

# Computation Graphs & Backpropagation

CMSC 723 / LING 723 / INST 725

Hal Daumé III [he/him]

12 Sep 2019

(Many slides c/o Marine Carpuat)

# Announcements, logistics

- HW1 is done!
- HW2:
  - Written portion will be posted in the next 24 hours
  - Programming portion will be posted before next class (hopefully earlier)
- Question: do you find the recorded class videos useful?

# Today

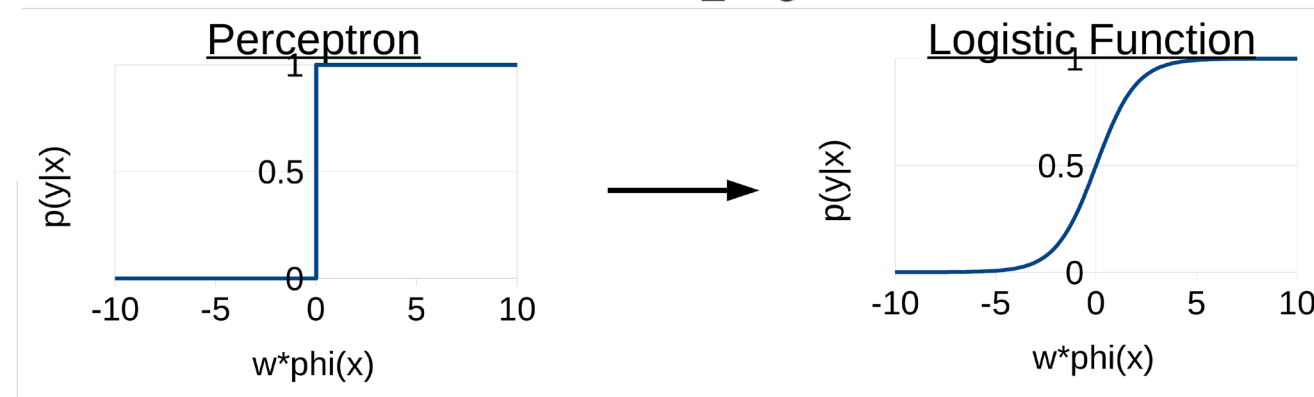
- Quick refresher on logistic regression
- Extension to non-linear models
  - aka neural networks
  - aka deep learning
  - aka artificial intelligence!!!!
- Framing: Much of NLP is about good
  - input representations
  - output representations
  - how to connect them

# Logistic Regression review

# The logistic function

- $x$ : the input
- $\boldsymbol{\varphi}(x)$ : vector of feature functions  $\{\varphi_1(x), \varphi_2(x), \dots, \varphi_l(x)\}$
- $\mathbf{w}$ : the weight vector  $\{w_1, w_2, \dots, w_l\}$
- $y$ : the prediction, +1 if “yes”, -1 if “no”

$$P(y=1|x) = \frac{e^{\mathbf{w} \cdot \boldsymbol{\varphi}(x)}}{1 + e^{\mathbf{w} \cdot \boldsymbol{\varphi}(x)}}$$



- Can account for uncertainty
- Differentiable

# Logistic regression: how to train?

- Train based on **conditional likelihood**
- Find parameters  $\mathbf{w}$  that maximize conditional likelihood of all answers  $y_i$  given examples  $x_i$

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}} \prod_i P(y_i | x_i; \mathbf{w})$$

# Stochastic gradient ascent (or descent)

- Online training algorithm for logistic regression
  - and other probabilistic models

```
create map  $w$   
for / iterations  
  for each labeled pair  $x$ ,  $y$  in the data  
     $w \ += \ \alpha \ * \ dP(y|x)/dw$ 
```

- Update weights for every training example
- Move in direction given by gradient
- Size of update step scaled by learning rate

# Gradient of the logistic function

$$\begin{aligned}\frac{d}{d \mathbf{w}} P(\mathbf{y}=1|\mathbf{x}) &= \frac{d}{d \mathbf{w}} \frac{e^{\mathbf{w} \cdot \boldsymbol{\varphi}(\mathbf{x})}}{1+e^{\mathbf{w} \cdot \boldsymbol{\varphi}(\mathbf{x})}} \\ &= \boldsymbol{\varphi}(\mathbf{x}) \frac{e^{\mathbf{w} \cdot \boldsymbol{\varphi}(\mathbf{x})}}{(1+e^{\mathbf{w} \cdot \boldsymbol{\varphi}(\mathbf{x})})^2}\end{aligned}$$

$$\begin{aligned}\frac{d}{d \mathbf{w}} P(\mathbf{y}=-1|\mathbf{x}) &= \frac{d}{d \mathbf{w}} \left(1 - \frac{e^{\mathbf{w} \cdot \boldsymbol{\varphi}(\mathbf{x})}}{1+e^{\mathbf{w} \cdot \boldsymbol{\varphi}(\mathbf{x})}}\right) \\ &= -\boldsymbol{\varphi}(\mathbf{x}) \frac{e^{\mathbf{w} \cdot \boldsymbol{\varphi}(\mathbf{x})}}{(1+e^{\mathbf{w} \cdot \boldsymbol{\varphi}(\mathbf{x})})^2}\end{aligned}$$

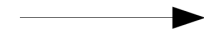


# Example: Person/not-person classification problem

Given an introductory sentence in Wikipedia  
predict whether the article is about a person

Given

Gonso was a Sanron sect priest (754-827)  
in the late Nara and early Heian periods.



Predict

Yes!

Shichikuzan Chigogataki Fudomyoo is  
a historical site located at Magura, Maizuru  
City, Kyoto Prefecture.



No!

# Example: initial update

- Set  $\alpha=1$ , initialize  $\mathbf{w}=\mathbf{0}$

$\mathbf{x}$  = A site , located in Maizuru , Kyoto       $y = -1$

$$\begin{aligned}\mathbf{w} \cdot \boldsymbol{\varphi}(\mathbf{x}) = 0 \quad \frac{d}{d\mathbf{w}} P(y = -1 | \mathbf{x}) &= -\frac{e^0}{(1+e^0)^2} \boldsymbol{\varphi}(\mathbf{x}) \\ &= -0.25 \boldsymbol{\varphi}(\mathbf{x})\end{aligned}$$

$$\mathbf{w} \leftarrow \mathbf{w} + -0.25 \boldsymbol{\varphi}(\mathbf{x})$$

$$\begin{aligned}w_{\text{unigram "Maizuru"}} &= -0.25 \\ w_{\text{unigram ","}} &= -0.5 \\ w_{\text{unigram "in"}} &= -0.25 \\ w_{\text{unigram "Kyoto"}} &= -0.25\end{aligned}$$

$$\begin{aligned}w_{\text{unigram "A"}} &= -0.25 \\ w_{\text{unigram "site"}} &= -0.25 \\ w_{\text{unigram "located"}} &= -0.25\end{aligned}$$

# Example: second update

$x$  = Shoken , monk born in Kyoto

$y = 1$

$$w \cdot \varphi(x) = -1 \quad \begin{array}{c} -0.5 \\ -0.25 \quad -0.25 \end{array} \quad \frac{d}{dw} P(y=1|x) = \frac{e^1}{(1+e^1)^2} \varphi(x) = 0.196 \varphi(x)$$

$$w \leftarrow w + 0.196 \varphi(x)$$

$w_{\text{unigram "Maizuru"}} = -0.25$   
 $w_{\text{unigram " , "}} = -0.304$   
 $w_{\text{unigram "in"}} = -0.054$   
 $w_{\text{unigram "Kyoto"}} = -0.054$

$w_{\text{unigram "A"}} = -0.25$   
 $w_{\text{unigram "site"}} = -0.25$   
 $w_{\text{unigram "located"}} = -0.25$

$w_{\text{unigram "Shoken"}} = 0.196$   
 $w_{\text{unigram "monk"}} = 0.196$   
 $w_{\text{unigram "born"}} = 0.196$

# Multiclass version

$$p(y \mid \boldsymbol{x}) = \frac{\exp(\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, y))}{\sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{\theta}^\top \boldsymbol{f}(\boldsymbol{x}, y'))}.$$

# Some models are better than others...

- Consider these 2 examples

-1 he saw a bird in the park  
+1 he saw a robbery in the park

- Which of the 2 models below is better?

## Classifier 1

he +3  
saw -5  
a +0.5  
bird -1  
robbery +1  
in +5  
the -3  
park -2

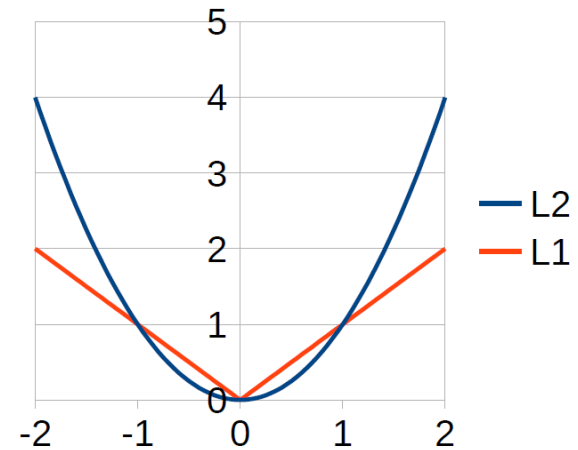
## Classifier 2

bird -1  
robbery +1

Classifier 2 will probably generalize better!  
It does not include irrelevant information  
=> Smaller model is better

# Regularization

- A penalty on adding extra weights
- L2 regularization:  $\|w\|_2$ 
  - big penalty on large weights
  - small penalty on small weights
- L1 regularization:  $\|w\|_1$ 
  - Uniform increase when large or small
  - Will cause many weights to become zero



Neural networks

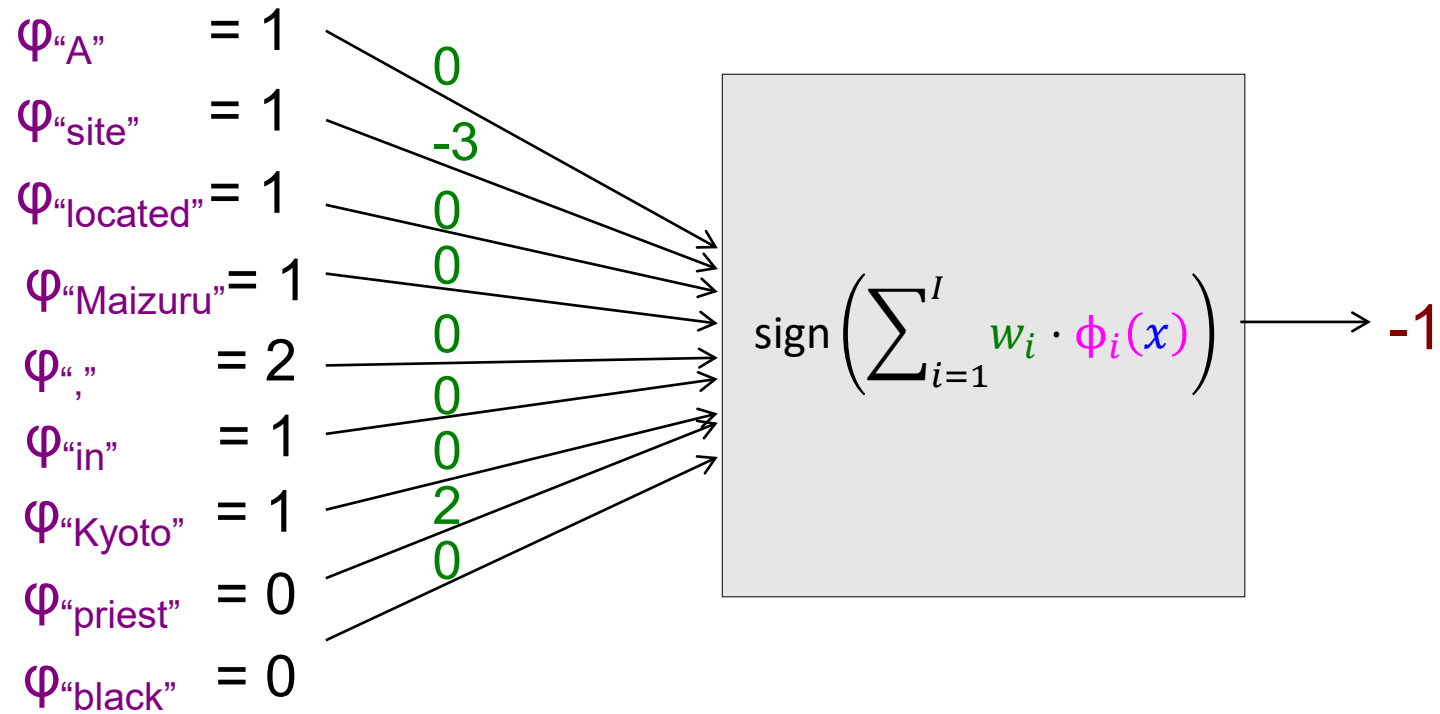
# Formalizing binary prediction

$$\begin{aligned} y &= \text{sign}(\mathbf{w} \cdot \boldsymbol{\varphi}(x)) \\ &= \text{sign}\left(\sum_{i=1}^I \mathbf{w}_i \cdot \varphi_i(x)\right) \end{aligned}$$

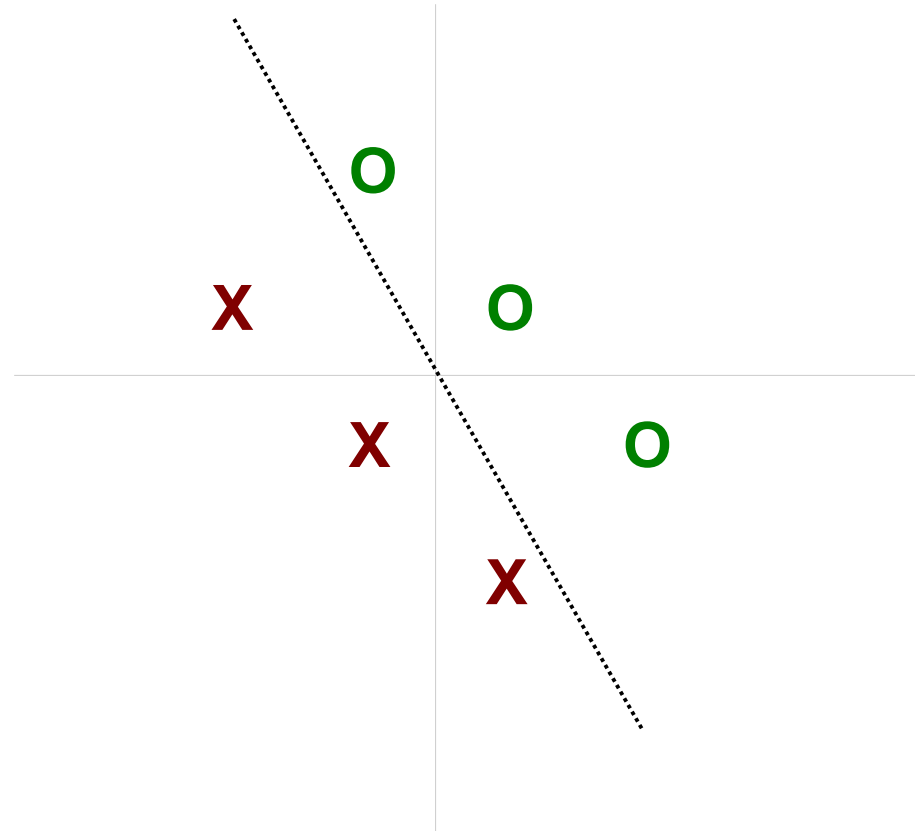
- $x$ : the input
- $\boldsymbol{\varphi}(x)$ : vector of feature functions  $\{\varphi_1(x), \varphi_2(x), \dots, \varphi_I(x)\}$
- $\mathbf{w}$ : the weight vector  $\{w_1, w_2, \dots, w_I\}$
- $y$ : the prediction, +1 if “yes”, -1 if “no”
  - ( $\text{sign}(v)$  is +1 if  $v \geq 0$ , -1 otherwise)



Linear models:  
a “machine” to calculate a weighted sum

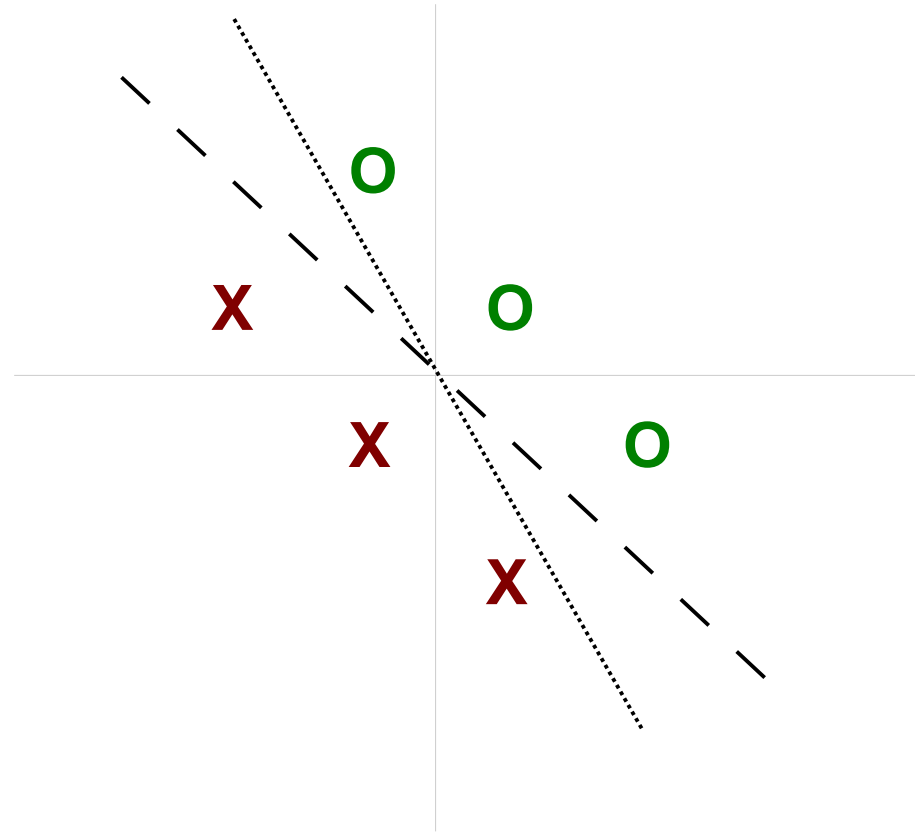


# Linear models : Geometric interpretation



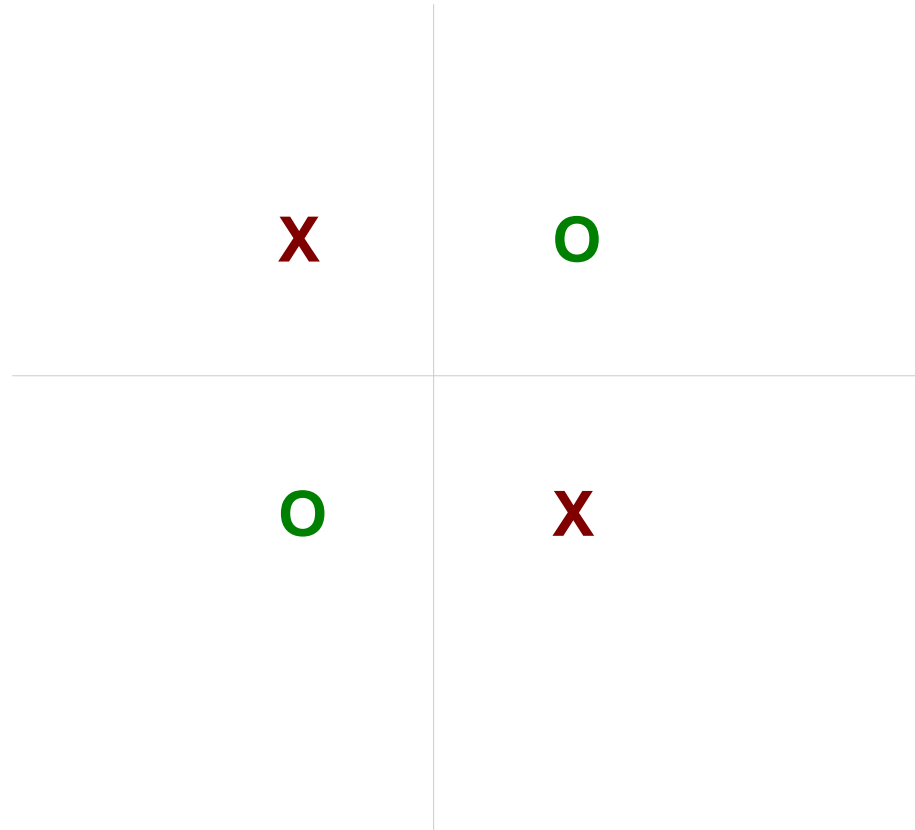
# Linear models :

## Geometric interpretation



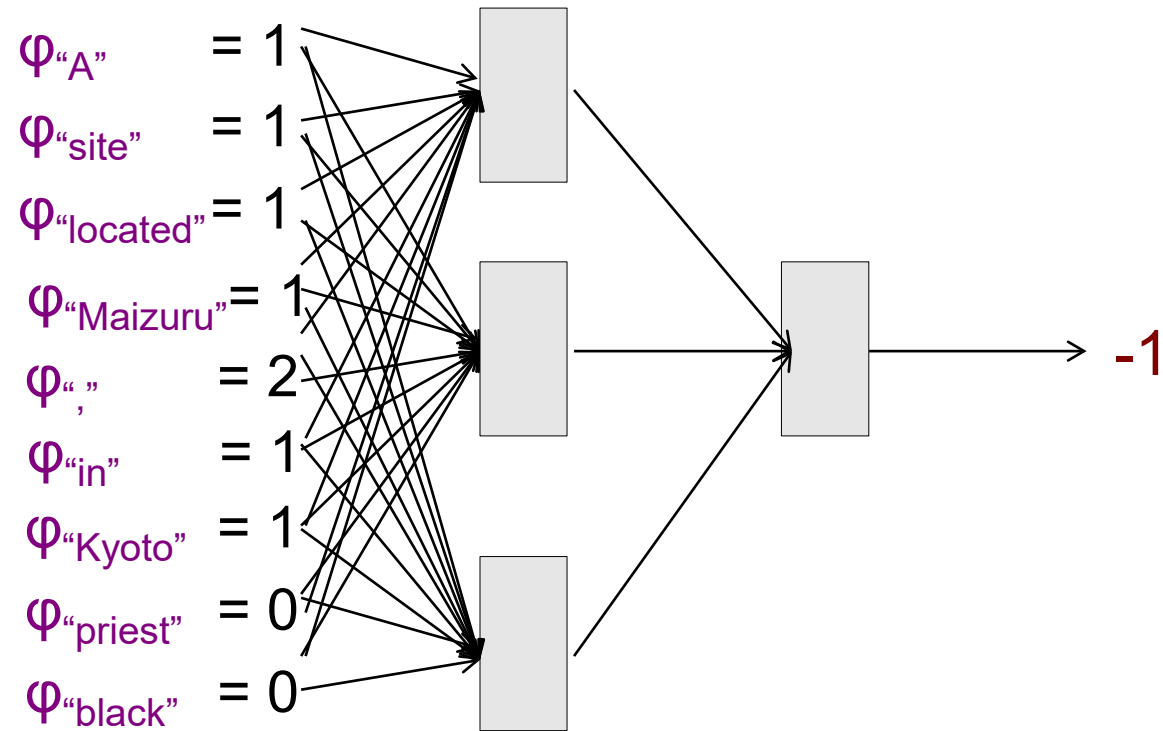
# Limitation of linear models

- can only find **linear separations** between positive and negative examples



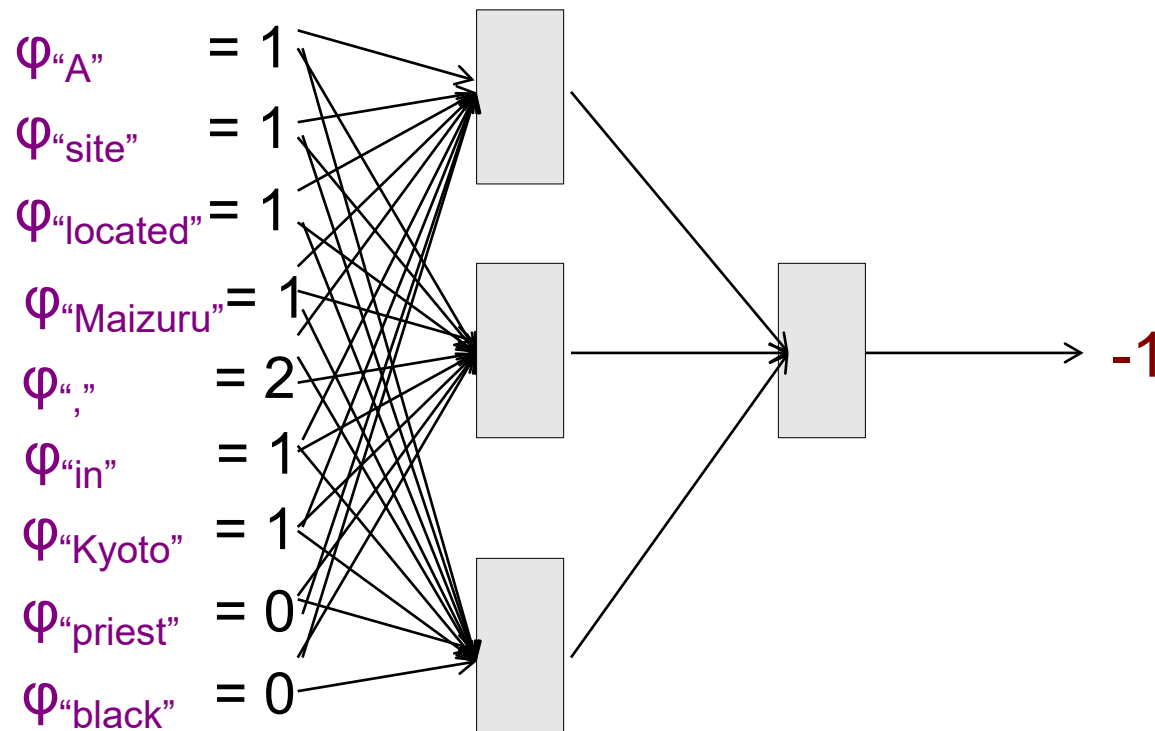
# Neural Networks

- Connect together multiple linear models with non-linear “activations”



- Motivation: Can represent non-linear functions!

# Neural Networks: key terms

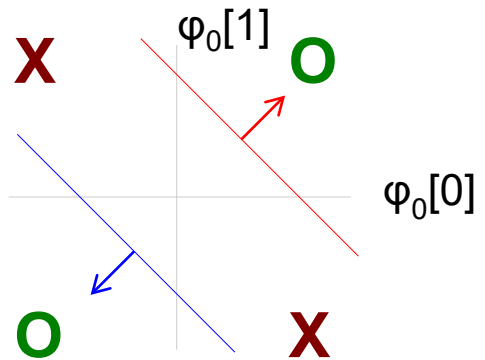


- Input (aka features)
- Output
- Nodes
- Layers
- Hidden layers
- Activation function (non-linear)
- Multi-layer perceptron

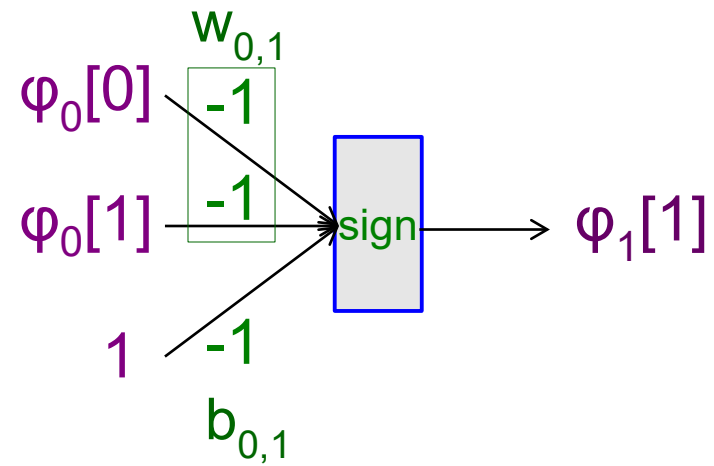
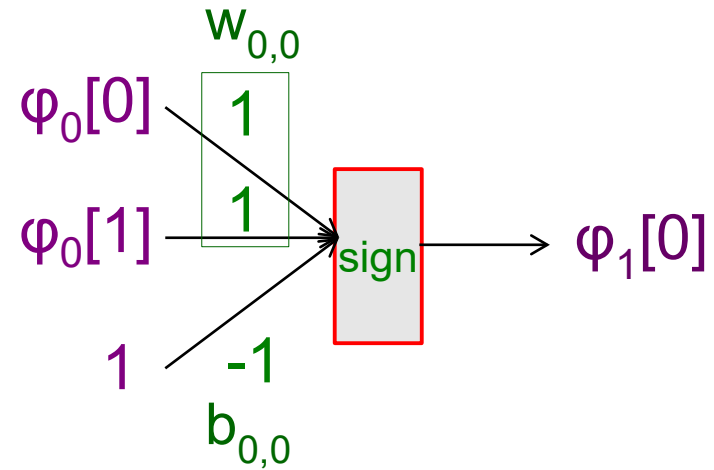
# Example

- Create two classifiers

$$\varphi_0(x_1) = \{-1, 1\} \quad \varphi_0(x_2) = \{1, 1\}$$

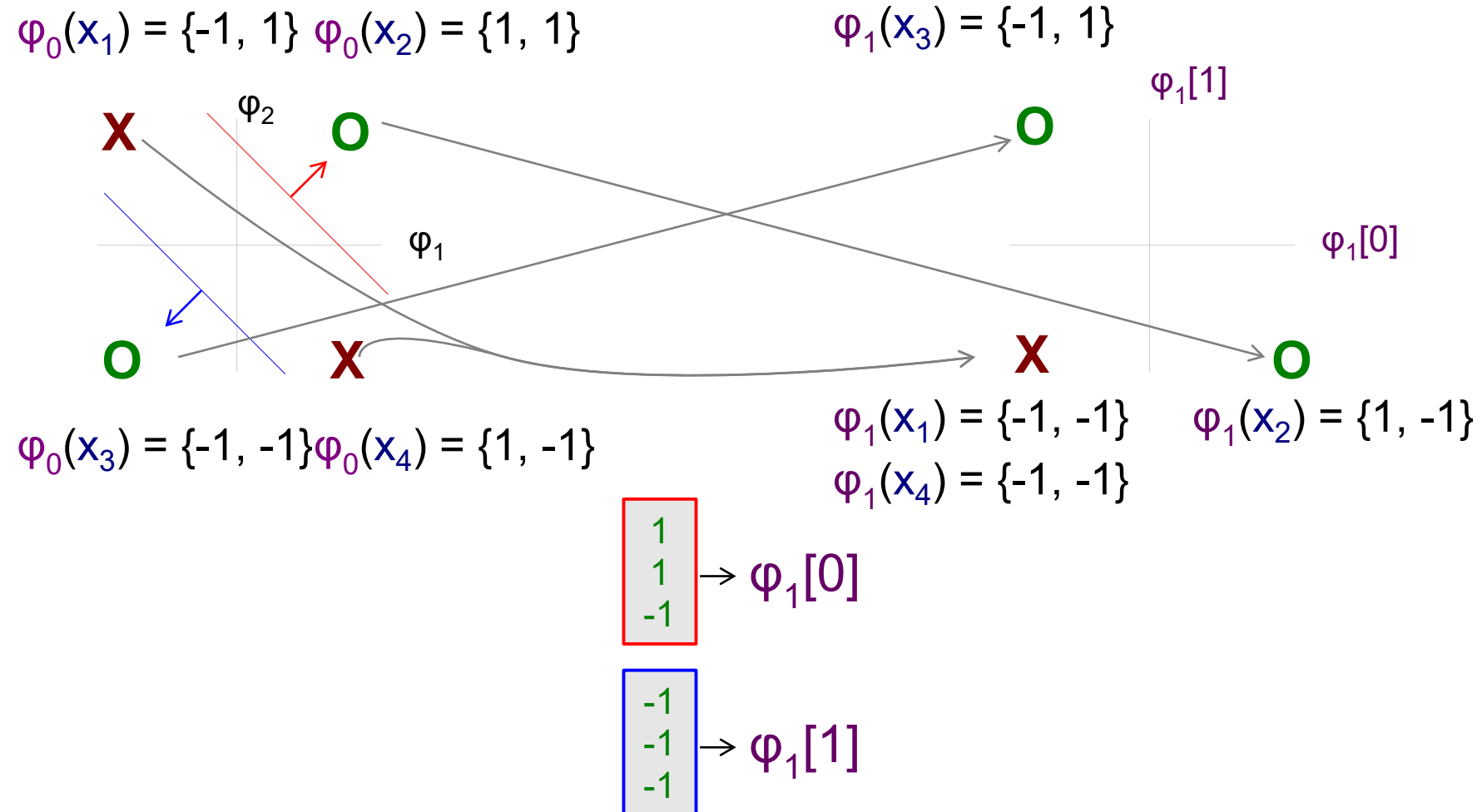


$$\varphi_0(x_3) = \{-1, -1\} \quad \varphi_0(x_4) = \{1, -1\}$$



# Example

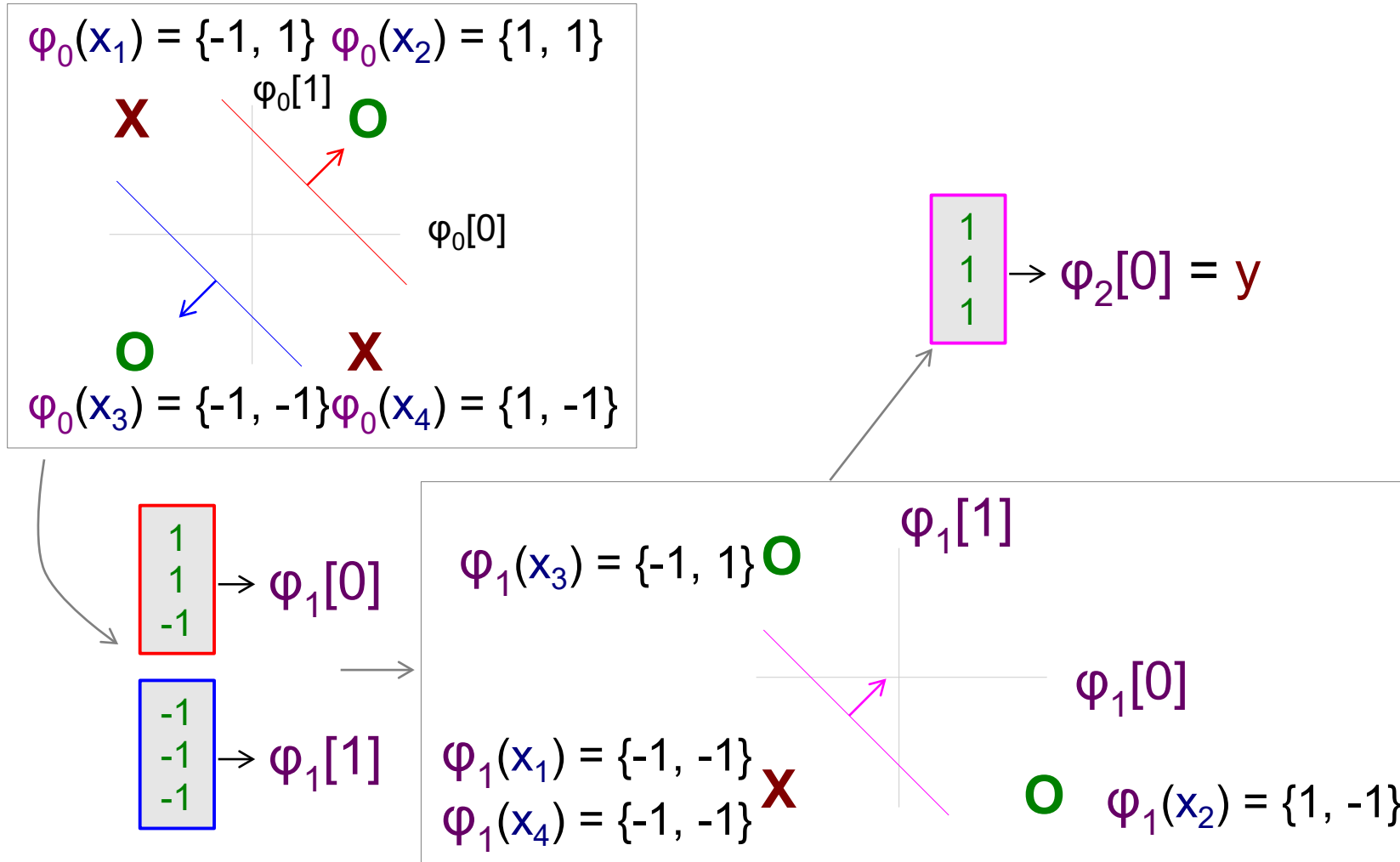
- These classifiers map to a new space





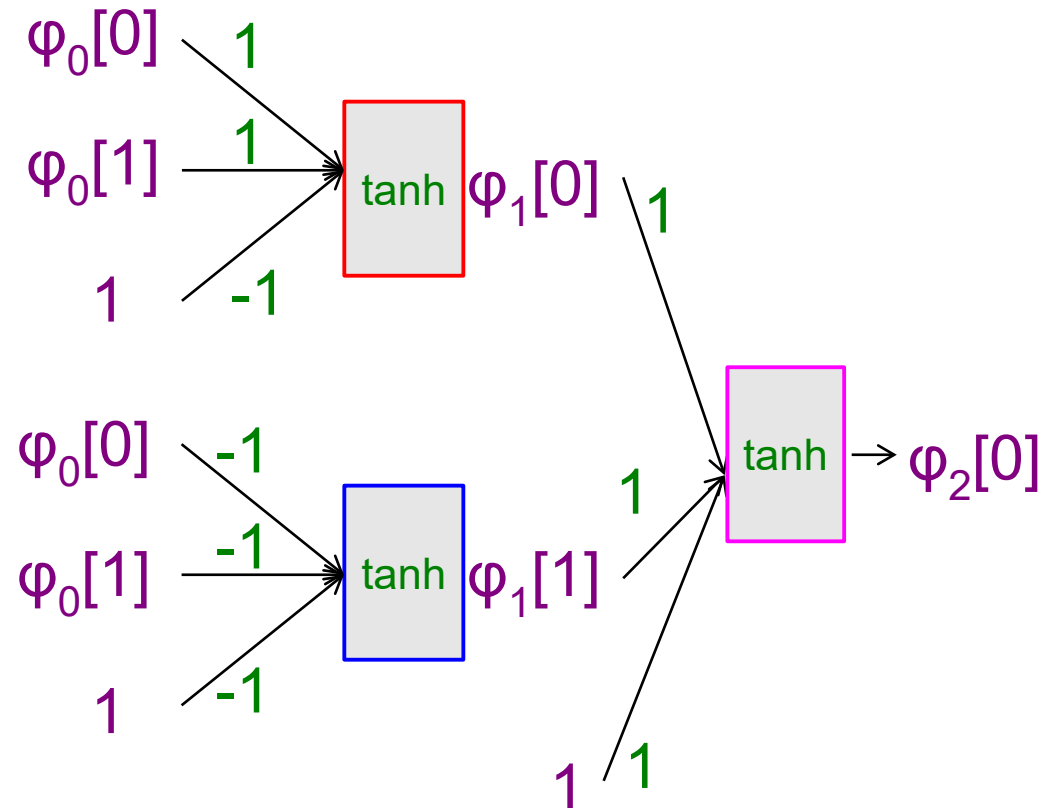
# Example

- In new space, the examples are linearly separable!



# Example wrap-up: Forward propagation

- The final net



# Softmax Function for multiclass classification

- Sigmoid function for multiple classes

$$P(\textcolor{red}{y} \mid \textcolor{blue}{x}) = \frac{e^{\textcolor{green}{\mathbf{w}} \cdot \textcolor{violet}{\phi}(\textcolor{blue}{x}, \textcolor{red}{y})}}{\sum_{\tilde{y}} e^{\textcolor{green}{\mathbf{w}} \cdot \textcolor{violet}{\phi}(\textcolor{blue}{x}, \tilde{y})}}$$

← Current class

← Sum of other classes

- Can be expressed using matrix/vector ops

$$\mathbf{r} = \exp(\textcolor{green}{\mathbf{W}} \cdot \textcolor{violet}{\phi}(\textcolor{blue}{x}, \textcolor{red}{y}))$$

$$\mathbf{p} = \mathbf{r} / \sum_{\tilde{r} \in \mathbf{r}} \tilde{r}$$

# Stochastic Gradient Descent

Online training algorithm for probabilistic models

```
 $w = 0$   
for / iterations  
  for each labeled pair  $x, y$  in the data  
     $w \ += \alpha * dP(y|x)/dw$ 
```

In other words

- For every training example, calculate the **gradient** (the direction that will increase the probability of  $y$ )
- **Move** in that direction, multiplied by learning rate  $\alpha$

# Gradient of the Sigmoid Function

Take the derivative of the probability

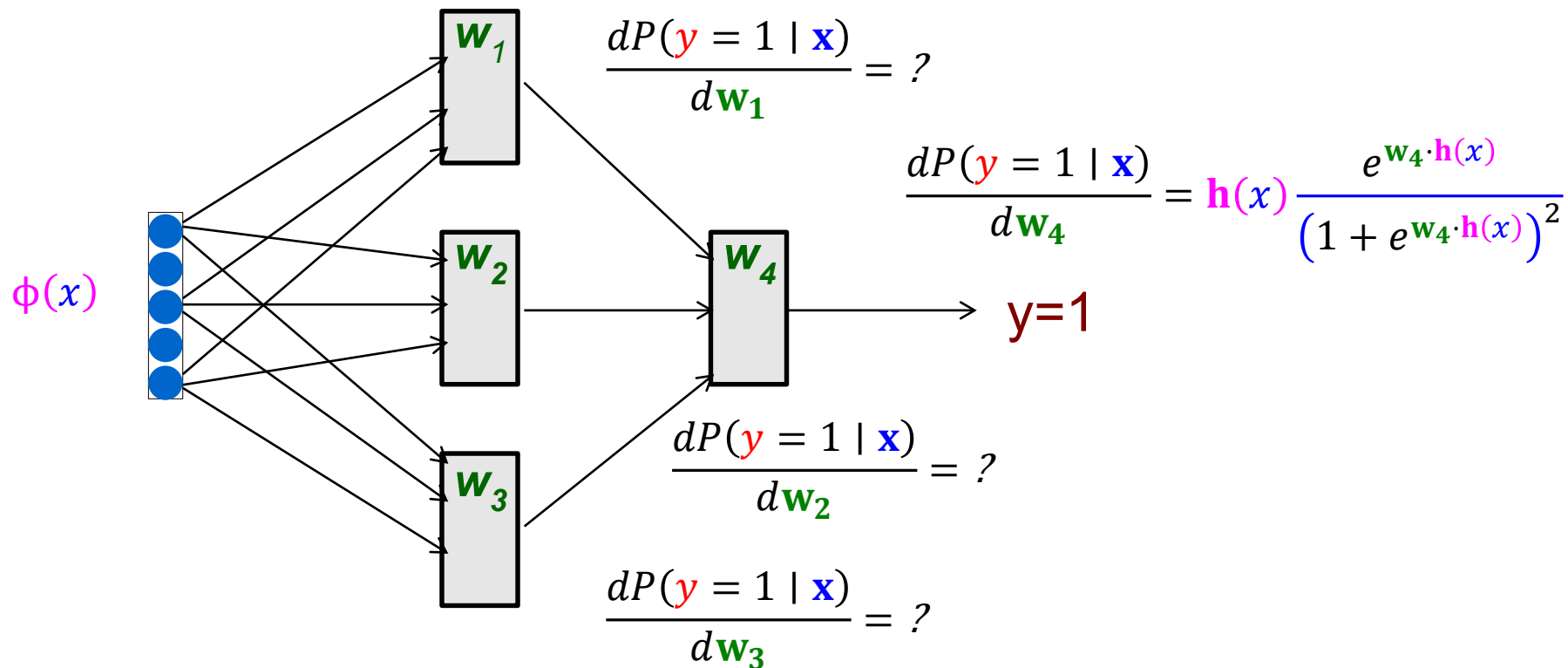
$$\begin{aligned}\frac{d}{d\mathbf{w}} P(\mathbf{y} = 1 \mid \mathbf{x}) &= \frac{d}{d\mathbf{w}} \frac{e^{\mathbf{w} \cdot \phi(\mathbf{x})}}{1 + e^{\mathbf{w} \cdot \phi(\mathbf{x})}} \\ &= \phi(\mathbf{x}) \frac{e^{\mathbf{w} \cdot \phi(\mathbf{x})}}{(1 + e^{\mathbf{w} \cdot \phi(\mathbf{x})})^2}\end{aligned}$$

$$\begin{aligned}\frac{d}{d\mathbf{w}} P(\mathbf{y} = -1 \mid \mathbf{x}) &= \frac{d}{d\mathbf{w}} \left( 1 - \frac{e^{\mathbf{w} \cdot \phi(\mathbf{x})}}{1 + e^{\mathbf{w} \cdot \phi(\mathbf{x})}} \right) \\ &= -\phi(\mathbf{x}) \frac{e^{\mathbf{w} \cdot \phi(\mathbf{x})}}{(1 + e^{\mathbf{w} \cdot \phi(\mathbf{x})})^2}\end{aligned}$$

# Learning: We Don't Know the Derivative for Hidden Units!

For NNs, only know correct tag for last layer

$\mathbf{h}(x)$



# Answer: Back-Propagation

Calculate derivative with chain rule

$$\frac{dP(\mathbf{y} = 1 \mid \mathbf{x})}{d\mathbf{w}_1} = \frac{dP(\mathbf{y} = 1 \mid \mathbf{x})}{d\mathbf{w}_4 \mathbf{h}(\mathbf{x})} \frac{d\mathbf{w}_4 \mathbf{h}(\mathbf{x})}{dh_1(\mathbf{x})} \frac{dh_1(\mathbf{x})}{d\mathbf{w}_1}$$

$\swarrow \qquad \searrow$

$$\frac{e^{\mathbf{w}_4 \cdot \mathbf{h}(\mathbf{x})}}{(1 + e^{\mathbf{w}_4 \cdot \mathbf{h}(\mathbf{x})})^2} \qquad \qquad \qquad w_{1,4}$$

$\downarrow \qquad \qquad \downarrow \qquad \downarrow$

Error of      Weight   Gradient of  
next unit ( $\delta_4$ )      this unit

In General

Calculate  $i$  based  
on next units  $j$ :

$$\frac{dP(\mathbf{y} = 1 \mid \mathbf{x})}{d\mathbf{w}_i} = \frac{dh_i(\mathbf{x})}{d\mathbf{w}_i} \sum_j \delta_j w_{i,j}$$

Backpropagation  
=  
Gradient descent  
+  
Chain rule



Note: for the mathematically inclined, this is “not quite true.”

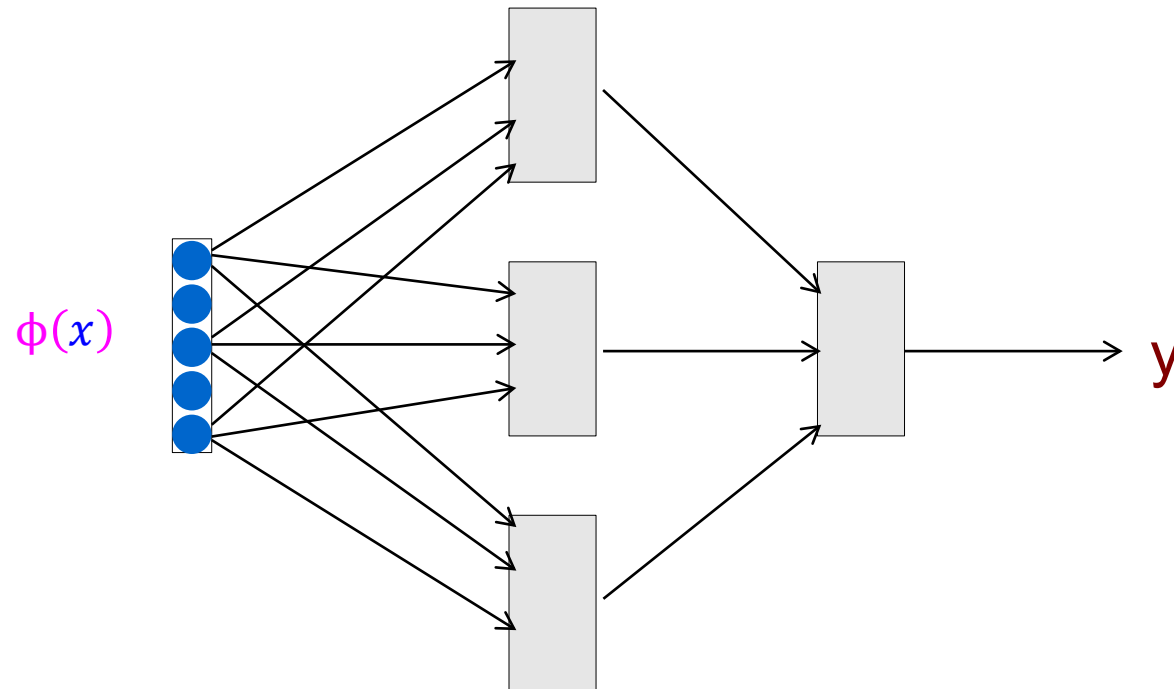
For details on why, see Tim Vieira’s nice blog post:

[timvieira.github.io/blog/post/2017/08/18/backprop-is-not-just-the-chain-rule](https://timvieira.github.io/blog/post/2017/08/18/backprop-is-not-just-the-chain-rule)



# Feed Forward Neural Nets

All connections point **forward**



It is a directed acyclic graph (DAG)

# Neural Networks summary

- Non-linear classification
- Prediction: forward propagation
  - Vector/matrix operations + non-linearities
- Training: backpropagation + stochastic gradient descent

For more details, see [CIML Chap 7](#)