Announcement

- Homework available on web site
  Naïve Bayes and Decision Trees
- Relevant Talk tomorrow (Friday)

    Prof. Jeff Siskind Purdue

    Embodied Intelligence

    3405 SC, 2PM

Perceptron Decision Boundary

Compare weighted sum of inputs to a threshold

$$\sum_{i=1}^{n} w_i \cdot x_i > \theta$$

Without loss of generality set x_0= -1 then w_0 is θ

$$\sum_{i=0}^{n} w_i \cdot x_i > 0$$

This defines a decision surface

$$\sum_{i=0}^{n} w_i \cdot x_i = \mathbf{w} \cdot \mathbf{x} = 0$$

Which is the equation of a hyperplane

Perceptron Learning

(Widrow-Hoff or delta rule)

$\text{percep}_{\mathbf{w}}(\mathbf{x})$ assigns + or 1 if $\mathbf{w} \cdot \mathbf{x} > 0$ (vector dot product)
else it assigns – or 0

err = label($\mathbf{x}$) – $\text{percep}_{\mathbf{w}}(\mathbf{x})$
0: correct -1: false pos 1: false neg

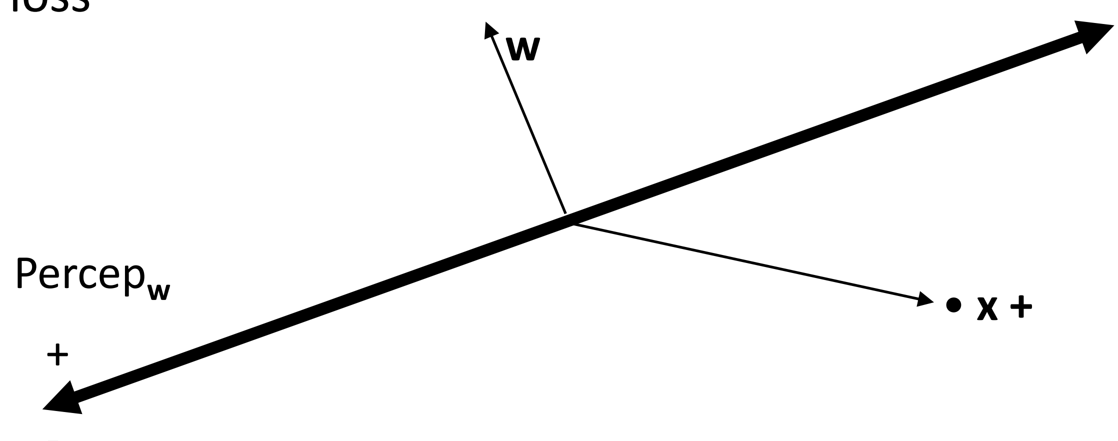
Here, false neg: $\mathbf{w} \cdot \mathbf{x} < 0$ but it should be > 0

loss = distance from boundary = - err $\mathbf{w} \cdot \mathbf{x}$

Want to adjust w_i's to reduce this loss

Loss fcn *gradient* is direction of greatest increase in loss with **w**

Want the opposite: step **w** in direction $-\nabla_w$ loss



Perceptron Learning

(Widrow-Hoff or delta rule)

Loss function = - err $\mathbf{w} \cdot \mathbf{x}$

Want the opposite: step $\mathbf{w}$ in direction $-\nabla_w$ loss

What is ∇_w loss?

View - err $\mathbf{w} \cdot \mathbf{x}$ as a function of $\mathbf{w}$

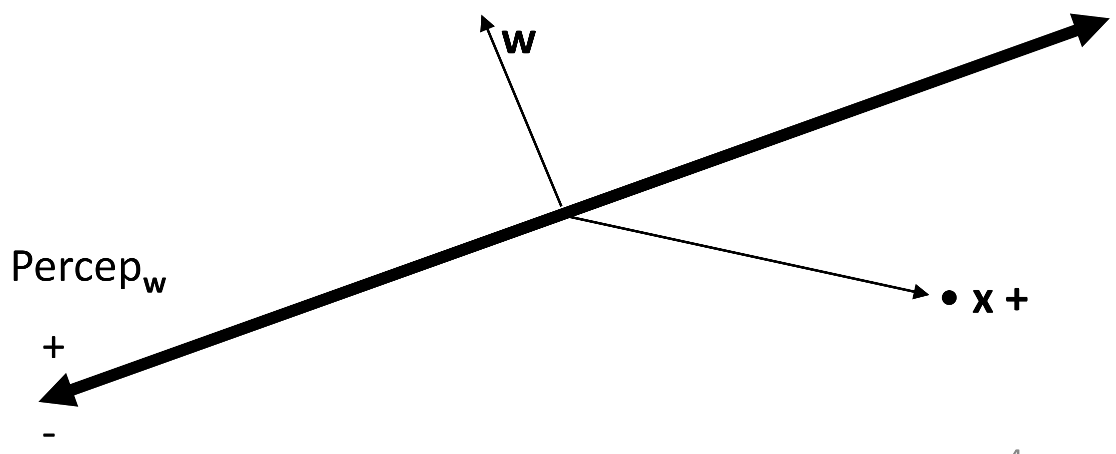
$\nabla_w(-\text{err}\ \mathbf{w} \cdot \mathbf{x}) = -\text{err}\ \mathbf{x}$

So $-\nabla_w(-\text{err}\ \mathbf{w} \cdot \mathbf{x}) = \text{err}\ \mathbf{x}$

Update $\mathbf{w}$ according to:

$$\Delta\mathbf{w} = \alpha\ \text{err}\ \mathbf{x}$$

where α is a learning rate



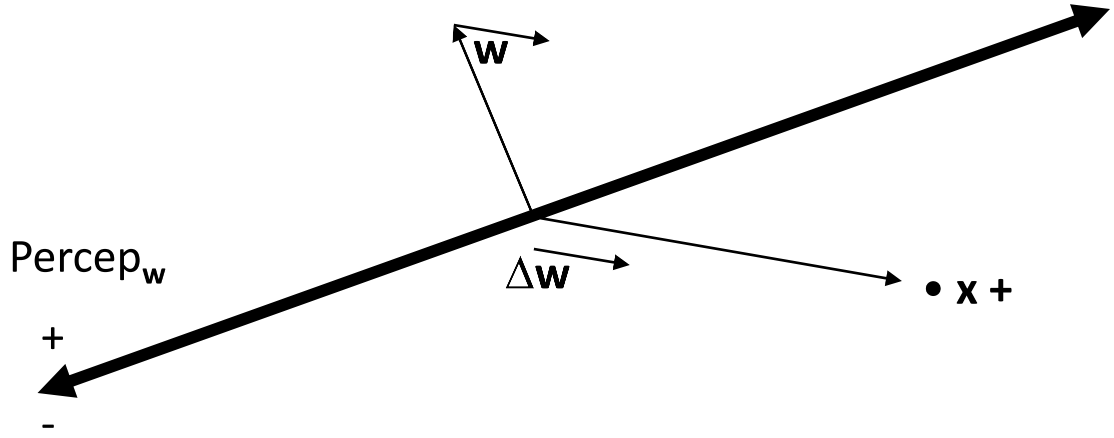
Perceptron Learning

(Widrow-Hoff or delta rule)

Choose a learning rate α

Compute $\Delta \mathbf{w} = \alpha \text{ err } \mathbf{x}$

Add $\Delta \mathbf{w}$ to $\mathbf{w}$



Perceptron Learning

(Widrow-Hoff or delta rule)

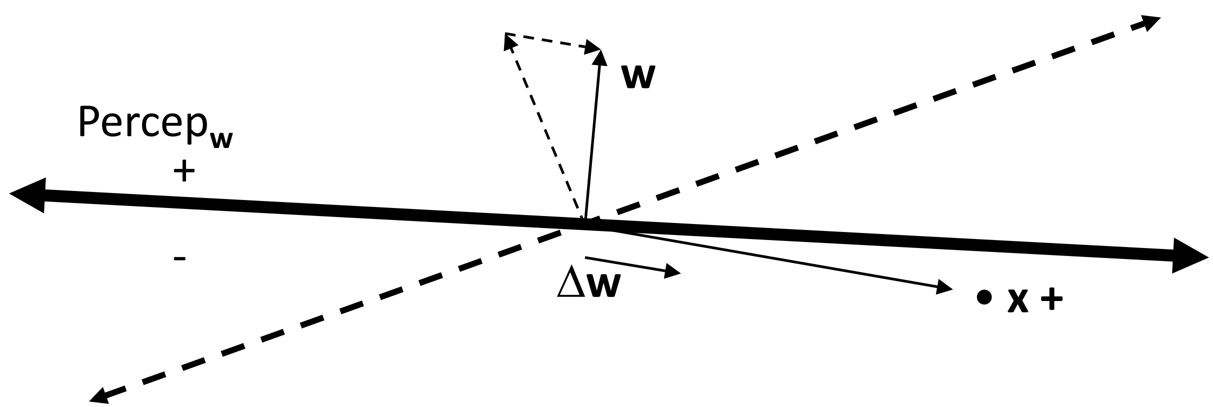
Choose a learning rate α; initialize **w** arbitrarily (small works best)

Compute $\Delta\mathbf{w} = \alpha$ err $\mathbf{x}$

Add Δ**w** to **w**

Repeatedly cycle through training examples

Learn (always and only) on errors



New perceptron rotates to reduce error

If **x** were a false positive…

Perceptron Learning

(Widrow-Hoff or delta rule)

If the points are linearly separable, the algorithm
a) will halt
b) will find a separator
(The celebrated Perceptron Convergence Theorem)

Choosing α wisely will speed convergence

If the points are not linearly separable, the algorithm may not halt.

WHY?

Bad if there is noise / uncertainty

First possibility: decay α (as in RL)

Better: accumulate $\Delta\mathbf{w}$ over an *epoch*
(one pass through the training set)

Epoch Perceptron Learning

Choose a convergence criterion (#epochs, min $|\Delta \mathbf{w}|$, …)
Choose a learning rate α, an initial $\mathbf{w}$

Repeat until converged:

$\Delta \mathbf{w} = \sum_{\mathbf{x}} \alpha \text{ err } \mathbf{x}$ (sum over training set holding **w**)

$\mathbf{w} \leftarrow \mathbf{w} + \Delta \mathbf{w}$ (update with accumulated changes)

Now it always converges

regardless of α (will influence the rate)

Whether or not training points are linearly separable

But it may not result in the fewest misclassified **x**'s

WHY?
(hint: what are we optimizing?)

Generative vs. Discriminative

- Is the perceptron generative or discriminative?
  - Models the decision boundary
  - Classify based on boundary side
  - Generally Boolean output concepts: assigns {+, -}
- Naïve Bayes is a generative classifier
  - Models membership in each label class
  - Classify based on goodness of fit
  - Works fine with more class labels
- Multi-class with discriminative learners
  - One vs. All (set of index functions)
  - All Pairs (vote)
  - List or Tree of distinctions
  - …

Features

- Each input is a vector of features
  - Set of name / value pairs
  - Sometimes "attributes"
  - Defines the example space
- Must encode events to be classified
- Aim for minimally adequate encoding
  - Asymetric "loss" for wrong feature choices
  - "Price is Right"
- Perceptron
  - Real feature values
  - Finite number
  - Fewer $\rightarrow$ less risk of overfitting
  - Too few $\rightarrow$ cannot adequately capture the target concept

Classifying Text
Simple NLP

- Distinguish text articles
  - Dog training vs. Iraq war
  - Blog vs. News
  - Spam vs. Useful email
  - Fox vs. CNN
  - Fed will raise prime vs. Fed will not
- Machine learning is best
  - For simple questions
  - Would be difficult to program directly
- Nuanced deep understanding may be in the future…
- Choose features wisely: success vs. failure

Bag of Words

- Text article is a sequence of words and punctuation
- 10,000 – 50,000 common words in English
- For simple problems: Bag of Words
  - No sequence information
  - Bag is like a set remembering repetition
  - Text representation = count of occurrences of each word

Bag of Words / Stemming

"Dogs and cats aren't natural enemies. A dog may chase a cat in fun but the cat is not eaten and seldom even killed."

Bag of Words Representation of Text

Naive:

a: 2   and: 2   arent: 1   dogs: 1   dog: 1 … democracy: 0 …

sparse vector notation; missing word $\rightarrow$ 0

Simple stemming:

a: 2   dog: 2   cat: 3   not: 1   kill: 1 …

More aggressive stemming:

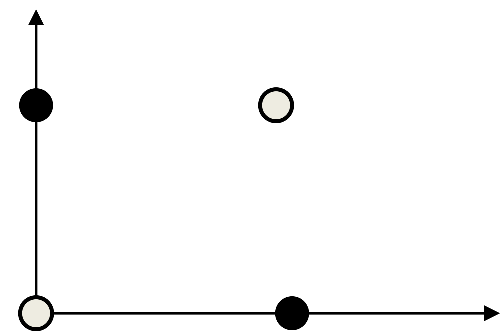
a: 2   be: 2   not: 2 …

Train a perceptron (note very high dimensional space)

Perhaps we should exclude certain words / types…

Perceptron Learning Algorithm

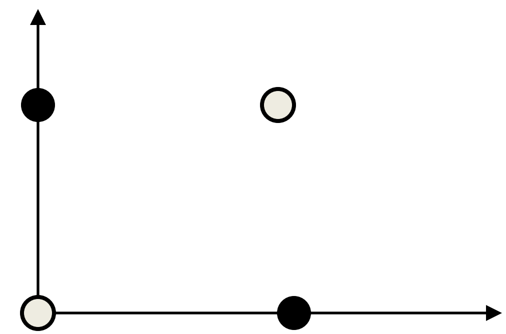
- Very limited expressiveness
- XOR on two Booleans:



- If only we could stack them
- What functions could we represent?

Limited Expressiveness of Perceptrons

- XOR on two Booleans:

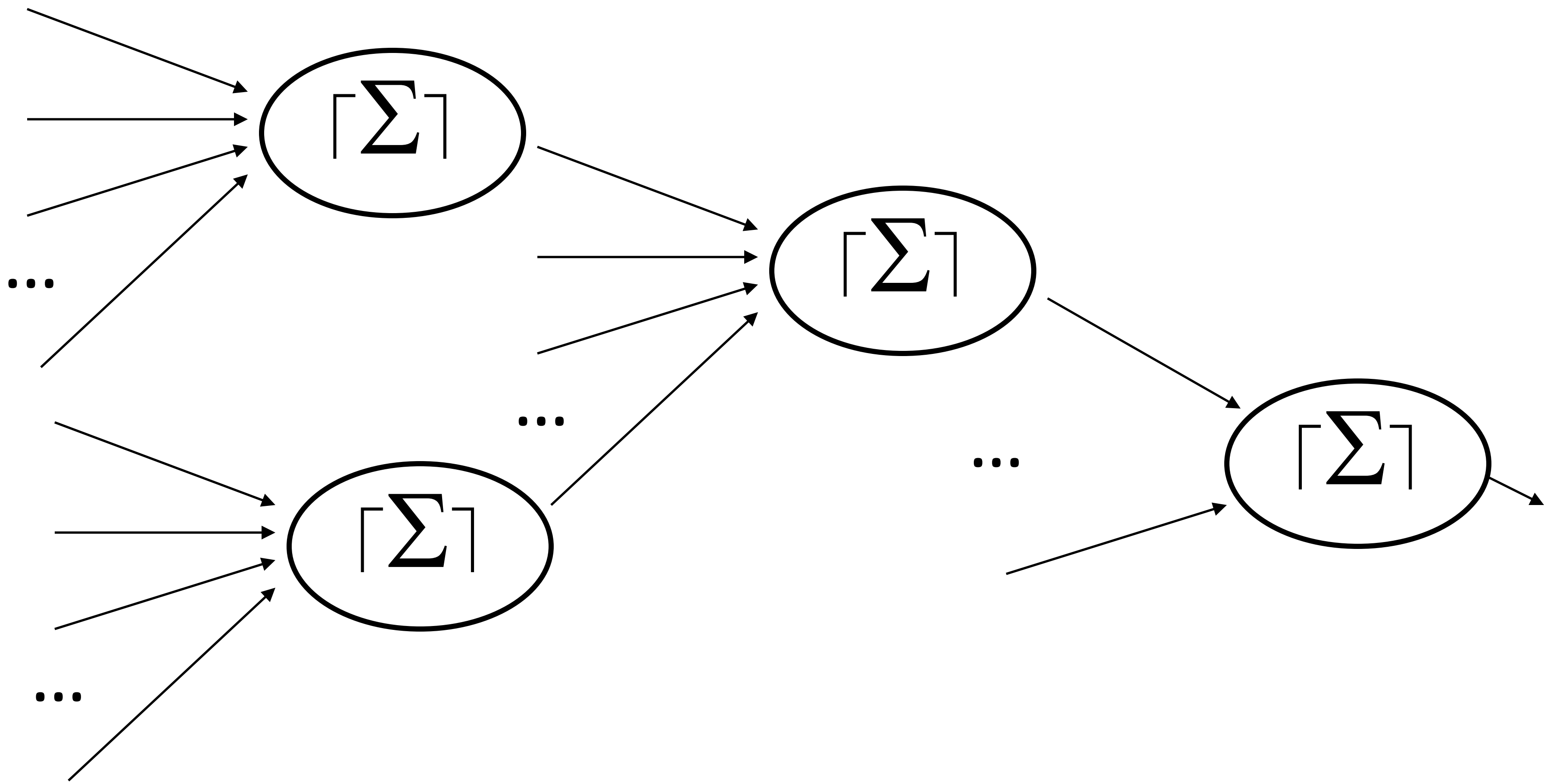


- Two important approaches:
  - Stacking them into multiple layers
  - Kernel methods
- What functions can we represent?
- How expressive is our hypothesis space?
- How hard is a learning problem?

Artificial Neural Networks:

multi-layer perceptrons

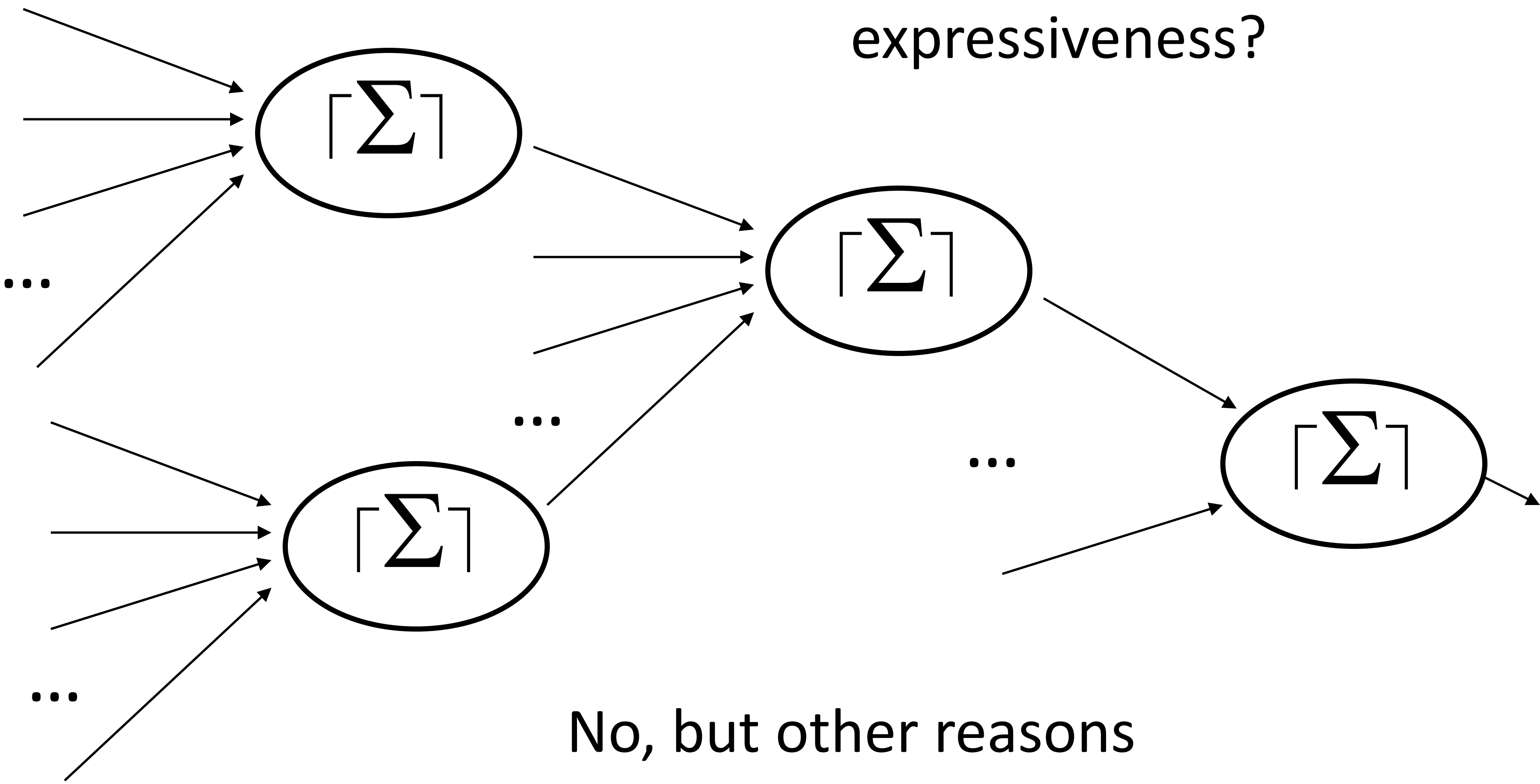
(can we represent XOR?)



Artificial Neural Networks:

multi-layer perceptrons

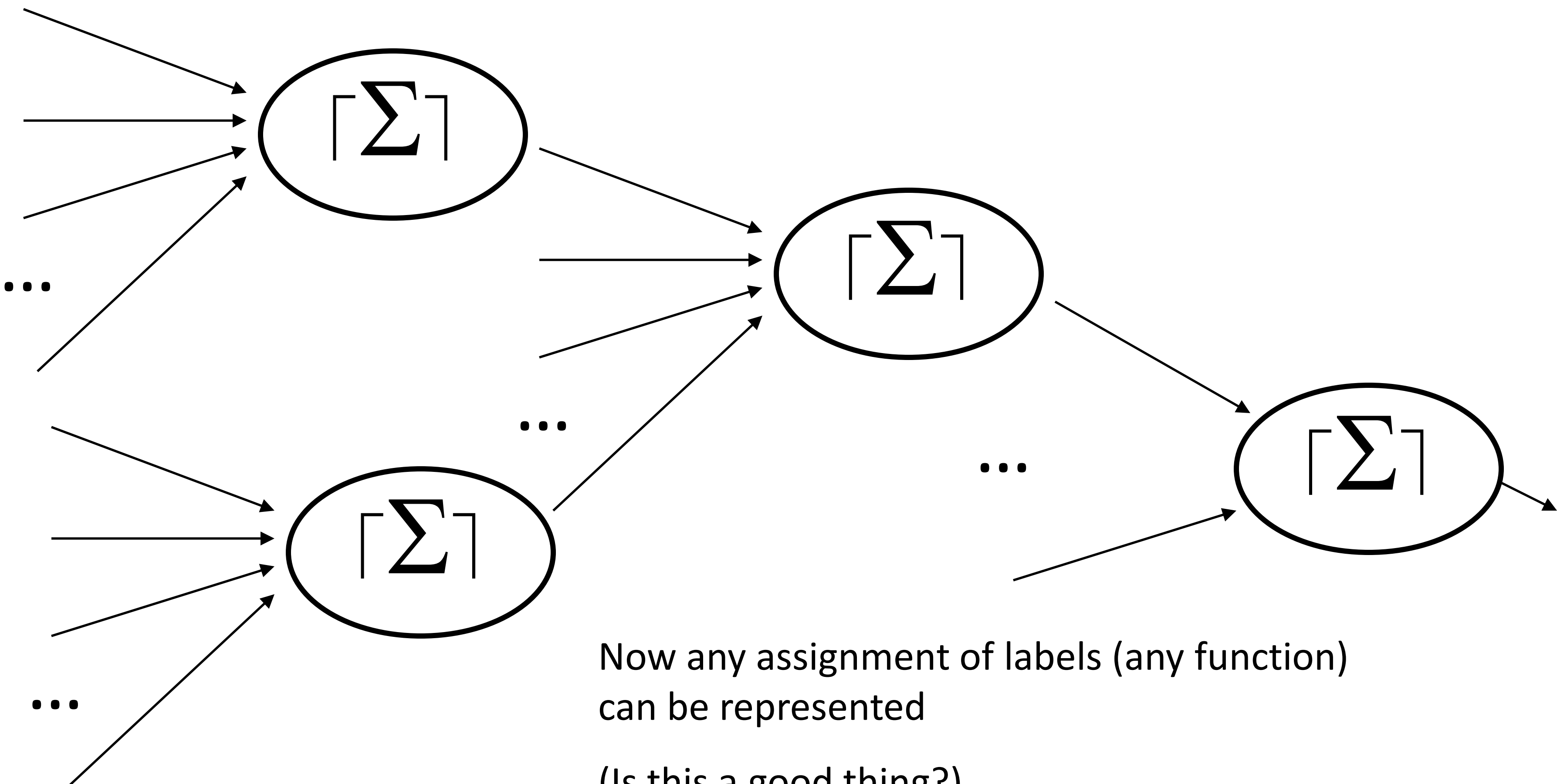
Add more levels for more expressiveness?



No, but other reasons ("deep" learning)

Can We Still Learn Efficiently?

(is there a generalized perceptron convergence theorem)



Now any assignment of labels (any function) can be represented

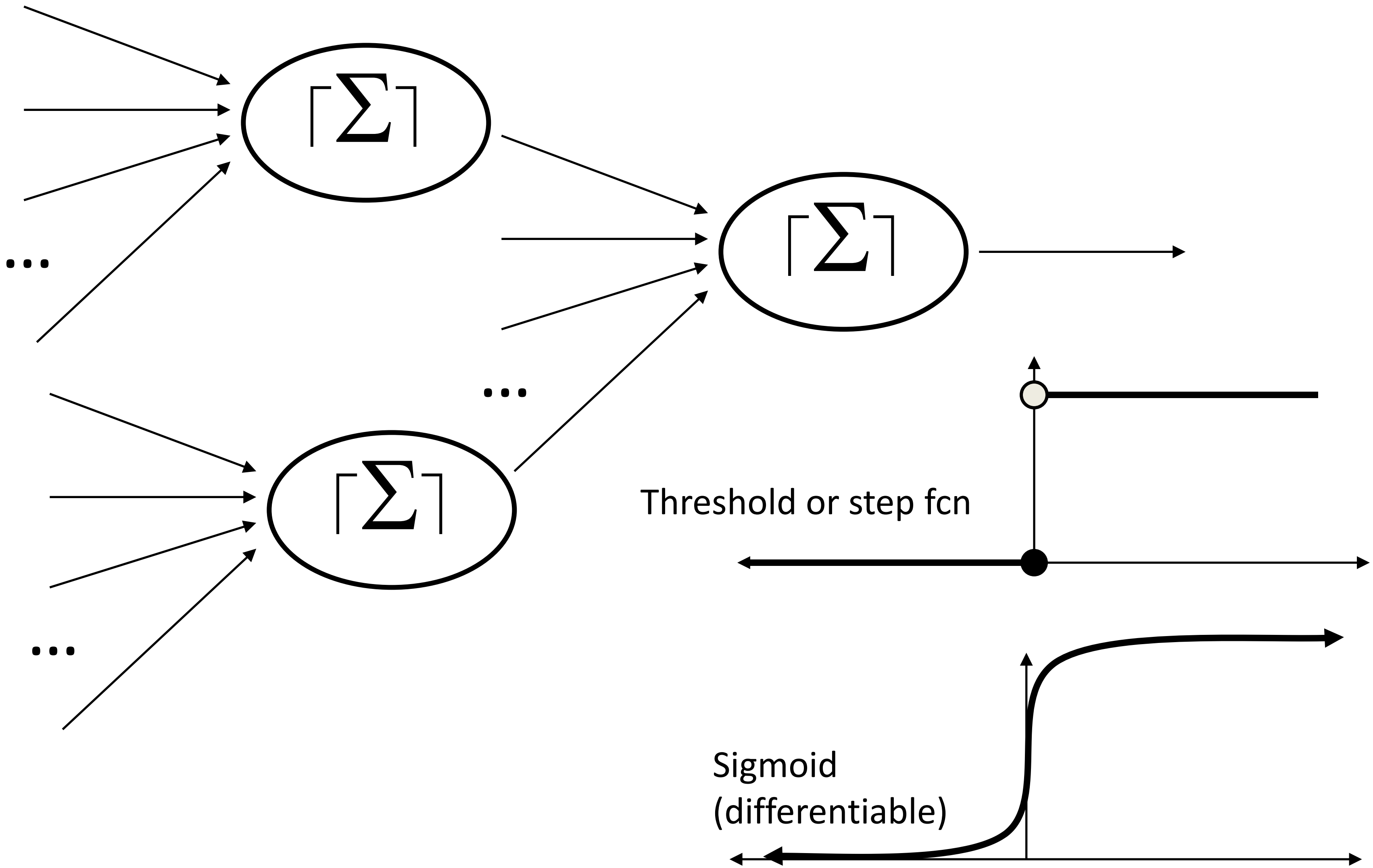
(Is this a good thing?)

No*

- Minsky and Papert suspected there was not in *Perceptrons* (1969)
- This largely killed off research interest (for nearly 20 years)
- Minsky and Papert were right

* but for a slightly modified linear device the answer becomes Yes!

Why “No”?



Back-Propogation

- Hinton, Rumlehart,…
- Common sigmoid: $g(x) = (1+e^{-x})^{-1}$
- Then $g' = g(1-g)$
- Compute $\Delta \mathbf{w} = \alpha$ err g' $\mathbf{x}$
- g’ apportions the error according to each unit’s ability to influence the error
- This is the missing factor in our weight update expression compared to eqn. 18.11
- Section 18.7.4 explains it well

Computational Learning Theory How Much Data is Enough?

- Training set is evidence for which $h \in H$ 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?
- What is the valuable resource?
- Training examples
- Hard learning problems require more training examples
- Hardest learning problems require the entire example space to be labeled

Simple Version

- Finite hypothesis space
- One of $h \in H$ actually generates the labels
- How hard is it to find?
- Impossible!
- What if we allow approximation?
  - Settle for accuracy $1-\varepsilon$
- Still Impossible!
- What if we allow occasional error $>1-\varepsilon$?
  - Settle for high confidence $1-\delta$
- Now Possible.

[Simple] Learning

- PAC formulation
- Probably Approximately Correct
- Example space X sampled with a fixed but unknown distribution $\mathcal{D}$
- Some target concept $h^* \in H$ is used to label an iid (according to $\mathcal{D}$) sample S of N examples
- Finite H
- Algorithm: return any $h \in H$ that agrees with all N training examples **S** |**S**| = N
- Choose N sufficiently large that with high confidence (1-δ) h has accuracy of at least 1-ε $0 < \varepsilon, \delta << 1$

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