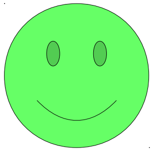
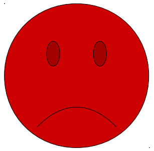
Vocabulary for Deep Learning (In 40 Minutes! With Pictures!)

Graham Neubig
Nara Institute of Science and Technology (NAIST)

Prediction Problems

Given x, predict y

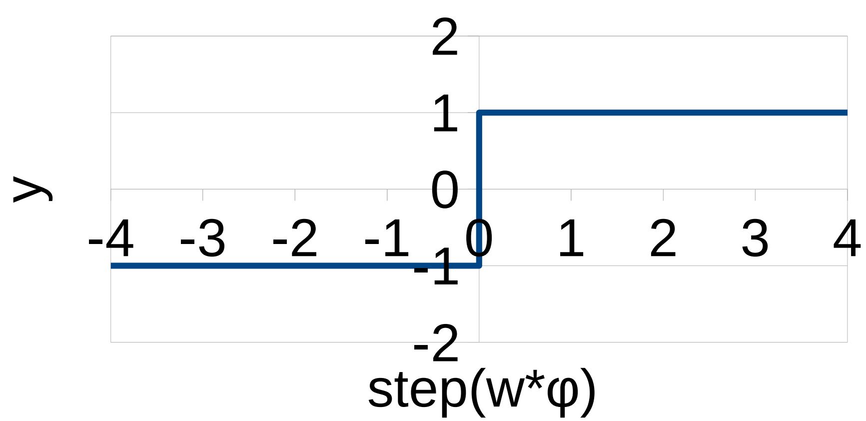
Example: Sentiment Analysis

x	y	
I am happy	+1	
I am sad	-1	

Linear Classifiers

$$y = \text{step}(\mathbf{w}^T \boldsymbol{\varphi}(x))$$

- x: the input
- **φ(x)**: vector of feature functions
- **w**: the weight vector
- y: the prediction, +1 if "yes", -1 if "no"



Example Feature Functions: Unigram Features

$$\varphi(\text{I am happy}) \rightarrow \begin{array}{c} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ \ldots \end{pmatrix} \end{array} \begin{array}{l} \text{I} \\ \text{was} \\ \text{am} \\ \text{sad} \\ \text{the} \\ \text{happy} \\ \\ \end{array} \qquad \varphi(\text{I am sad}) \rightarrow \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ \ldots \end{pmatrix}$$

The Perceptron

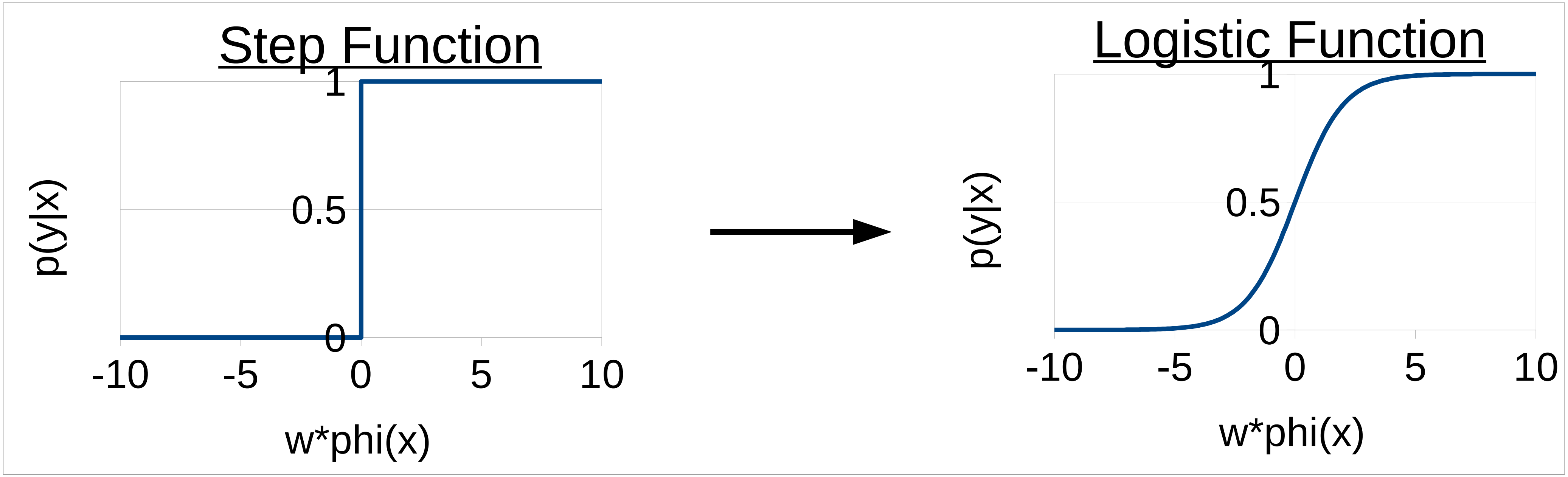
- Think of it as a “machine” to calculate a weighted sum and insert it into an activation function

$$\varphi(x) \longrightarrow \boxed{\begin{array}{l} \boldsymbol{w} \\ \mathrm{step}(\boldsymbol{w}^T \varphi(x)) \end{array}} \longrightarrow y$$

Sigmoid Function (Logistic Function)

- The sigmoid function is a “softened” version of the step function

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



- Can account for uncertainty
- Differentiable

Logistic Regression

- Train based on conditional likelihood
- Find the parameters **w** that maximize the conditional likelihood of all answers y_i given the example x_i

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

- How do we solve this?

Stochastic Gradient Descent

- Online training algorithm for probabilistic models (including logistic regression)

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

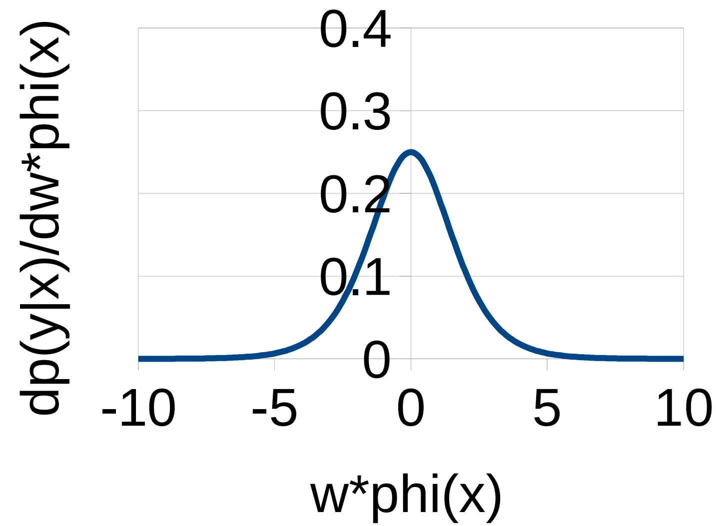
- In other words
  - For every training example, calculate the gradient (the direction that will increase the probability of y)
  - Move in that direction, multiplied by learning rate α

Gradient of the Sigmoid Function

- Take the derivative of the probability

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

$$= \varphi(x)\frac{e^{w\cdot\varphi(x)}}{(1+e^{w\cdot\varphi(x)})^2}$$



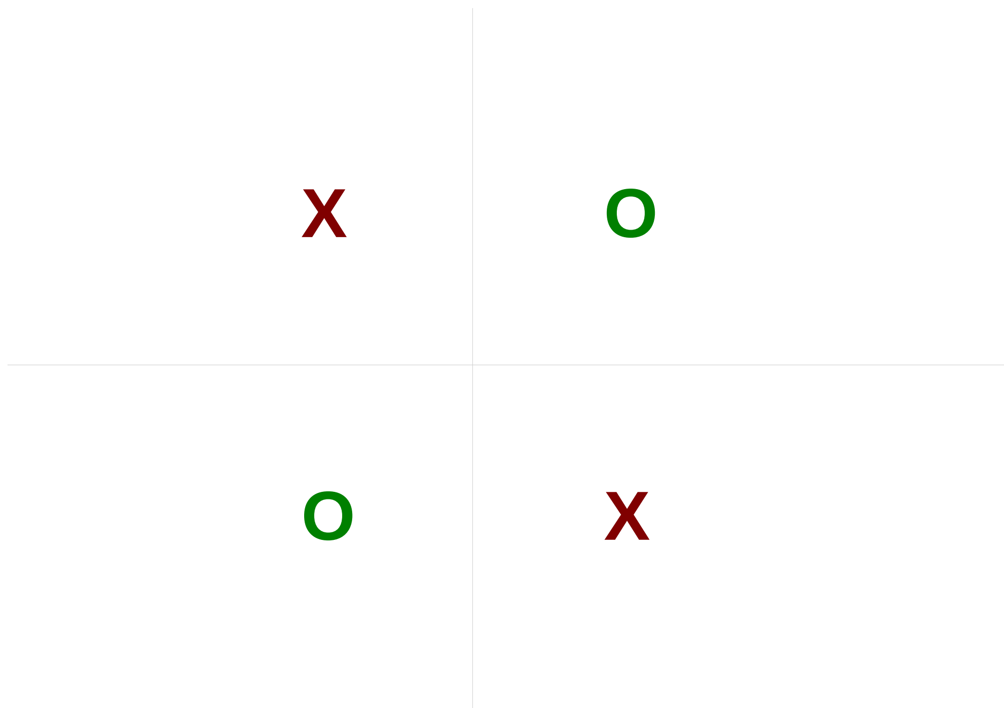
$$\frac{d}{dw}P(y=-1|x) = \frac{d}{dw}\left(1-\frac{e^{w\cdot\varphi(x)}}{1+e^{w\cdot\varphi(x)}}\right)$$

$$= -\varphi(x)\frac{e^{w\cdot\varphi(x)}}{(1+e^{w\cdot\varphi(x)})^2}$$

Neural Networks

Problem: Linear Constraint

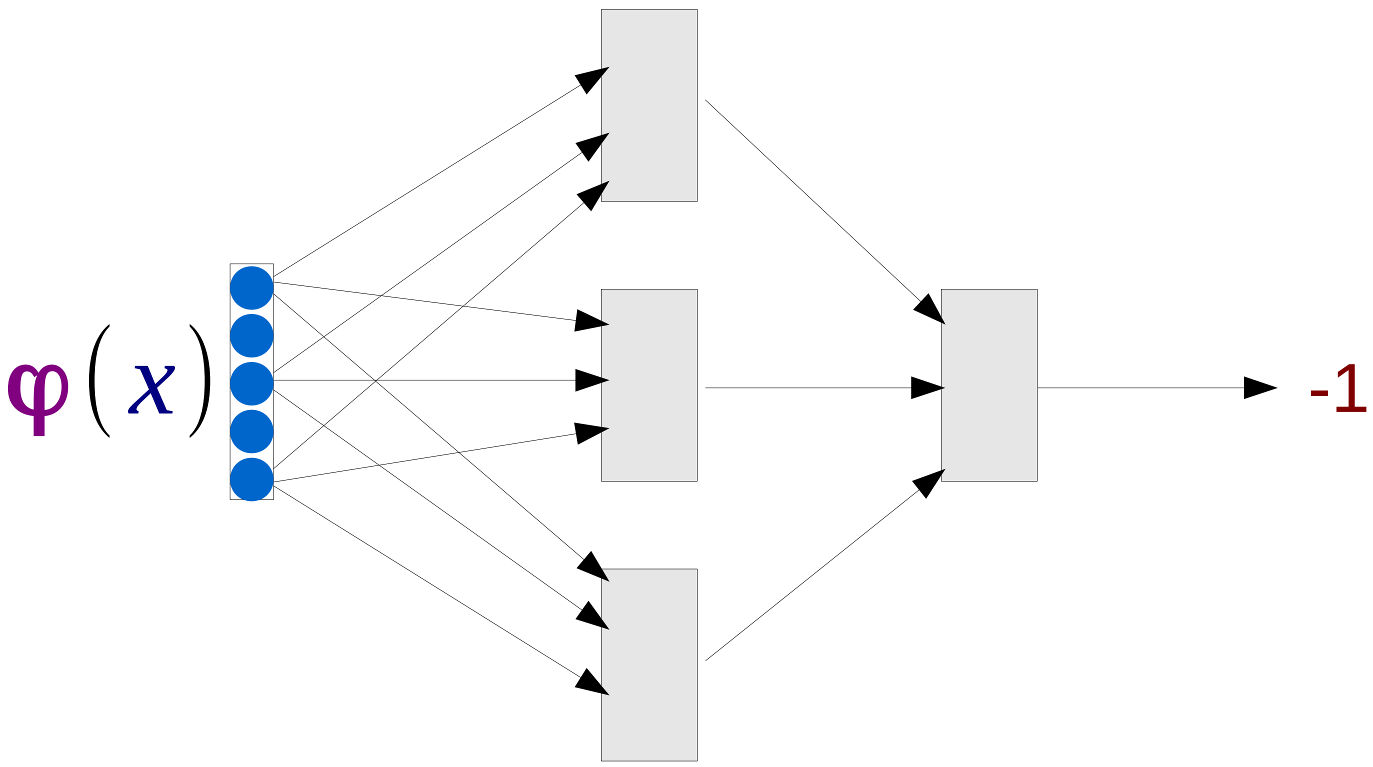
- Perceptron cannot achieve high accuracy on non-linear functions



- Example: “I am **not** happy”

Neural Networks (Multi-Layer Perceptron)

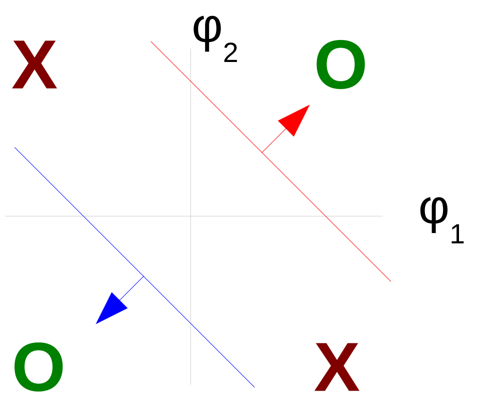
- Neural networks connect multiple perceptrons together



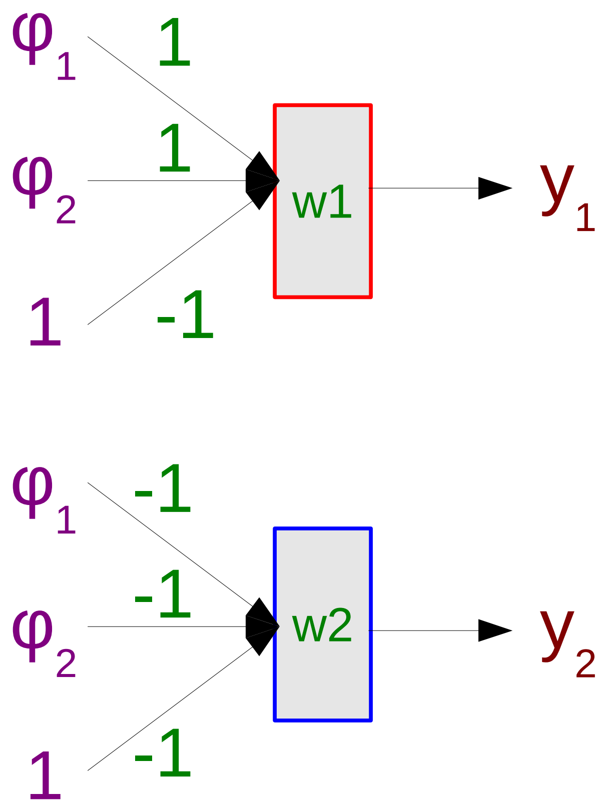
Example:

- Build two classifiers:

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



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

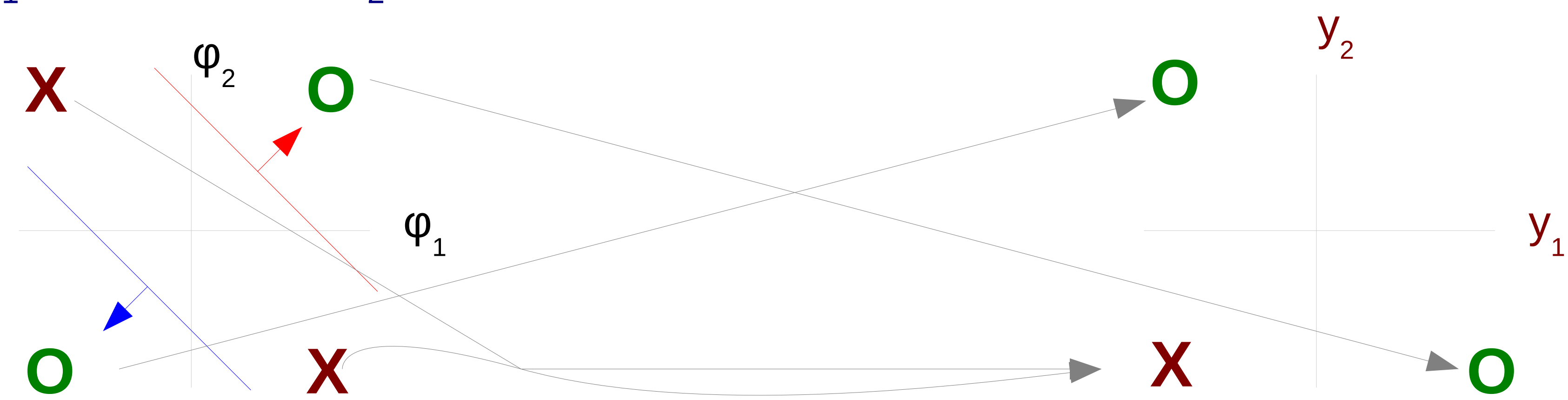


Example:

- These classifiers map the points to a new space

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

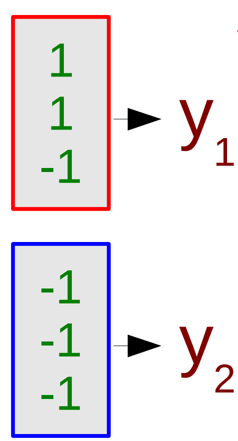
$y(x_3) = \{-1, 1\}$



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

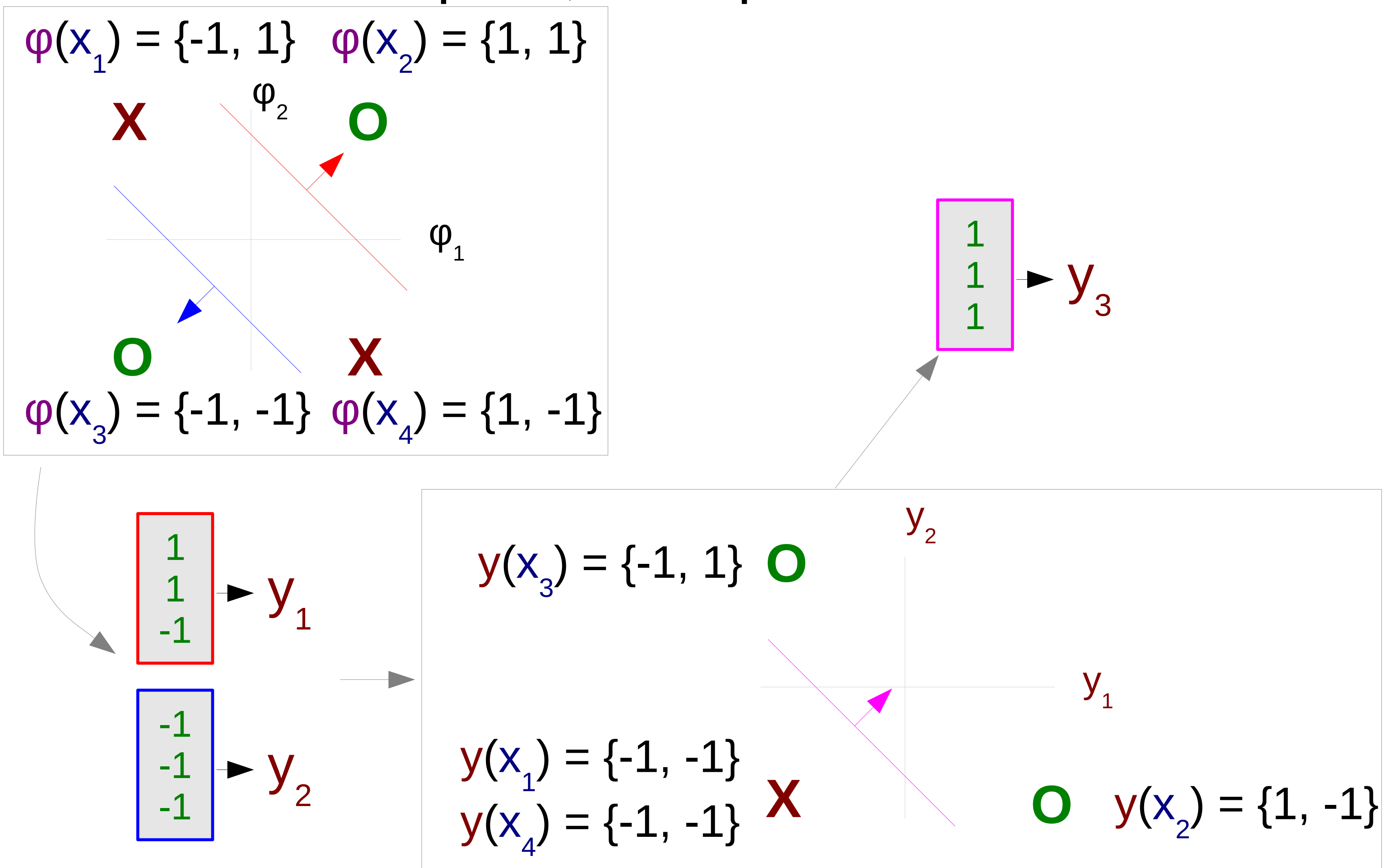
$y(x_1) = \{-1, -1\}$ $y(x_2) = \{1, -1\}$

$y(x_4) = \{-1, -1\}$



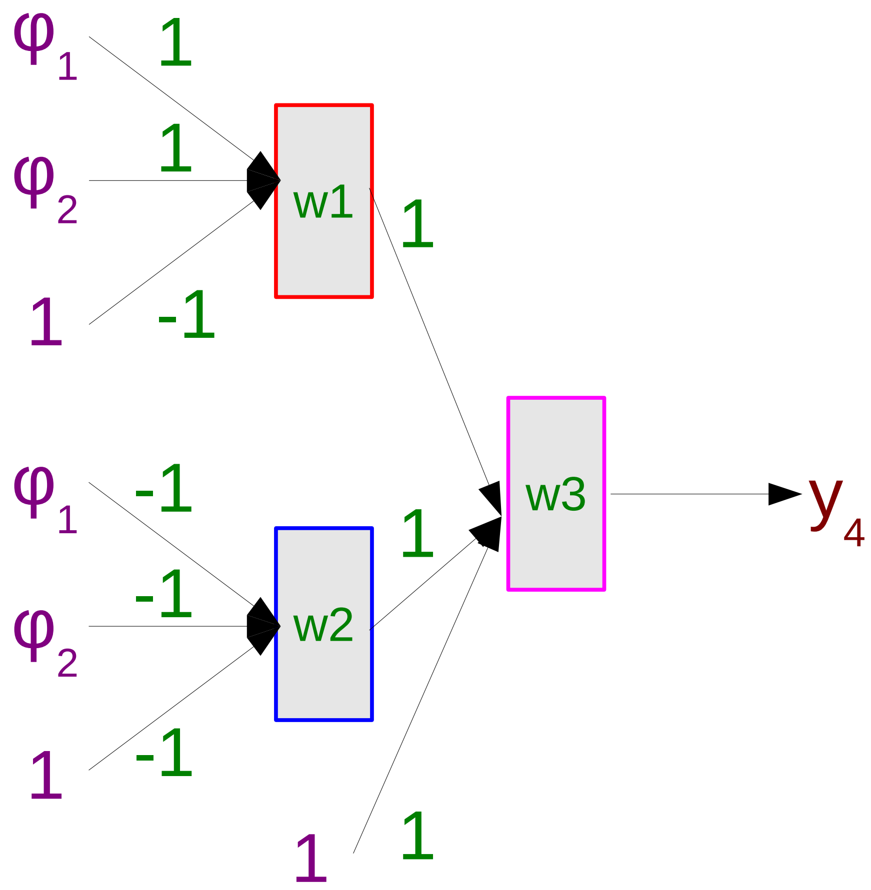
Example:

- In the new space, examples are classifiable!



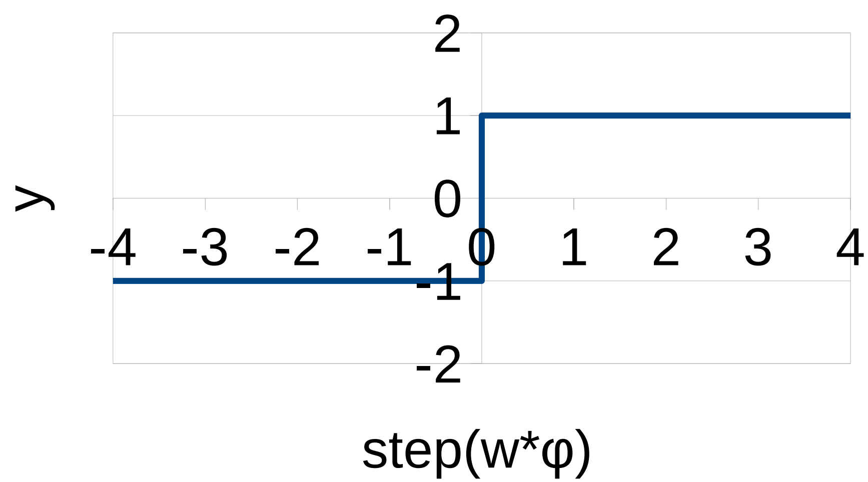
Example:

- Final neural network:

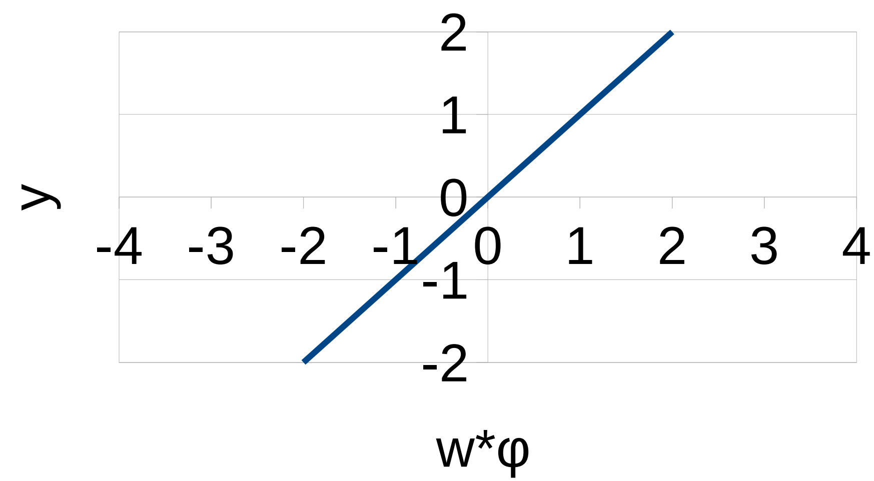


Hidden Layer Activation Functions

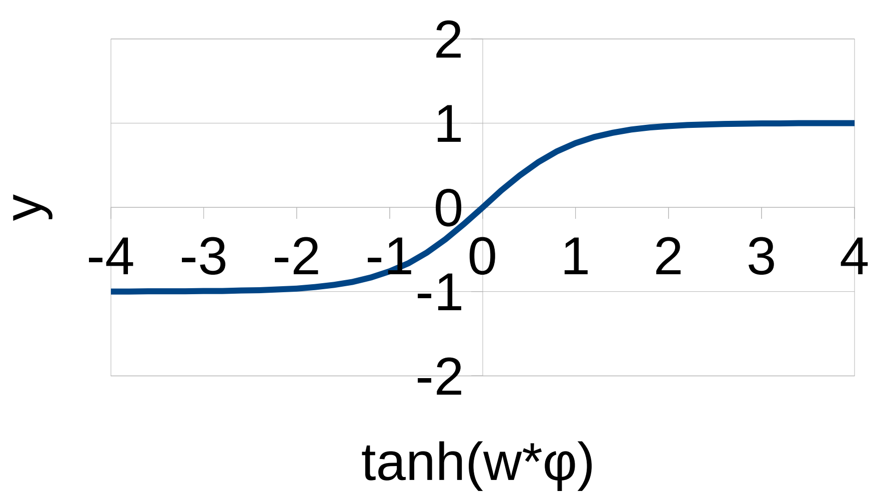
Step Function:
- Cannot calculate gradient



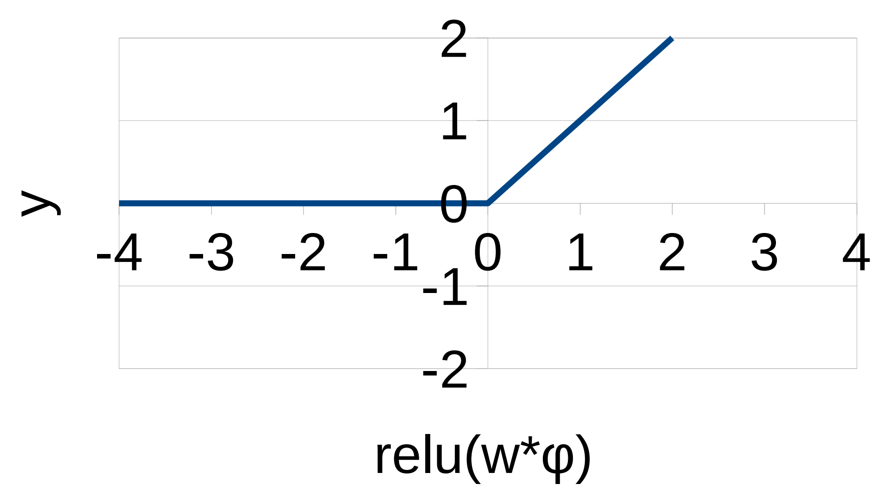
Linear Function:
- Whole net become linear



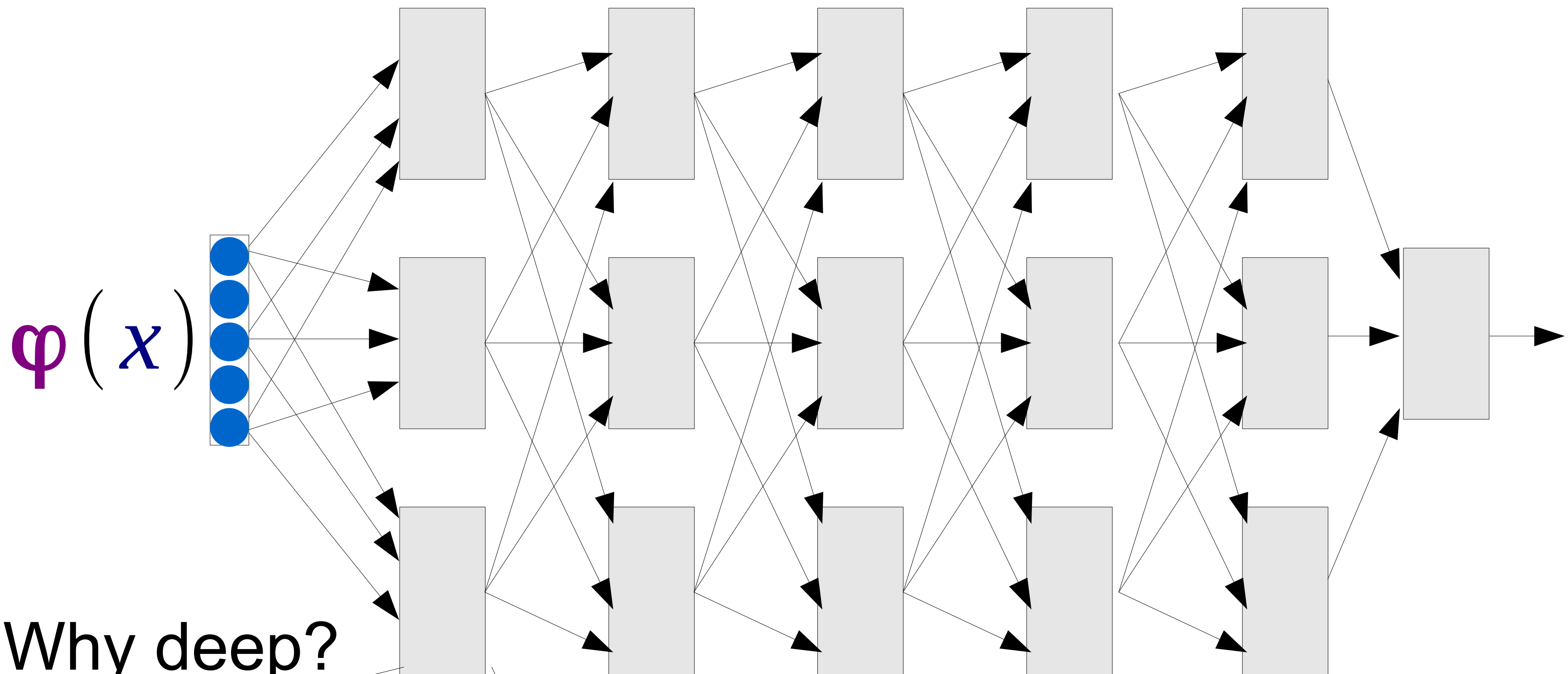
Tanh Function:
Standard (also 0-1 sigmoid)



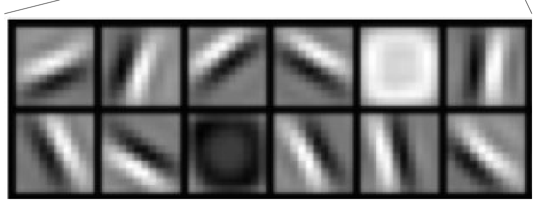
Rectified Linear Function:
+ Gradients at large vals.

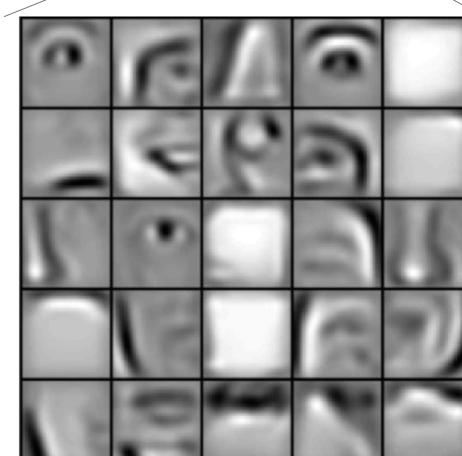


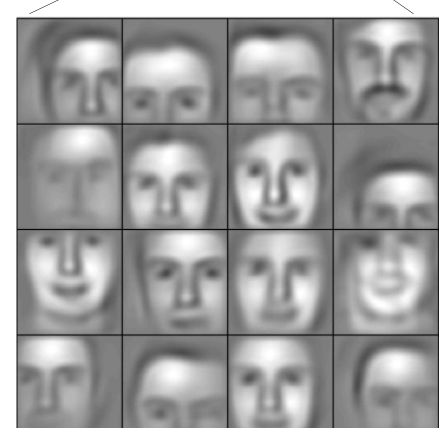
Deep Networks



- Why deep?





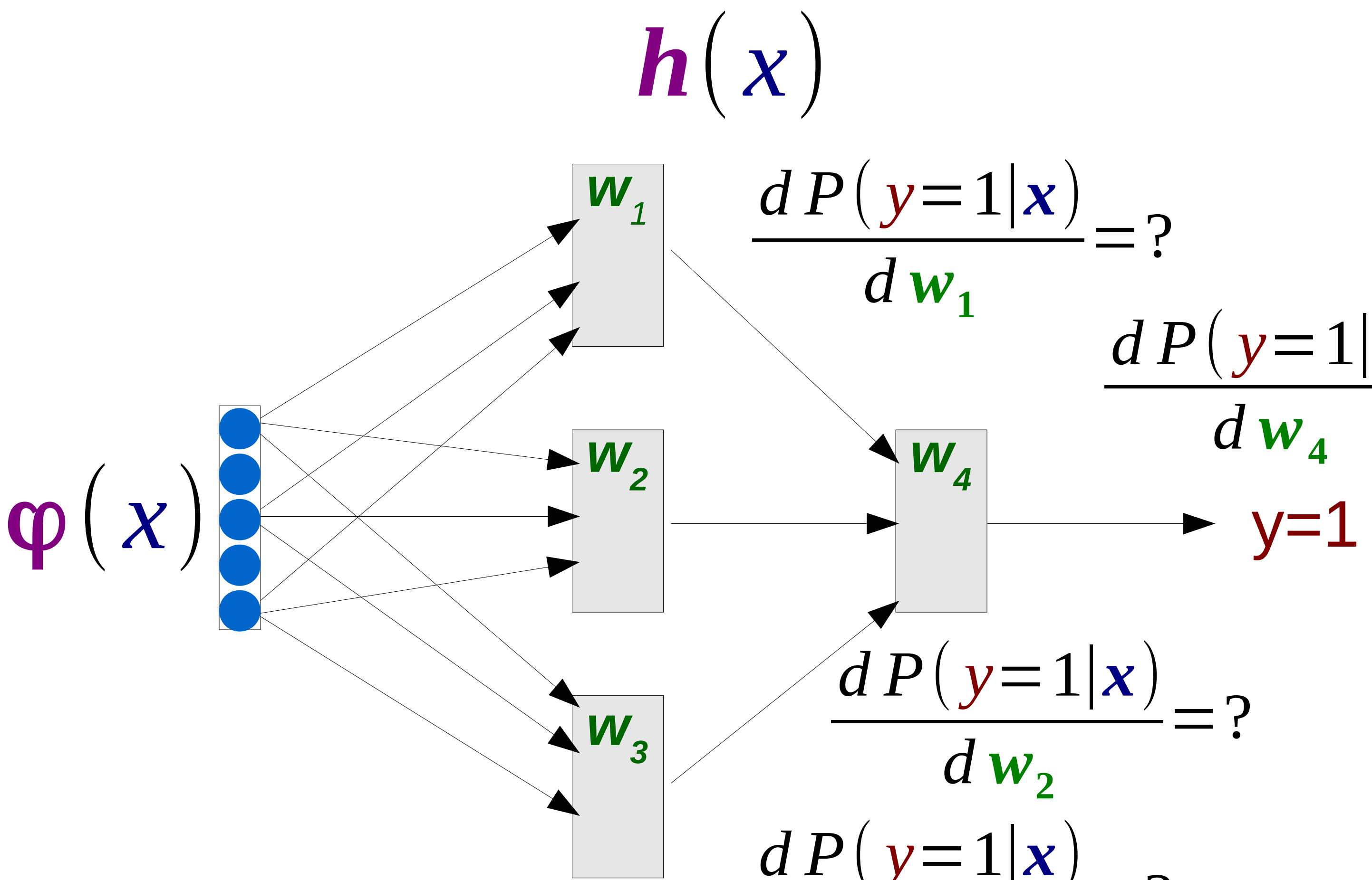


- Gradually more detailed functions (e.g. [Lee+ 09])

Learning Neural Nets

Learning: Don't Know Derivative for Hidden Units!

- For NNs, only know correct tag for last layer



$$\frac{d P(y=1|\mathbf{x})}{d \mathbf{w_1}} = ?$$

$$\frac{d P(y=1|\mathbf{x})}{d \mathbf{w_4}} = \mathbf{h}(x) \frac{e^{\mathbf{w_4}\cdot \mathbf{h}(x)}}{(1+e^{\mathbf{w_4}\cdot \mathbf{h}(x)})^2}$$

$$\frac{d P(y=1|\mathbf{x})}{d \mathbf{w_2}} = ?$$

$$\frac{d P(y=1|\mathbf{x})}{d \mathbf{w_3}} = ?$$

Answer: Back-Propogation

- Calculate derivative w/ chain rule

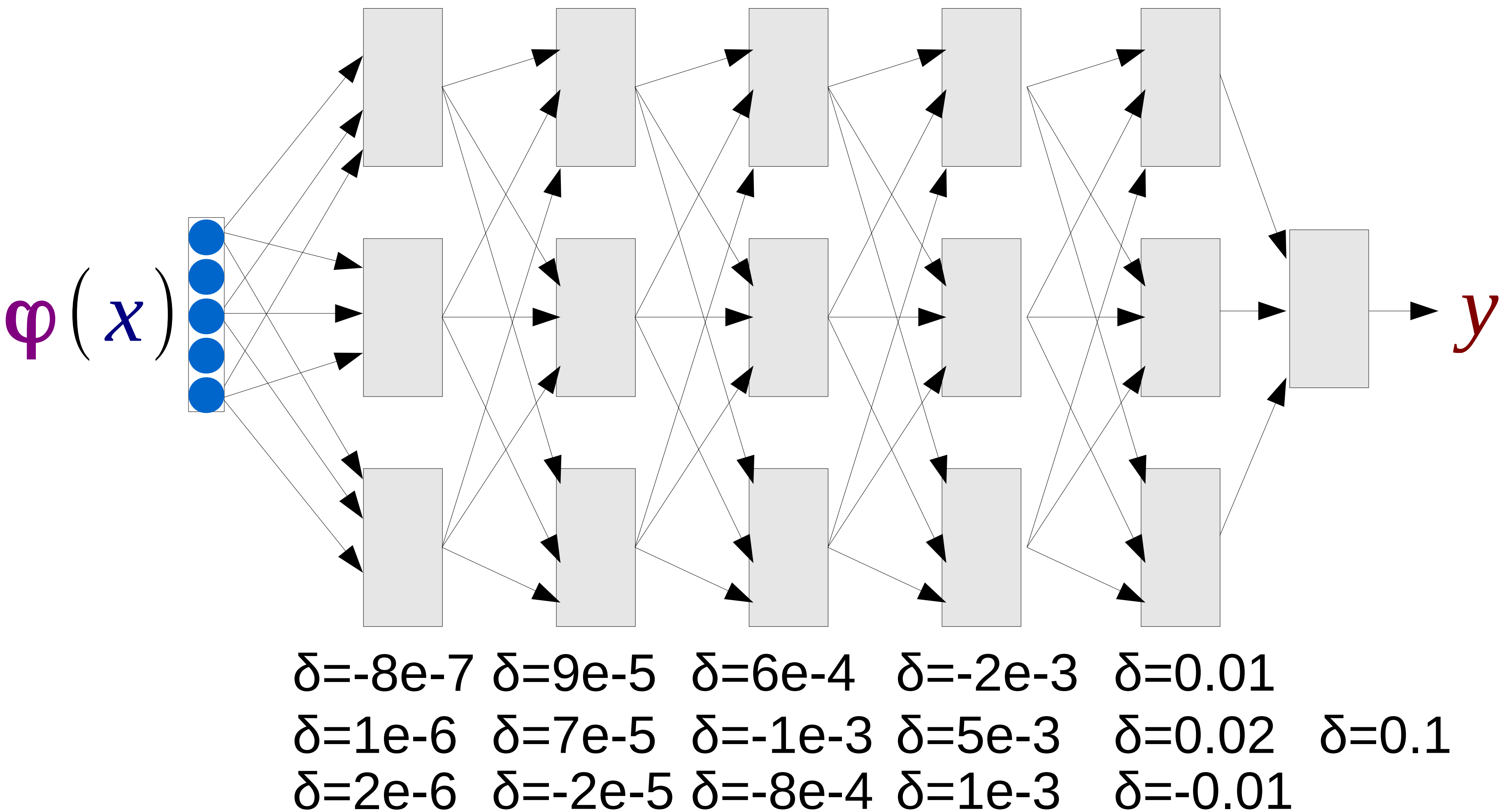
$$\frac{d P(y=1|x)}{d \boldsymbol{w_1}} = \frac{d P(y=1|x)}{d \boldsymbol{w_4 h}(\boldsymbol{x})} \frac{d \boldsymbol{w_4 h}(\boldsymbol{x})}{d h_1(\boldsymbol{x})} \frac{d h_1(\boldsymbol{x})}{d \boldsymbol{w_1}}$$

$$\underbrace{\frac{e^{\boldsymbol{w_4} \cdot \boldsymbol{h}(x)}}{(1+e^{\boldsymbol{w_4} \cdot \boldsymbol{h}(x)})^2}}_{\text{Error of next unit } (\delta_4)} \quad \underbrace{w_{1,4}}_{\text{Weight}} \quad \text{Gradient of this unit}$$

<u>In General</u>
Calculate *i* based on next units *j*:

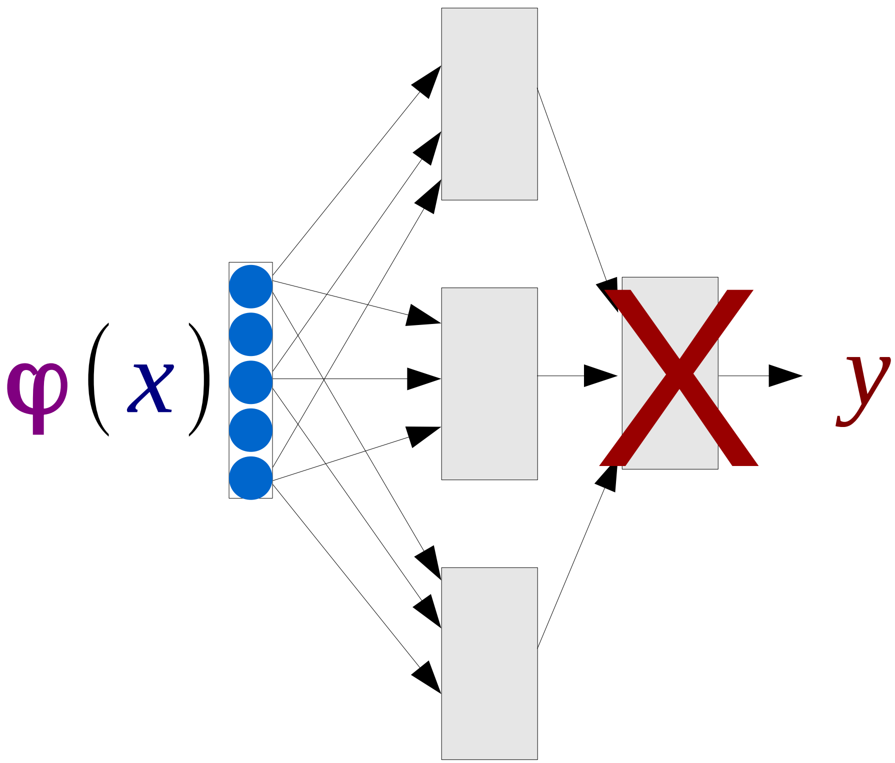
$$\frac{d P(y=1|\boldsymbol{x})}{\boldsymbol{w_i}} = \frac{d h_i(\boldsymbol{x})}{d \boldsymbol{w_i}} \sum_j \delta_j w_{i,j}$$

BP in Deep Networks: Vanishing Gradients



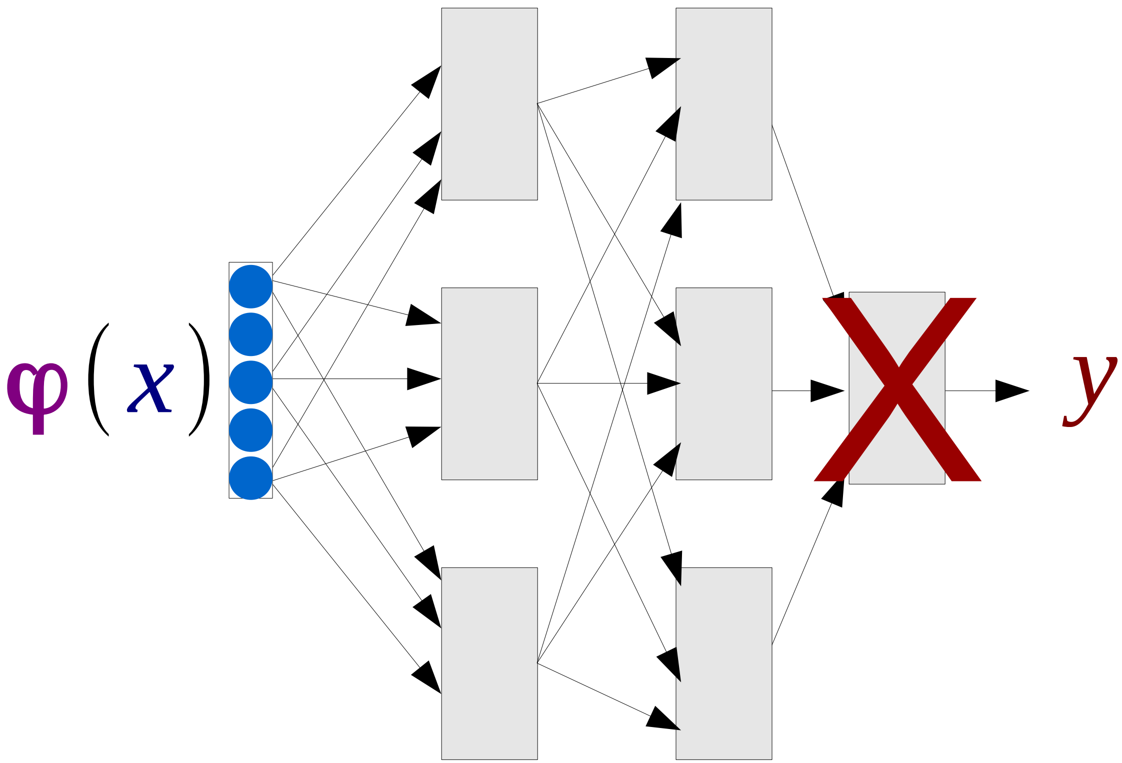
- Exploding gradient as well

Layerwise Training



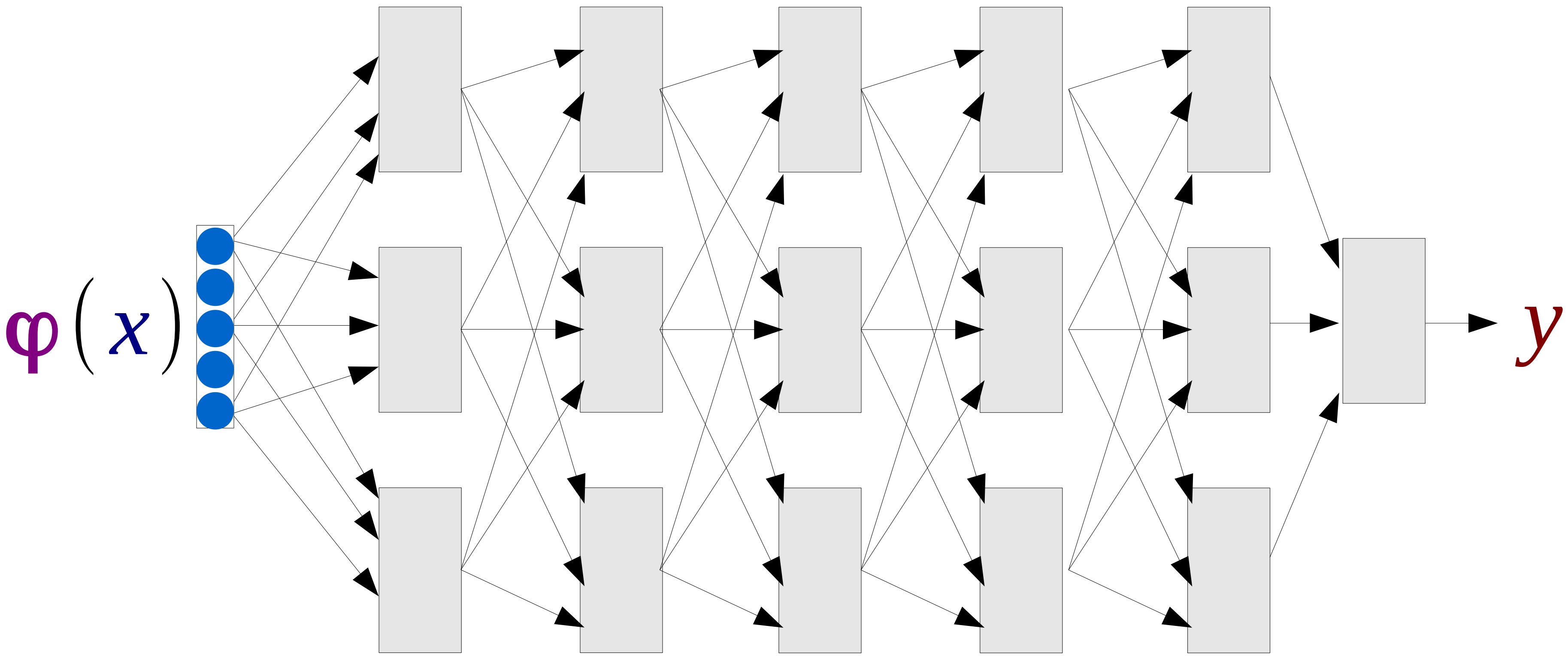
- Train one layer at a time

Layerwise Training



- Train one layer at a time

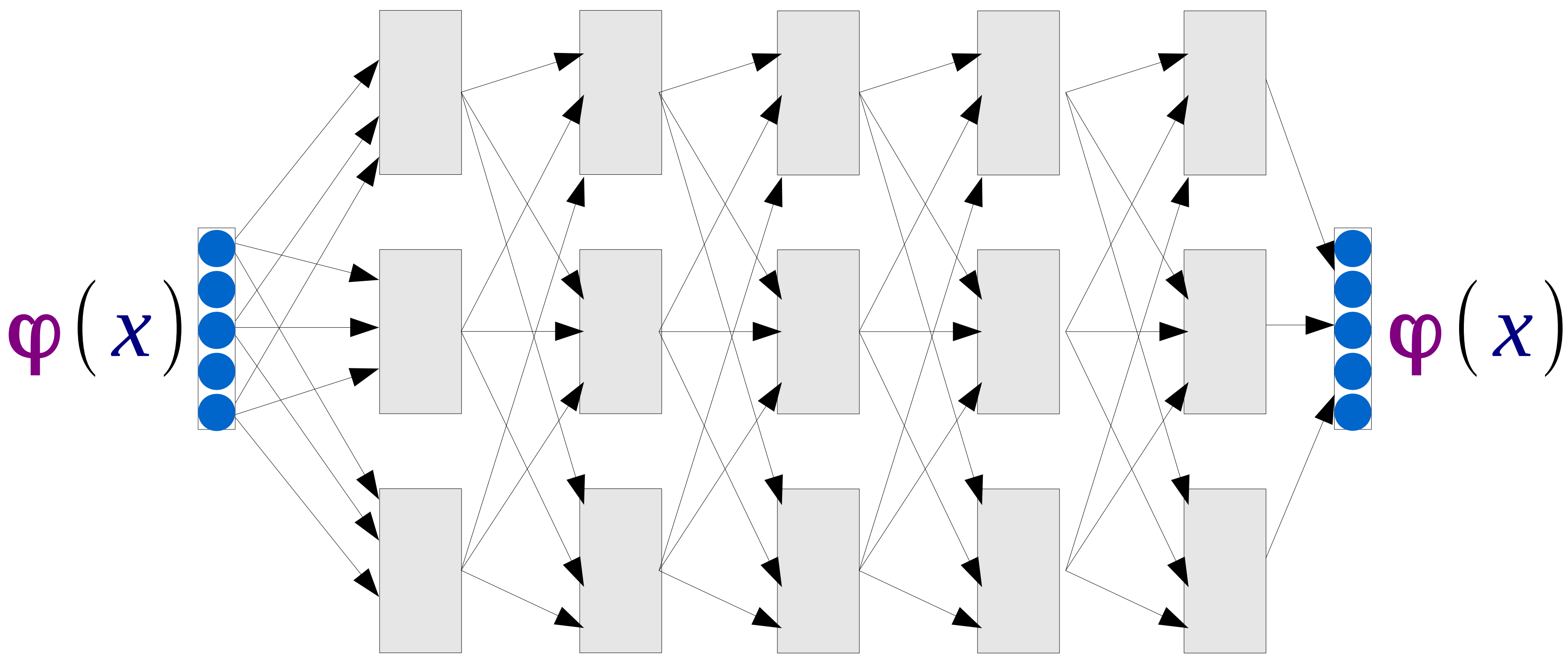
Layerwise Training



- Train one layer at a time

Autoencoders

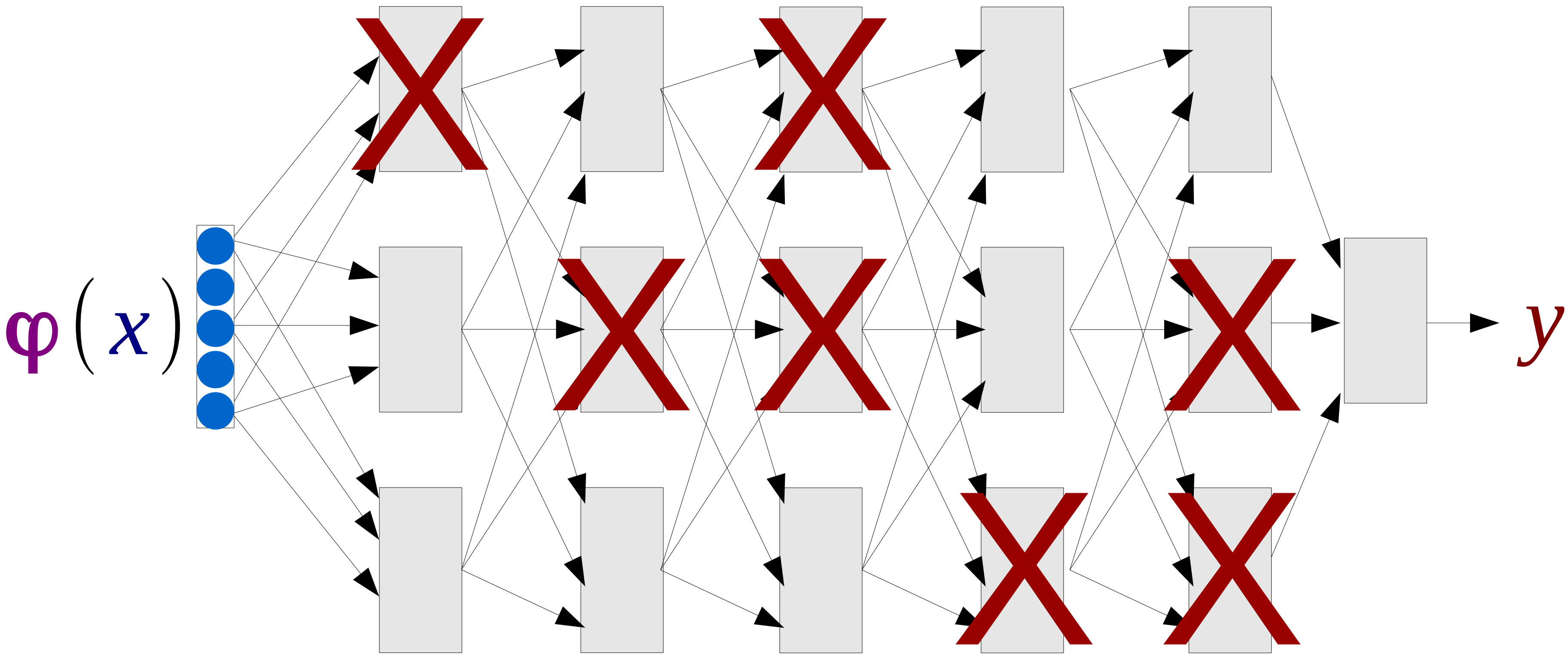
- Initialize the NN by training it to reproduce itself



- Advantage: No need for labels y!

Dropout

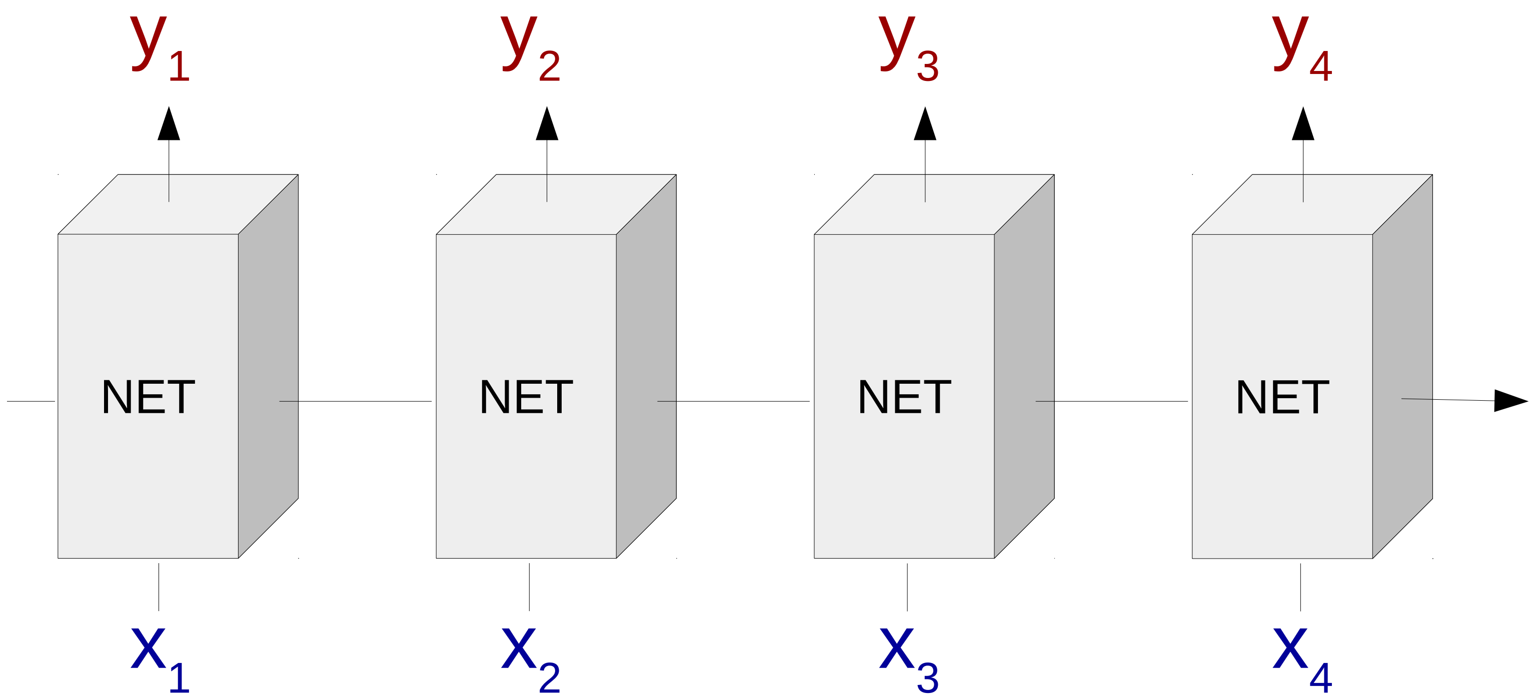
- Problem: Overfitting



- Deactivate part of the network on each update
- Can be seen as an ensemble method

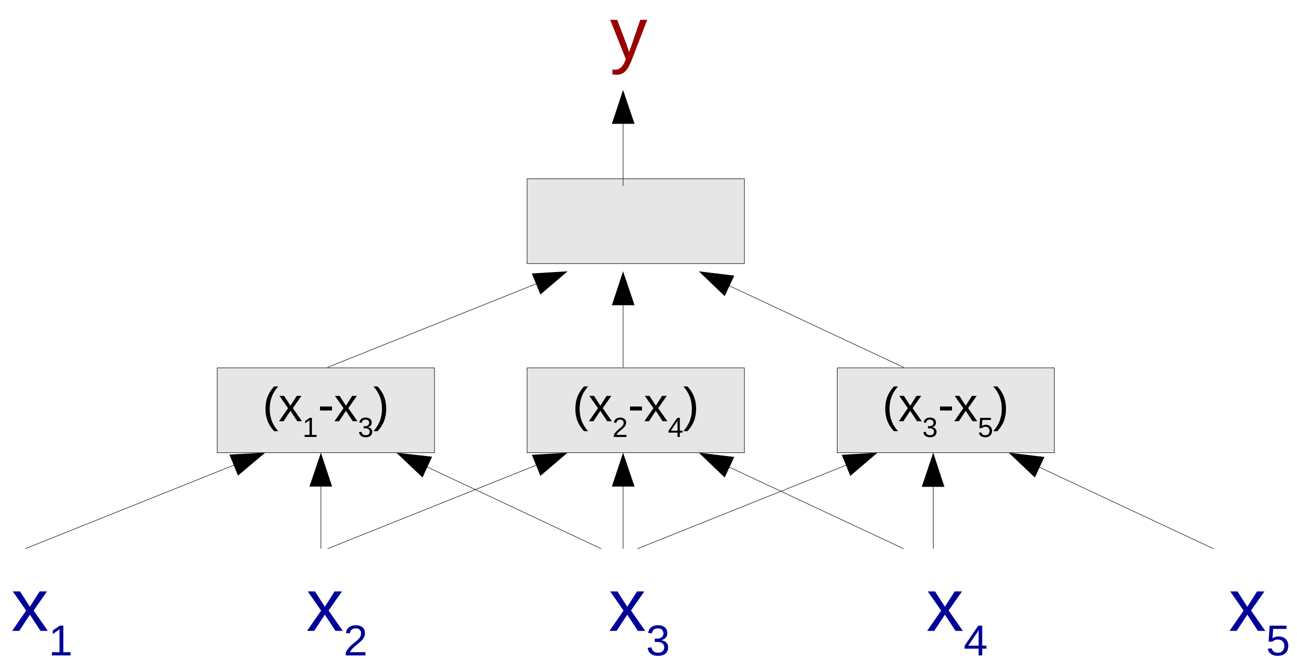
Network Architectures

Recurrent Neural Nets



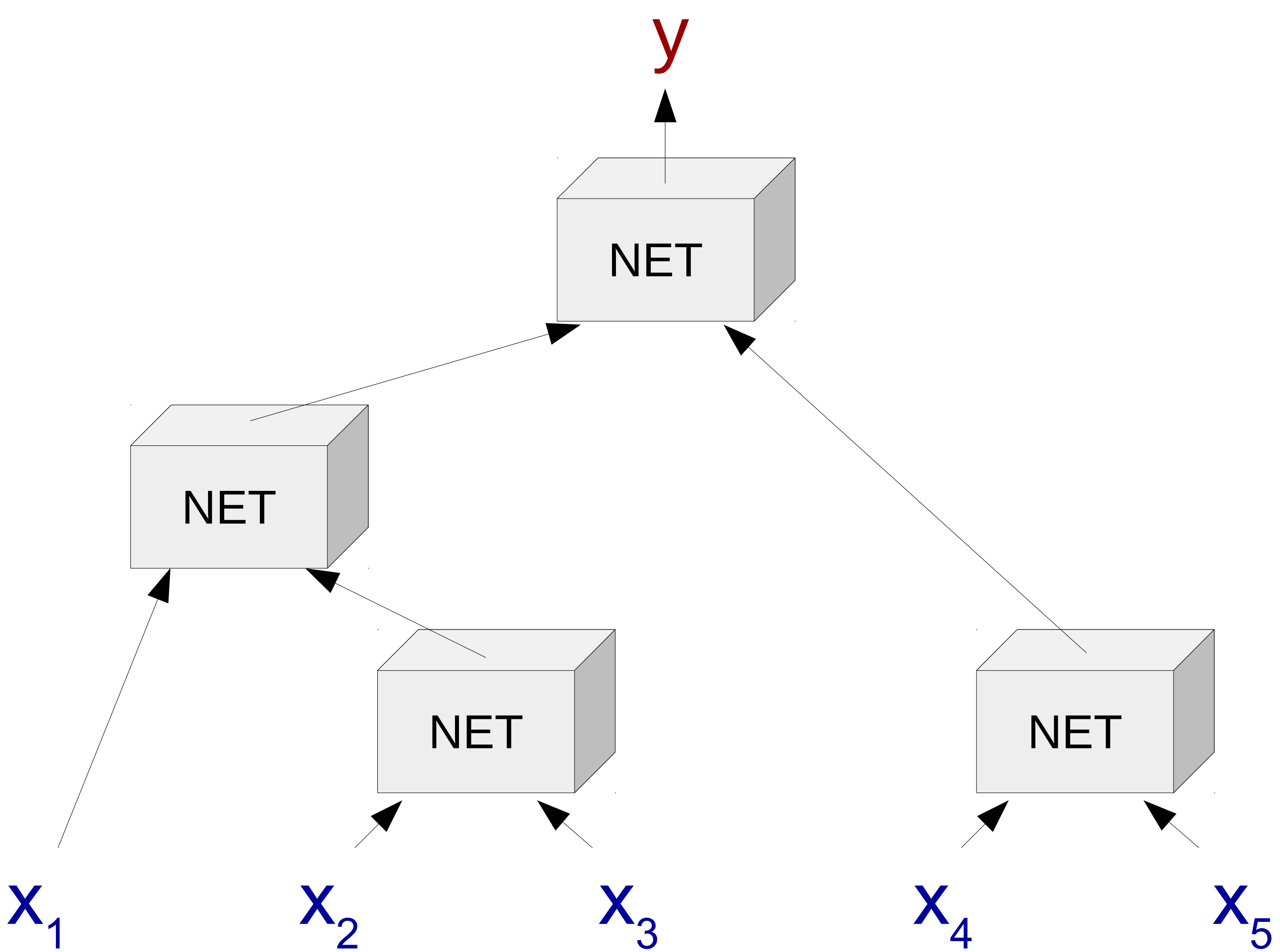
- Good for modeling sequence data
- e.g. Speech

Convolutional Neural Nets



- Good for modeling data with spatial correlation
- e.g. Images

Recursive Neural Nets



- Good for modeling data with tree structures
- e.g. Language

Other Topics

Other Topics

- Deep belief networks
  - Restricted Boltzmann machines
  - Contrastive estimation
- Generating with Neural Nets
- Visualizing internal states
- Batch/mini-batch update strategies
- Long short-term memory
- Learning on GPUs
- Tools for implementing deep learning