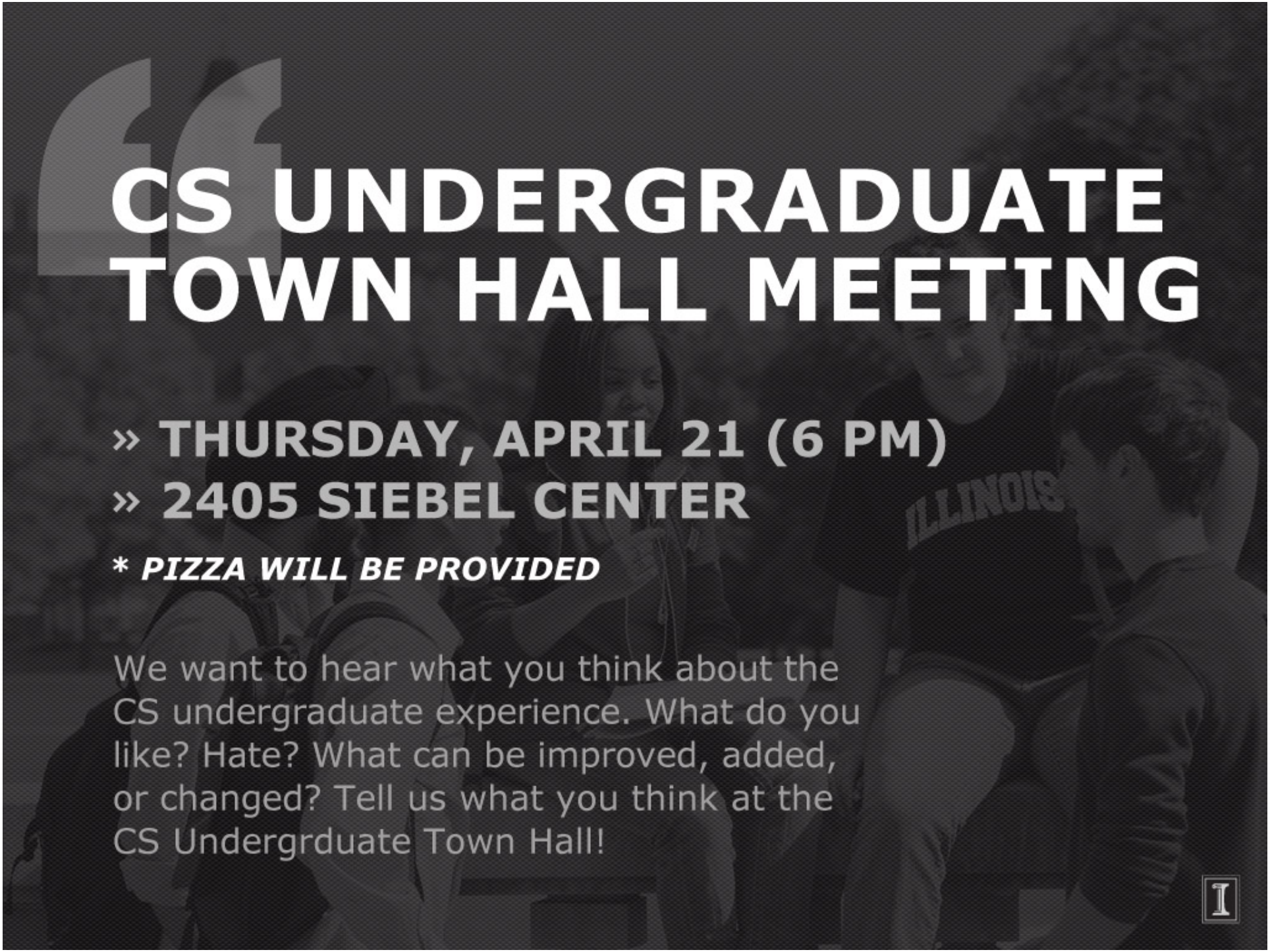
CS440/ECE448: Intro to Artificial Intelligence

Lecture 26: A bit about learning theory

Prof. Julia Hockenmaier
juliahmr@illinois.edu

http://cs.illinois.edu/fa11/cs440


CS UNDERGRADUATE TOWN HALL MEETING
» THURSDAY, APRIL 21 (6 PM)
» 2405 SIEBEL CENTER
* PIZZA WILL BE PROVIDED
We want to hear what you think about the CS undergraduate experience. What do you like? Hate? What can be improved, added, or changed? Tell us what you think at the CS Undergrduate Town Hall!

Monday, May 2, 4 pm in 1404 Siebel Center
Natural Language Applications Across Genres: From News to Novels
Prof. Kathleen McKeown, Columbia University

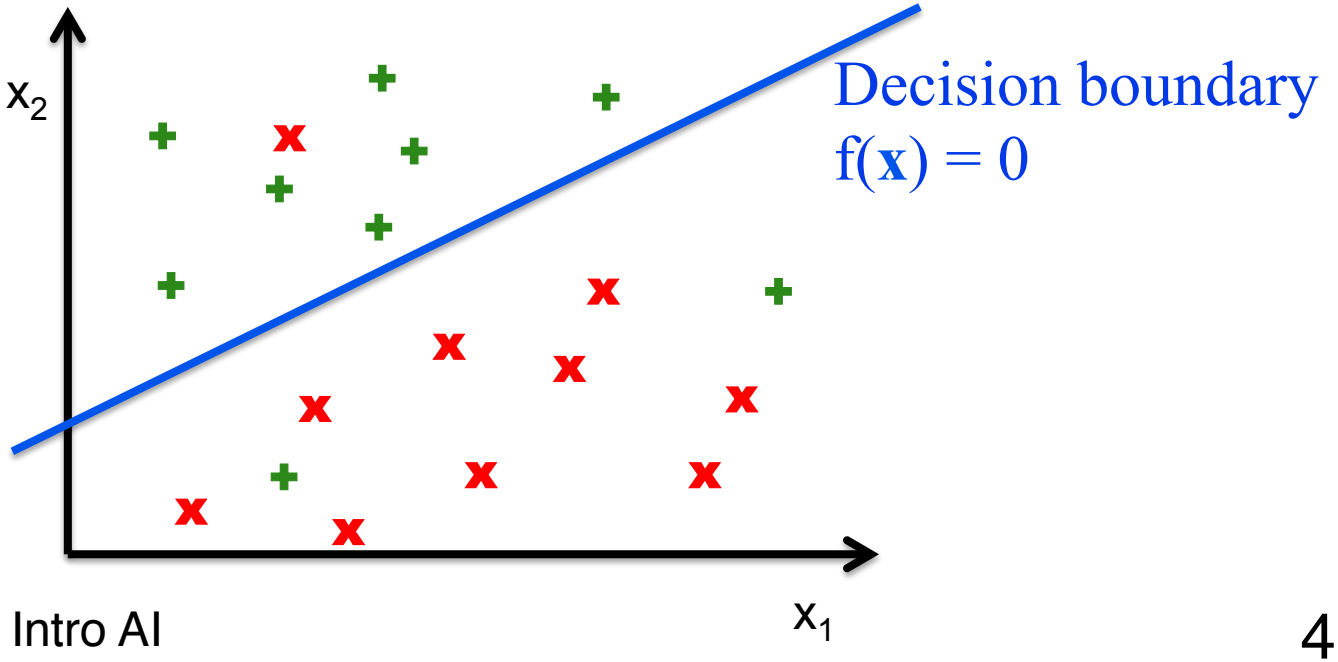
Monday, May 2, 6 pm in 2405 Siebel Center
Attending Graduate School: A panel discussion

Tuesday, May 3, 10 am 2405 Siebel Center
Machine Learning - Modern Times
Dr Corinna Cortes (Head of Google Research, NY)

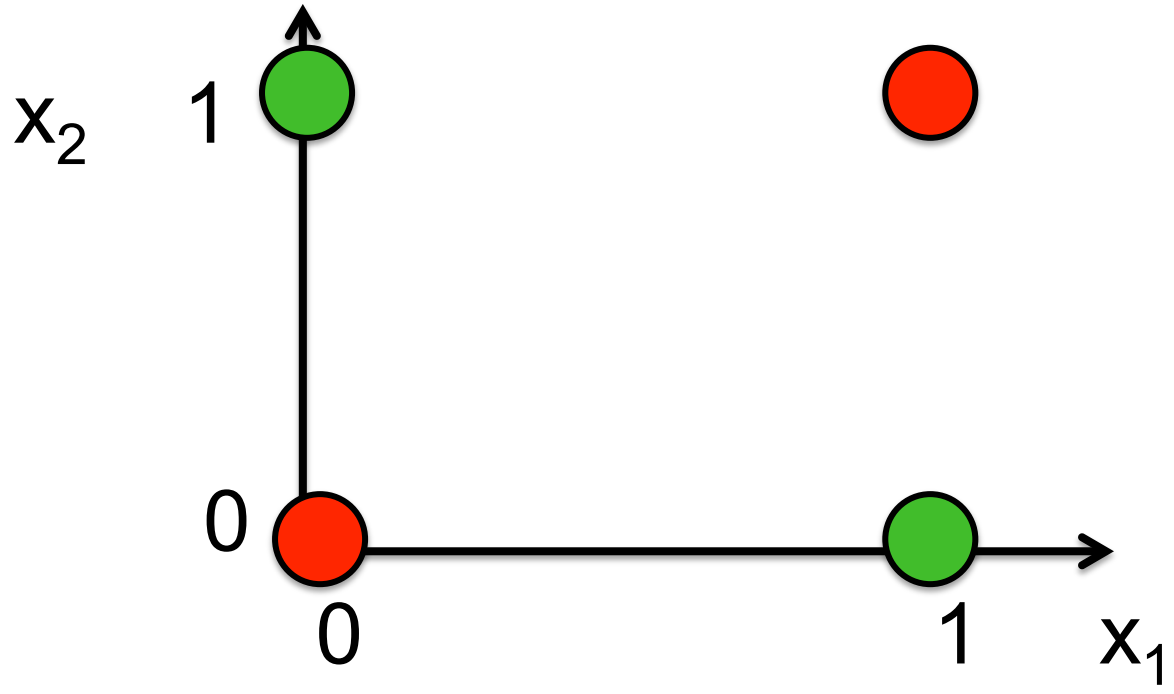
Binary classification: training

Input: $\{(\mathbf{x}^i, y^i)\}$ with $(x_{1\ldots} x_d) \in R^d$ $y^i \in \{+1, -1\}$

Task: Find weights $\mathbf{w} = (w_0\ w_{1\ldots} w_d) \in R^{d+1}$ that define $f(\mathbf{x}) = \mathbf{wx}$



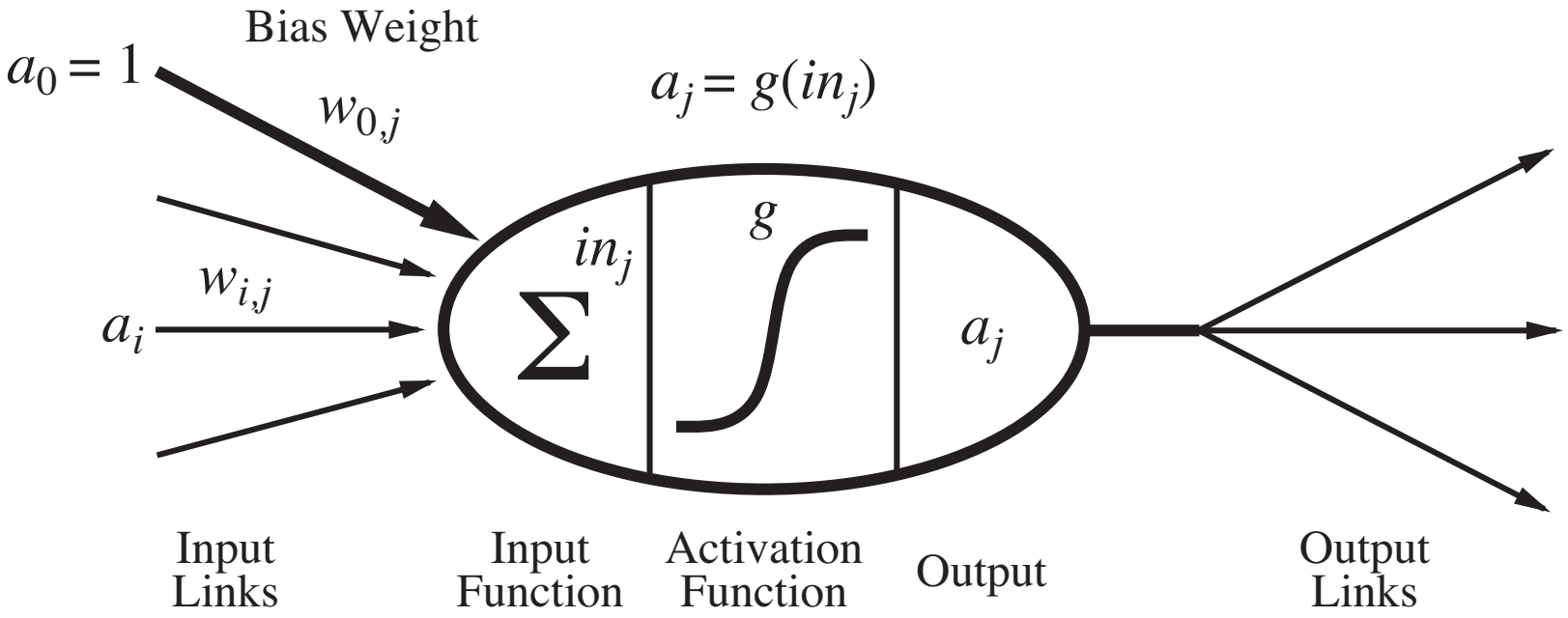
Boolean XOR



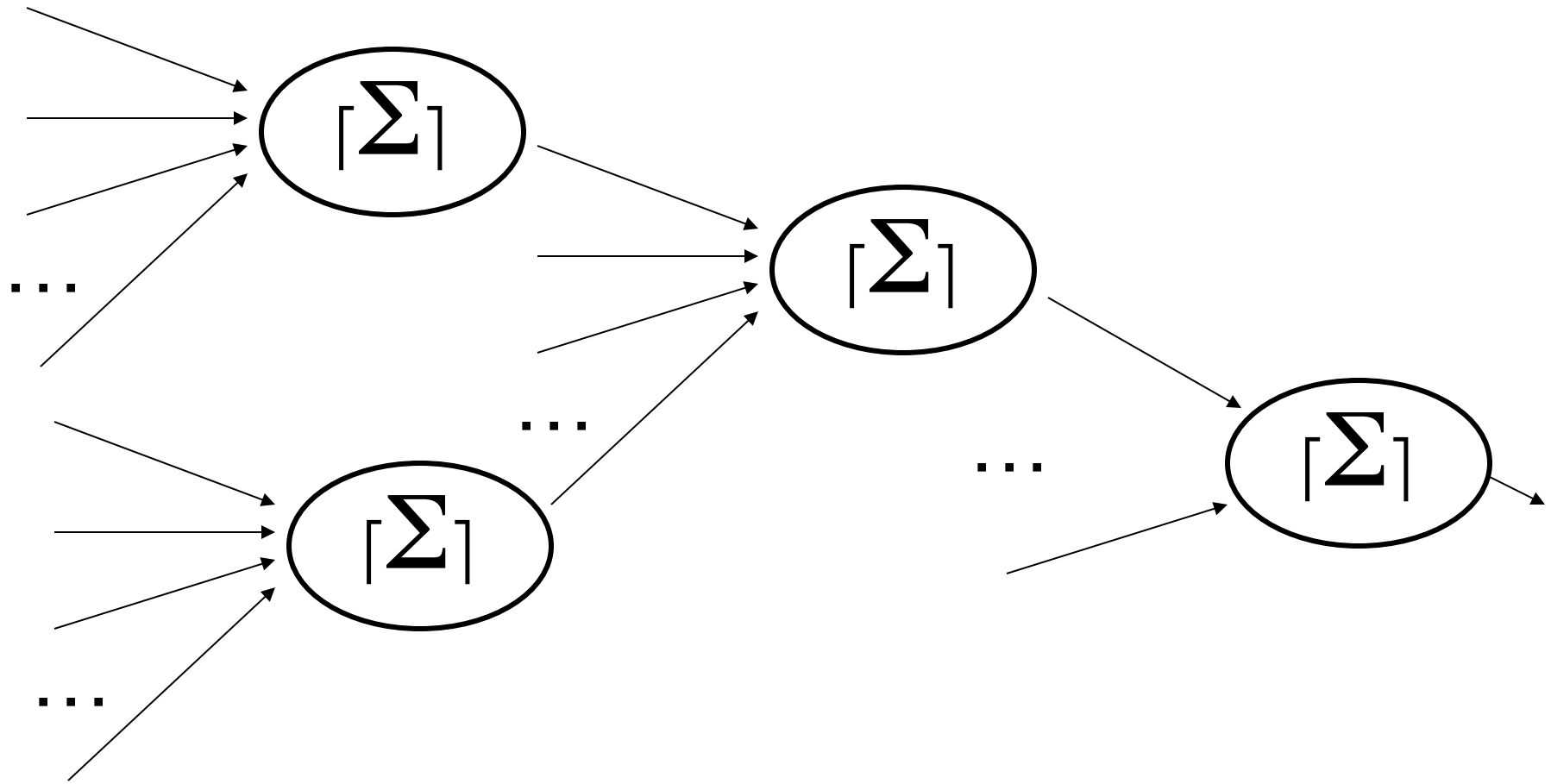
XOR is not linearly separable

From perceptrons to neural networks

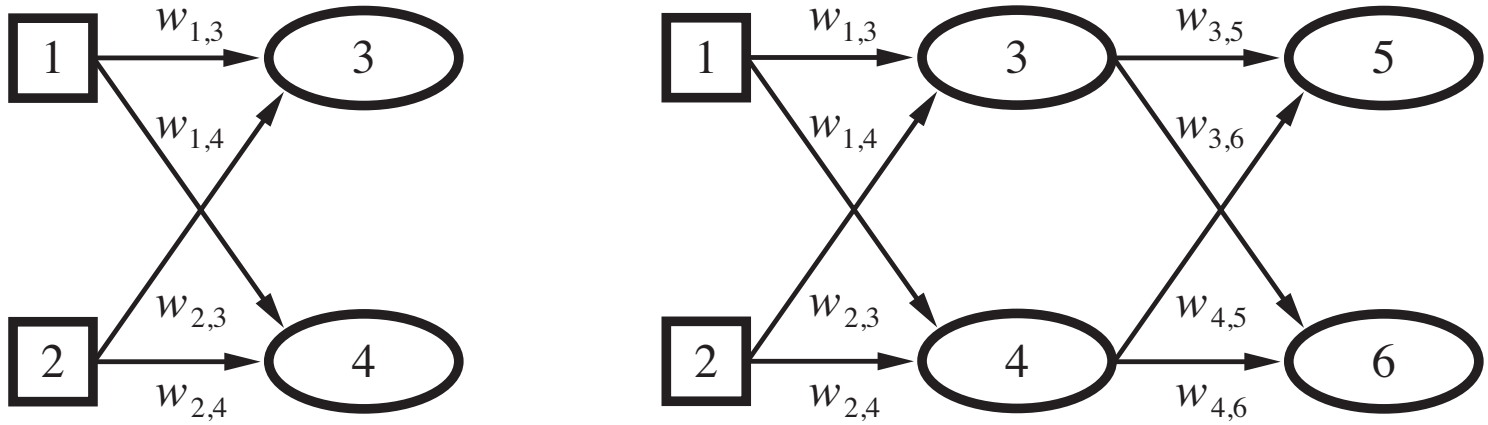
We can think of a single perceptron as one neuron



Artificial Neural Networks: Multi-layer perceptrons



From perceptrons to neural nets



A neural net consists of nodes connected by directed links.

Each node has an **activation a_i**

Links a_{ij} propagate the activation a_i from i to j.

Each link has a **weight w_{ij}** that determines the strength and sign of the connection

From perceptrons to neural networks

Each unit computes a weighted sum of its inputs: $in_j = \sum_j w_{ij}\, a_{ij}$

and applies an **activation function** to this input

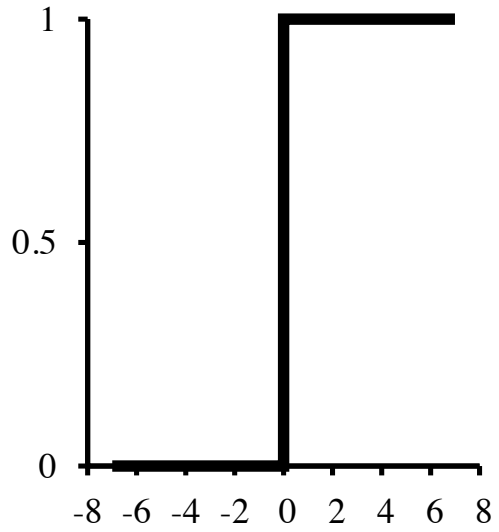
$$a_j = g(in_j) = g(\sum_j w_{ij}\, a_{ij})$$

activation function: linear threshold or sigmoid threshold

The perceptron threshold

The perceptron uses a hard threshold function:
$h_{\mathbf{w}}(\mathbf{x})$: if $f(\mathbf{x}) = \mathbf{wx} > 0$ return $y = 1$, else return $y = 0$

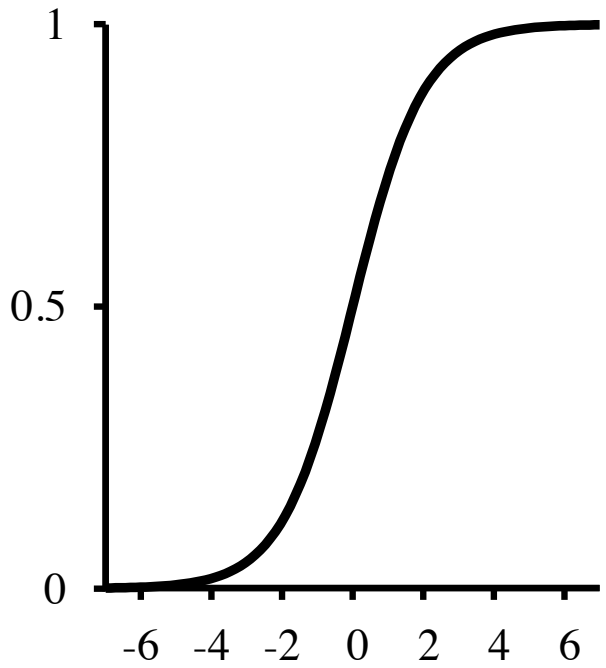
This is a non-differentiable function, so we cannot use gradient descent.



The sigmoid threshold

The logistic (sigmoid) function is differentiable. This is necessary to train multilayer networks (backpropagation)

$$h_{\mathbf{w}}(x) = \frac{1}{1 + e^{-\mathbf{wx}}}$$



Computational learning theory

Computational Learning Theory
How Much Data is Enough?

- Training set is evidence for which h∈H is
  - Correct: [Simple, Proper, Realizable??] learning
  - Best: *Agnostic* learning
- Remember: training = labeled independent sampling from an underlying population
- Suppose we perform well on the training set
- How well will perform on the underlying population?
- This is the *test accuracy* or *utility* of a concept (not how well it classifies the training set)

What Makes a Learning Problem Hard?

How do we measure “hard”?

- Computation time?
- Space complexity?
- Number of training examples required?

Hard learning problems require more training examples

The hardest learning problems require the entire example space to be labeled

PAC learning
(PAC = probably approximately correct)

Assumptions:

– The hypothesis space H is finite

– One hypothesis $h \in$ H generates the labels

– Training data is sampled i.i.d
according to an unknown distribution D

Question:

How many training examples do we need
to see to find *h*?

PAC learning
(PAC = probably approximately correct)

Original question: How many training examples do we need to find the correct hypothesis *h* ?

A simpler task: Find an approximately correct hypothesis *h'* that makes no more than ε errors.

A simpler question: How many training examples do we need to find with high probability an approximately correct hypothesis *h'* ?

PAC learning
(PAC = probably approximately correct)

How many examples do we need to find with high probability an approximately correct hypothesis *h'* ?

How many examples do we need to find with probability $p > 1 - \delta$ a hypothesis *h'* that has accuracy $> 1 - \varepsilon$?

$$N \geq \frac{1}{\varepsilon}\left(\ln\frac{1}{\delta} + \ln|H|\right)$$

PAC learning: intuition

A hypothesis h is *bad* if its true error $> \varepsilon$

$\forall x \in \mathbf{X}$: $\Pr_D(h(x) \neq h^*(x)) > \varepsilon$

A hypothesis h *looks good* if it is correct on our training set **S**

$\forall s \in \mathbf{S}$: $h(s) = h^*(s)$ $\quad |\mathbf{S}| = N$

We want the probability that a bad hypothesis looks good to be smaller than δ

PAC learning: intuition

We want the probability that a bad hypothesis looks good to be smaller than δ

Probability of a bad h getting one $x \sim \mathbf{X}_D$ correct:

$$P_D(h(x) = h^*(x)) \leq 1-\varepsilon$$

Probability of a bad h getting N $x \sim \mathbf{X}_D$ correct:

$$P_D(h(x) = h^*(x)) \leq (1-\varepsilon)^N$$

PAC learning: intuition

Probability that **one** bad h gets N $x \sim \mathbf{X}_D$ correct:
$P_D(h(x) = h^*(x)) \leq (1-\varepsilon)^N$

Probability that **any** of the $|H|$ h gets N $x \sim \mathbf{X}_D$ correct
$\leq |H|(1-\varepsilon)^N$
Exclusive union bound: $P(A \vee B) \leq Pr(A) + Pr(B)$

We want this to happen with probability $\leq \delta$
Set $|H|(1-\varepsilon)^N \leq \delta$, solve for N

PAC learning: intuition

It is sufficient that $|H|(1-\varepsilon)^N \leq \delta$
Or $\ln(|H|) + N \ln(1-\varepsilon) \leq \ln(\delta)$

Recall $e^{-y} > 1-y$ (for $y > 0$) so $\ln(1-\varepsilon) < -\varepsilon$

$N \geq (\ln \delta - \ln |H|)/ -\varepsilon$
$N \geq (1/\varepsilon) (-\ln \delta + \ln |H|)$
$N \geq (1/\varepsilon) (\ln (1/\delta) + \ln |H|)$
(very loose bound)

See text section 18.5

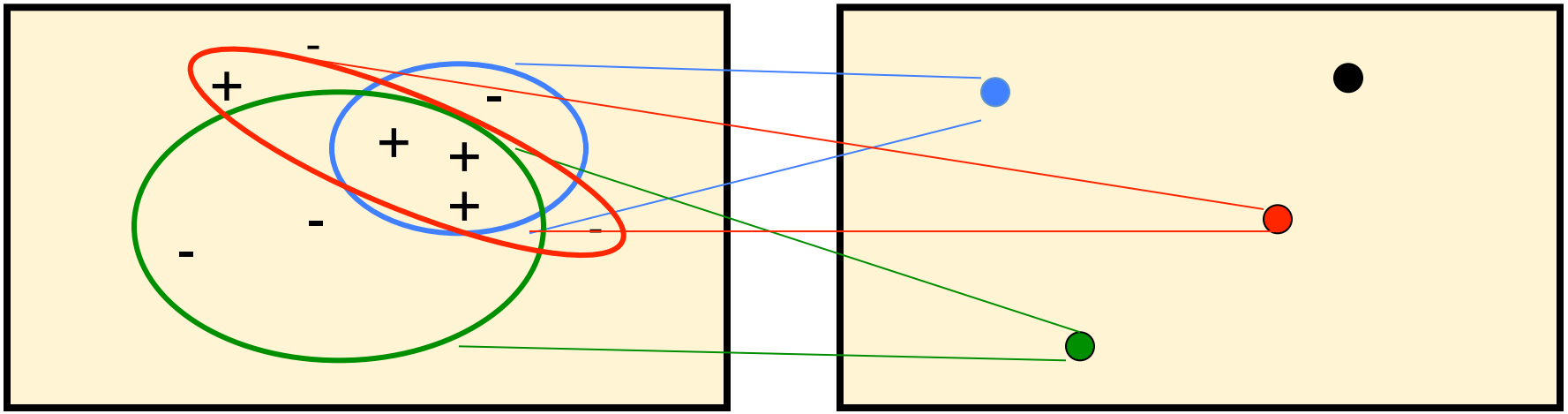
PAC-Learnability

Often the hypothesis space (or concept class) is syntactically parameterized

n-Conjuncts, k-DNF, k-CNF, m of n, ANN w/ k units,…

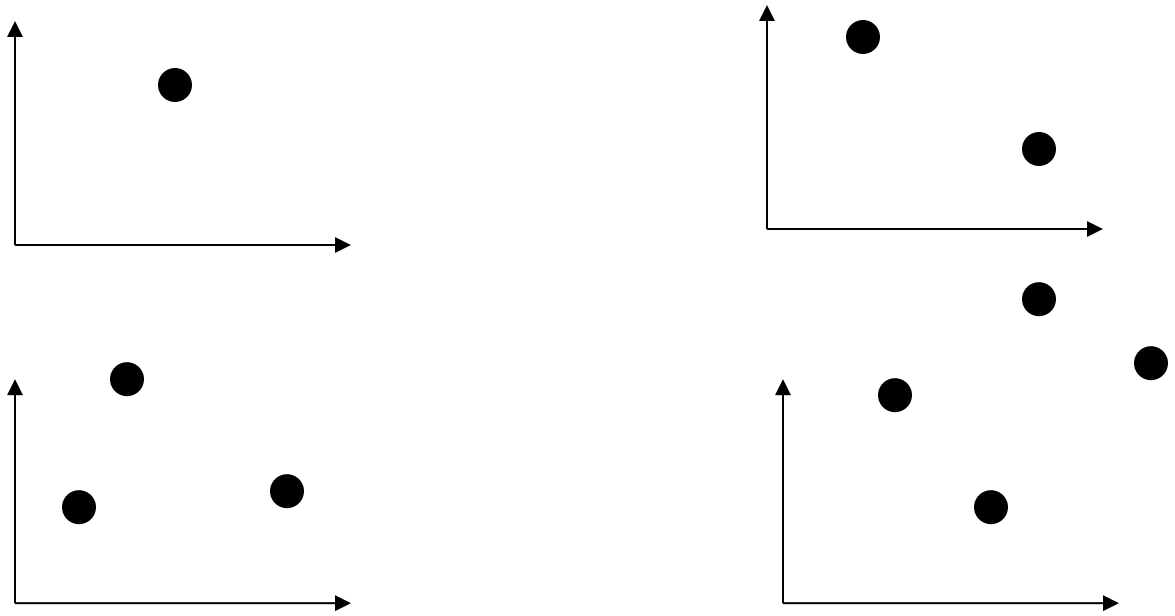
The concept class C is *PAC learnable* if there exists an algorithm that will return for any element c in C with probability $p > 1 - \delta$ an approximately correct hypothesis *h'* that has $\text{accuracy} > 1 - \varepsilon$

Hypotheses as partitioning functions h_i: X →{+,-}



Given a set of *N* labeled examples, is there a consistent hypothesis?
If we change the labels, is there still a consistent hypothesis?
What is the largest *N* for which the answer is always “yes”?
This is the **Vapnik-Chervonenkis dimension** of the hypothesis space **VC(H)**

VC Dimension of a 2D- Perceptron



Thus the VC dimension of a 2-d perceptron is 3

The largest set of points that can be labeled arbitrarily

Note that |H| is infinite, but expressiveness is quite low.

Capacity & VC Dimension

VC(H) is the cardinality of the largest set of examples *shattered* by H

An example set is shattered by a hypothesis set *iff* every label assignment of the examples is consistent with some element of H

VC is most common but there are other measures of *capacity*

VC Dimension

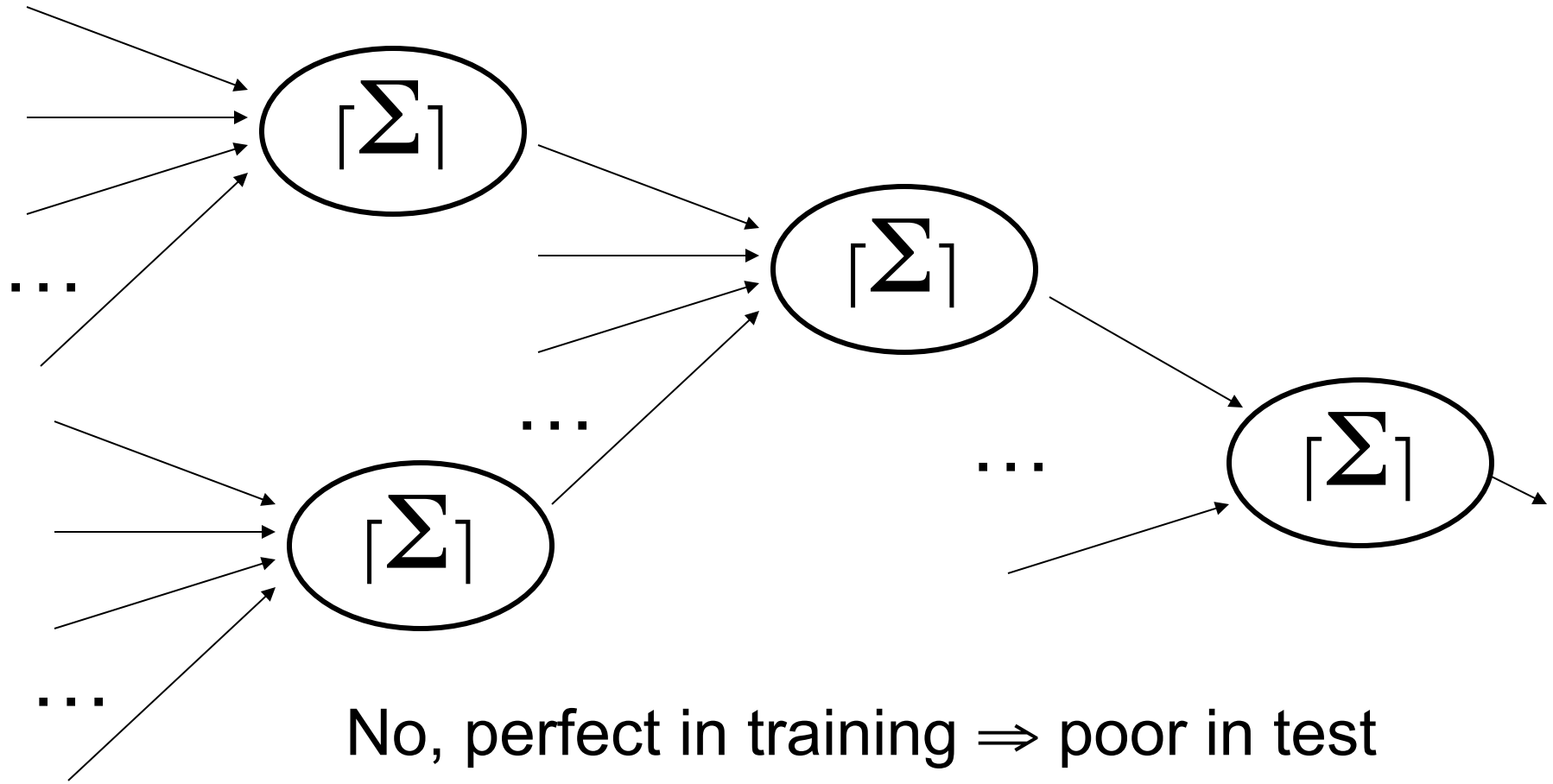
VC dimension can be challenging to prove, non-intuitive

Examples:

- H = intervals; X= real line: VC(H) = 2
- H = linear half-spaces; X = plane: VC(H) = 3
  d dimensional hyperplane: *d+1*
- H = feed-forward neural net: O(v·s·log(s))
  s units; v is VC of each component

With enough units, an ANN can learn any assignment of training labels

Is this a good thing?



No, perfect in training ⇒ poor in test

What about PAC learning with infinite H?

Essentially, VC(H) plays the role of ln|H|
For learning w/ finite H:

$$N \geq \frac{1}{\varepsilon}\left(\ln\frac{1}{\delta} + \ln|H|\right)$$

For learning w/ infinite H:

$$N \geq \frac{c}{\varepsilon}\left(\ln\frac{1}{\delta} + VC(H)\cdot\ln\frac{1}{\varepsilon}\right)$$