

Neural Networks II

CMSC 422

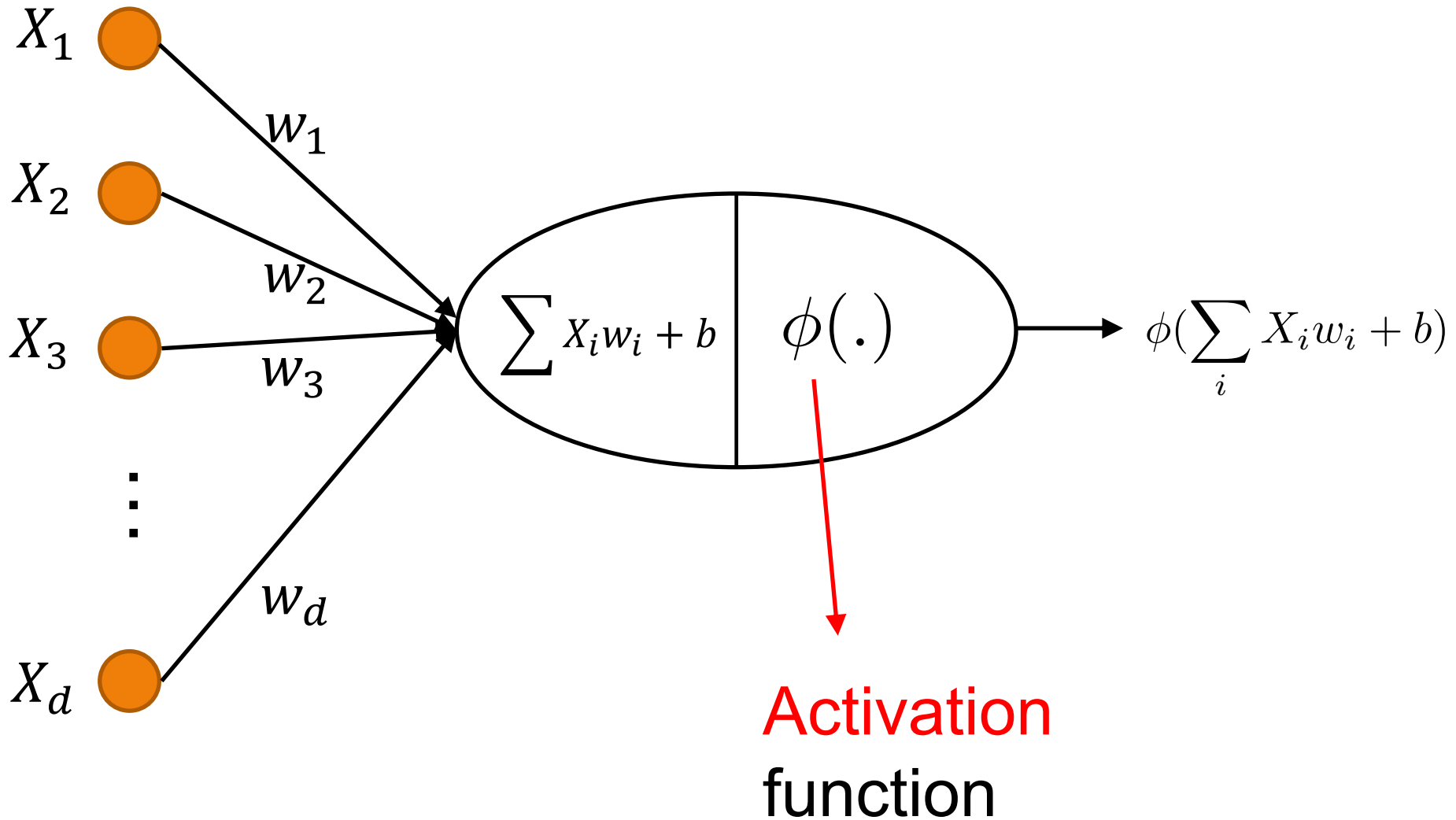
SOHEIL FEIZI

sfeizi@cs.umd.edu

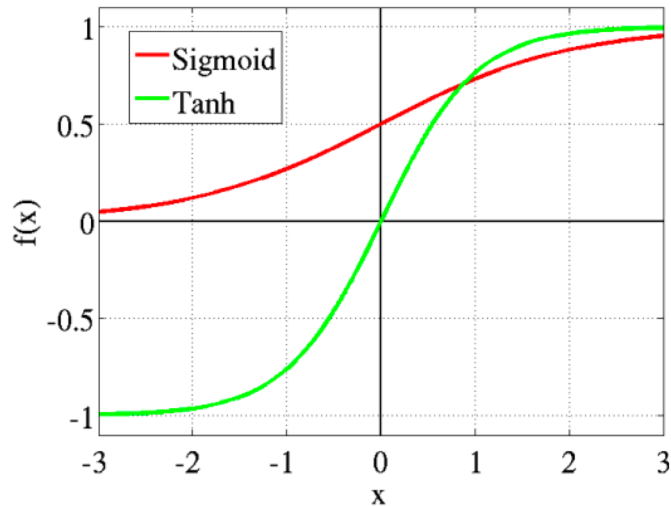
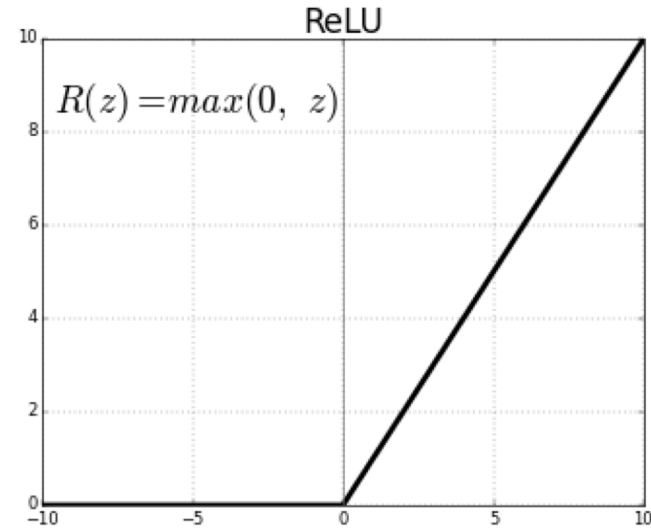
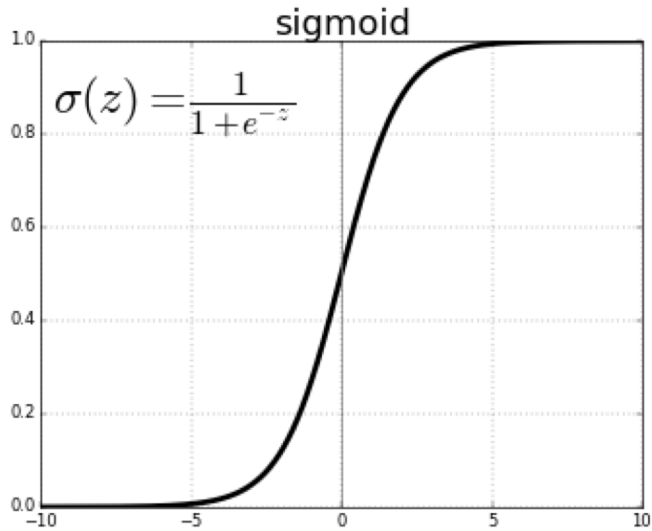
Today's topics

- Neural Networks for the regression problem
- Back Propagation: SGD+ Chain rule

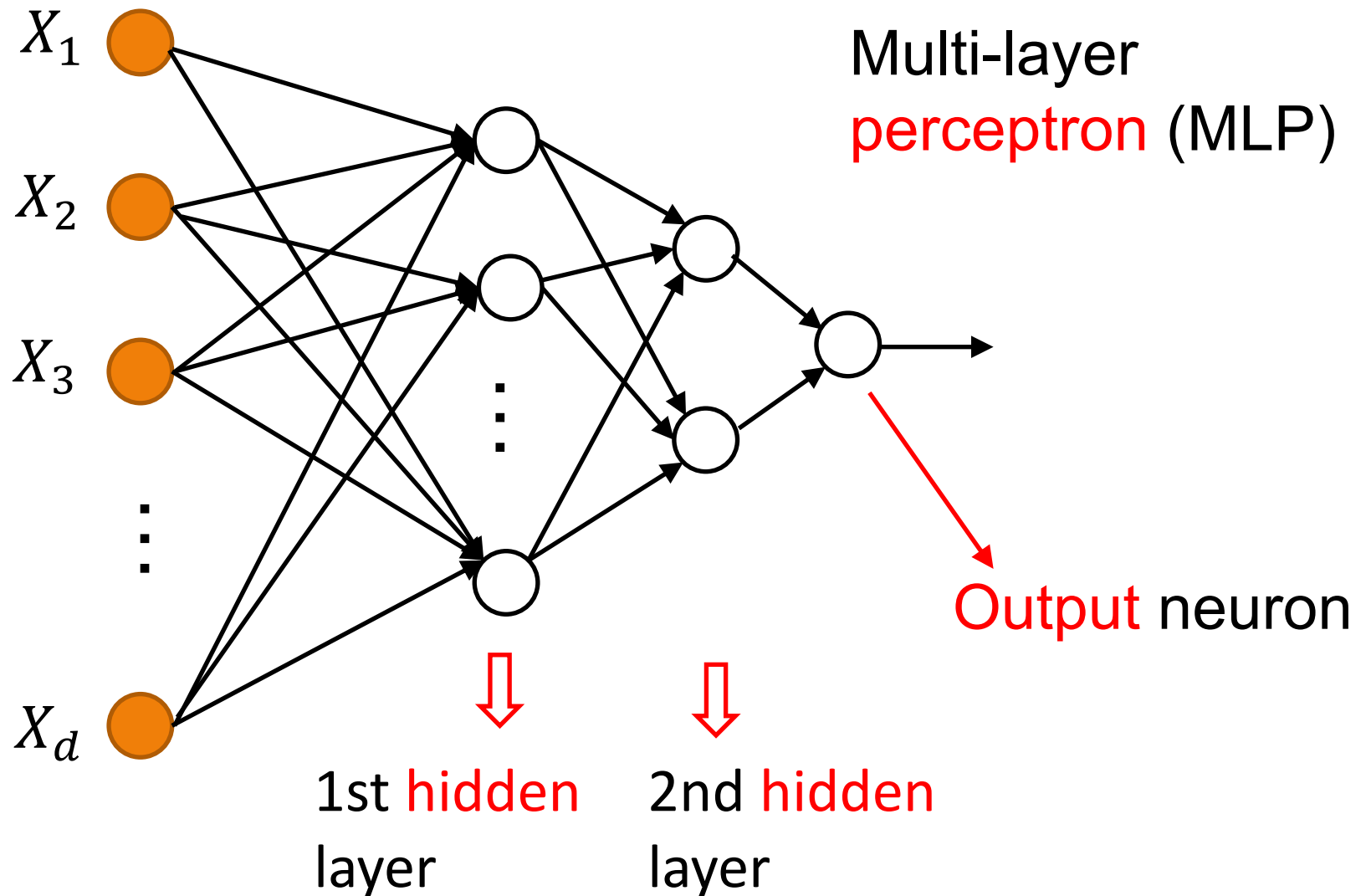
Neural Unit



Popular Activation Functions



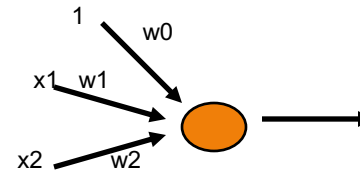
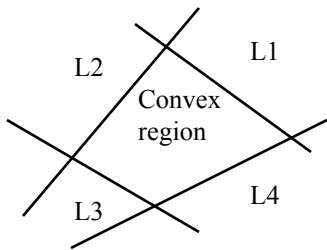
Multi-Layer Neural Network



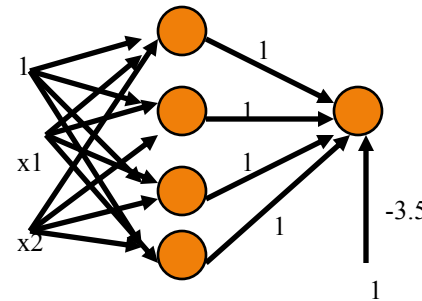
Types of decision regions

$$w_0 + w_1x_1 + w_2x_2 > 0$$

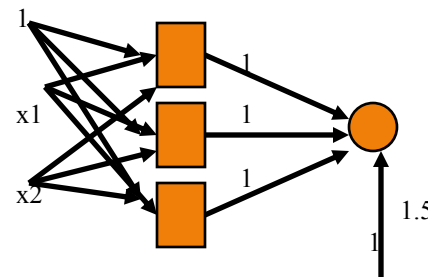
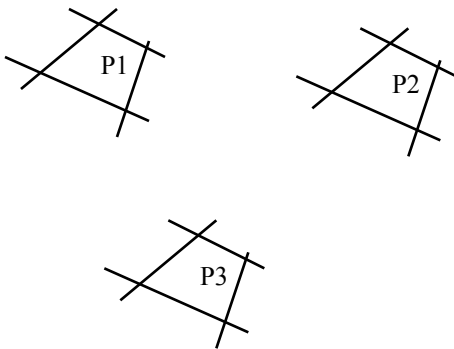
$$w_0 + w_1x_1 + w_2x_2 < 0$$



Network with a single node



One-hidden layer network that realizes the convex region: each hidden node realizes one of the lines bounding the convex region

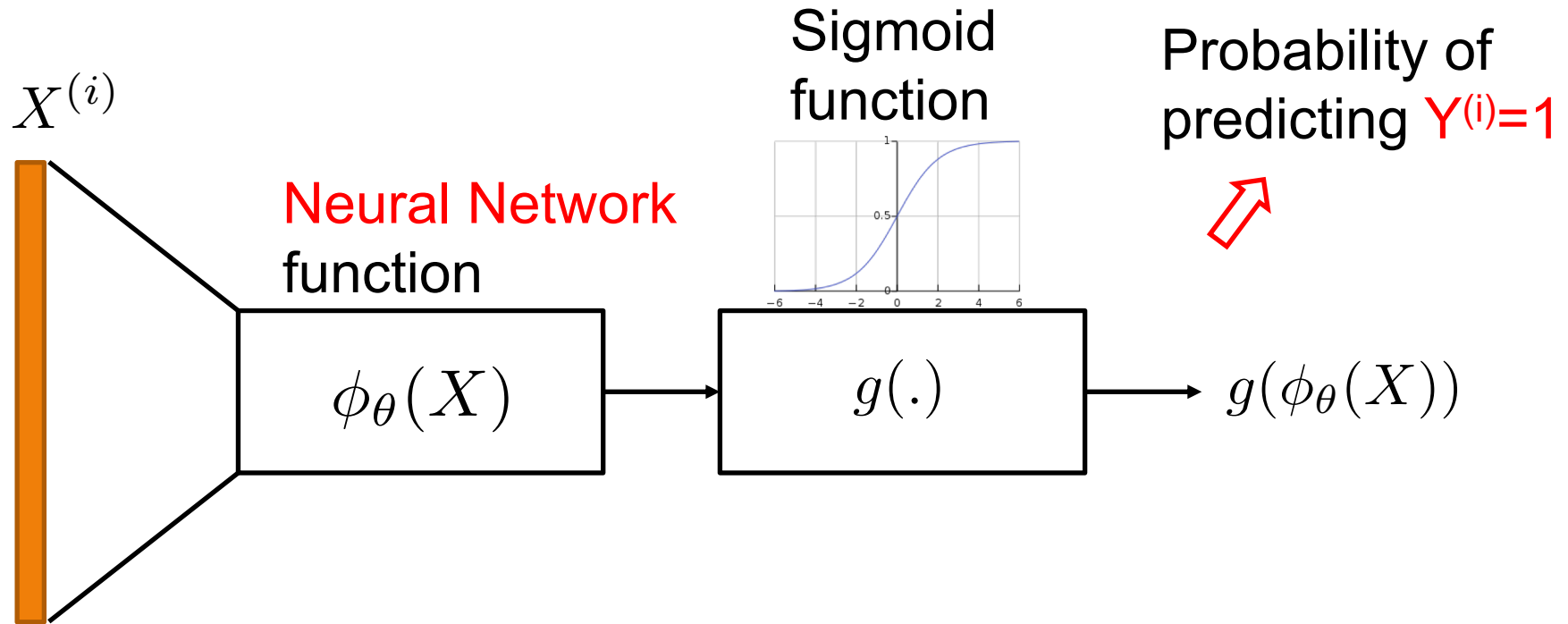


two-hidden layer network that realizes the union of three convex regions: each box represents a one hidden layer network realizing one convex region

Discussion

- 2-layer perceptron lets us
 - Discover more complex decision boundaries than perceptron
 - Learn combinations of features that are useful for classification
- Key design question: How many hidden units?
 - More hidden units yield more complex functions
 - Fewer hidden units requires fewer examples to train

Classification using Neural Network



What is θ ?

Compute model parameters using cross-entropy loss opt:

$$\max_{\theta} \sum_{i=1}^N Y^{(i)} \log g(\phi_{\theta}(X^{(i)})) + (1 - Y^{(i)}) \log(1 - g(\phi_{\theta}(X^{(i)})))$$

Regression Problem

- Learning a functional relationship about a real-valued number, i.e., when y is tomorrow's temperature.

- Training data: $\{(X^{(i)}, Y^{(i)})\}$

$$Y^{(i)} \in \mathbb{R}$$

Regression examples

Stock market



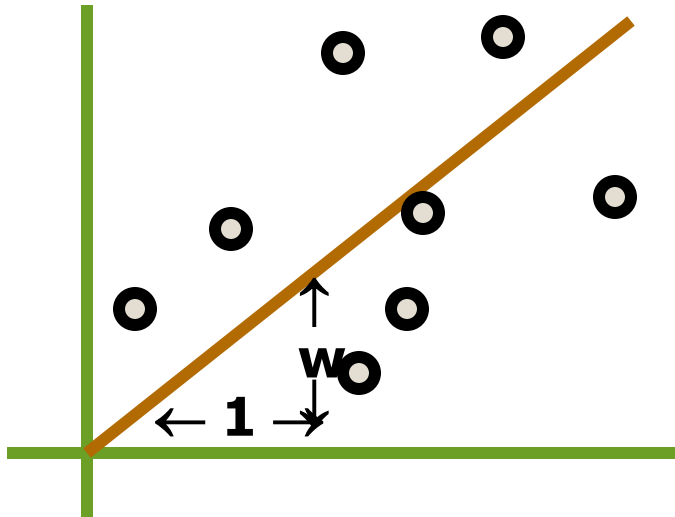
Weather prediction



Temperature
72° F

Predict the temperature at any given location

Linear Regression



DATASET

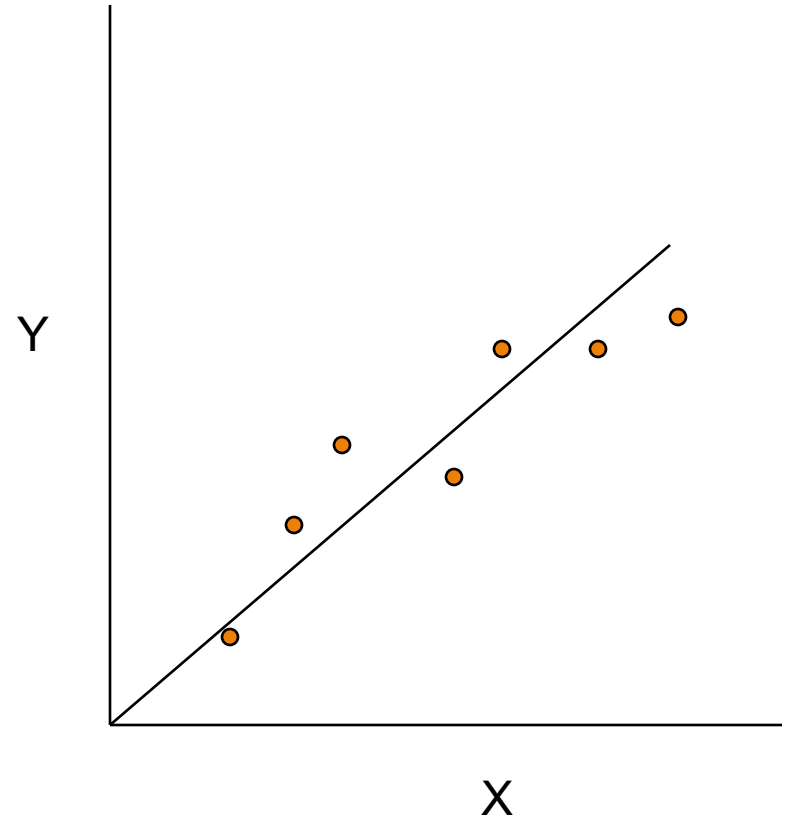
inputs	outputs
$x_1 = 1$	$y_1 = 1$
$x_2 = 3$	$y_2 = 2.2$
$x_3 = 2$	$y_3 = 2$
$x_4 = 1.5$	$y_4 = 1.9$
$x_5 = 4$	$y_5 = 3.1$

Simplest case: $\text{Out}(x) = w^t x + b$ for some unknown w, b .

Given the data, we can estimate w, b .

Linear regression

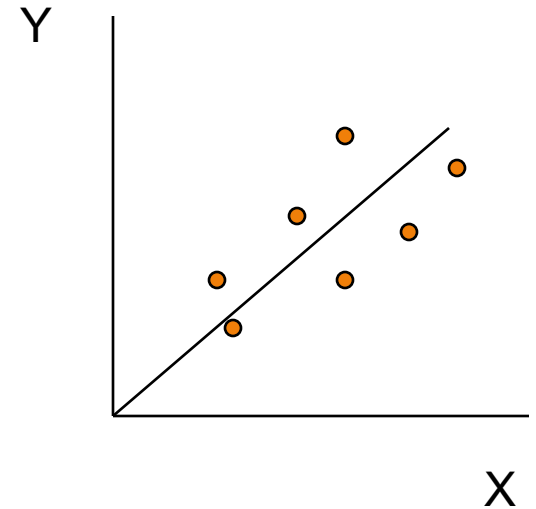
- Given an input x we would like to compute an output y
- For example:
 - Predict height from age
 - Predict Google's price from Apple's price
 - Predict distance from wall from sensors
 - BMI based on height and weight
 - Papers published based on age



Linear regression

Our goal is to estimate w, b from a training dataset

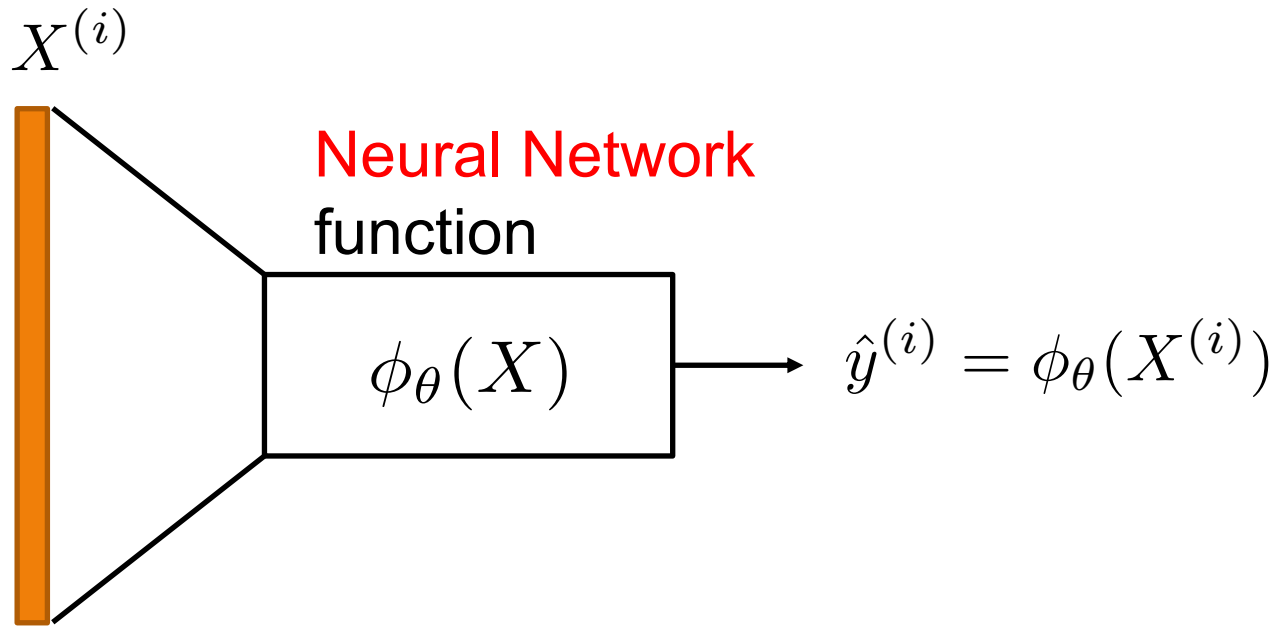
Optimization: minimize **squared** error (least squares)



$$\min_{w,b} \sum_{i=1}^N (y^{(i)} - \hat{y}^{(i)})^2$$
$$\hat{y}^{(i)} = w^t X^{(i)} + b$$

Why **quadratic**?

Nonlinear Regression using Neural Network



What is θ ?

Compute model parameters using **quadratic** loss opt:

$$\min_{\theta} \sum_{i=1}^N (y^{(i)} - \phi_{\theta}(X^{(i)}))^2$$

Loss functions

Classification Problem

- ❖ Hinge loss
- ❖ Cross entropy loss

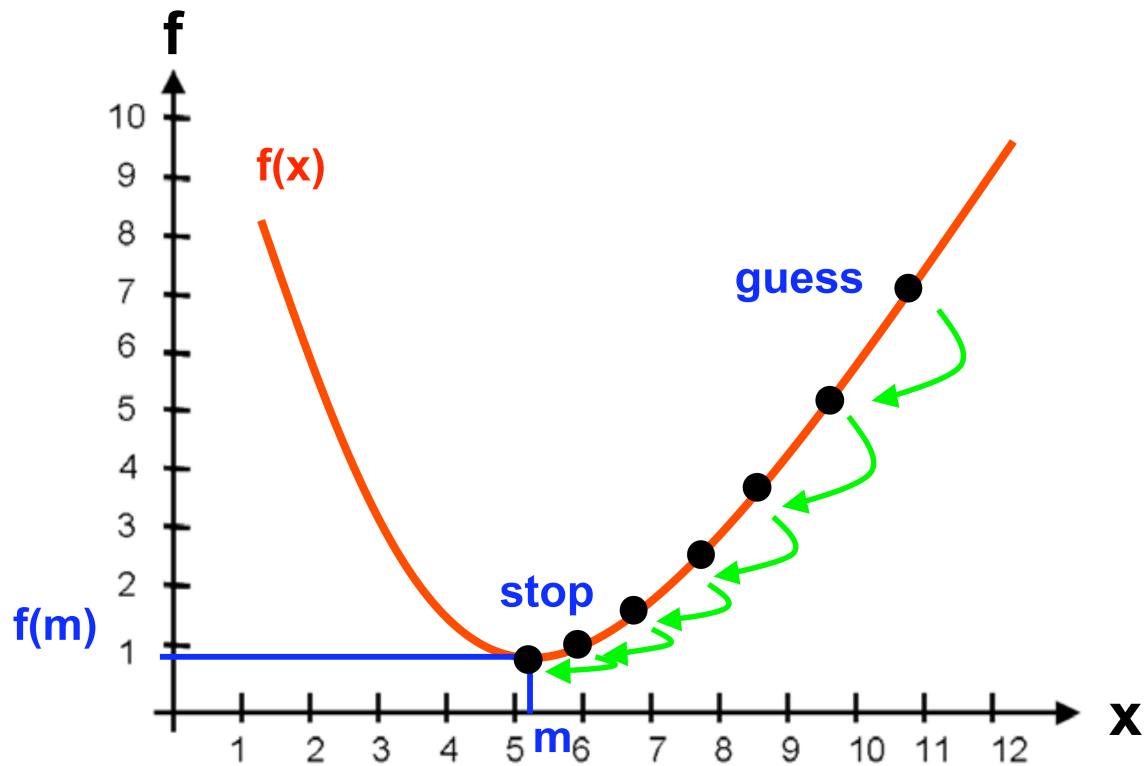
Regression Problem

- ❖ Quadratic loss

How to find optimal model parameters?

Stochastic Gradient Descent

If the objective of optimization is **convex**



Stochastic Gradient Descent

If the objective of optimization is **non-convex**



In practice, SGD still performs well. **Why?**

Stochastic Gradient Descent

What do we need to be able to use SGD in deep learning?

Computation of the **gradient** of the loss function with respect to model parameters

Training a Neural Network

The Backpropagation Algorithm

=

Gradient descent + Chain rule

Review of Chain Rule

X, Y, Z in \mathbb{R}

$Y = g(X)$

$Z = f(Y)$

$$\frac{dZ}{dX} = \frac{dZ}{dY} \frac{dY}{dX}$$

Practice:

$Y = X^2$ and $Z = 2Y + 1$

compute

$$\frac{dZ}{dX} = ?$$

Review of Chain Rule

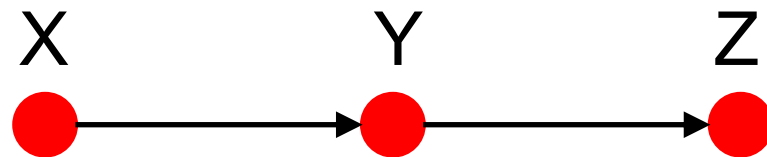
X, Y, Z in \mathbb{R}

$Y = g(X)$

$Z = f(Y)$

$$\frac{dZ}{dX} = \frac{dZ}{dY} \frac{dY}{dX}$$

Graph representation:



This is NOT a neural network

Review of Chain Rule

$$X \in \mathbb{R}^n, Y \in \mathbb{R}^m, Z \in \mathbb{R}$$

$$Y=g(X), Z=f(Y)$$

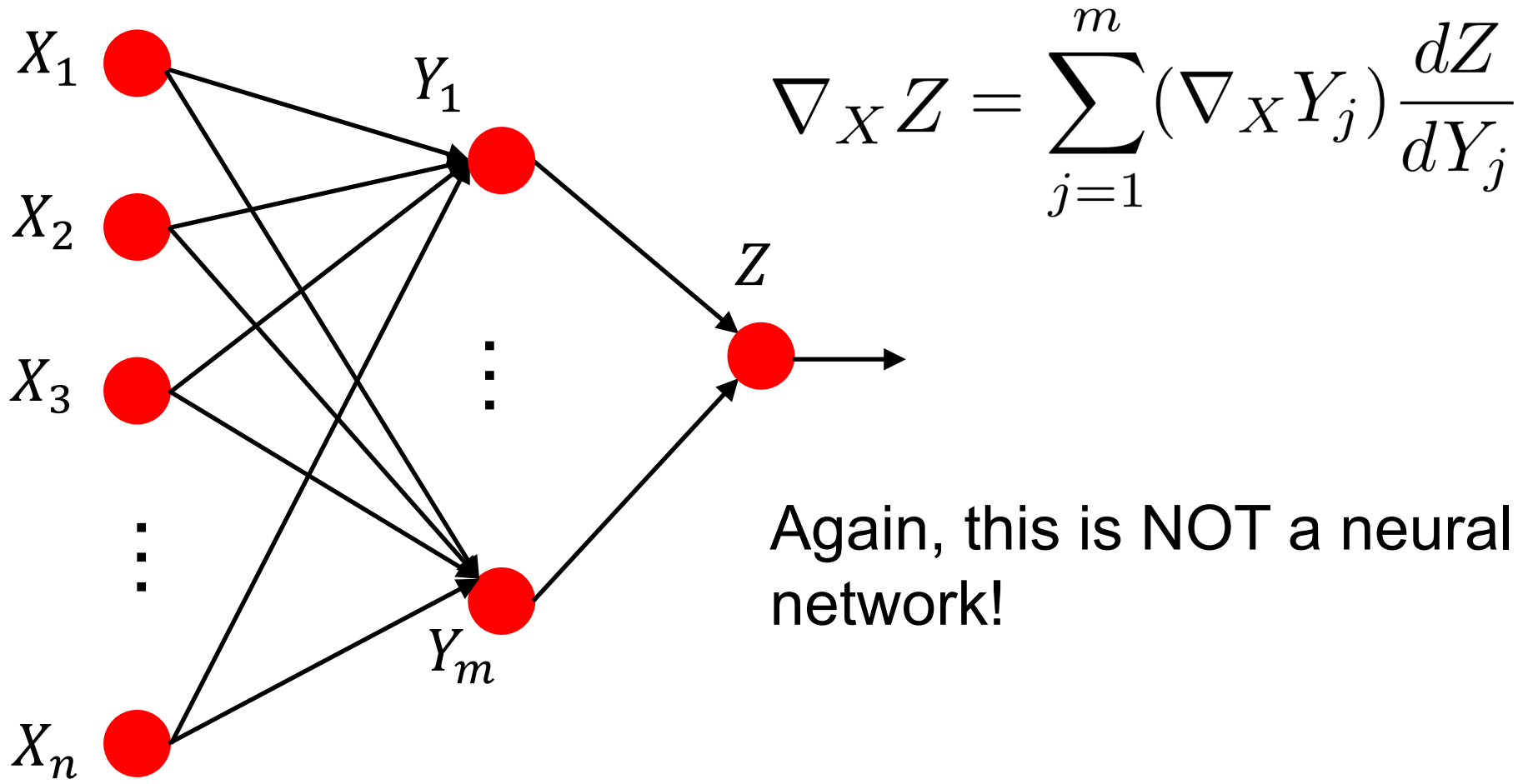
$$\nabla_X Z = \underbrace{\left(\frac{dY}{dX}\right)}_{\text{Jacobian matrix}} \nabla_Y Z = \sum_{j=1}^m (\nabla_X Y_j) \frac{dZ}{dY_j}$$

Jacobian matrix

Practice: $Y_1=X_1+X_2^2, Y_2=X_2+X_3^2, Z=Y_1^2+Y_2^2$

Compute $\nabla_X Z$

Graph Representation

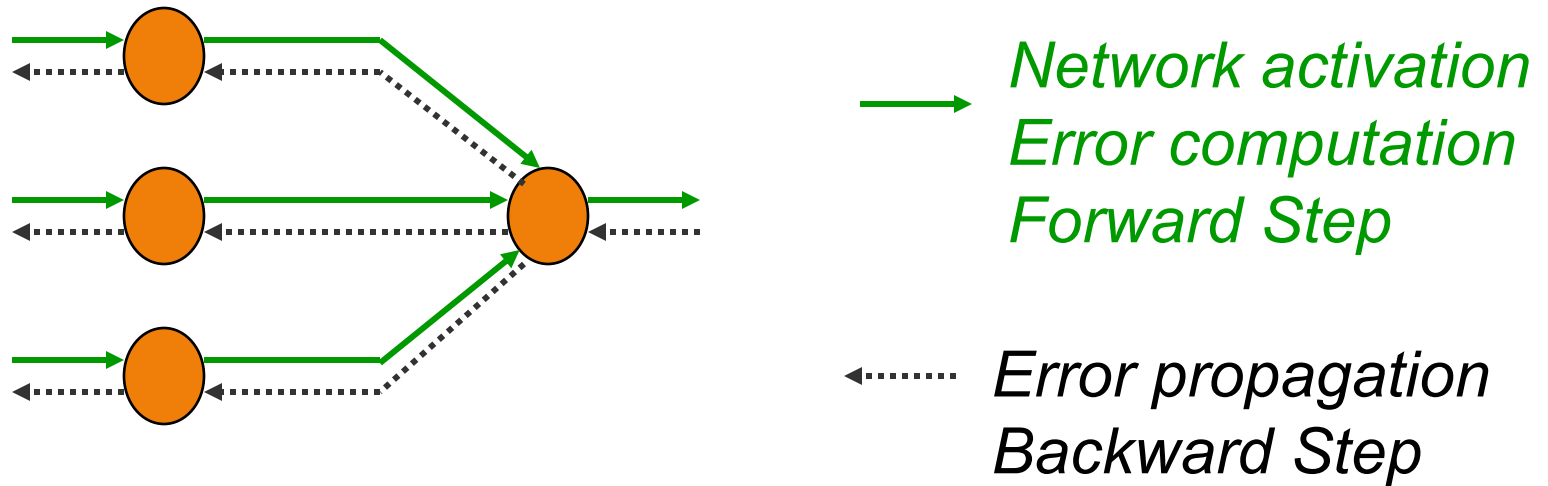


Training: Backpropagation Algorithm

- Searches for weight values that **minimize the total error of the network** over the set of training examples.
- **Repeated** procedures of the following two passes:
 - **Forward pass:** Compute the **outputs** of all units in the network, and the **error** of the output layers.
 - **Backward pass:** The network error is used for updating the weights.
 - Starting at the output layer, **the error is propagated backwards through the network, layer by layer.** This is done by recursively computing the local gradient of each neuron.

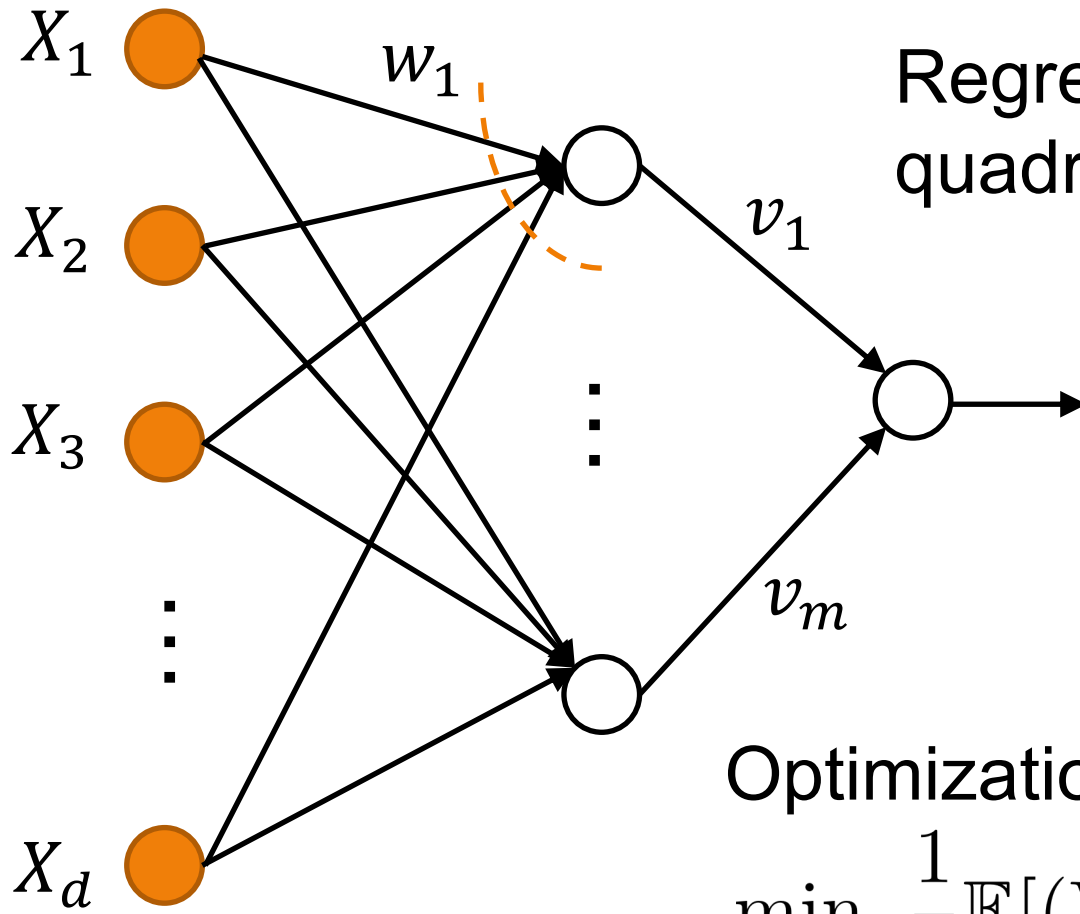
The Backpropagation Algorithm

Back-propagation training algorithm, illustrated:



Backprop adjusts the weights of the NN in order to minimize the network total mean squared error.

Example: two layer NN



Regression problem with quadratic loss function

Optimization:

$$\min_{W, v} \frac{1}{2} \mathbb{E} \left[\left(Y - \sum_{i=1}^m \underbrace{v_i \phi(w_i^t X)}_{\phi_i} \right)^2 \right]$$

Gradient of objective w.r.t. output layer weights v

$$\nabla_{v_i} L = \mathbb{E} \left[(Y - \sum_{i=1}^m v_i \phi(w_i^t X)) \phi(w_i^t X) \right]$$

Error term:
 $Y - \hat{Y}$

activation of hidden
unit i

Gradient of objective w.r.t. hidden unit weights w_i

$$\nabla_L w_1 = \left(\frac{dL}{d\phi_1} \right) \nabla_{w_1} \phi_1$$

$$= -\mathbb{E} \left[\left(Y - \sum_{i=1}^m v_i \phi(w_i^t X) \right) v_1 \right]$$

$$= \mathbb{E} \left[\phi'(w_1^t X) X \right]$$

Multi-Label Classification

Q: how to extend our method for multi-label classification?

Recall: Multi-Label Classification, Logistic Regression

