leq R} \sum_j \left(\lambda_j - \frac{1}{2} \sum_k \lambda_j \lambda_k y^{(j)} y^{(k)} x^{(j)} \cdot x^{(k)} \right)$ s.t. $\sum_j \lambda_j y^{(j)} = 0$
  - Optimally: $w^* = \sum_j \lambda_j y^{(j)} x^{(j)}$; to handle b: add constant feature $x_0 = 1$
  - Support vector = points on or inside margin = $\lambda_j > 0$
  - Gram matrix = $K_{jk} = x^{(j)} \cdot x^{(k)}$ = similarity of every pair of instances

Today's lecture

Multi-Layer Perceptrons

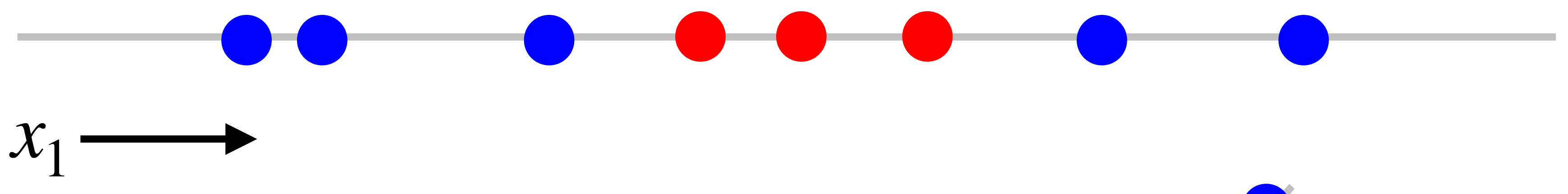
Support Vector Machines

Lagrangian and duality

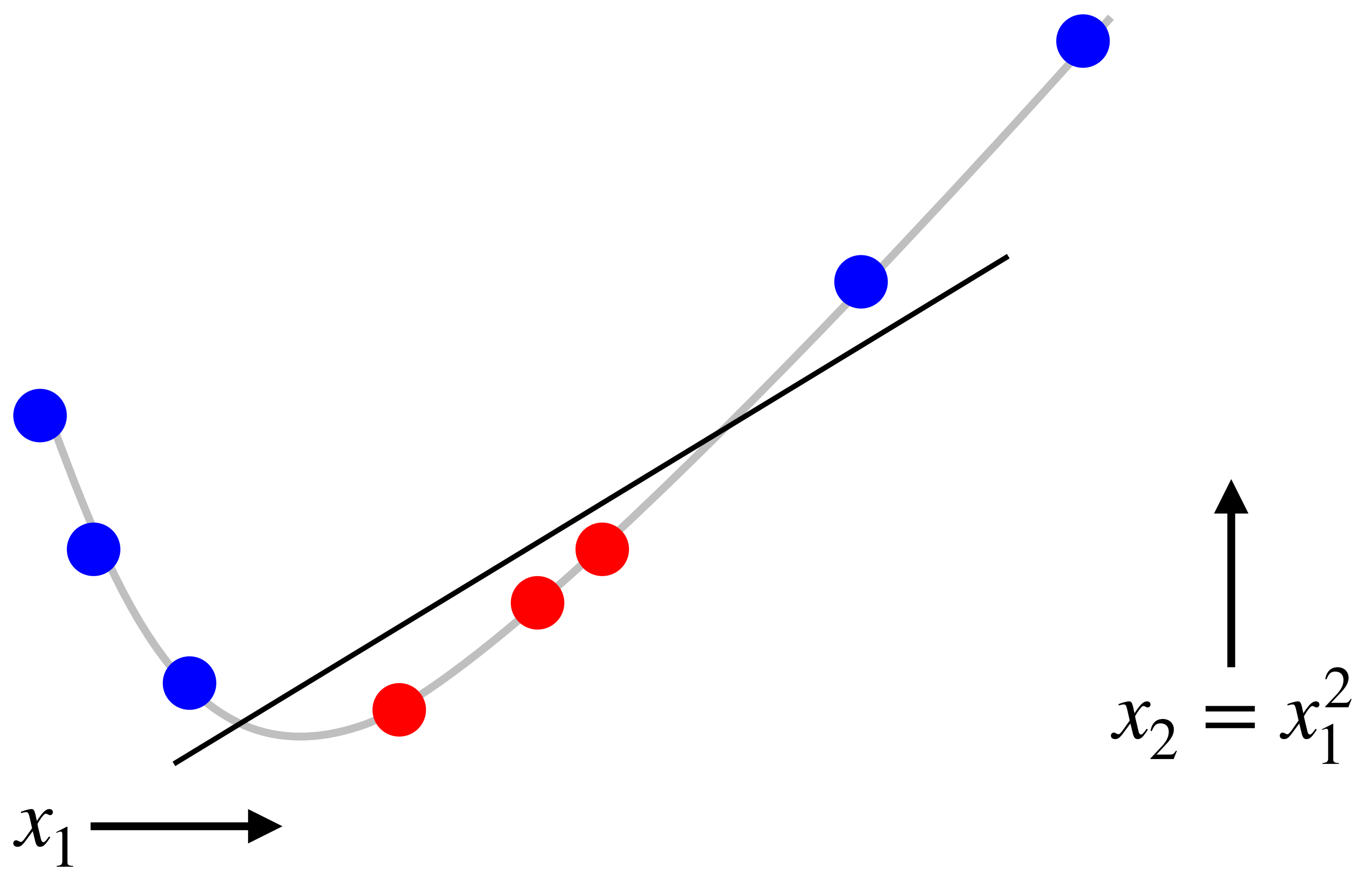
Kernel Machines

Adding features

- So far: linear SVMs, not very expressive
  - $\Longrightarrow$ add features $x \mapsto \Phi(x)$
- Linearly non-separable:



- Linearly separable in quadratic features:



Adding features

- Prediction: $\hat{y}(x) = \text{sign}(w \cdot \Phi(x) + b)$
- Dual problem: $\max_{0 \le \lambda \le R} \sum_j \left(\lambda_j - \frac{1}{2} \sum_k \lambda_j \lambda_k y^{(j)} y^{(k)} \Phi(x^{(j)}) \cdot \Phi(x^{(k)}) \right)$ s.t. $\sum_j \lambda_j y^{(j)} = 0$
- Example: quadratic features $\Phi(x) = \begin{bmatrix} 1 & \sqrt{2}x_i & x_i^2 & \sqrt{2}x_i x_{i'} \end{bmatrix}$
  - n features $\mapsto O(n^2)$ features
  - Why $\sqrt{2}$? Next slide... But just scale corresponding weights

Implicit features

- For dual problem, we need $K_{jk} = \Phi(x^{(j)}) \cdot \Phi(x^{(k)})$

- Kernel trick: with $\Phi(x) = \begin{bmatrix} 1 & \sqrt{2}x_i & x_i^2 & \sqrt{2}x_i x_{i'} \end{bmatrix}$:

$$K_{jk} = 1 + \sum_i 2x_i^{(j)}x_i^{(k)} + \sum_i (x_i^{(j)}x_i^{(k)})^2 + \sum_{i<i'} 2(x_i^{(j)}x_i^{(k)})(x_{i'}^{(j)}x_{i'}^{(k)})$$

$$= \left(1 + \sum_i x_i^{(j)}x_i^{(k)}\right)^2$$

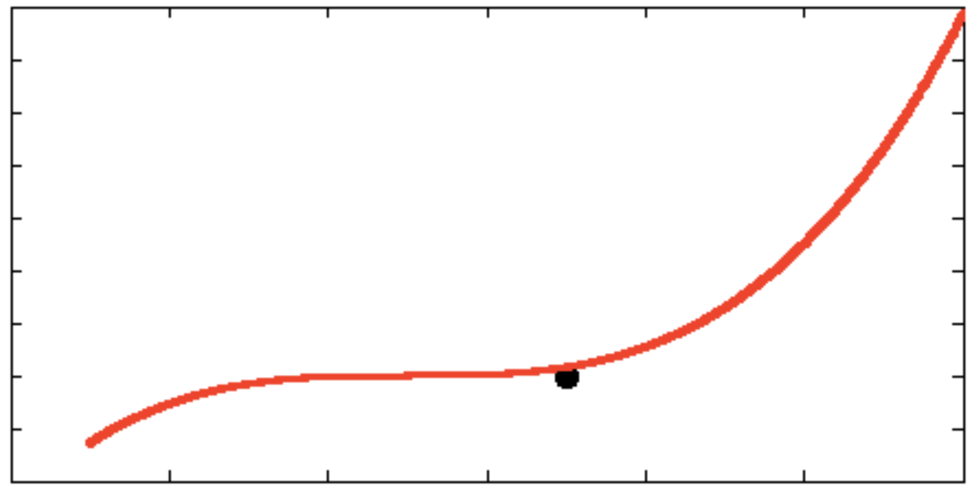
  - Each of m^2 elements computed in $O(n)$ time (instead of $O(n^2)$)

Mercer's Theorem

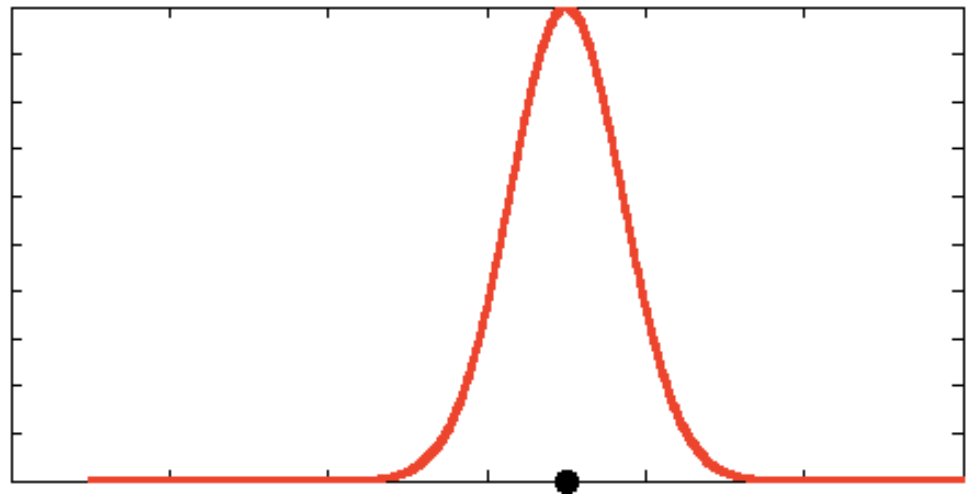
- Reminder: positive semidefinite matrix $A \succeq 0$: $v^\intercal A v \geq 0$ for all vectors v
- Positive semidefinite kernel $K \succeq 0$: matrix $K(x^{(j)}, x^{(k)}) \succeq 0$ for all datasets
- Mercer's Theorem: if $K \succeq 0 \implies K(x, x') = \Phi(x) \cdot \Phi(x')$ for some $\Phi(x)$
- Φ may be hard to calculate
  - May even be infinite dimensional (Hilbert space)
  - Not an issue, only the kernel $K(x, x')$ should be easy to compute ($O(m^2)$ times)

Common kernel functions

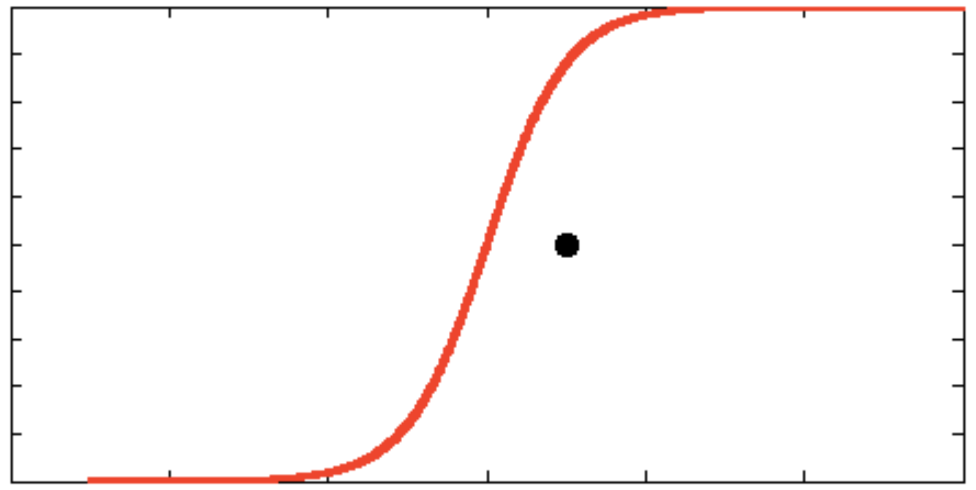
- Polynomial: $K(x, x') = (1 + x \cdot x')^d$



- Radial Basis Functions (RBF): $K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$



- Saturating: $K(x, x') = \tanh(ax \cdot x' + c)$



- Domain-specific: textual similarity, genetic code similarity, ...
  - May not be positive semidefinite, and still work well in practice

Kernel SVMs

- Define kernel $K : (x, x') \mapsto \mathbb{R}$
- Solve dual QP: $\max_{0 \le \lambda \le R} \sum_j \left(\lambda_j - \frac{1}{2} \sum_k \lambda_j \lambda_k y^{(j)} y^{(k)} K(x^{(j)}, x^{(k)}) \right)$ s.t. $\sum_j \lambda_j y^{(j)} = 0$
- Learned parameters = λ (m parameters)
  - But also need to store all support vectors (having $\lambda_j > 0$)
- Prediction: $\hat{y}(x) = \operatorname{sign}(w \cdot \Phi(x))$

$$= \operatorname{sign}\left(\sum_j \lambda_j y^{(j)} \Phi(x^{(j)}) \cdot \Phi(x) \right) = \operatorname{sign}\left(\sum_j \lambda_j y^{(j)} K(x^{(j)}, x) \right)$$

Demo

- https://cs.stanford.edu/people/karpathy/svmjs/demo/

Linear vs. kernel SVMs

- Linear SVMs
  - $\hat{y} = \text{sign}(w \cdot x + b) \Longrightarrow n + 1$ parameters
  - Alternatively: represent by indexes of SVs; usually, #SVs = #parameters
- Kernel SVMs
  - $K(x, x')$ may correspond to high- (possibly infinite-) dimensional $\Phi(x)$
  - Typically more efficient to store the SVs $x^{(j)}$ (not $\Phi(x^{(j)})$)
    - And their corresponding λ_j

Recap

- Maximize margin for separable data
  - Primal QP: maximize $||w||^2$ subject to linear constraints
  - Dual QP: m variables, m^2 dot products
- Soft margin for non-separable data
  - Primal problem: regularized hinge loss
  - Dual problem: m-dimensional QP
- Kernel Machines
  - Dual form involves only pairwise similarity
  - Mercer kernels: equivalent to dot products in implicit high-dimensional space

Logistics

assignments

- Assignment 4 will be published soon, due Fri, Nov 12

project

- Project abstract due Tue, Nov 16

midterm

- Midterm exam on Thu, Nov 4, 11am–12:20 in **SH 128**
- If you're eligible to be remote — let us know immediately