University of California, Irvine

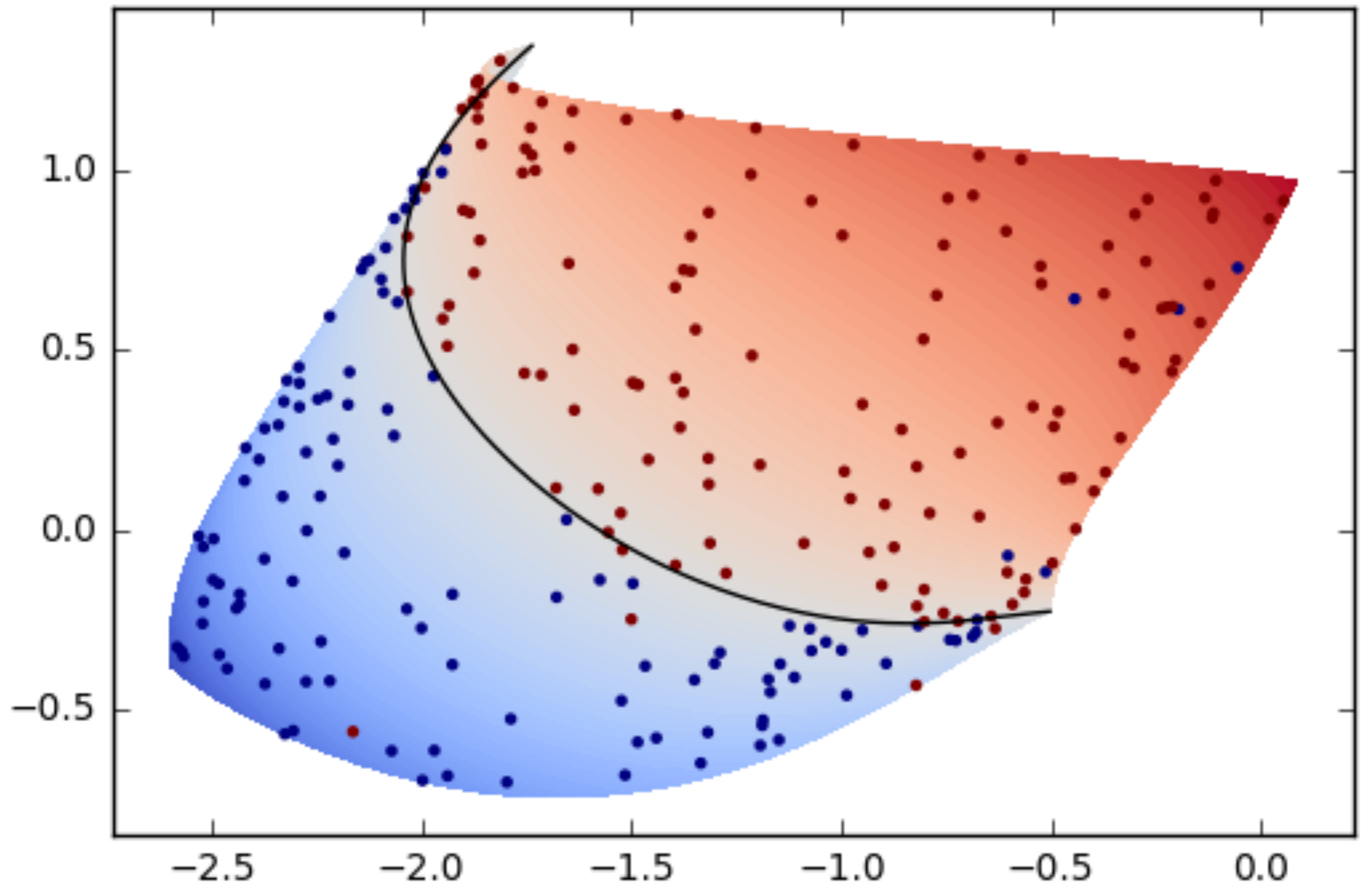
CS 273A: Machine Learning

Fall 2021

Lecture 10: VC Dimension

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 3 due next Tuesday, Nov 2

midterm

- Midterm exam on Nov 4, 11am–12:20 in **SH 128**
- If you're eligible to be remote — let us know by Oct 28
- If you're eligible for more time — let us know by Oct 28
- Review during lecture this Thursday

Today's lecture

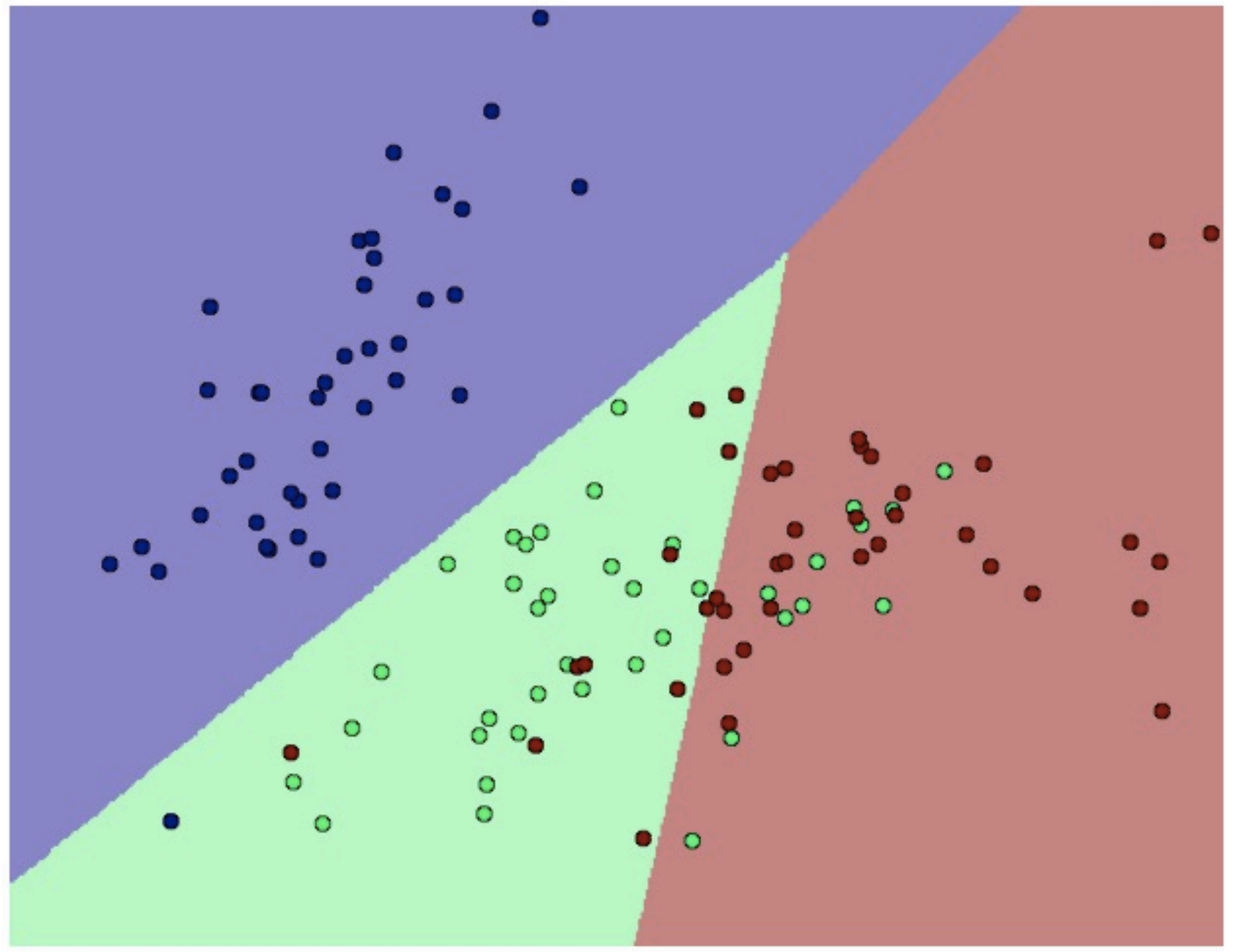
Multi-class classifiers

VC dimension

Multilayer perceptrons

Multi-class linear models

- How to predict multiple classes?
- Idea: have a linear response per class $r_c = \theta_c^\intercal x$
  - Choose class with largest response: $f_\theta(x) = \arg\max_c \theta_c^\intercal x$
- Linear boundary between classes c_1, c_2:
  - $\theta_{c_1}^\intercal x \lesseqgtr \theta_{c_2}^\intercal x \iff (\theta_{c_1} - \theta_{c_2})^\intercal x \lesseqgtr 0$



Multi-class linear models

- More generally: add features — can even depend on y!

$$f_\theta(x) = \arg\max_y \theta^\intercal \Phi(x, y)$$

- Example: $y = \pm 1$
  - $\Phi(x, y) = xy$

$$\implies f_\theta(x) = \arg\max_y y\theta^\intercal x = \begin{cases} +1 & +\theta^\intercal x > -\theta^\intercal x \\ -1 & +\theta^\intercal x < -\theta^\intercal x \end{cases}$$

$$= \operatorname{sign}(\theta^\intercal x) \longleftarrow \textbf{perceptron!}$$

Multi-class linear models

- More generally: add features — can even depend on y!

$$f_\theta(x) = \arg\max_y \theta^\intercal \Phi(x, y)$$

- Example: $y \in \{1,2,\ldots,C\}$
  - $\Phi(x, y) = [0 \;\; 0 \;\; \cdots \;\; x \;\; \cdots \;\; 0] =$ one-hot$(y) \otimes x$
  - $\theta = [\theta_1 \;\; \cdots \;\; \theta_C]$

$$\implies f_\theta(x) = \arg\max_c \theta_c^\intercal x \longleftarrow \text{largest linear response}$$

Multi-class perceptron algorithm

- While not done:
  - For each data point $(x, y) \in \mathcal{D}$:
    - Predict: $\hat{y} = \arg\max_c \theta_c^\intercal x$
    - Increase response for true class: $\theta_y \leftarrow \theta_y + \alpha x$
    - Decrease response for predicted class: $\theta_{\hat{y}} \leftarrow \theta_{\hat{y}} - \alpha x$
- More generally:
  - Predict: $\hat{y} = \arg\max_y \theta^\intercal \Phi(x, y)$
  - Update: $\theta \leftarrow \theta + \alpha(\Phi(x, y) - \Phi(x, \hat{y}))$

Multilogit Regression

- Define multi-class probabilities: $p_\theta(y|x) = \dfrac{\exp(\theta_y^\intercal x)}{\sum_c \exp(\theta_c^\intercal x)} = \operatorname{soft\,max}_c \theta_c^\intercal x \Big|_y$

  "logit" for c

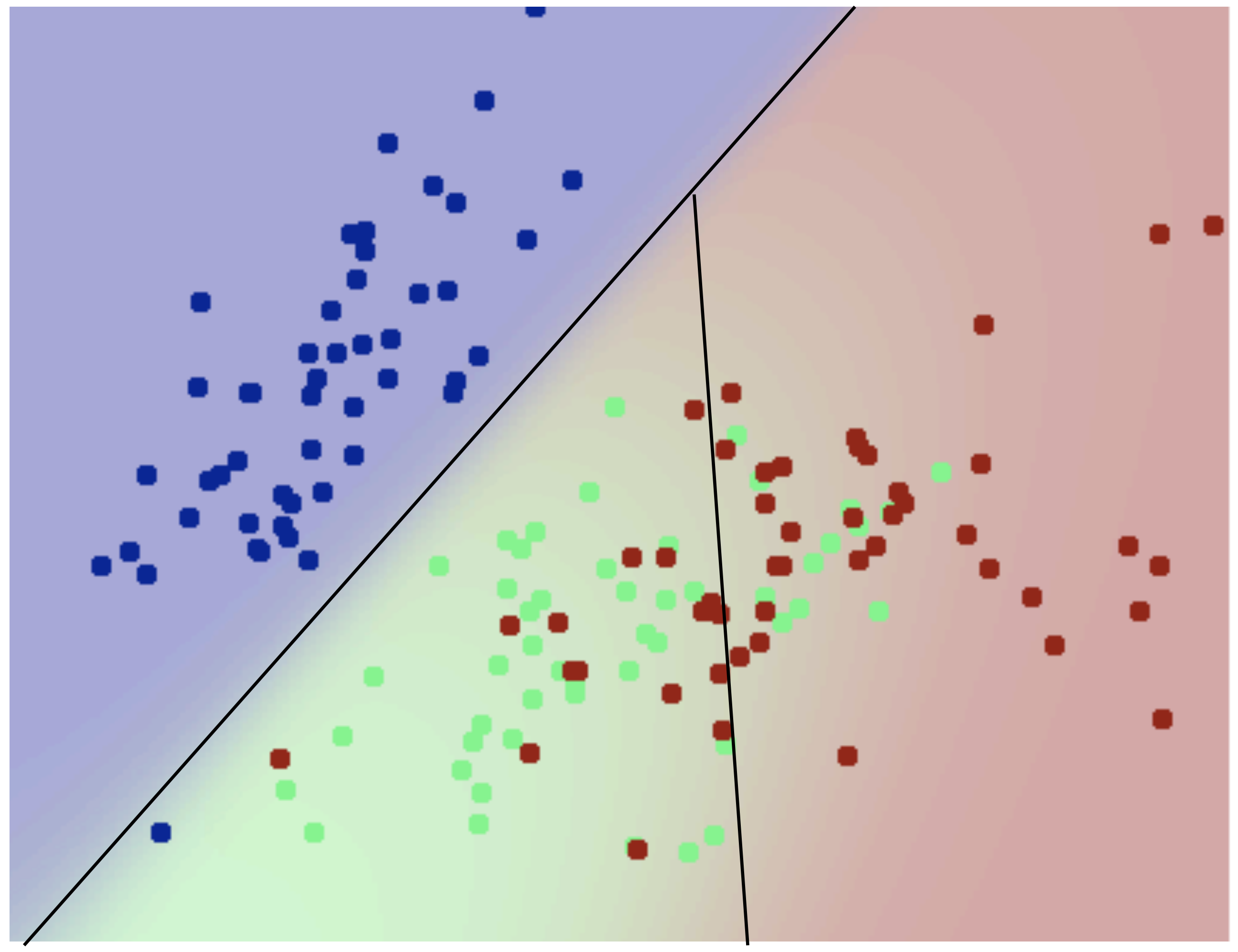
  - For binary y:

$$p_\theta(y=1|x) = \frac{\exp(\theta_1^\intercal x)}{\exp(\theta_1^\intercal x) + \exp(\theta_2^\intercal x)}$$

$$= \frac{1}{1+\exp((\theta_2-\theta_1)^\intercal x)} = \sigma((\theta_1-\theta_2)^\intercal x)$$

Logistic Regression with $\theta = \theta_1 - \theta_2$

- Benefits:
  - Probabilistic predictions: knows its confidence
  - Linear decision boundary: $\arg\max_y \exp(\theta_y^\intercal x) = \arg\max_y \theta_y^\intercal x$
  - NLL is convex



Multilogit Regression: gradient

- NLL loss: $\mathscr{L}_\theta(x,y) = -\log p_\theta(y\,|\,x) = -\theta_y^\intercal x + \log \sum_c \exp(\theta_c^\intercal x)$

- Gradient:

$$-\nabla_{\theta_c}\mathscr{L}_\theta(x,y) = \delta(y=c)x - \frac{\nabla_{\theta_c}\sum_{c'}\exp(\theta_{c'}^\intercal x)}{\sum_{c'}\exp(\theta_{c'}^\intercal x)}$$

$$= \left(\delta(y=c) - \frac{\exp(\theta_c^\intercal x)}{\sum_{c'}\exp(\theta_{c'}^\intercal x)}\right)x$$

$$= (\delta(y=c) - p_\theta(c\,|\,x))x$$

make true class more likely

make all other classes less likely

- Compare to multi-class perceptron: $(\delta(y=c) - \delta(\hat{y}=c))x$

Today's lecture

Multi-class classifiers

VC dimension

Multilayer perceptrons

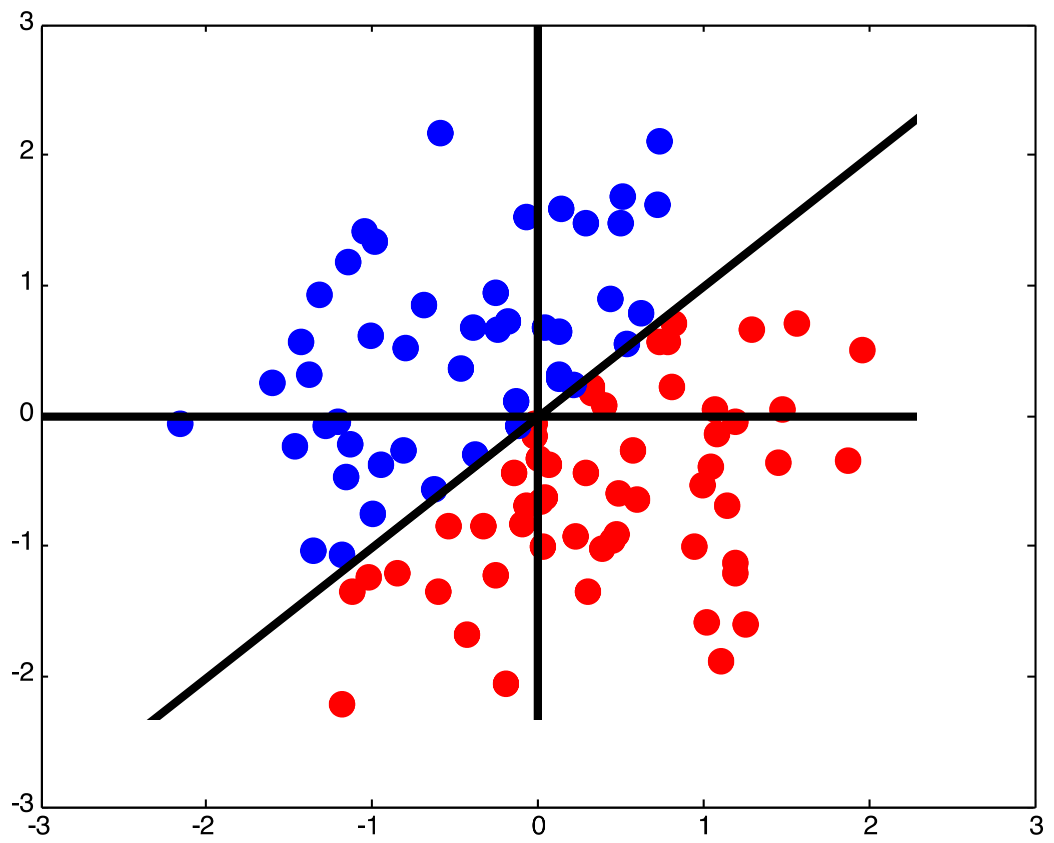
Complexity measures

- What are we looking for in a measure of model class complexity?
  - Tell us something about generalization error $\mathscr{L}_{\text{test}} - \mathscr{L}_{\text{training}}$
  - Tell us how error depends on amount of data m

    also called: risk – empirical risk
  - Have a recipe for finding the complexity of a given model class
- Ideally: a way to select model complexity (other than validation)
  - Akaike Information Criterion (AIC) — roughly: loss + #parameters
  - Bayesian Information Criterion (BIC) — roughly: loss + #parameters $\cdot \log m$
    - But what's the #parameters, effectively? Should $f_{\theta_1,\theta_2} = g_{\theta=h(\theta_1,\theta_2)}$ change the complexity?

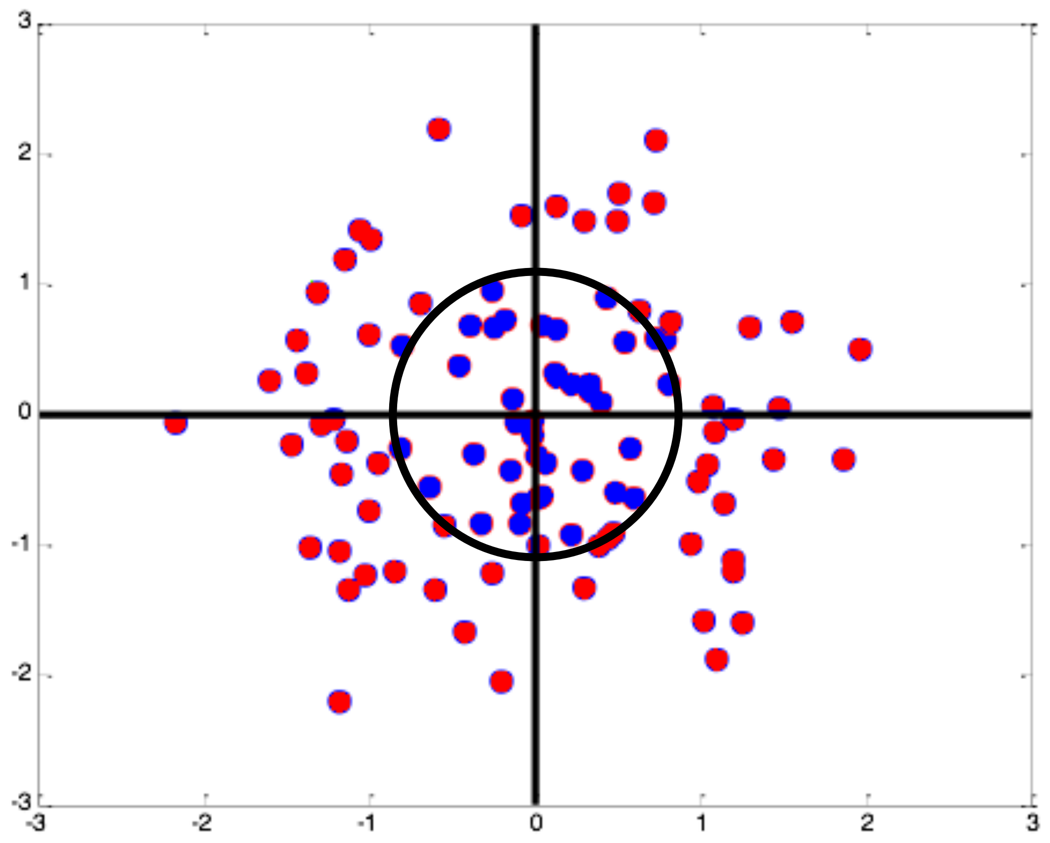
Model expressiveness

- Model complexity also measures expressiveness / representational power
- Tradeoff:
  - More expressive $\implies$ can reduce error, but may also overfit to training data
  - Less expressive $\implies$ may not be able to represent true pattern / trend
- Example: $\text{sign}(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$



Model expressiveness

- Model complexity also measures expressiveness / representational power
- Tradeoff:
  - More expressive $\implies$ can reduce error, but may also overfit to training data
  - Less expressive $\implies$ may not be able to represent true pattern / trend
- Example: $\text{sign}(x_1^2 + x_2^2 - \theta)$



Shattering

- Separability / realizability: there's a model that classifies all points correctly
- Shattering: the points are separable <u>regardless of their labels</u>
  - Our model class can shatter points $x^{(1)}, \ldots, x^{(h)}$

    if for <u>any</u> labeling $y^{(1)}, \ldots, y^{(h)}$

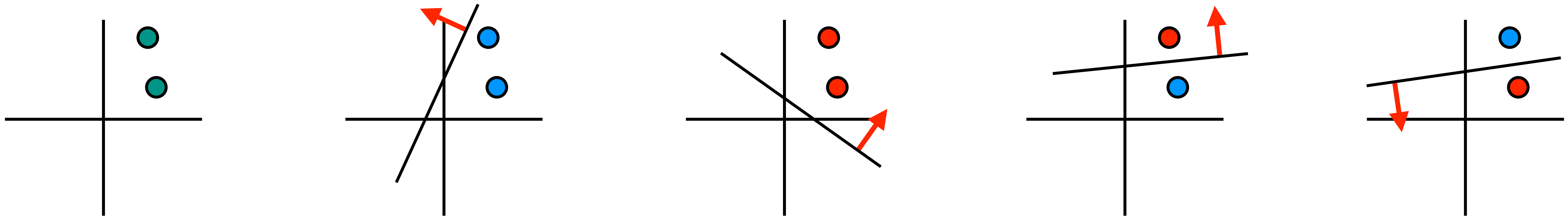
    there <u>exists</u> a model that classifies all of them correctly
    - The model class must have at least as many models as labelings C^h

Shattering

- Separability / realizability: there's a model that classifies all points correctly
- Shattering: the points are separable regardless of their labels
  - Our model class can shatter points $x^{(1)}, \ldots, x^{(h)}$

    if for any labeling $y^{(1)}, \ldots, y^{(h)}$

    there exists a model that classifies all of them correctly
- Example: can $f_\theta(x) = \text{sign}(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$ shatter these points?

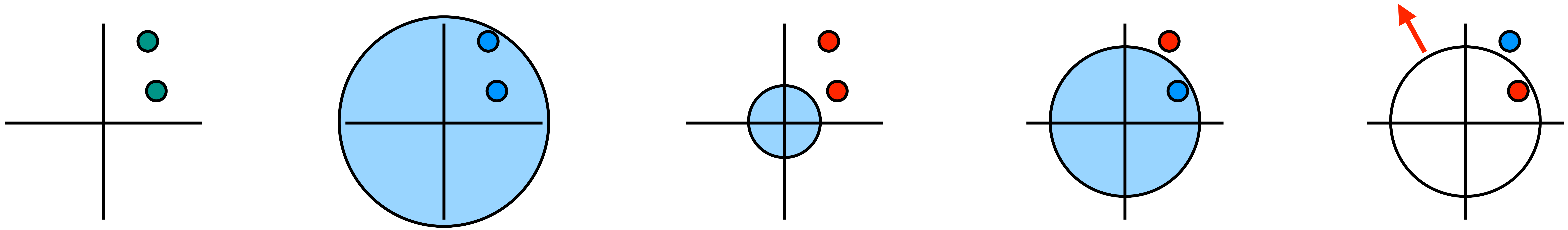


Shattering

- Separability / realizability: there's a model that classifies all points correctly
- Shattering: the points are separable regardless of their labels
    - Our model class can shatter points $x^{(1)}, \dots, x^{(h)}$

      if for any labeling $y^{(1)}, \dots, y^{(h)}$

      there exists a model that classifies all of them correctly
- Example: can $f_\theta(x) = \text{sign}(x_1^2 + x_2^2 - \theta)$ shatter these points?

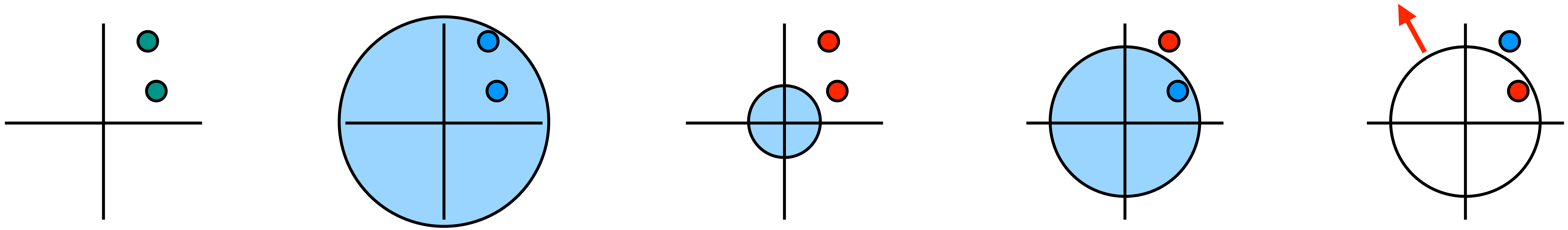


Vapnik–Chervonenkis (VC) dimension

- VC dimension: maximum number H of points that can be shattered by a class
- A game:
  - Fix a model class $f_\theta : x \to y \quad \theta \in \Theta$
  - Player 1: choose h points $x^{(1)}, \ldots, x^{(h)}$
  - Player 2: choose labels $y^{(1)}, \ldots, y^{(h)}$
  - Player 1: choose model θ
  - Are all $y^{(j)} = f_\theta(x^{(j)})$? $\implies$ Player 1 wins $\quad \exists x^{(1)}, \ldots, x^{(h)} : \ \forall y^{(1)}, \ldots, y^{(h)} : \ \exists \theta : \ \forall j : \ y^{(j)} = f_\theta(x^{(j)})$
- $h \leq H \implies$ Player 1 can win, otherwise cannot win

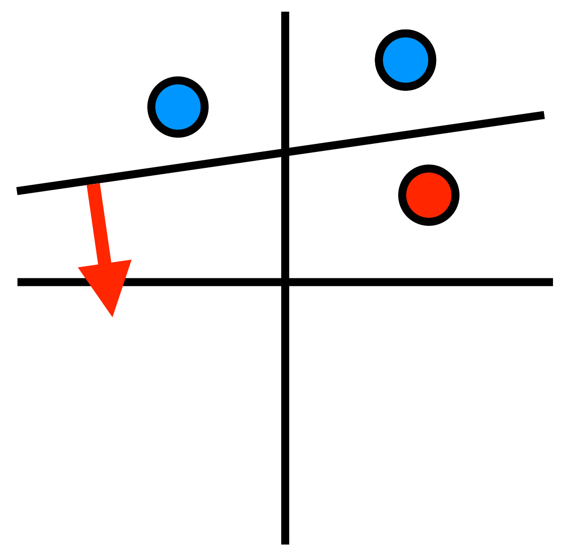
VC dimension: example (1)

- VC dimension: maximum number H of points that can be shattered by a class
- To find H, think like the winning player: 1 for $h \leq H$, 2 for $h > H$
- Example: $f_\theta(x) = \operatorname{sign}(x_1^2 + x_2^2 - \theta)$
    - We can place one point and ”shatter” it
    - We can prevent shattering any two points: make the distant one blue
    - $H = 1$

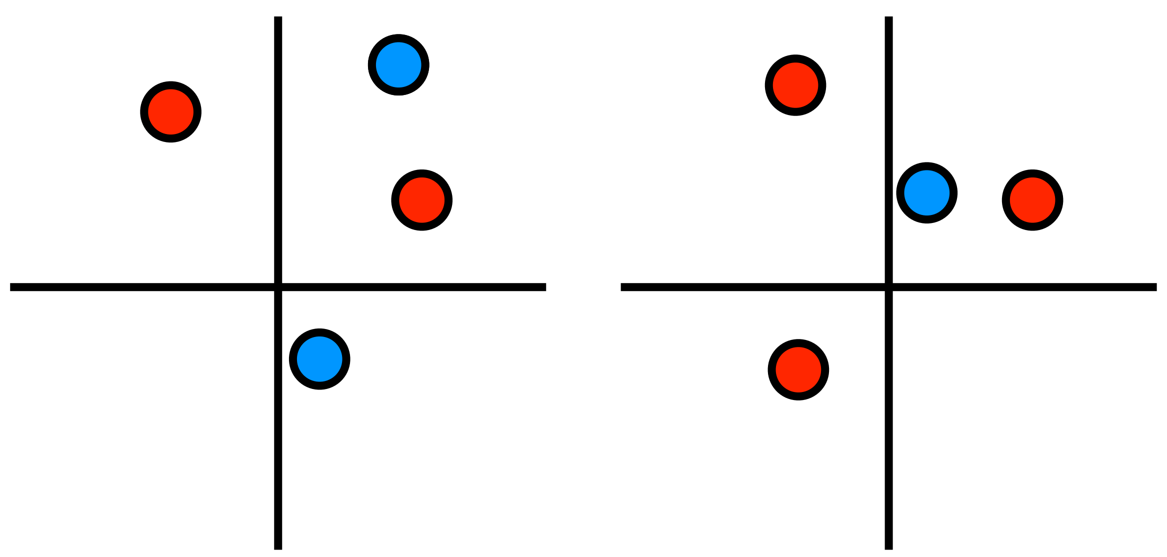


VC dimension: example (2)

- Example: $f_\theta(x) = \text{sign}(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$
  - We can place 3 points and shatter them



  - We can prevent shattering any 4 points:
    - If they form a convex shape, alternate labels
    - Otherwise, label differently the point in the triangle



  - $H = 3$
- Linear classifiers (perceptrons) of d features have VC-dim $d + 1$
  - But VC-dim is generally not #parameters

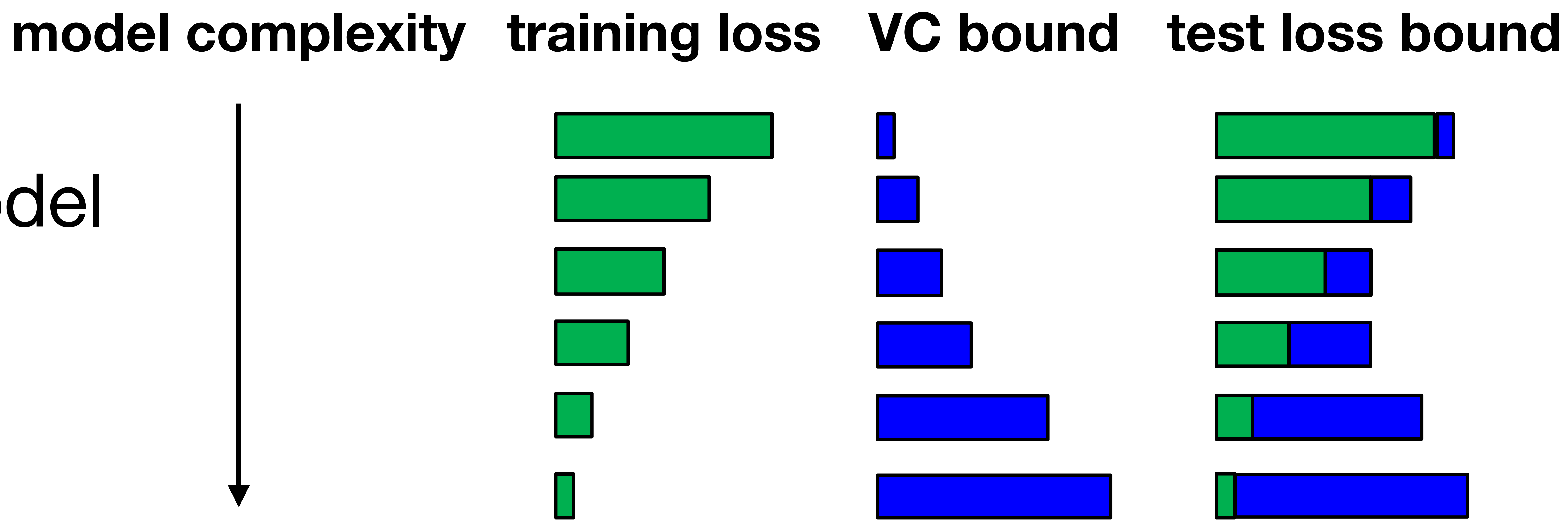
VC Generalization bound

- VC-dim of a model class can be used to bound generalization loss:
    - With probability at least $1 - \eta$, we will get a "good" dataset, for which
- $$\underbrace{\text{test loss} - \text{training loss}}_{\text{generalization loss}} \leq \sqrt{\frac{H \log(2m/H) + H - \log(\eta/4)}{m}}$$
- We need larger training size m:
    - The better generalization we need
    - The more complex (higher VC-dim) our model class
    - The more likely we want to get a good training sample

Model selection with VC-dim

- Using validation / cross-validation:
  - Estimate loss on held out set
  - Use validation loss to select model

- Using VC dimension:
  - Use generalization bound to select model
  - Structural Risk Minimization (SRM)
  - Bound not tight, much too conservative



Today's lecture

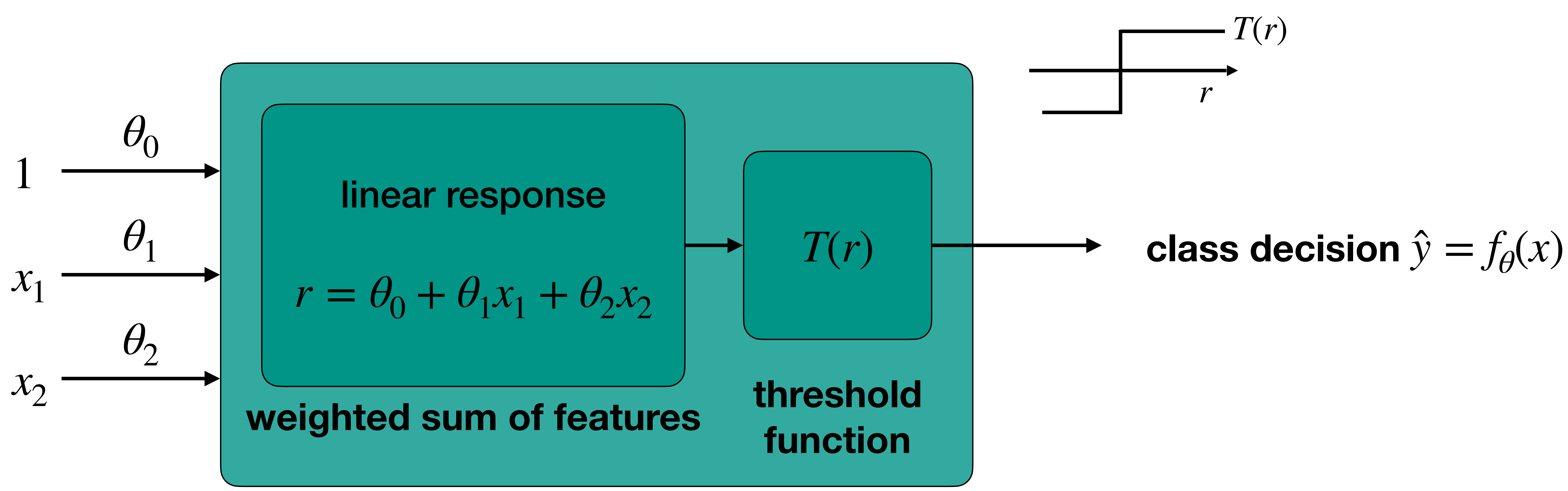
Multi-class classifiers

VC dimension

Multilayer perceptrons

Linear classifiers

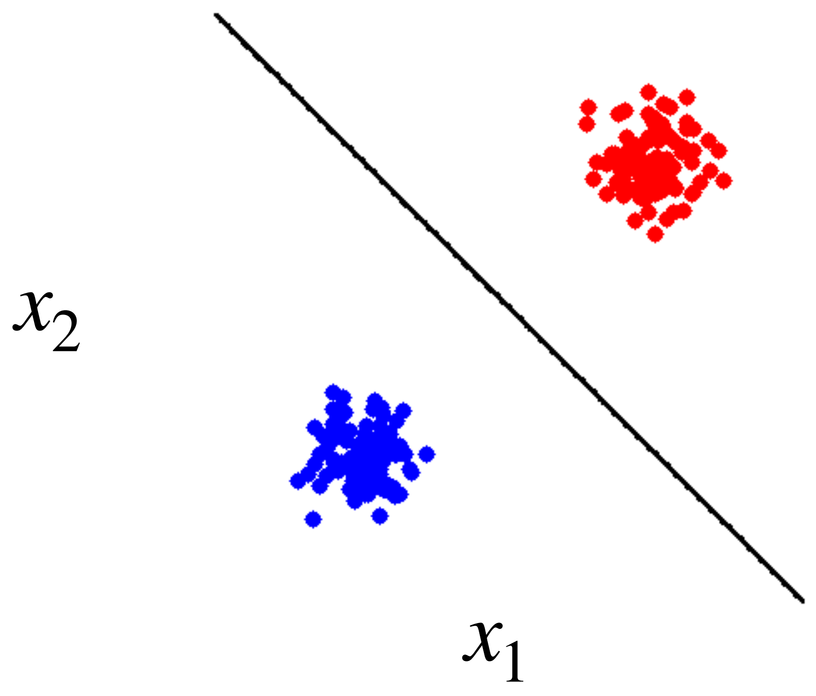
- Perceptron = use hyperplane to partition feature space → classes
  - Soft classifiers (logistic) = sensitive to margin from decision boundary



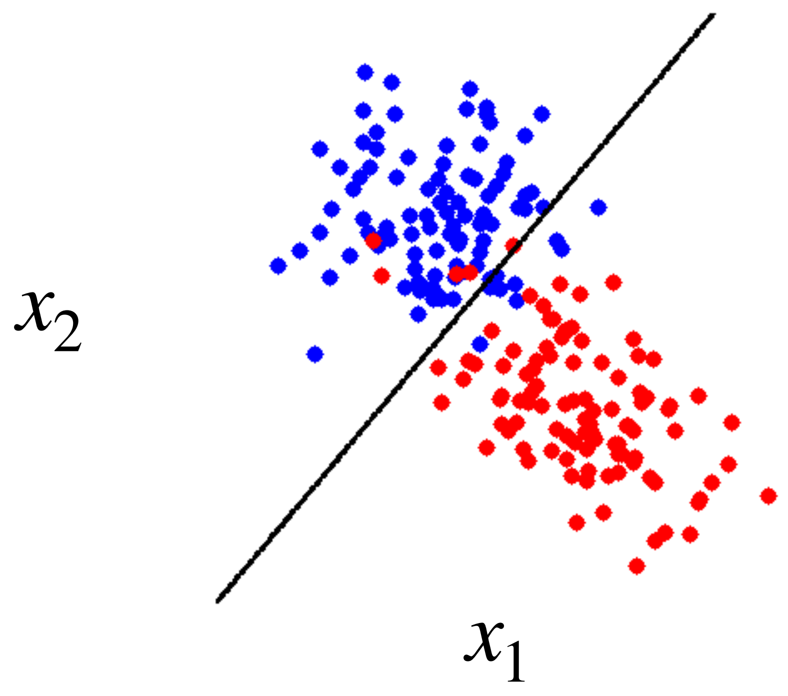
Adding features

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

Linearly separable data



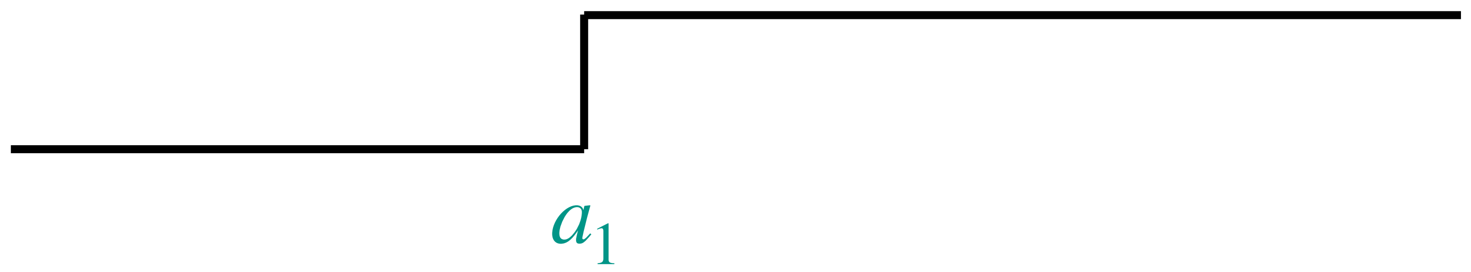
Linearly non-separable data



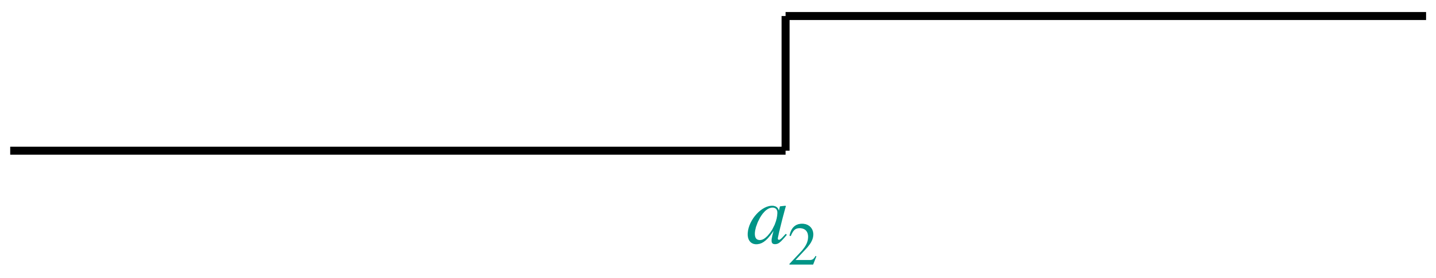
Combining step functions

- Combinations of step functions allow more complex decision boundaries

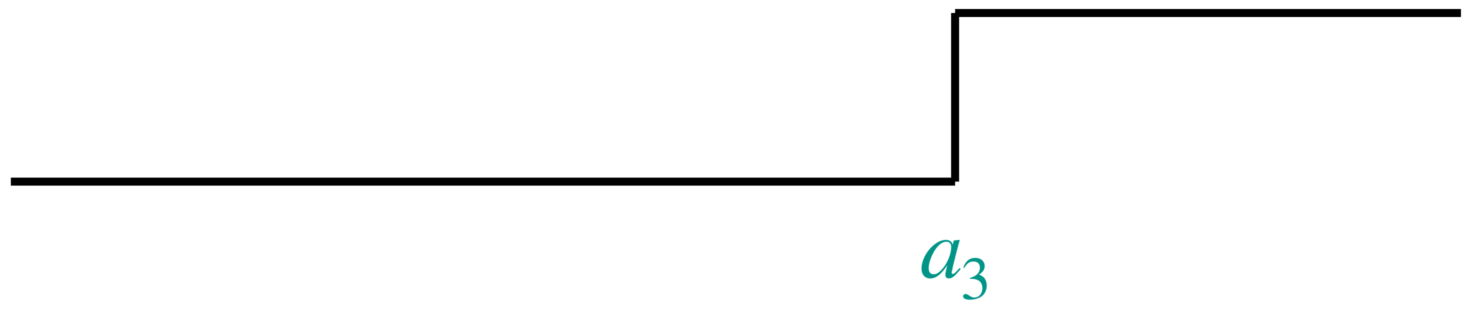
$F_1(x) = T(x - a_1)$



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



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



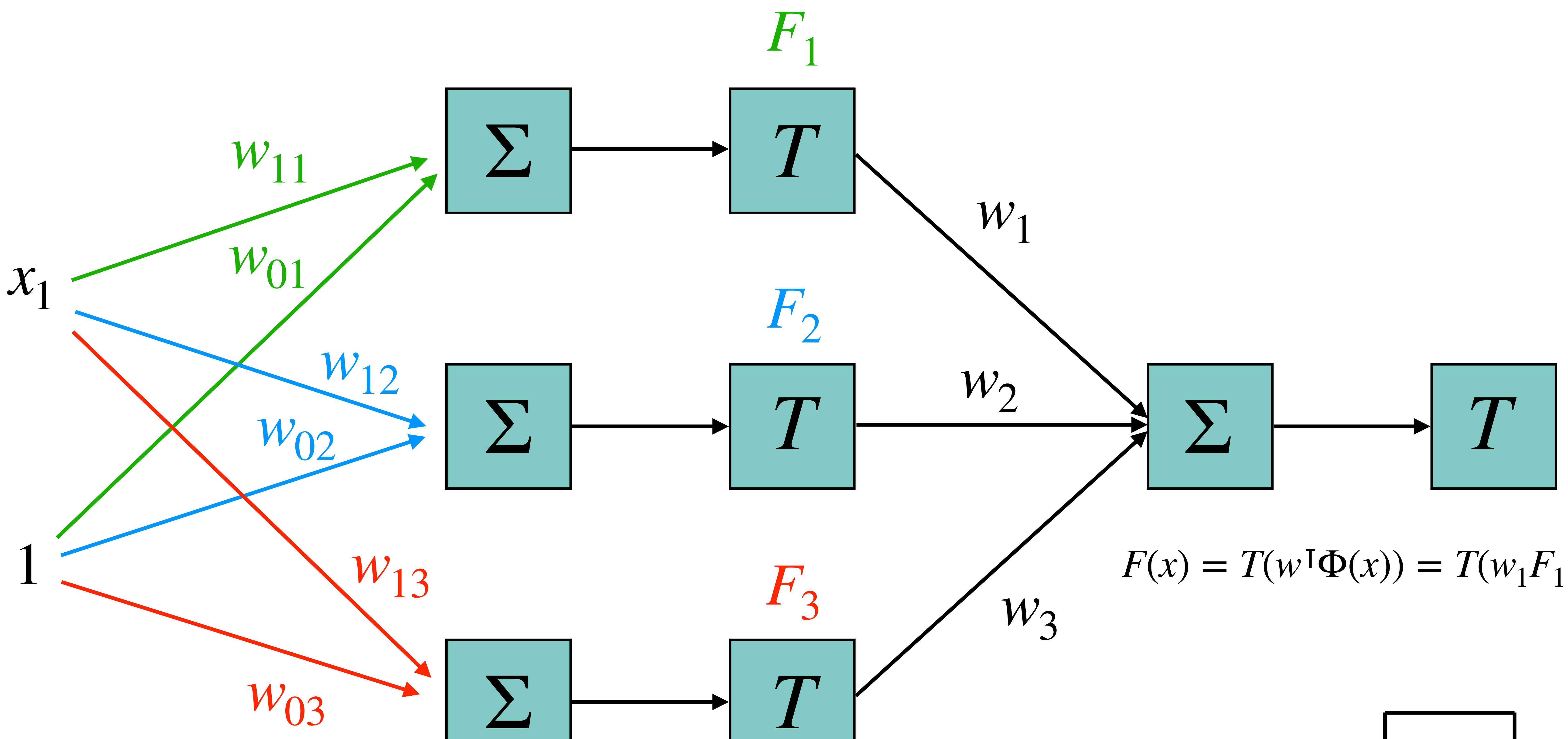
$\Phi(x) = \begin{bmatrix} F_1(x) & F_2(x) & F_3(x) \end{bmatrix}$

is piecewise constant

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

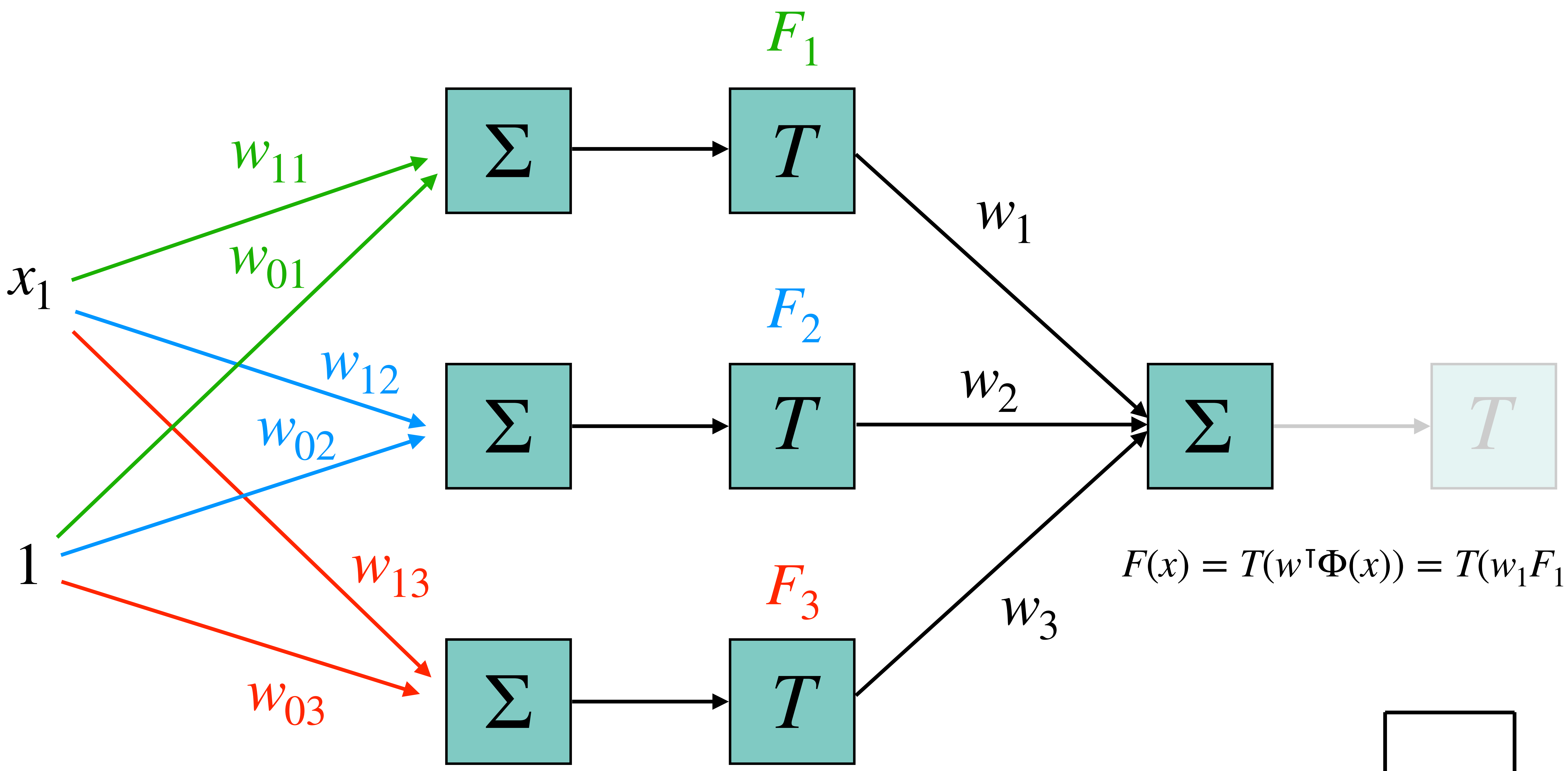
- Need to learn:
  - Thresholds a_1, a_2, a_3
  - Weights w_1, w_2, w_3, w_4

Multi-Layer Perceptron (MLP)

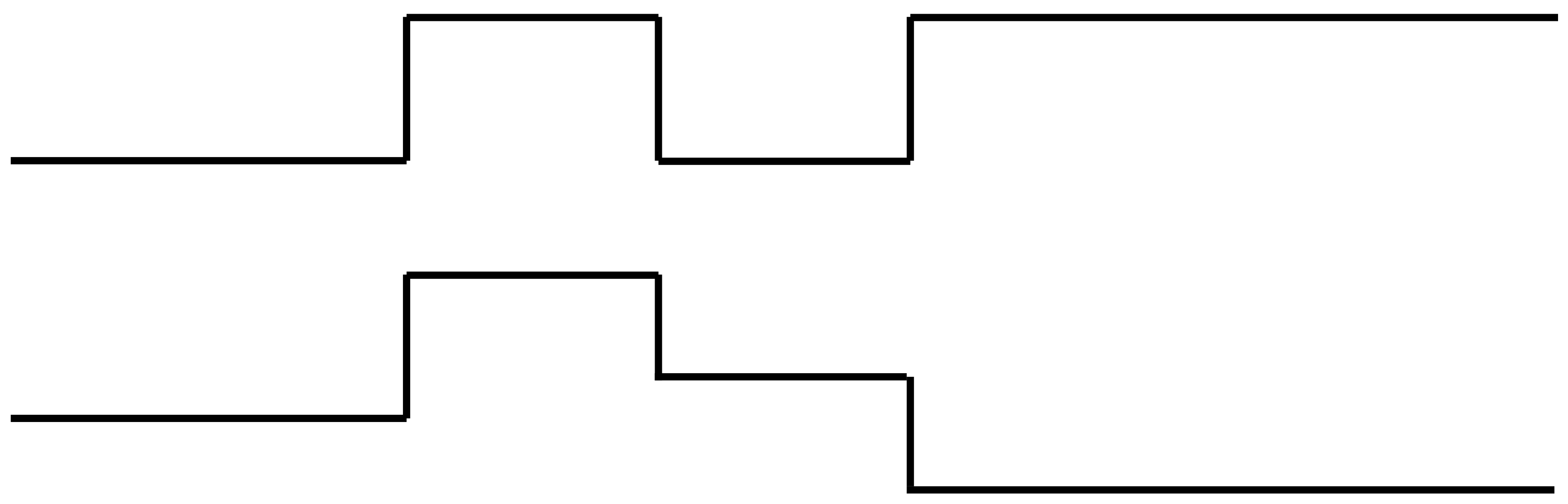


$$F(x) = T(w^\intercal \Phi(x)) = T(w_1 F_1(x) + w_2 F_2(x) + w_3 F_3(x) + w_4)$$

Multi-Layer Perceptron (MLP)



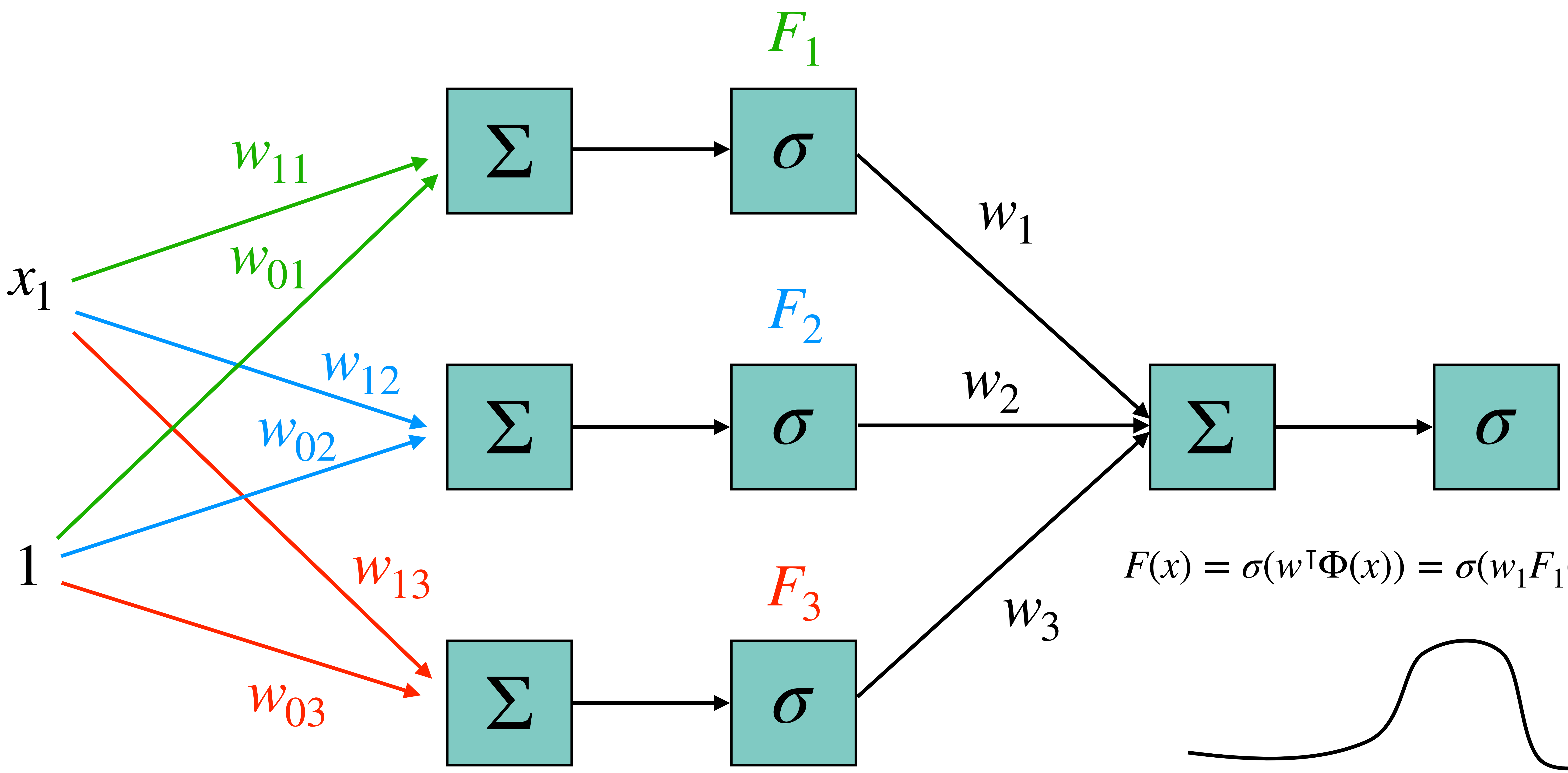
$F(x) = T(w^\intercal \Phi(x)) = T(w_1 F_1(x) + w_2 F_2(x) + w_3 F_3(x) + w_4)$



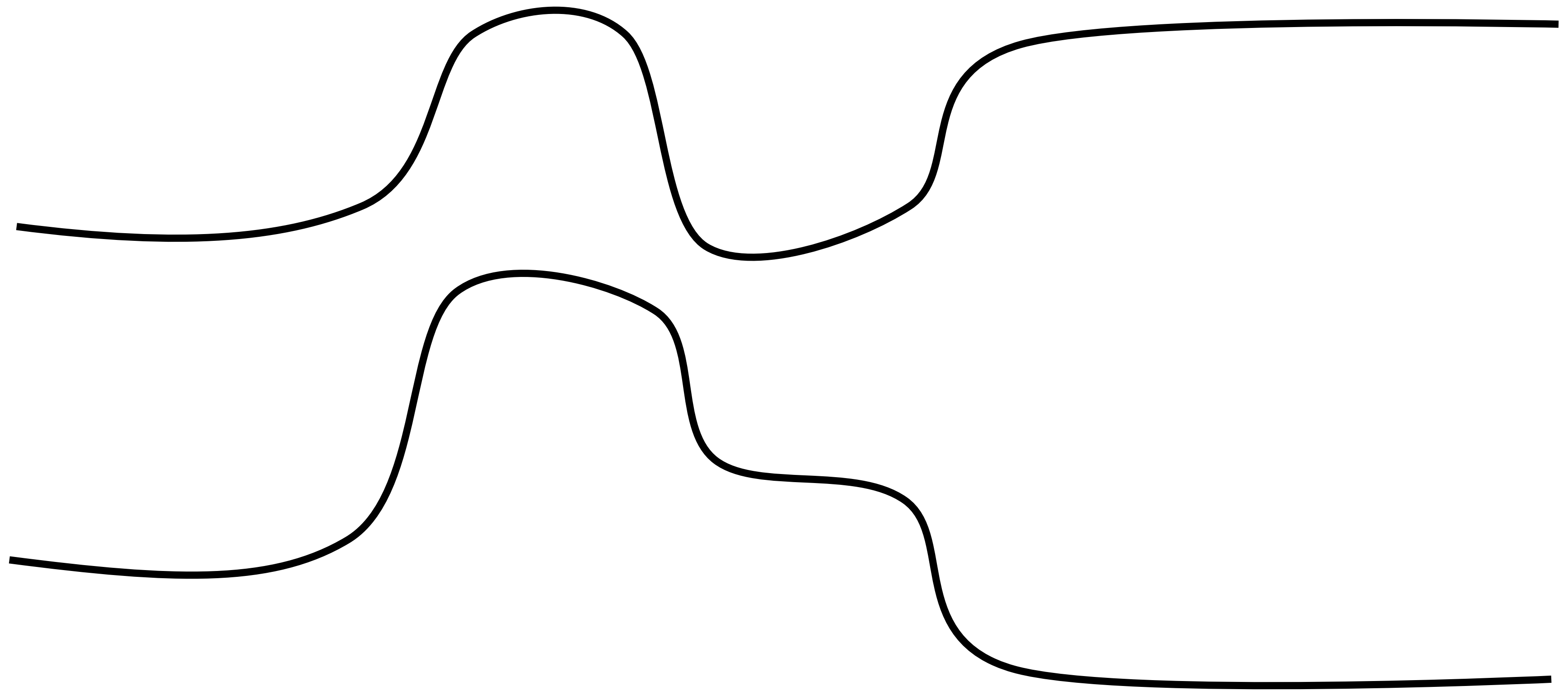
regression

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

Multi-Layer Perceptron (MLP)



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

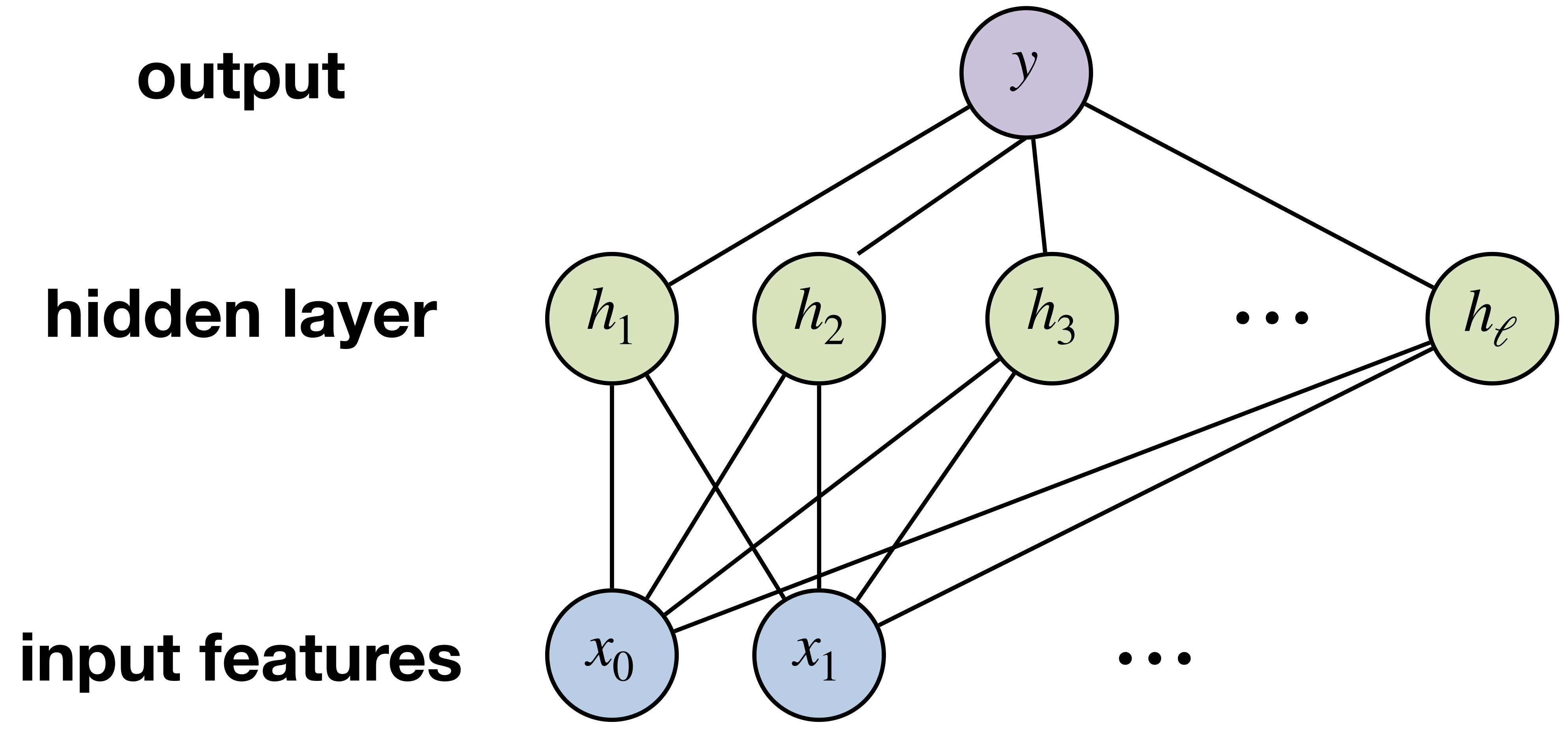


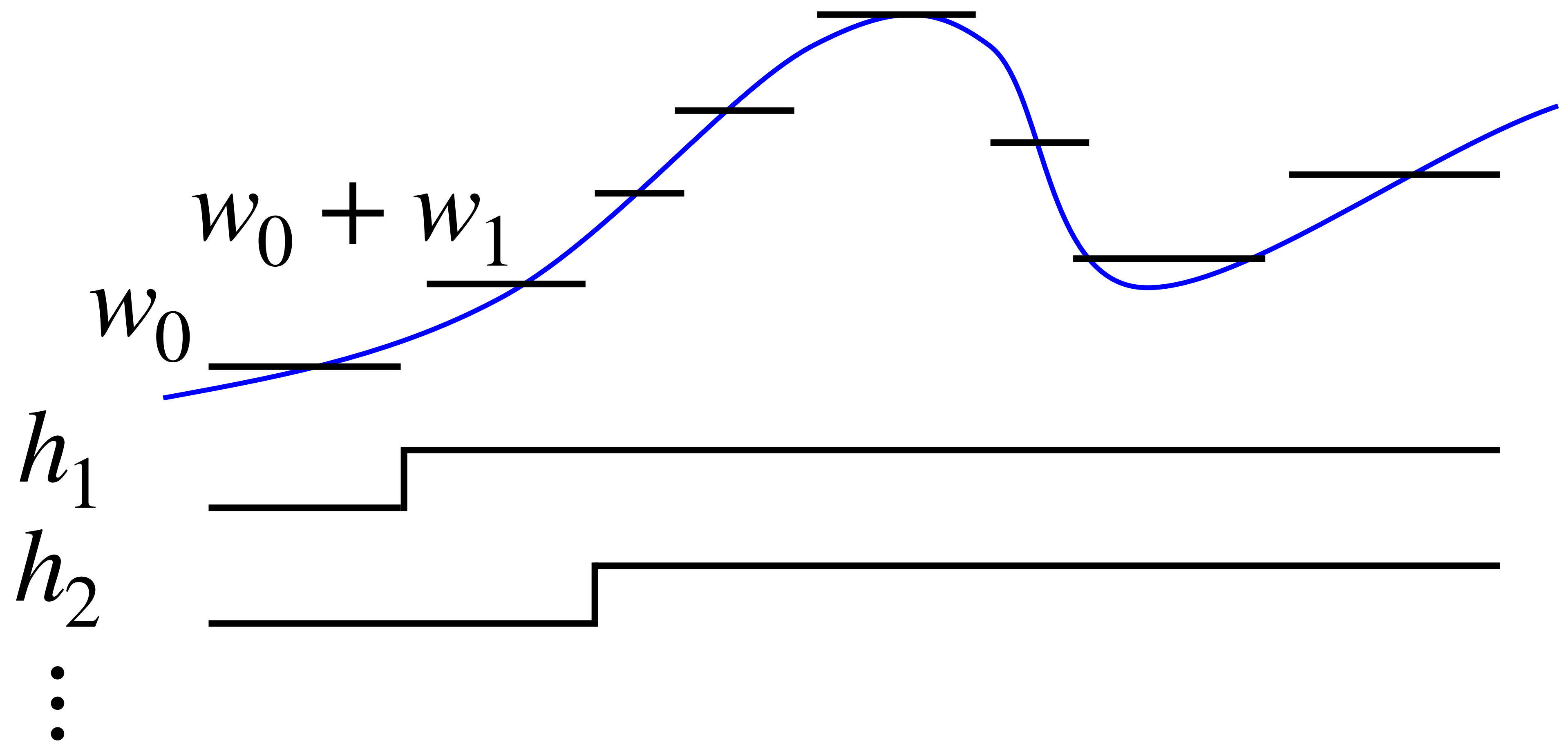
regression

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

MLPs: properties

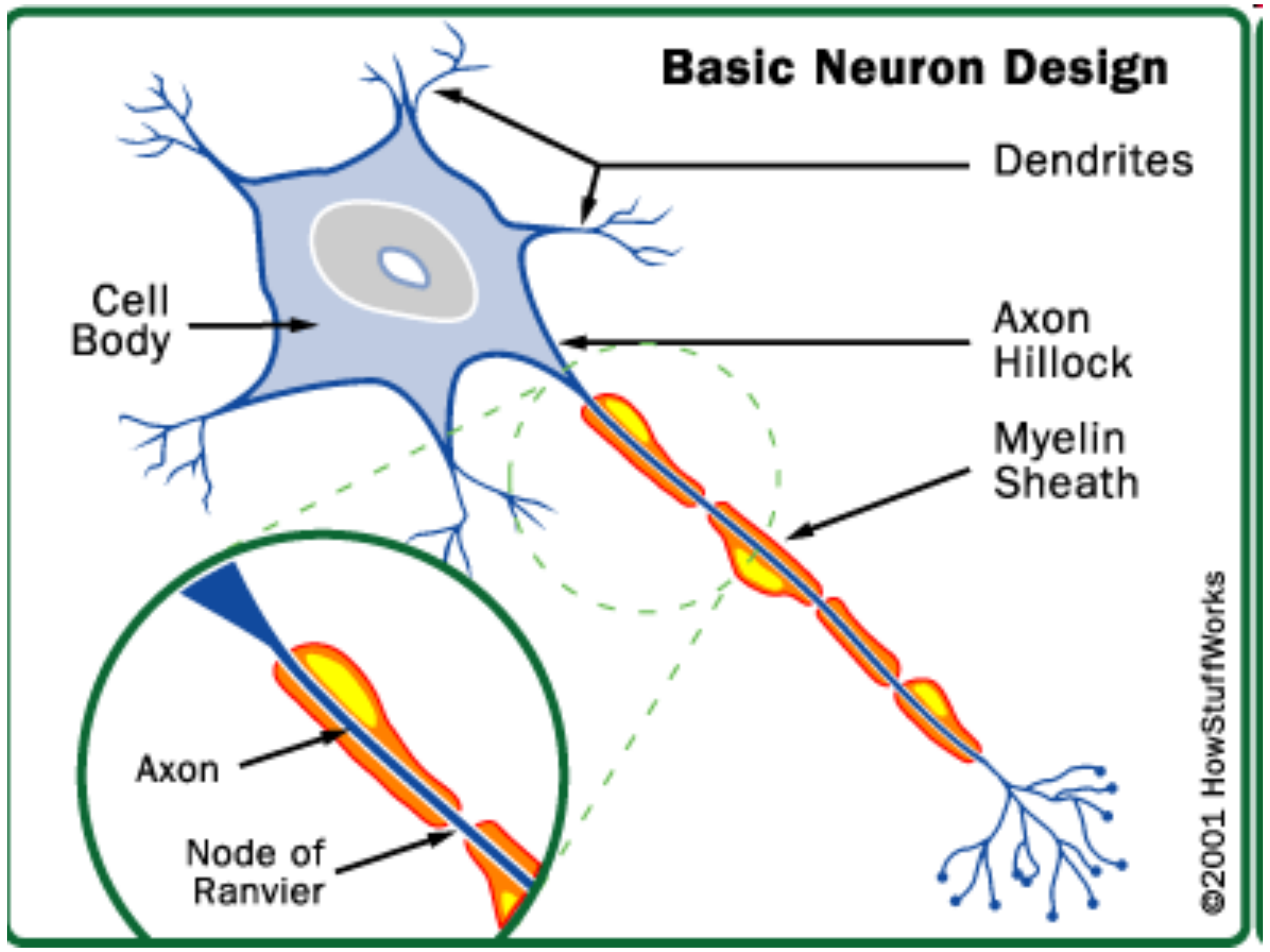
- Simple building blocks
  - Each unit is a perceptron: linear response → non-linear activation
- MLPs are universal approximators:
  - Can approximate any function arbitrarily well, with enough units





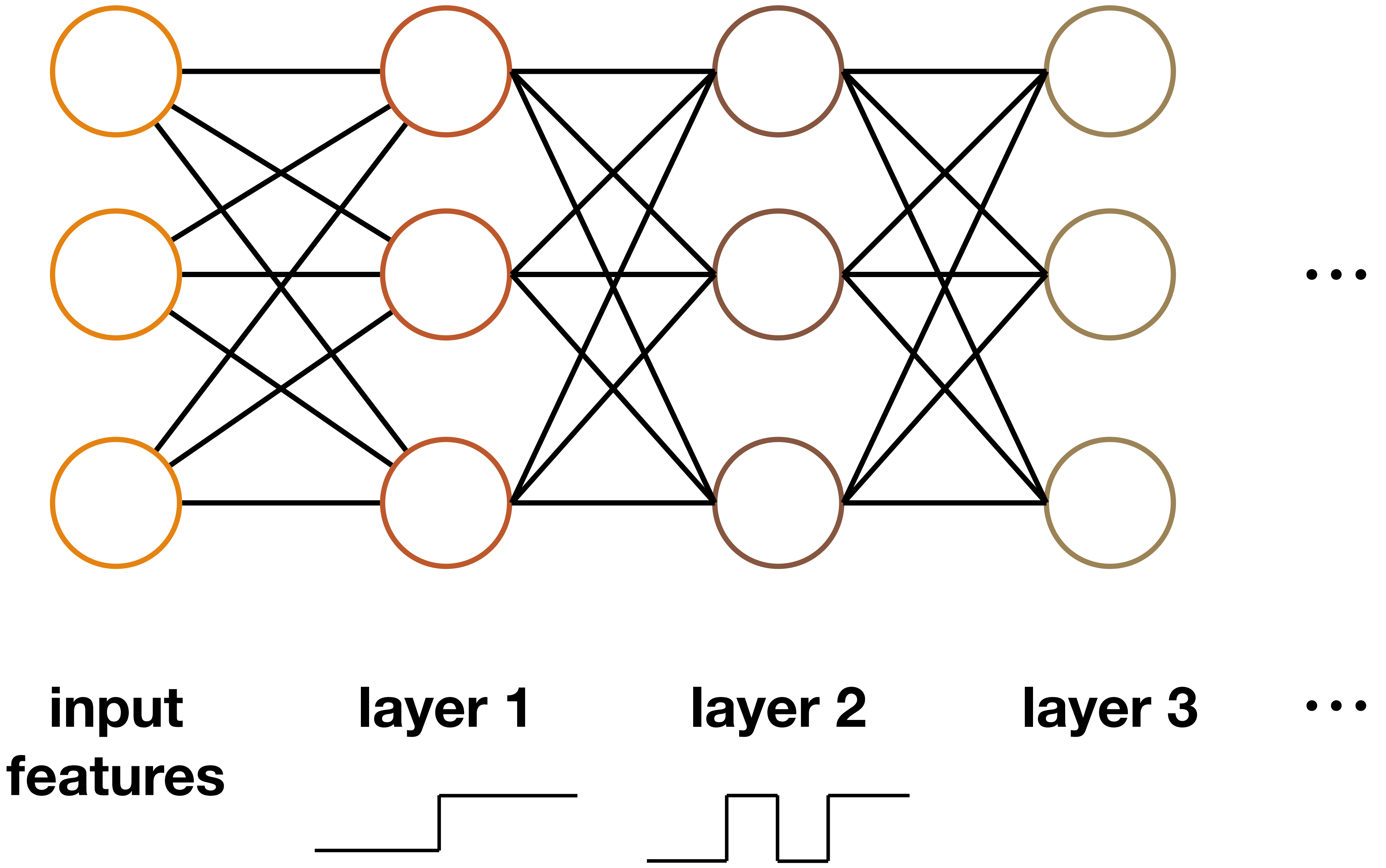
“Neural” Networks

- 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



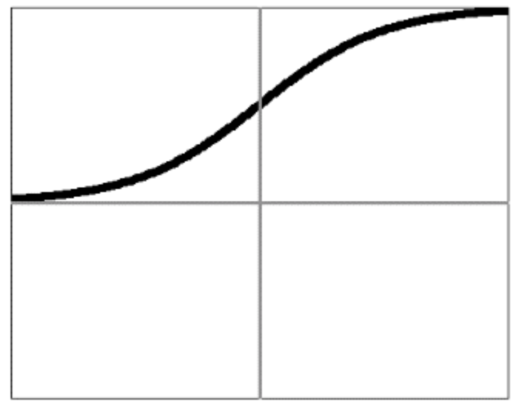

```
r1 = w[0].T @ x + b[0]    # linear response
h1 = sig(r1)              # activation function

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

                          # ...
```

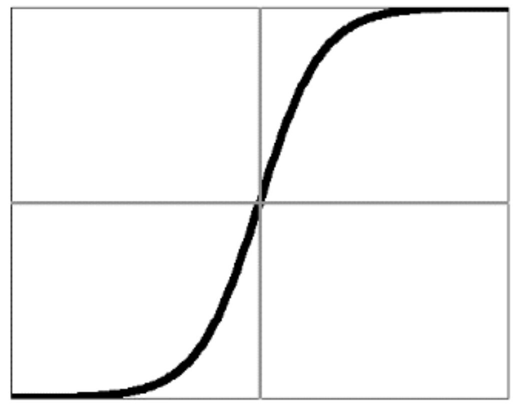
Activation functions

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



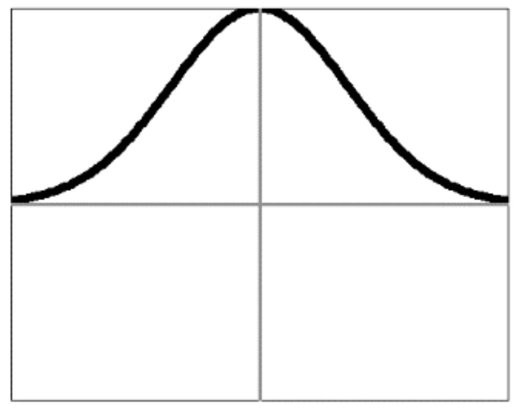
$\sigma'(z) = \sigma(z)(1 - \sigma(z))$

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



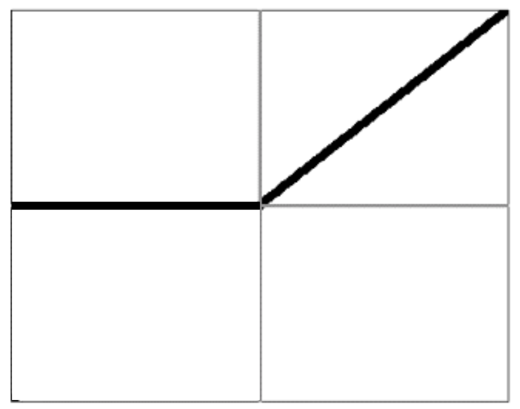
$\sigma'(z) = 1 - \sigma^2(z)$

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



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

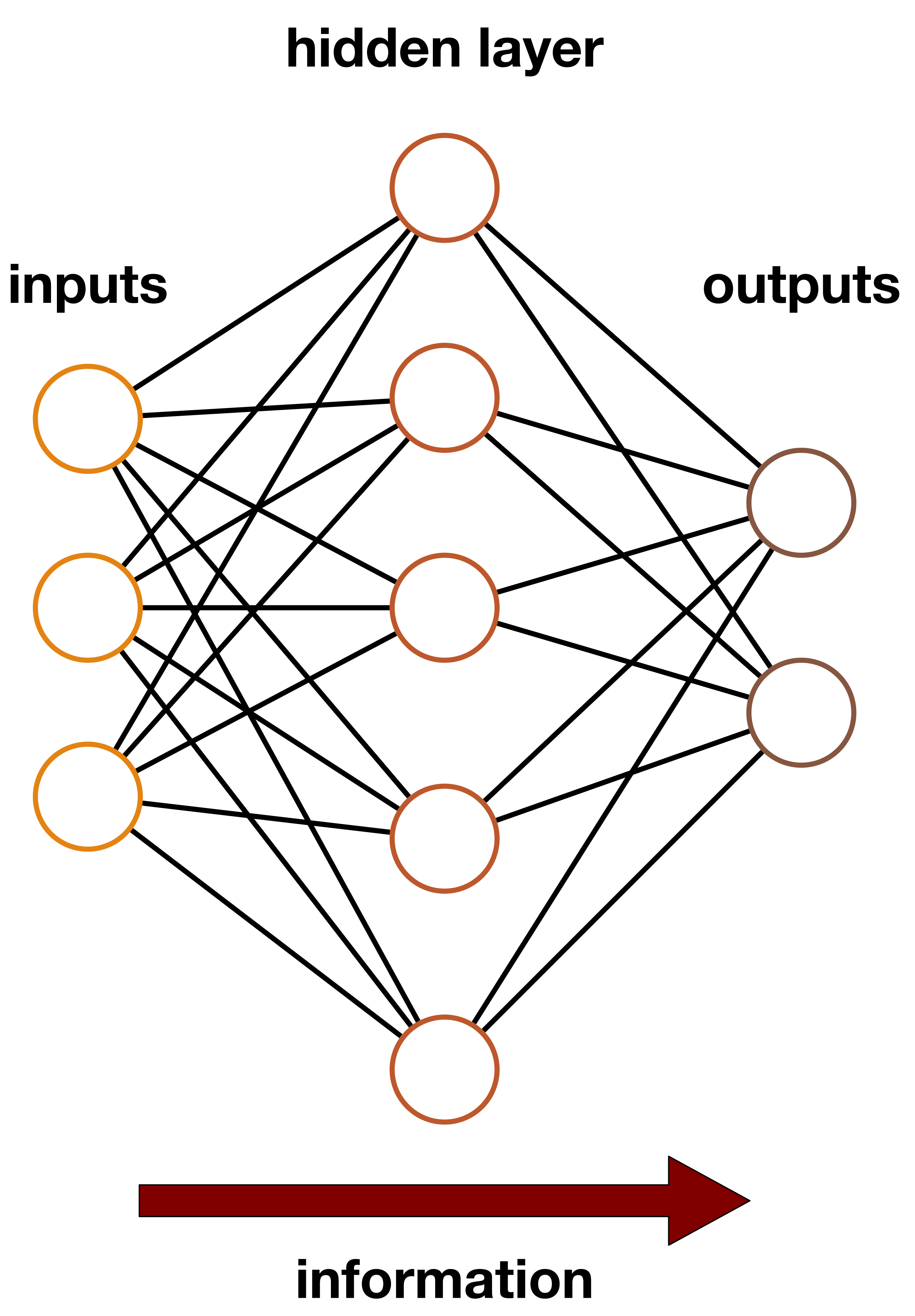
- Rectified linear (ReLU) $\quad \sigma(z) = \max(0,z)$



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

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 $\implies$ 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): $\hat{y}_c = \dfrac{\exp(h_c)}{\sum_{\bar{c}} \exp(h_{\bar{c}})}$



Logistics

assignments

- Assignment 3 due next Tuesday, Nov 2

midterm

- Midterm exam on Nov 4, 11am–12:20 in **SH 128**
- If you're eligible to be remote — let us know by Oct 28
- If you're eligible for more time — let us know by Oct 28
- Review during lecture this Thursday