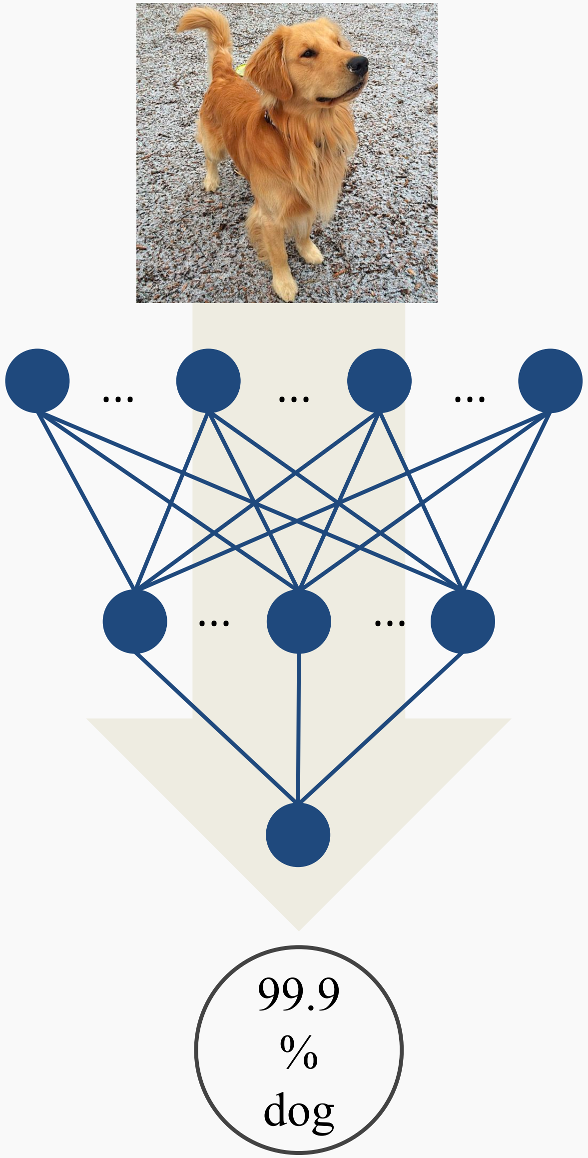
Lecture 5: Review of Neural Networks

CS109B Data Science 2

Pavlos Protopapas and Mark Glickman

Artificial Neural Networks

1. Machine Learning Algorithm
2. Very simplified parametric models of our brain
3. Networks of basic processing units: neurons
4. Neurons store information which has to be learned (the weights or the state for RNNs)
5. Many types of architectures to solve different tasks
6. They can scale to massive data



Outline

Anatomy of a NN

Design choices

Learning

Review of Feed Forward Artificial Neural Networks

Anatomy of a NN

Design choices

- Activation function
- Loss function
- Output units
- Architecture

Learning

Review of Feed Forward Artificial Neural Networks

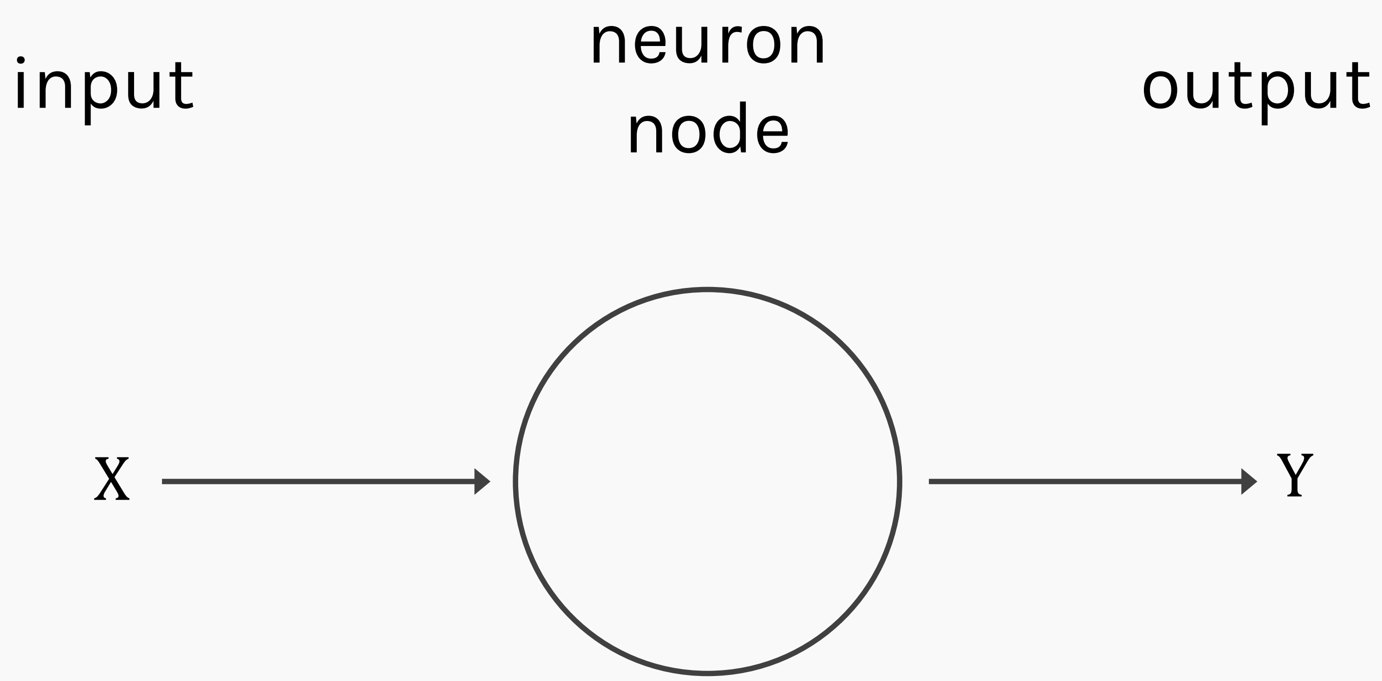
Anatomy of a NN

Design choices

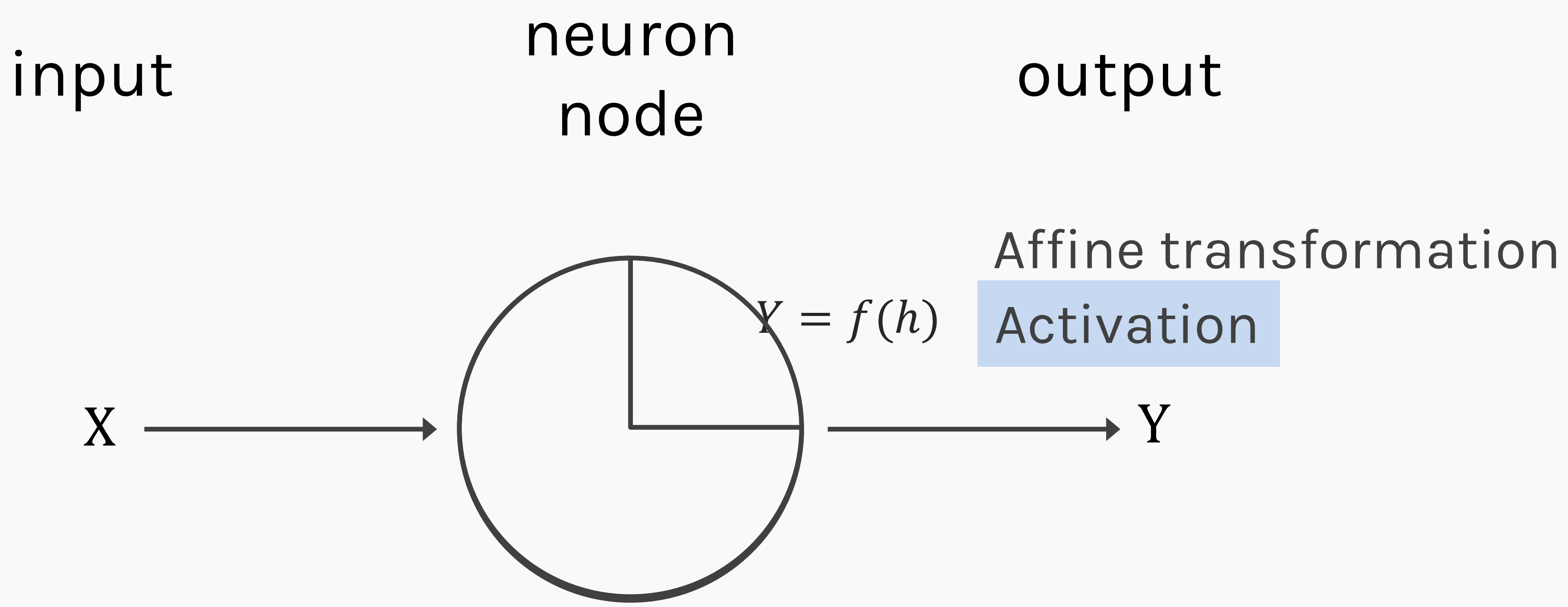
- Activation function
- Loss function
- Output units
- Architecture

Learning

Anatomy of artificial neural network (ANN)



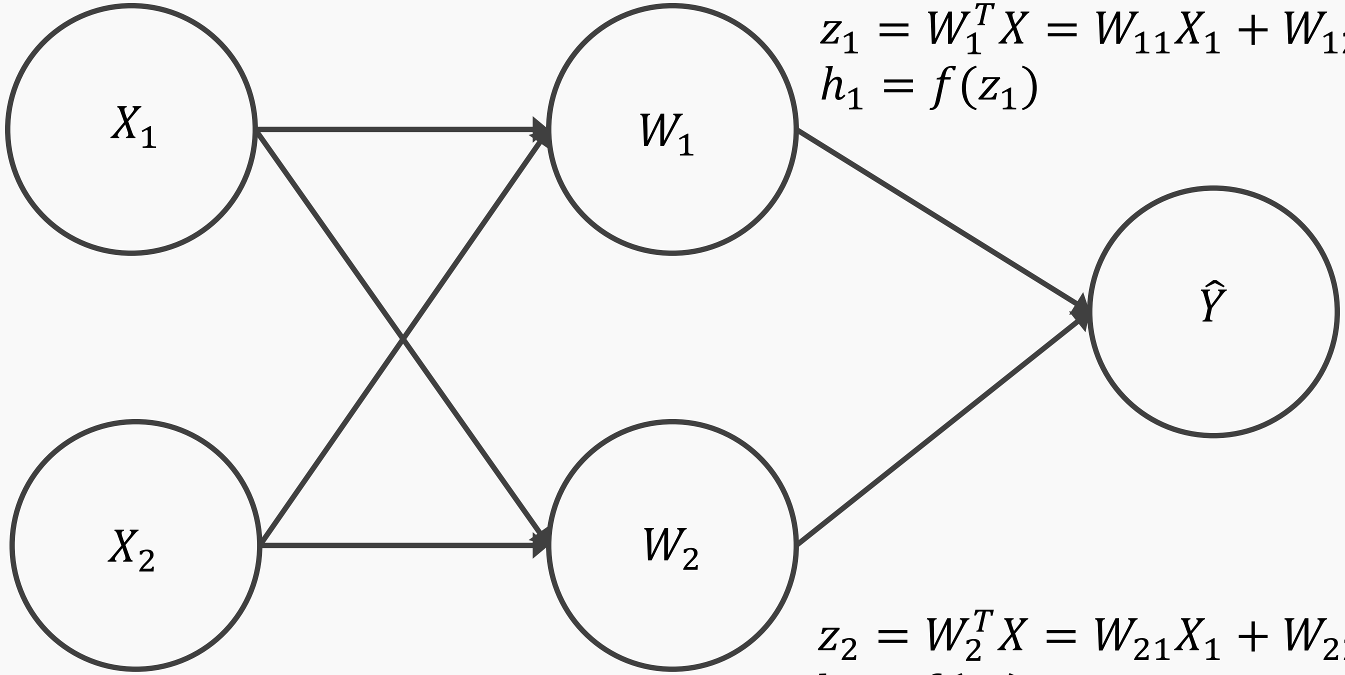
Anatomy of artificial neural network (ANN)



We will talk later about the choice of activation function.

Anatomy of artificial neural network (ANN)

Input layer hidden layer output layer



$z_1 = W_1^T X = W_{11}X_1 + W_{12}X_2 + W_{10}$
$h_1 = f(z_1)$

$z_2 = W_2^T X = W_{21}X_1 + W_{22}X_2 + W_{20}$
$\mathrm{h}_2 = f(z_2)$

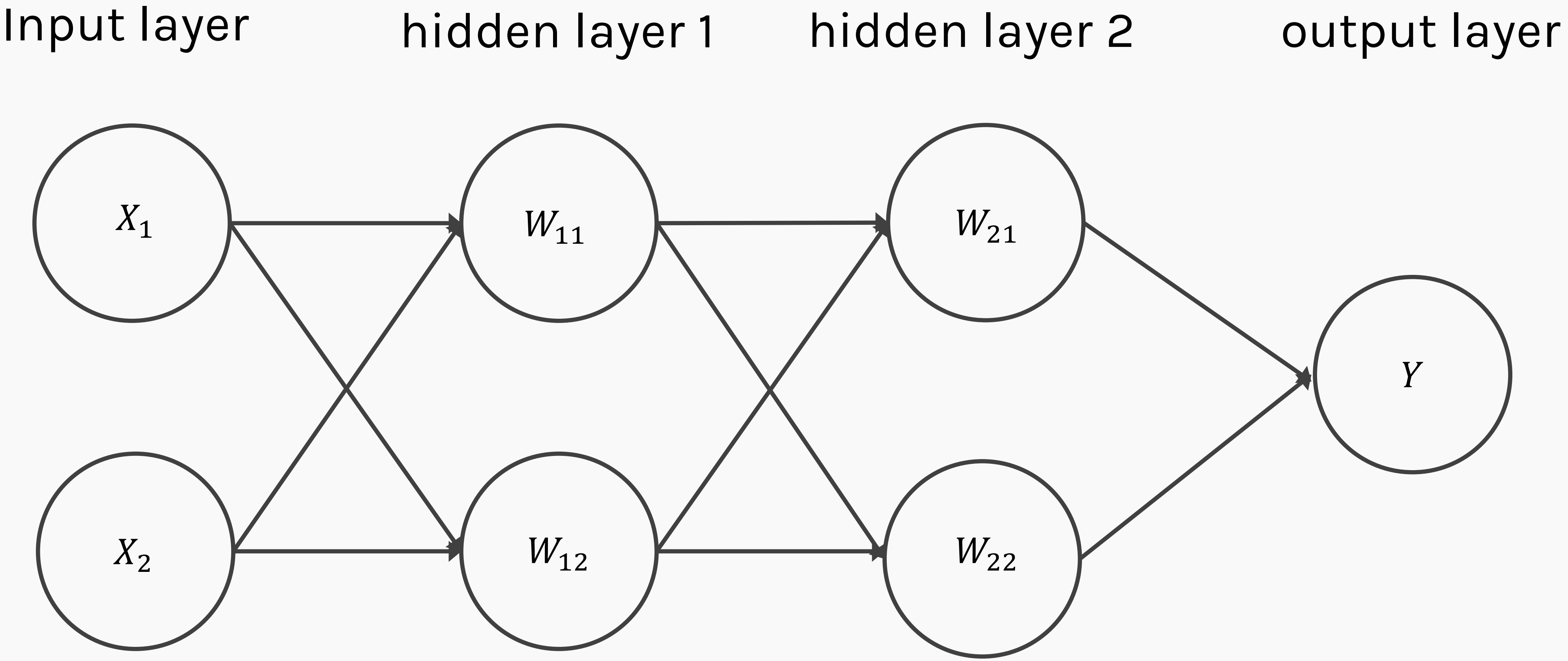
$\hat{Y} = g(h_1, h_2) \longrightarrow J = \mathcal{L}(\hat{Y}, Y)$

Output function

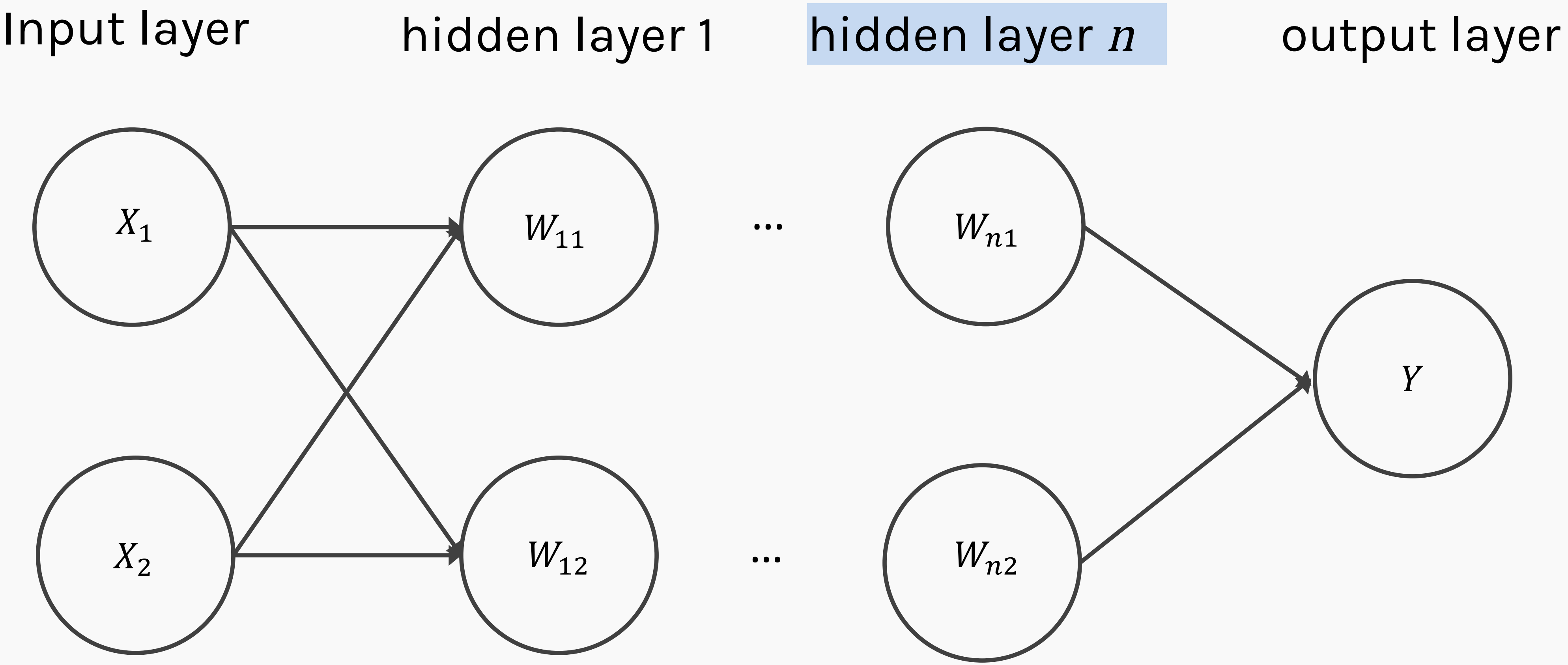
Loss function

We will talk later about the choice of the output layer and the loss function.

Anatomy of artificial neural network (ANN)

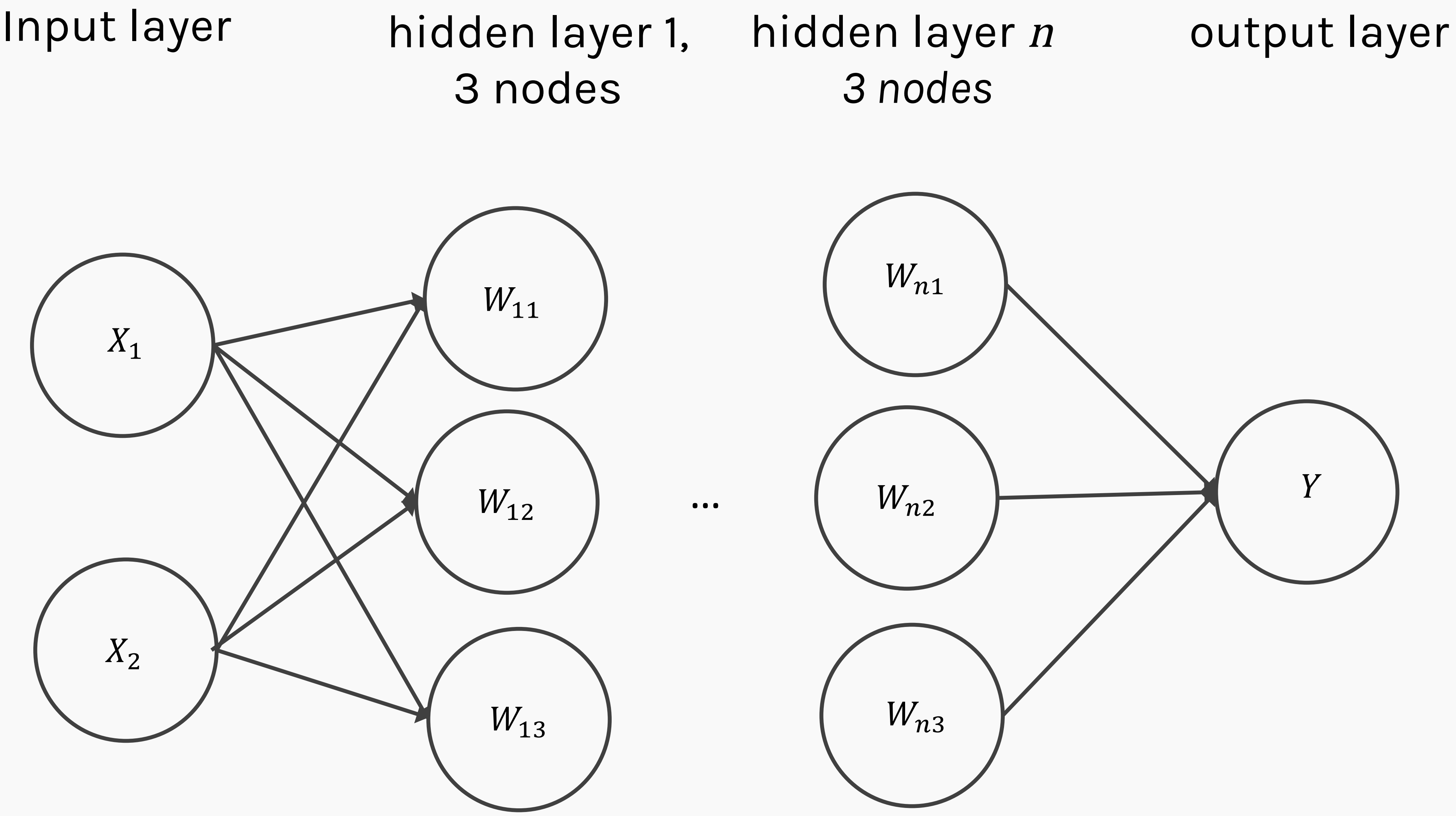


Anatomy of artificial neural network (ANN)



We will talk later about the choice of the number of layers.

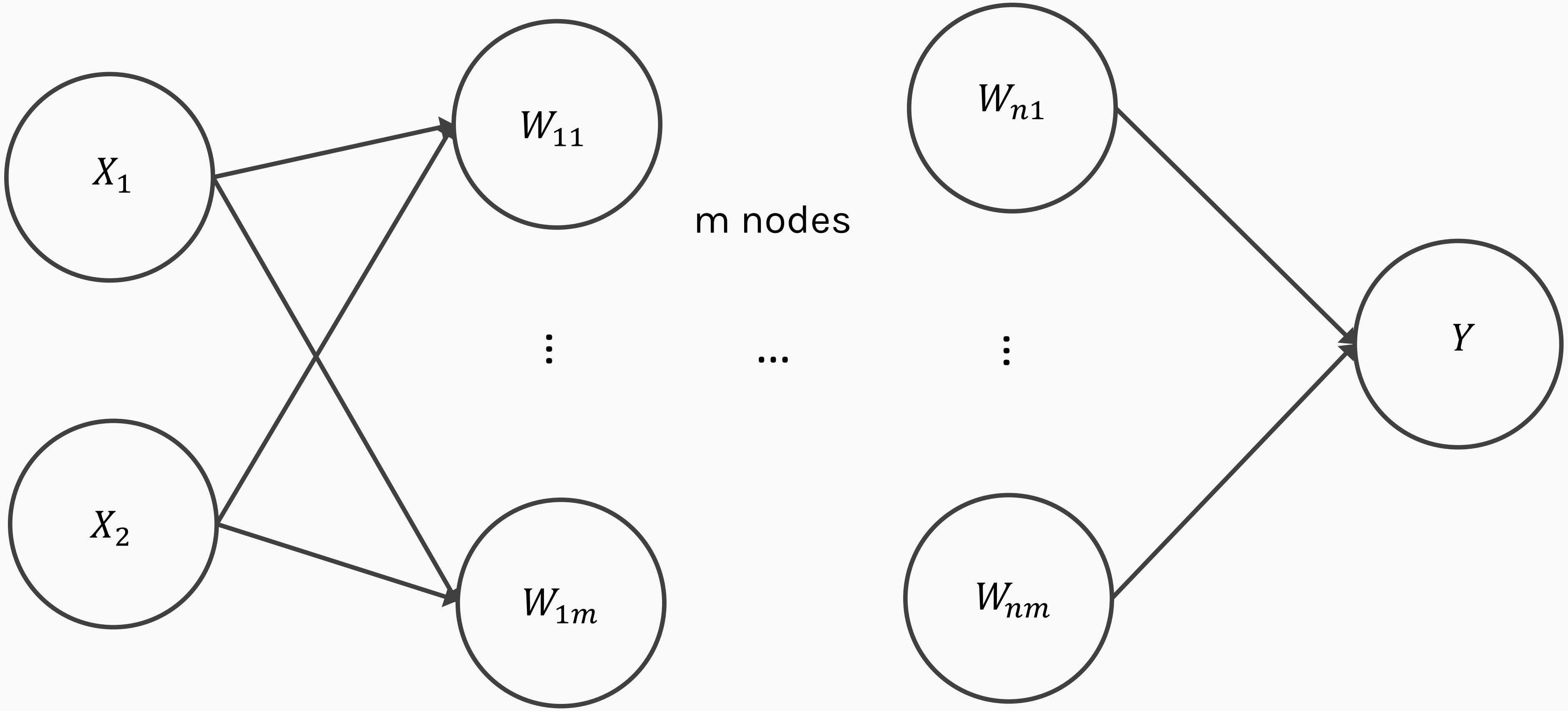
Anatomy of artificial neural network (ANN)



Anatomy of artificial neural network (ANN)

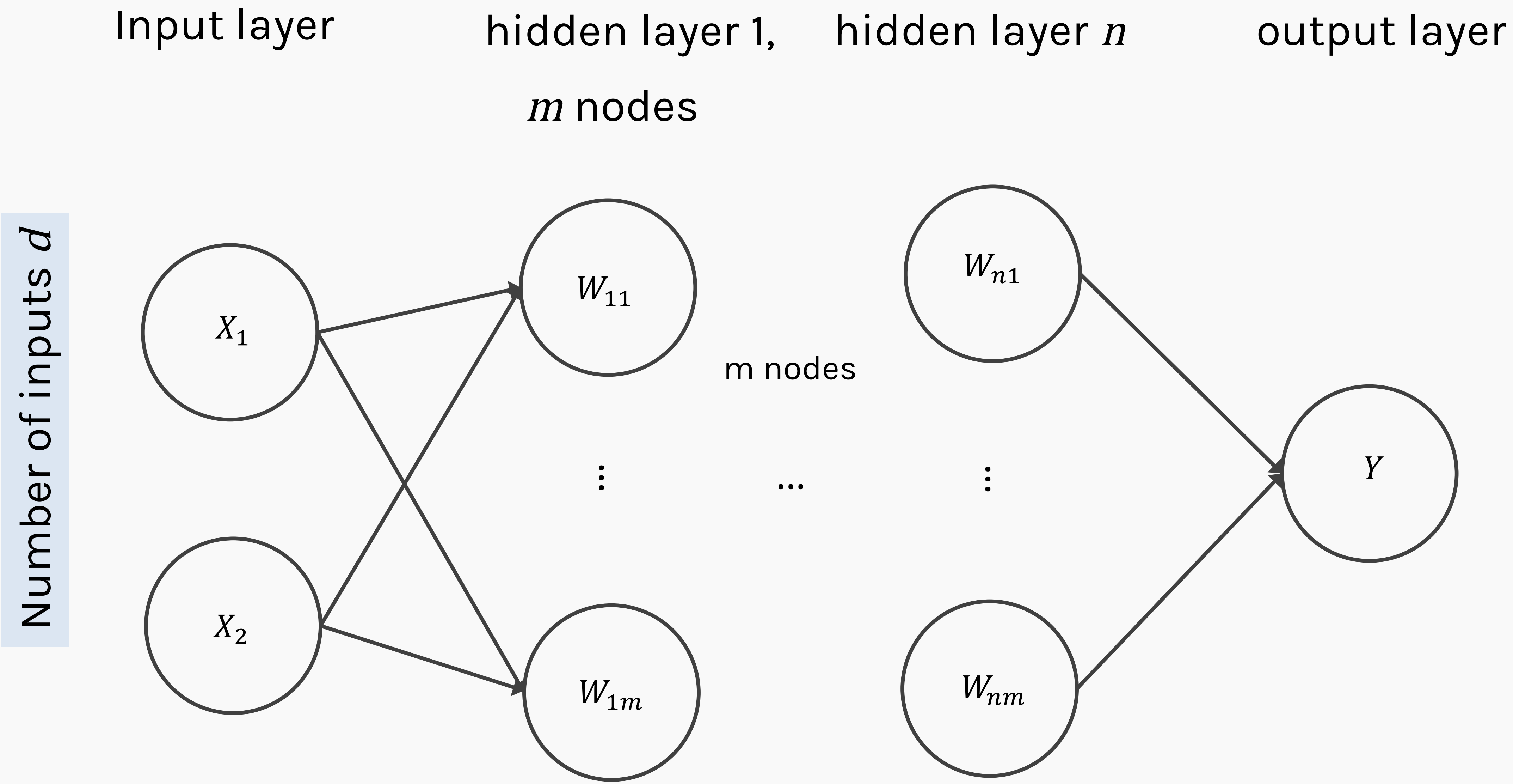
Input layer　　hidden layer 1,　　hidden layer n　　output layer

m nodes



We will talk later about the choice of the number of nodes.

Anatomy of artificial neural network (ANN)



Number of inputs is specified by the data

Review of Feed Forward Artificial Neural Networks

Anatomy of a NN

Design choices

- **Activation function**
- Loss function
- Output units
- Architecture

Learning

Activation function

$$h = f(W^T X + b)$$

The activation function should:

- Ensures not linearity
- Ensure gradients remain large through hidden unit

Common choices are

- Sigmoid
- Relu, leaky ReLU, Generalized ReLU, MaxOut
- softplus
- tanh
- swish

Activation function

$$h = f(W^T X + b)$$

The activation function should:

- Ensures **not linearity**
- Ensure gradients remain large through hidden unit

Common choices are

- Sigmoid
- Relu, leaky ReLU, Generalized ReLU, MaxOut
- softplus
- tanh
- swish

Beyond Linear Models

Linear models

- Can be fit efficiently (via convex optimization)
- Limited model capacity

Alternative:

$$f(x) = w^T \phi(x)$$

Where ϕ is a *non-linear transform*

Traditional ML

Manually engineer ϕ

- Domain specific, enormous human effort

Generic transform

- Maps to a higher-dimensional space
- Kernel methods: e.g. RBF kernels
- Over fitting: does not generalize well to test set
- Cannot encode enough prior information

Deep Learning

- Directly learn ϕ

$$f(x;\theta) = W^T \phi(x;\theta)$$

- where θ are parameters of the transform
- ϕ defines hidden layers
- Can encode prior beliefs, generalizes well

Activation function

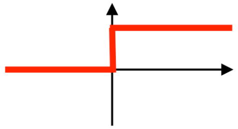
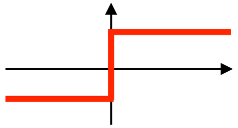
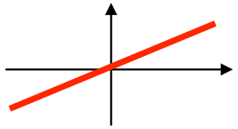
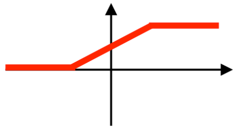
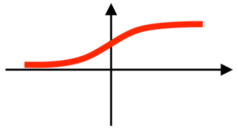
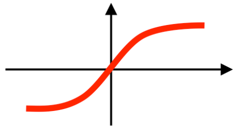
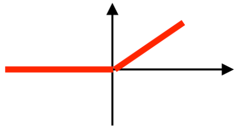
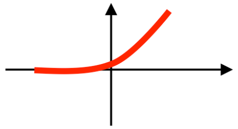
$$h = f(W^T X + b)$$

The activation function should:

- Ensures not linearity
- Ensure gradients remain large through hidden unit

Common choices are

- Sigmoid
- Relu, leaky ReLU, Generalized ReLU, MaxOut
- softplus
- tanh
- swish

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks	

Review of Feed Forward Artificial Neural Networks

Anatomy of a NN

Design choices

- Activation function
- **Loss function**
- Output units
- Architecture

Learning

Loss Function

Cross-entropy between training data and model distribution (i.e. negative log-likelihood)

$$J(W) = -\mathbb{E}_{x,y\sim\hat{p}_{\text{data}}} \log p_{\text{model}}(\mathrm{y}|\mathrm{x})$$

Do not need to design separate loss functions.

Gradient of cost function must be large enough

Review of Feed Forward Artificial Neural Networks

Anatomy of a NN

Design choices

- Activation function
- Loss function
- **Output units**
- Architecture

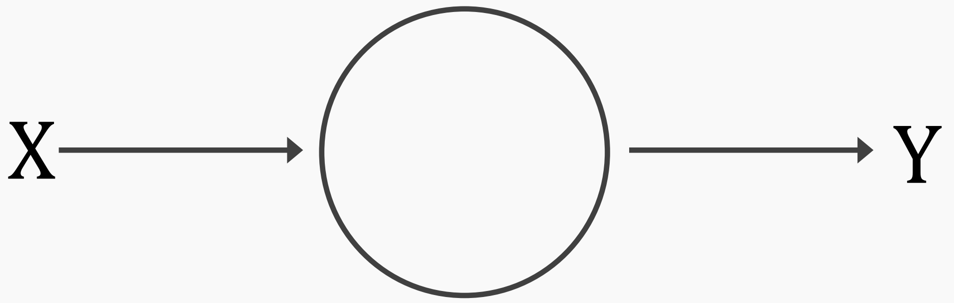
Learning

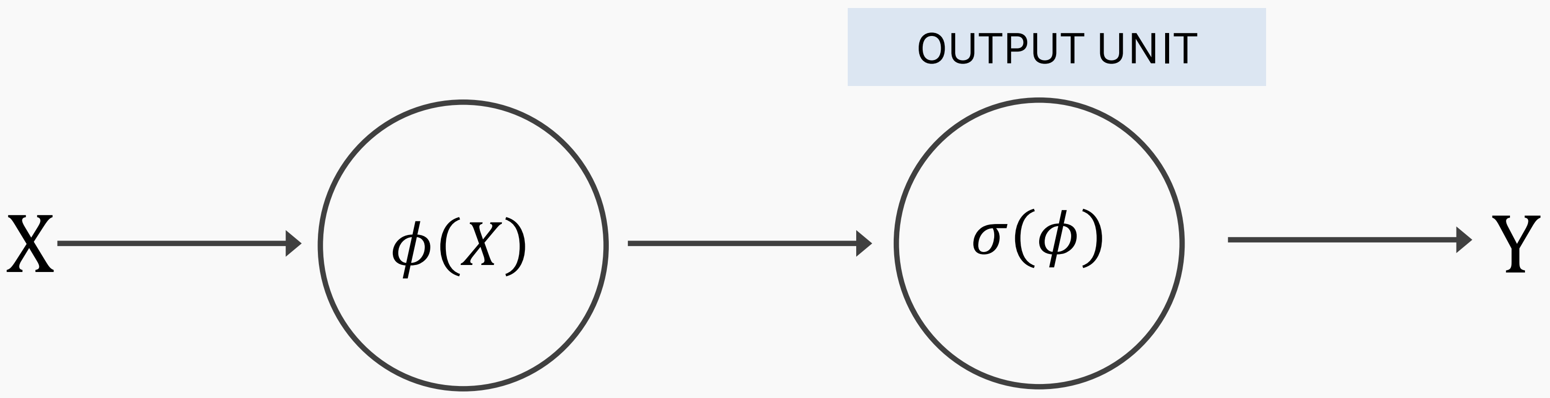
Output Units

Output Type	Output Distribution	Output layer	Cost Function
Binary			

Link function

$$X \Longrightarrow \phi(X) = W^T X \Longrightarrow P(y=0) = \frac{1}{1+e^{\phi(X)}}$$





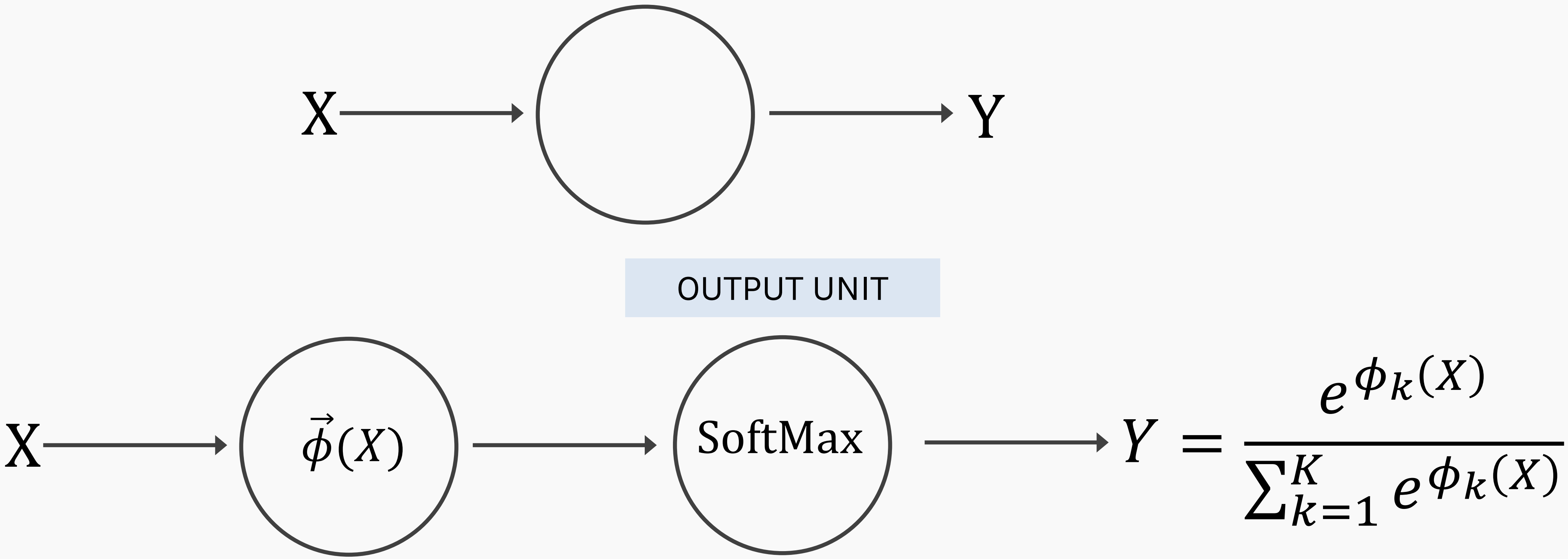
Output Units

Output Type	Output Distribution	Output layer	Cost Function
Binary	Bernoulli	Sigmoid	Binary Cross Entropy

Output Units

Output Type	Output Distribution	Output layer	Cost Function
Binary	Bernoulli	Sigmoid	Binary Cross Entropy
Discrete			

Link function multi-class problem



Output Units

Output Type	Output Distribution	Output layer	Cost Function
Binary	Bernoulli	Sigmoid	Binary Cross Entropy
Discrete	Multinoulli	Softmax	Cross Entropy

Output Units

Output Type	Output Distribution	Output layer	Cost Function
Binary	Bernoulli	Sigmoid	Binary Cross Entropy
Discrete	Multinoulli	Softmax	Cross Entropy
Continuous	Gaussian	Linear	MSE

Output Units

Output Type	Output Distribution	Output layer	Cost Function
Binary	Bernoulli	Sigmoid	Binary Cross Entropy
Discrete	Multinoulli	Softmax	Cross Entropy
Continuous	Gaussian	Linear	MSE
Continuous	Arbitrary	-	GANS

Review of Feed Forward Artificial Neural Networks

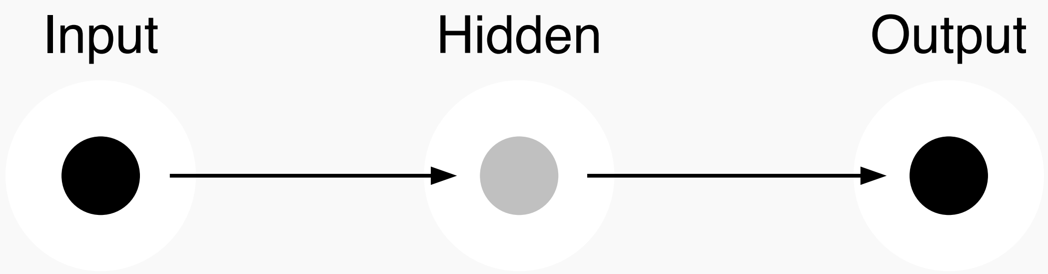
Anatomy of a NN

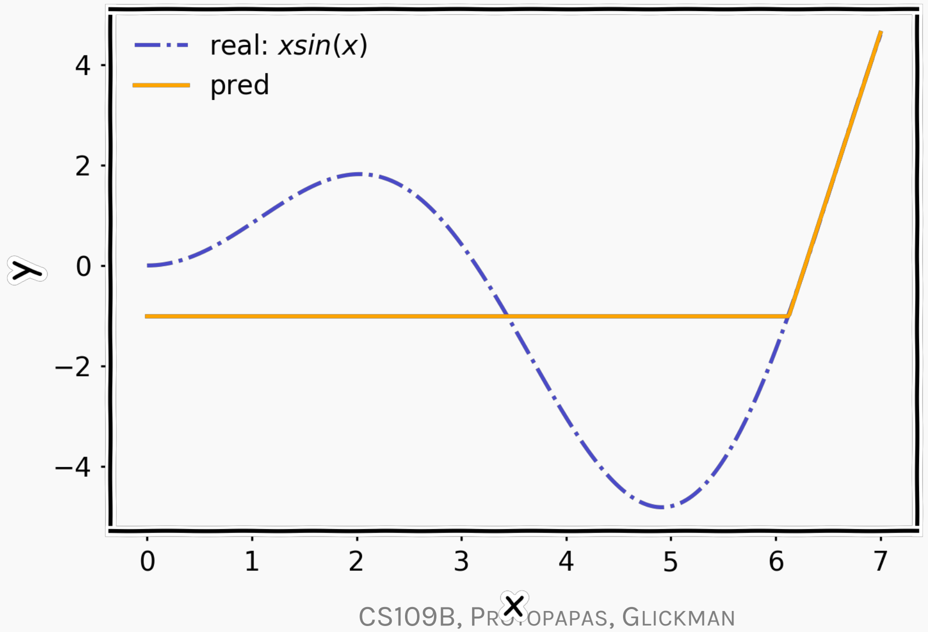
Design choices

- Activation function
- Loss function
- Output units
- **Architecture**

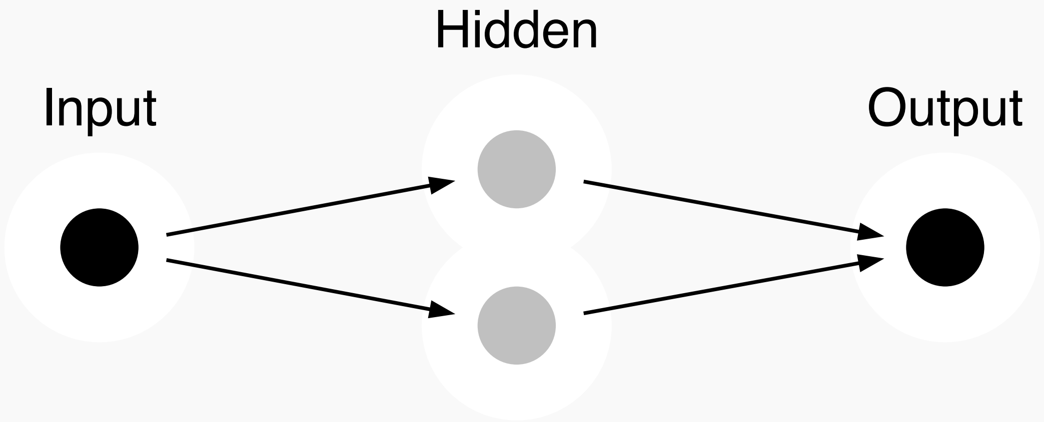
Learning

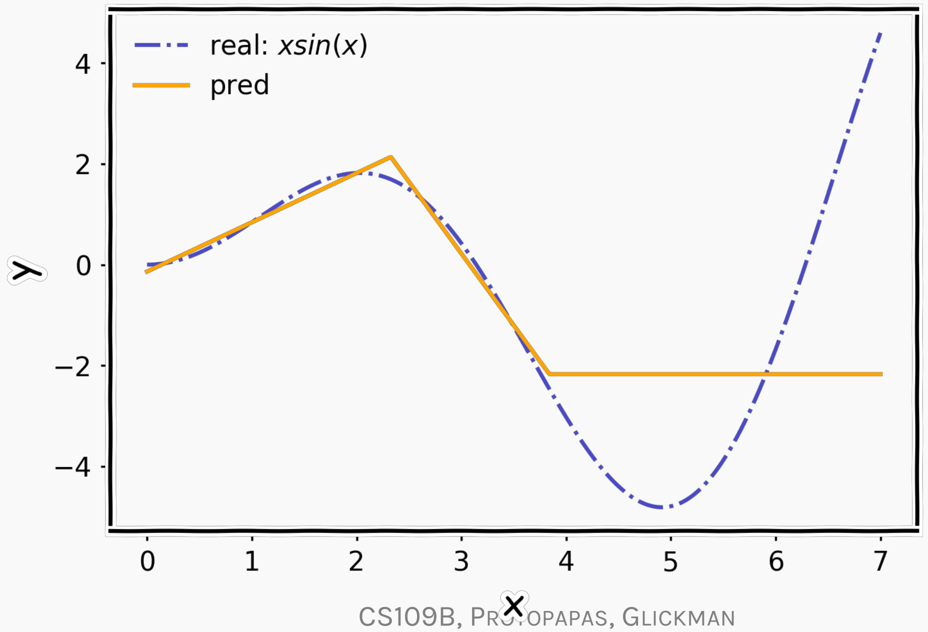
Architecture



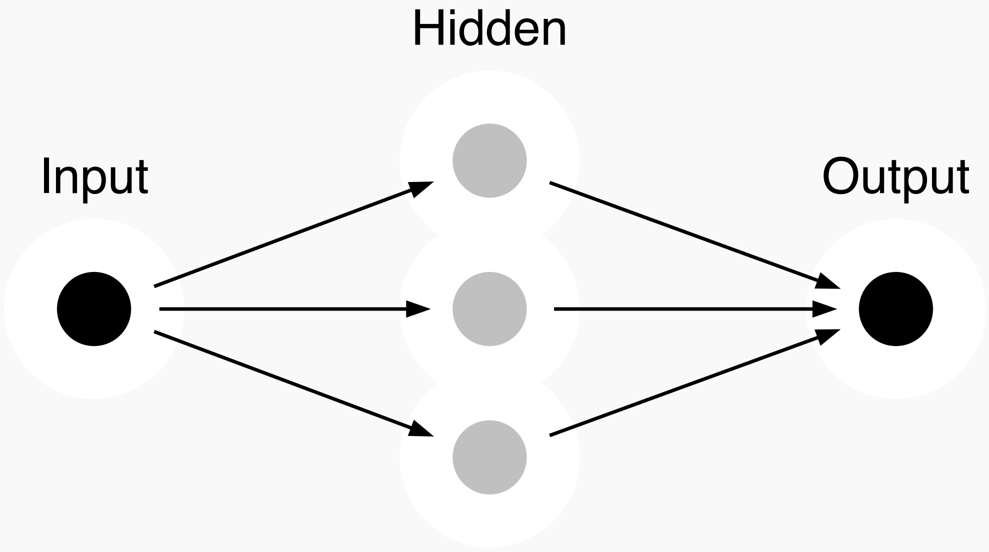


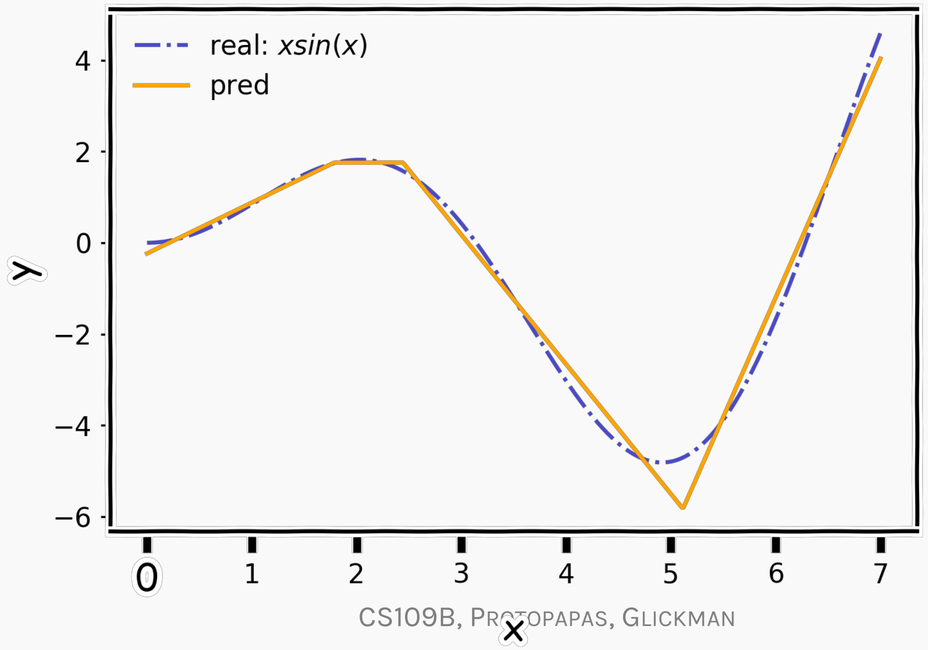
Architecture (cont)



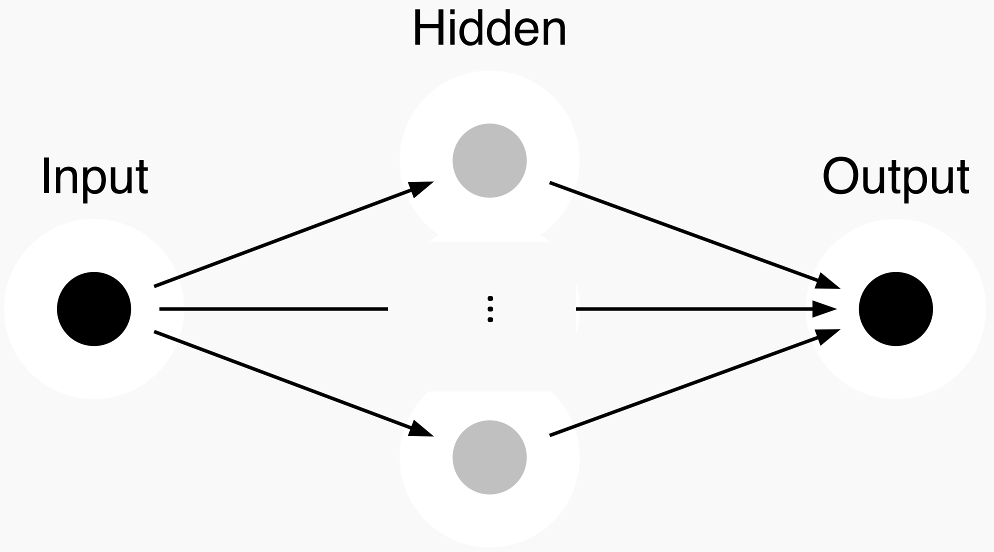


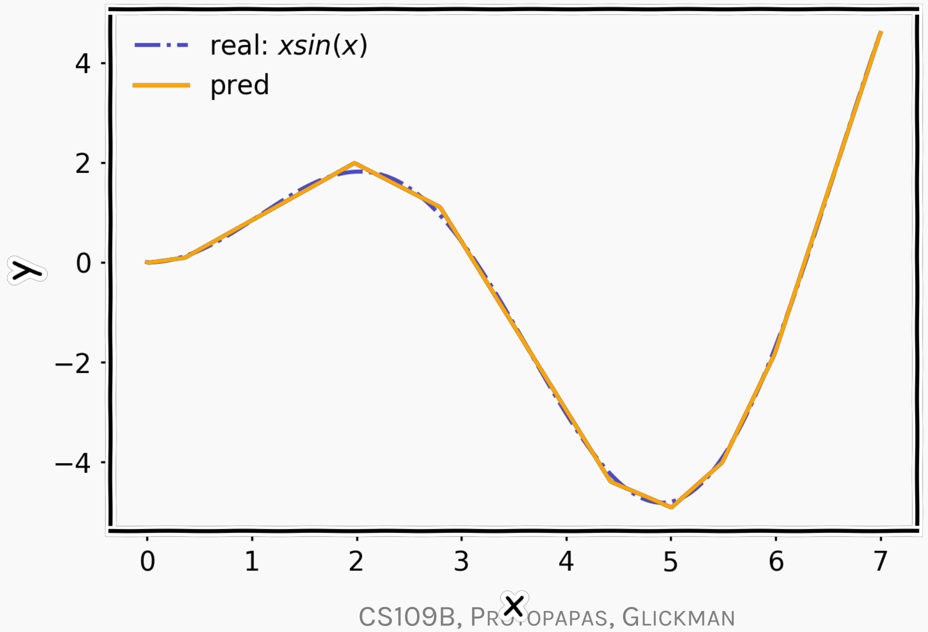
Architecture (cont)



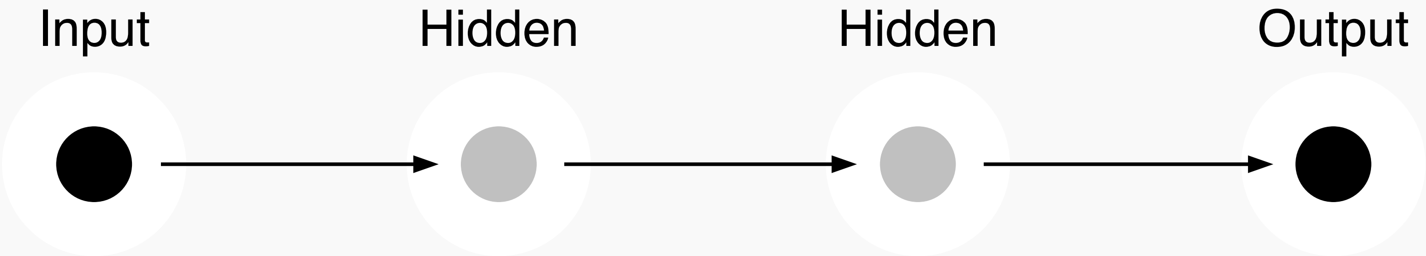


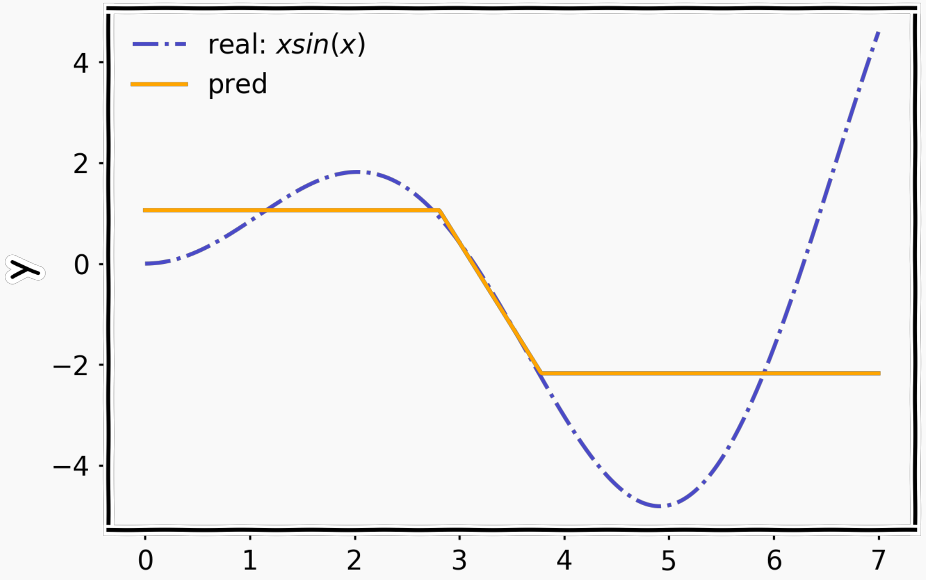
Architecture (cont)



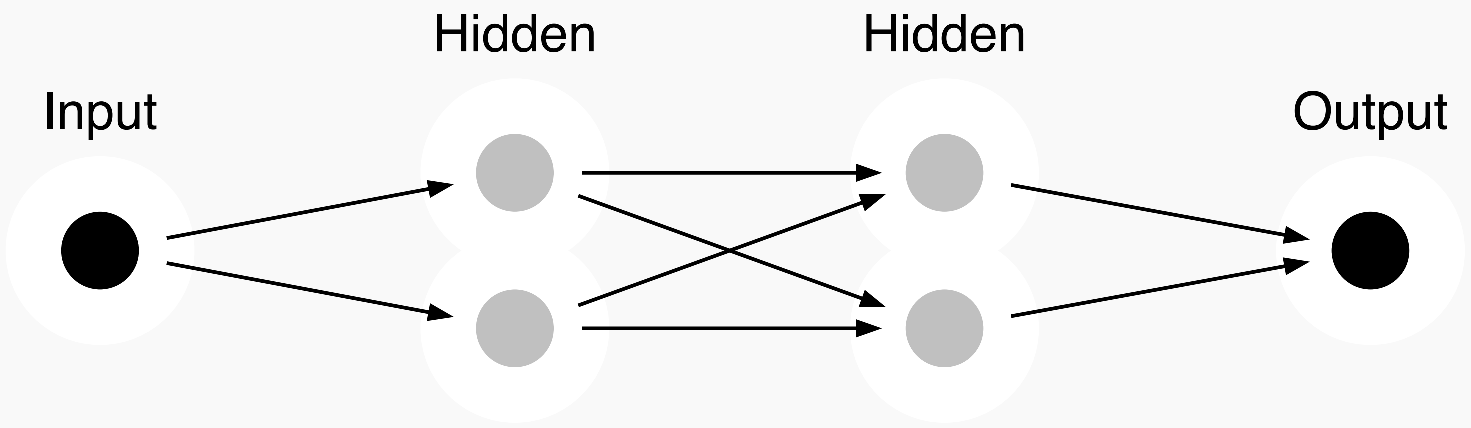


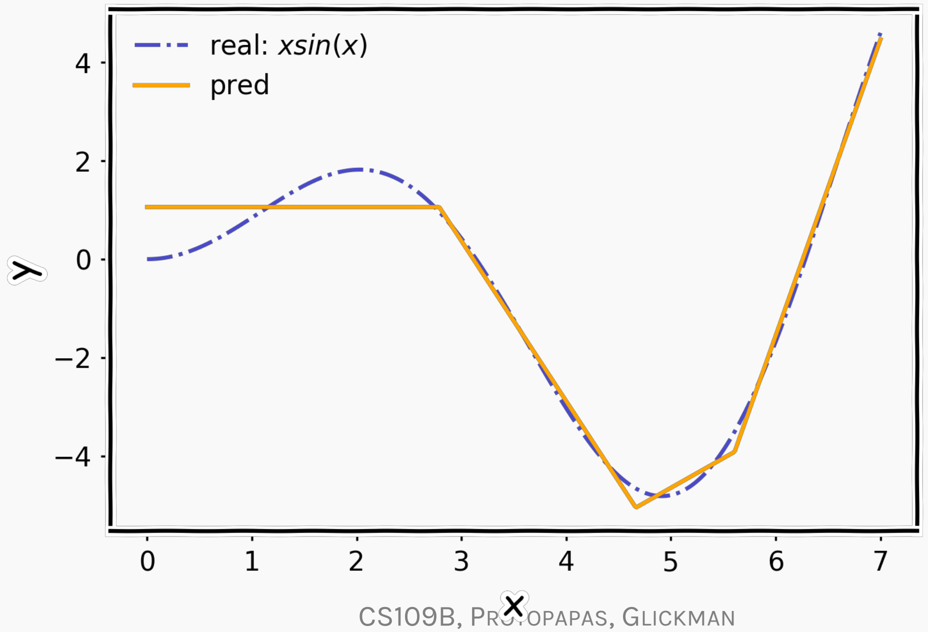
Architecture (cont)



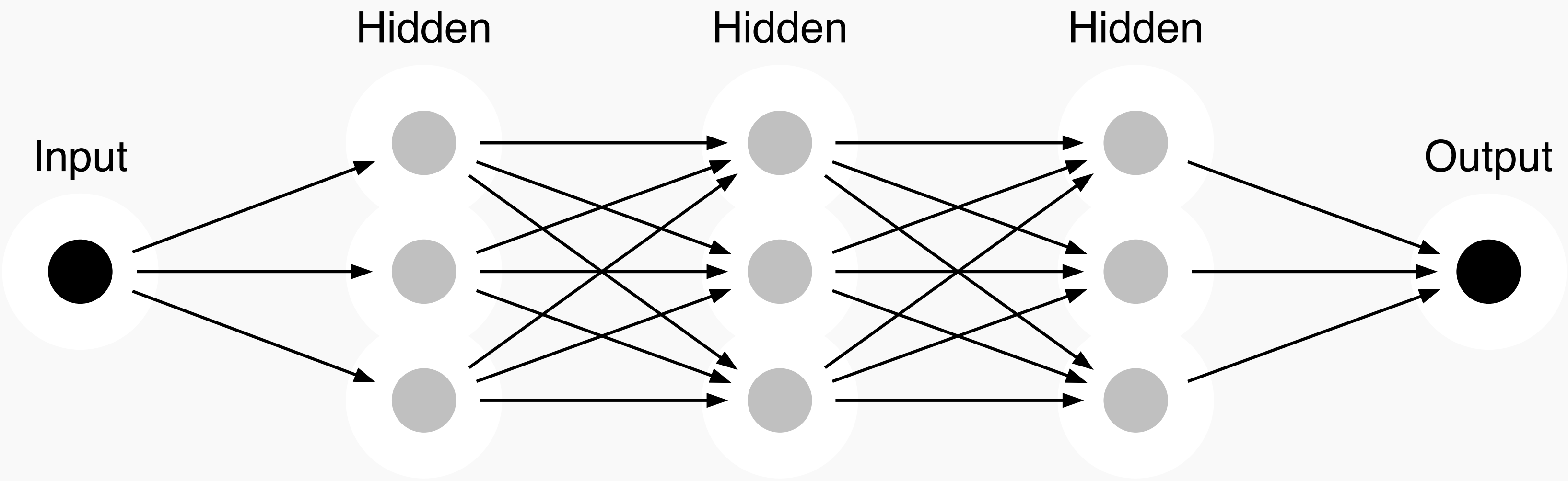


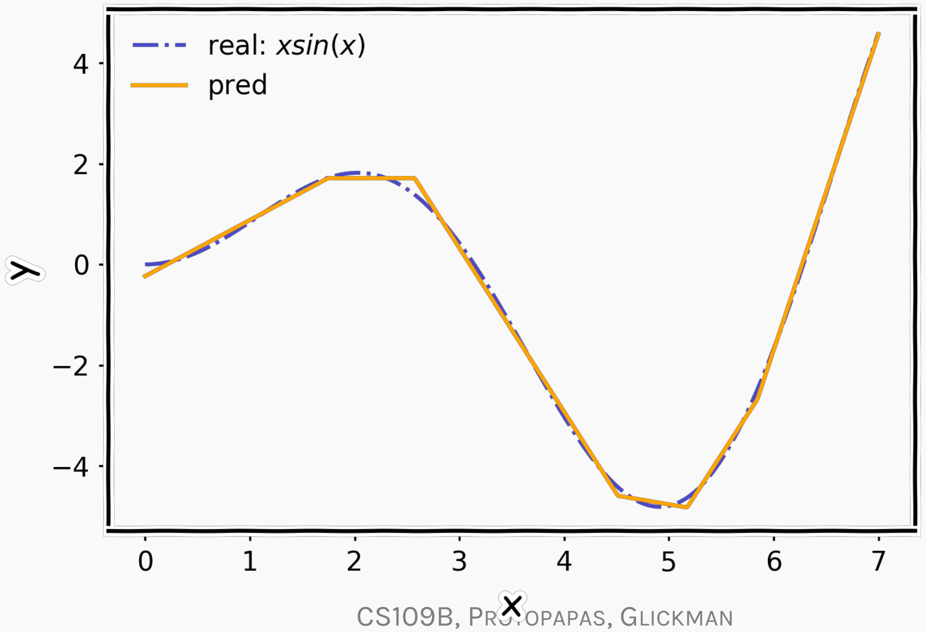
Architecture (cont)





Architecture (cont)





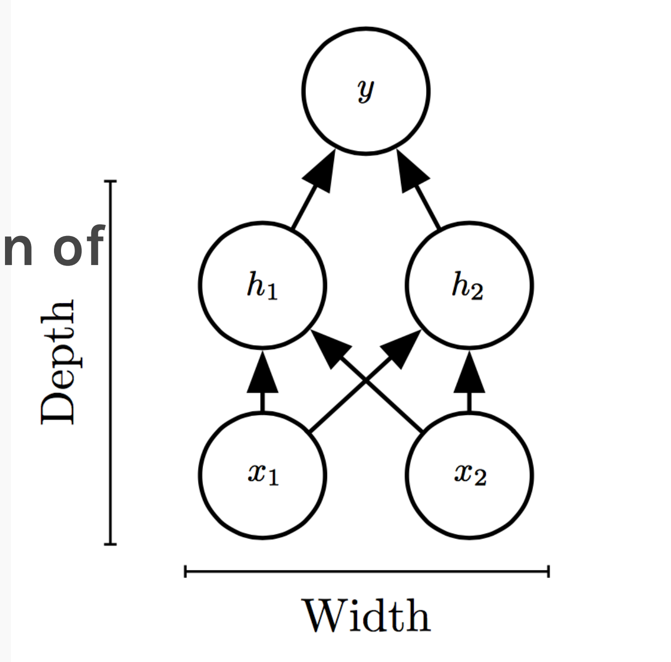
Universal Approximation Theorem

Think of Neural Network as function approximation.

$$Y = f(x) + \epsilon$$

$$Y = \hat{f}(x) + \epsilon$$

NN: $\Longrightarrow \hat{f}(x)$



One hidden layer is enough to *represent* an approximation of any function to an arbitrary degree of accuracy

So why deeper?

- Shallow net may need (exponentially) more width
- Shallow net may overfit more

What does an astronomer blow with gum?

Hubbles

Auditors: Volunteers

LiveSlides web content

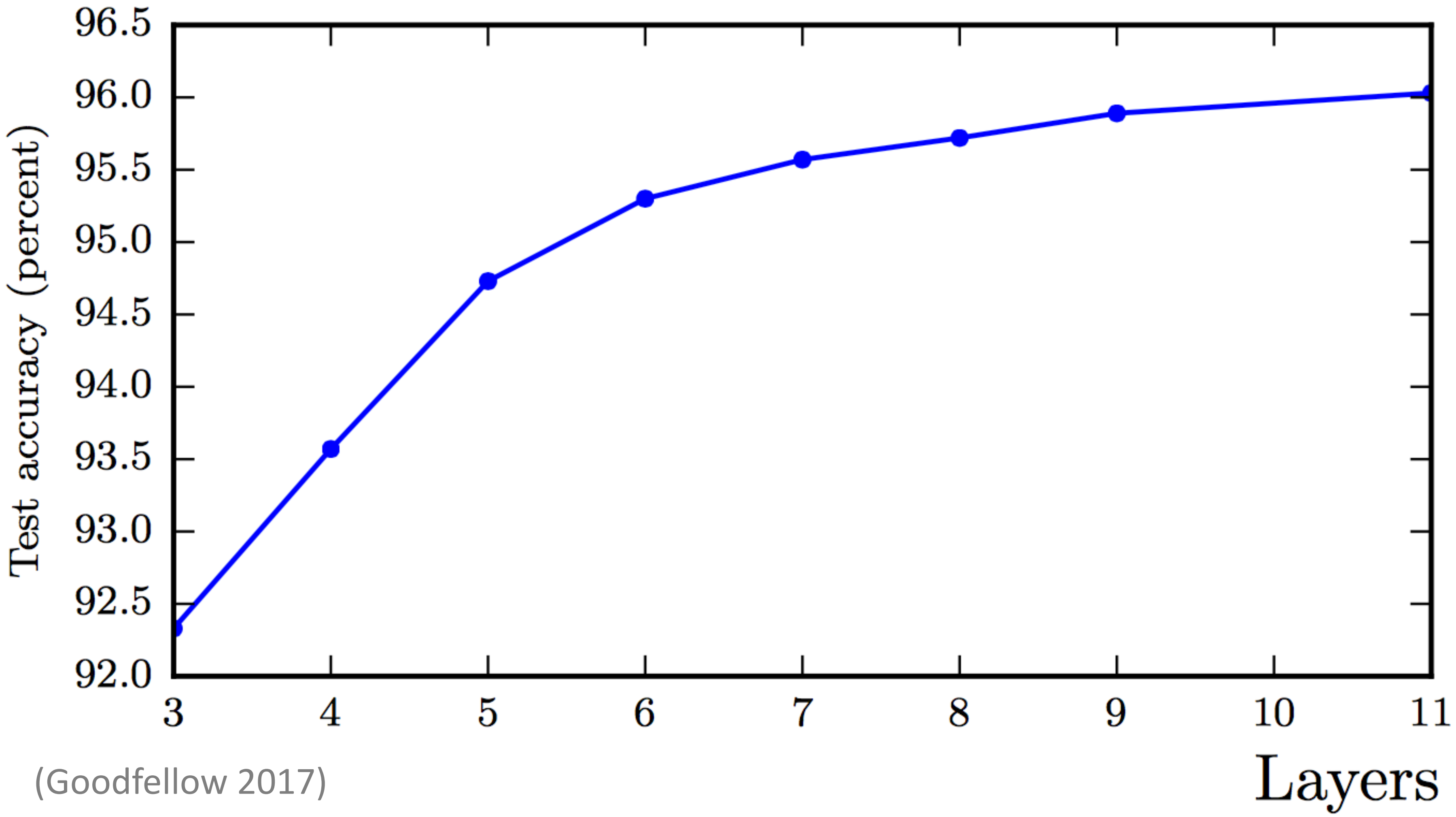
To view

Download the add-in.

liveslides.com/download

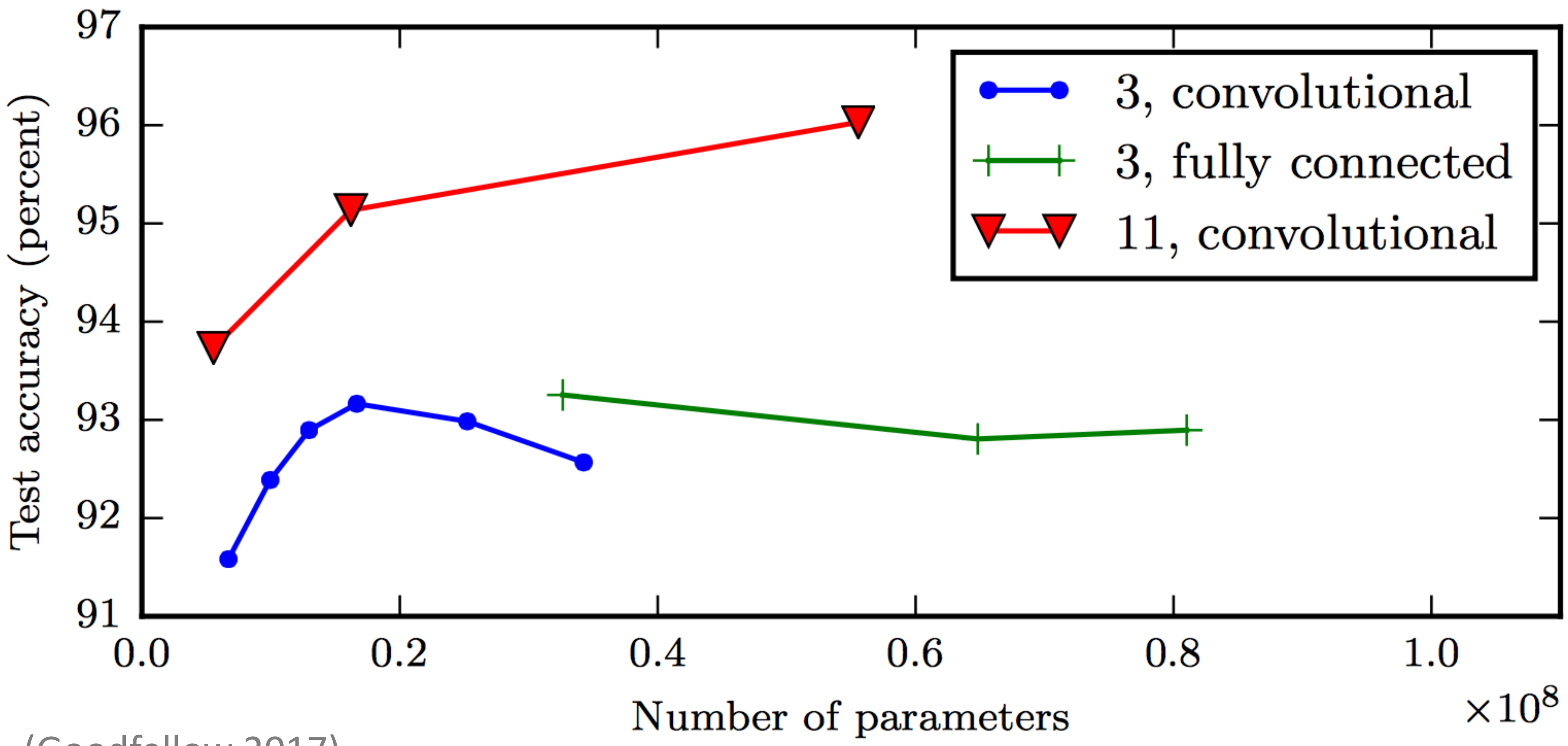
Start the presentation.

Better Generalization with Depth



(Goodfellow 2017)

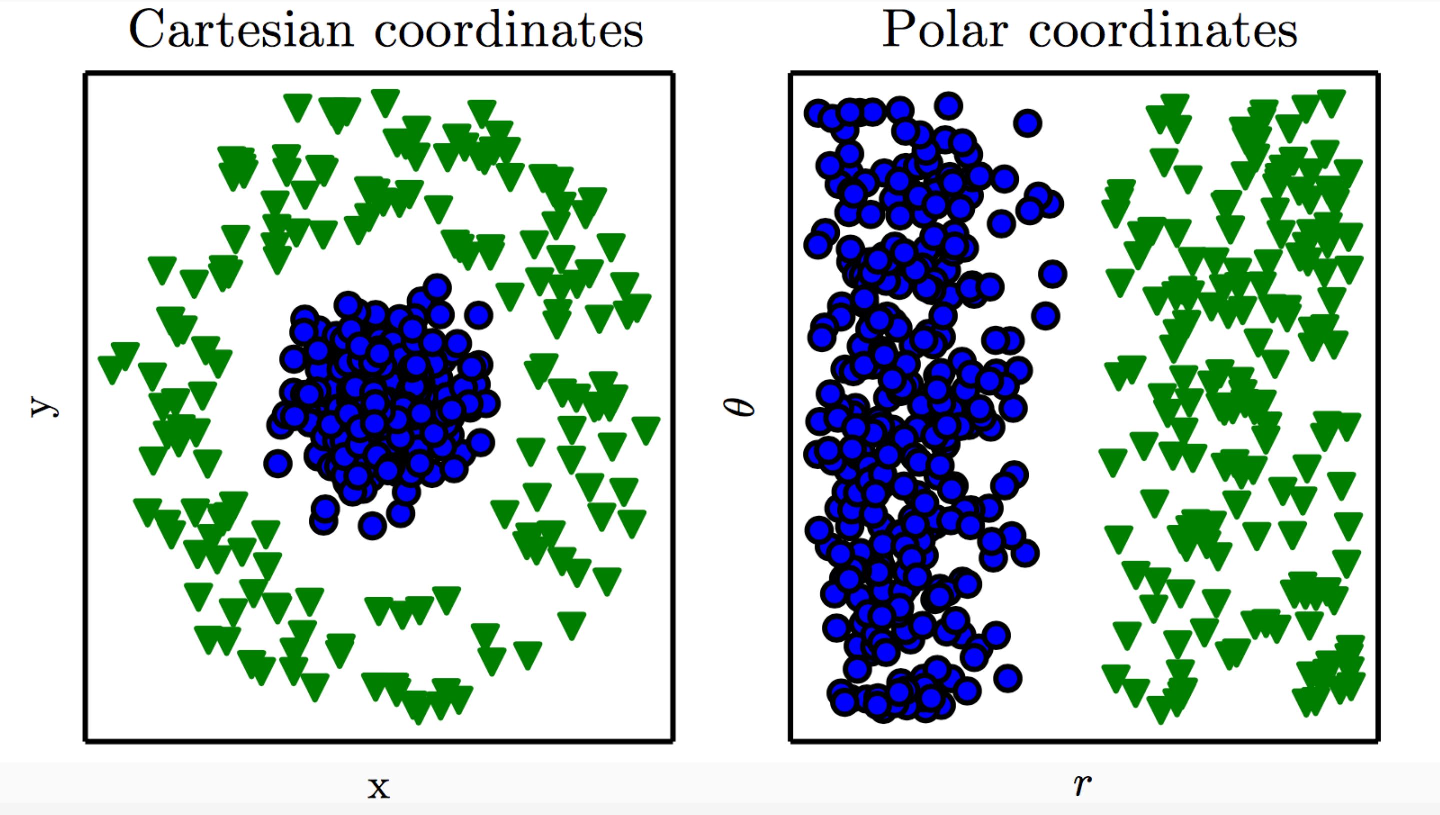
Large, Shallow Nets Overfit More



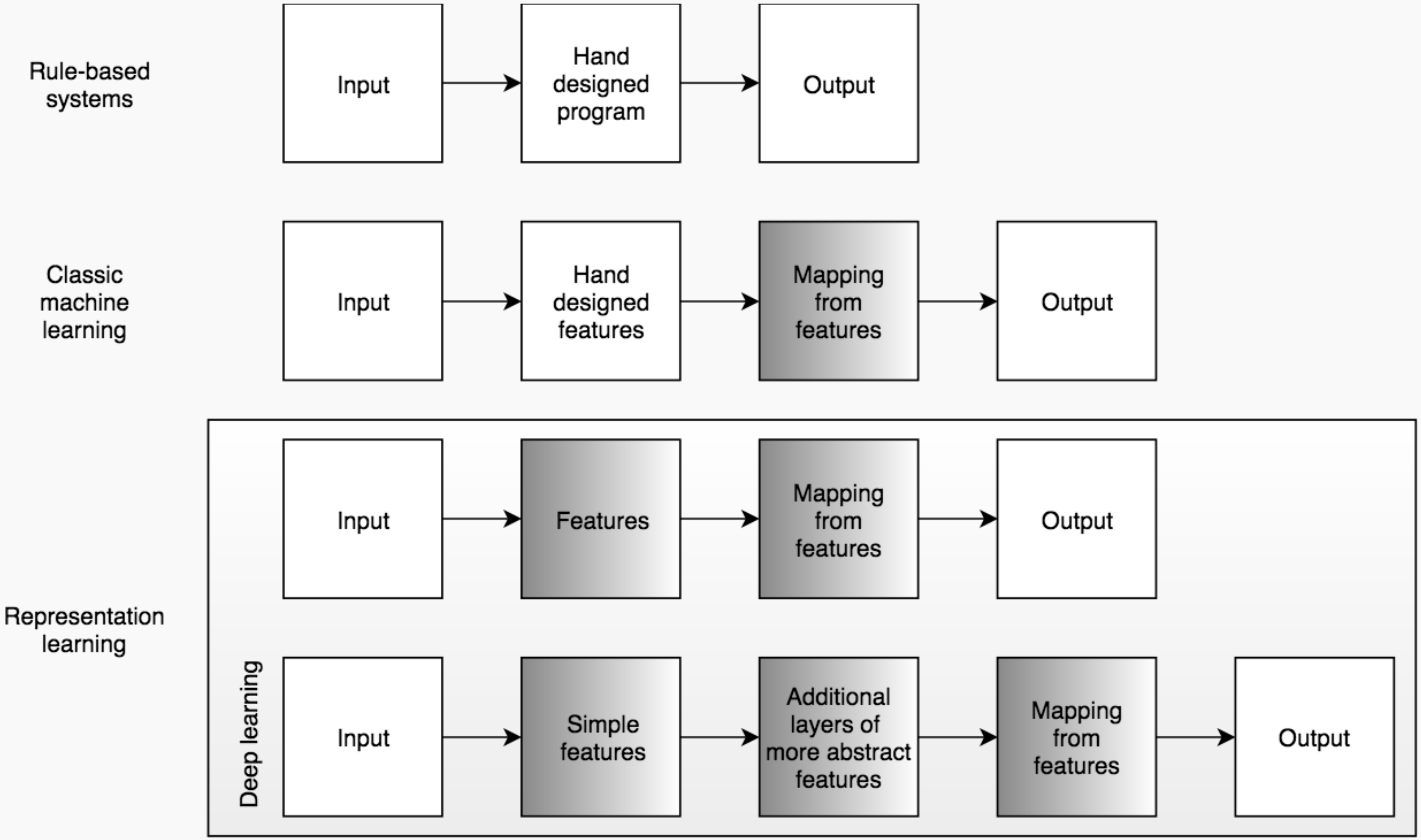
(Goodfellow 2017)

Why layers? Representation

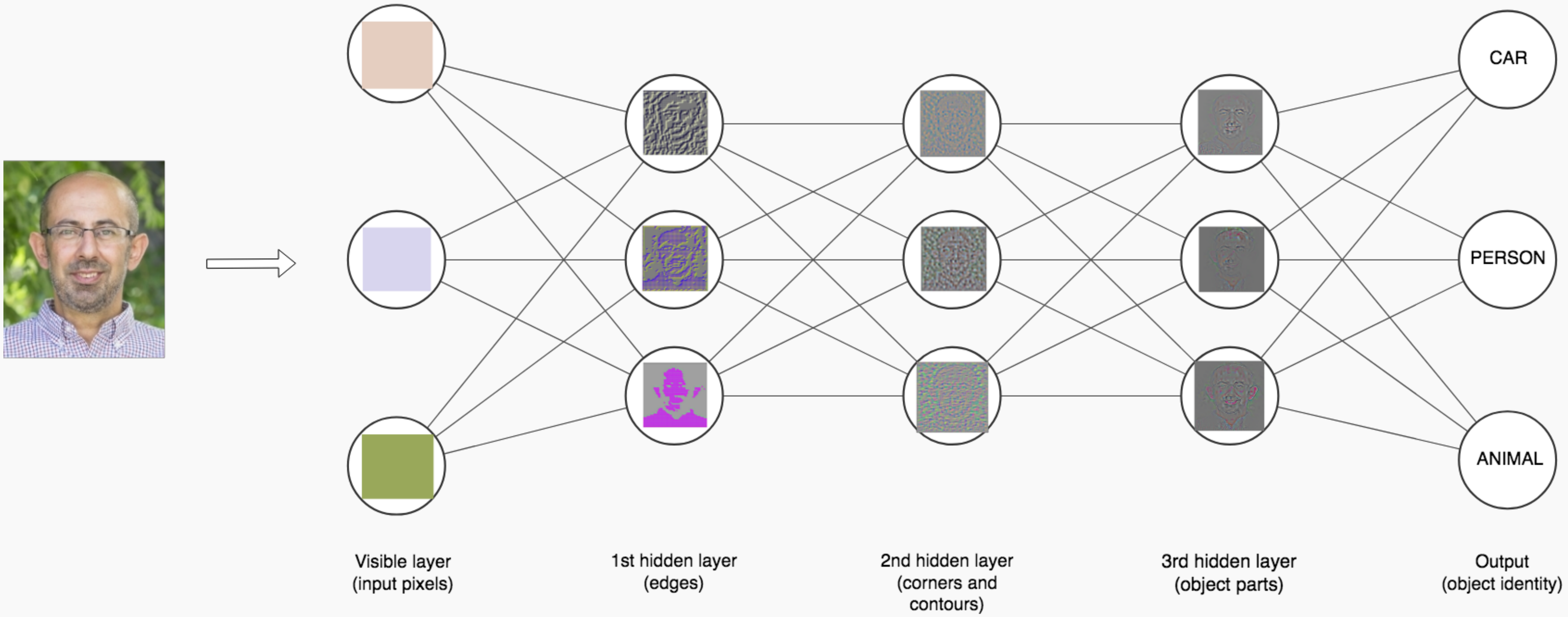
Representation Matters



Learning Multiple Components



Depth = Repeated Compositions



Review of Feed Forward Artificial Neural Networks

Anatomy of a NN

Design choices

- Activation function
- Loss function
- Output units
- Architecture

Learning

Review of Feed Forward Artificial Neural Networks

Anatomy of a NN

Design choices

- Activation function
- Loss function
- Output units
- Architecture

Learning(more next lecture)

Basics ideas of optimizer

Backprop

Heart Data

response variable *Y*
is Yes/No

Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
63	1	typical	145	233	1	2	150	0	2.3	3	0.0	fixed	No
67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.0	normal	Yes
67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.0	reversable	Yes
37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.0	normal	No
41	0	nontypical	130	204	0	2	172	0	1.4	1	0.0	normal	No

Basic idea of learning

Start with Regression or Logistic Regression

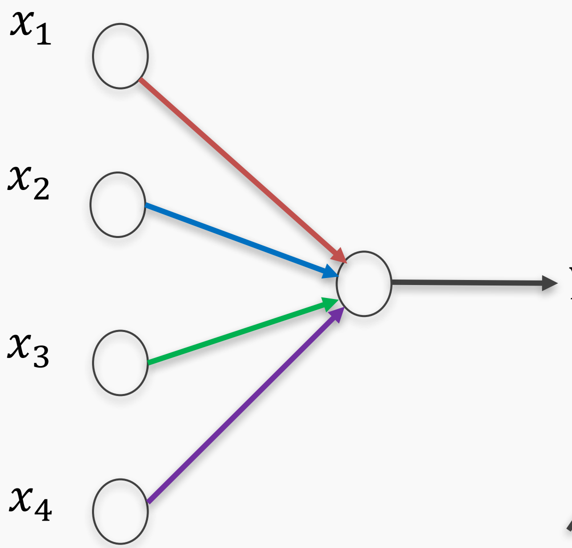
Classification

$f(X) = \frac{1}{1+e^{-W^T X}}$

Regression

$f(X) = W^T X$

$$W^T = [W_0, W_1, \dots, W_4] = [\beta_0, \beta_1, \dots, \beta_4]$$



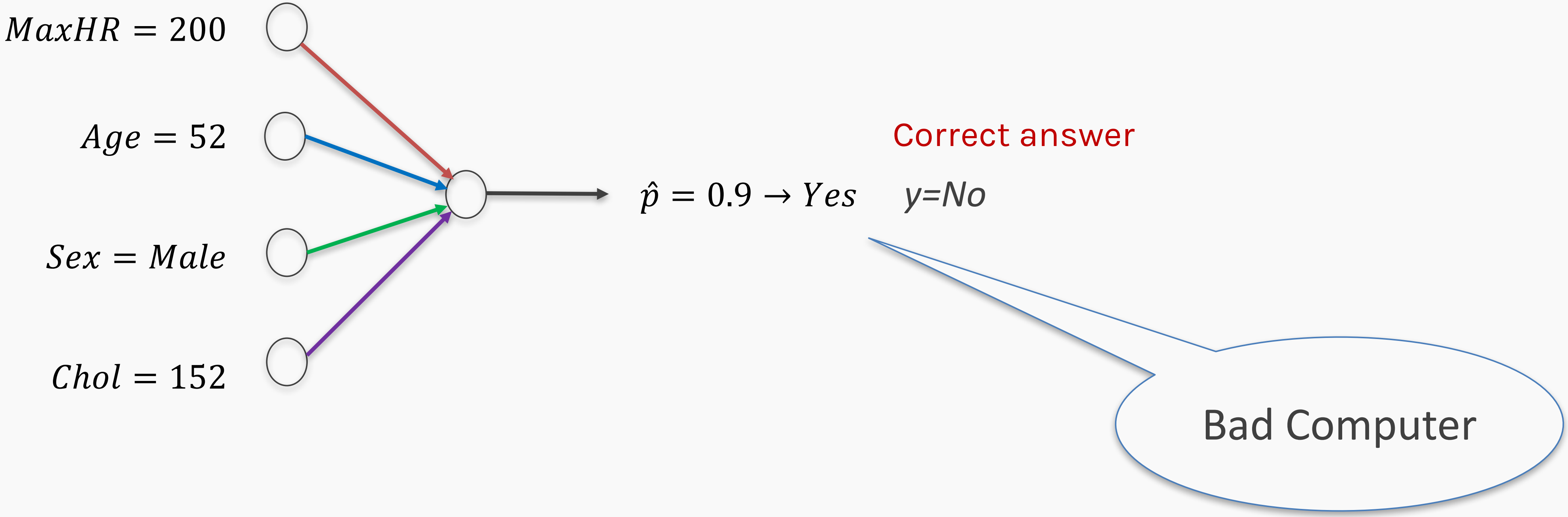
$$Y = f(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4)$$

Intercept or **Bias**

Coefficients or **Weights**

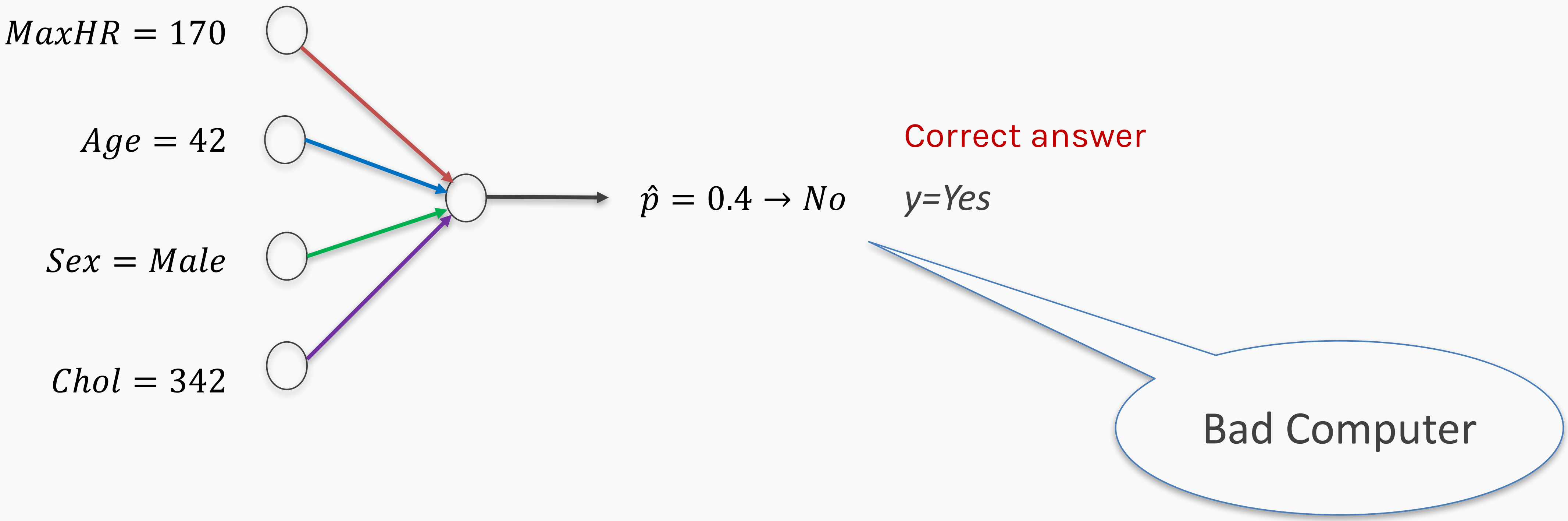
But what is the idea?

Start with all randomly selected weights. Most likely it will perform horribly. For example, in our heart data, the model will be giving us the wrong answer.



But what is the idea?

Start with all randomly selected weights. Most likely it will perform horribly. For example, in our heart data, the model will be giving us the wrong answer.



But what is the idea?

- **Loss Function:** Takes all of these results and averages them and tells us how bad or good the computer or those weights are.
- Telling the computer how **bad** or **good** is, does not help.
- You want to tell it how to change those weights so it gets better.

Loss function: $\mathcal{L}(w_0, w_1, w_2, w_3, w_4)$

For now let's only consider one weight, $\mathcal{L}(w_1)$

But what is the idea?

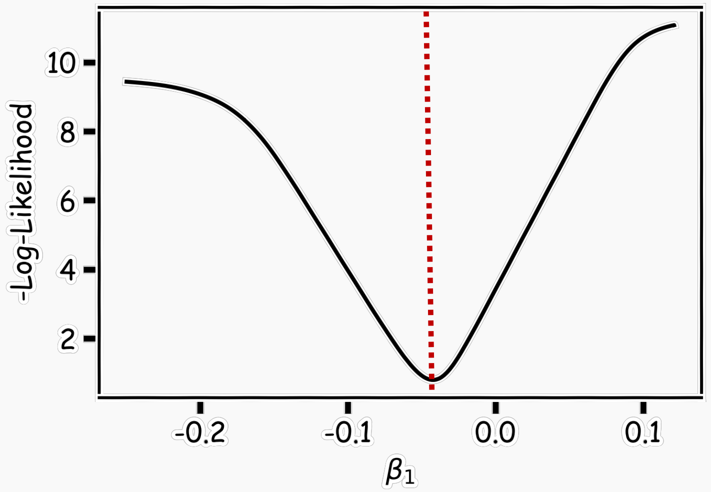
Trial and error:

Change the weights and see the effect.

This can take long long time especially in NN where we have millions of weights to adjust.

Minimizing the Loss function

Ideally we want to know the value of w_1 that gives the minimul $\mathcal{L}(W)$

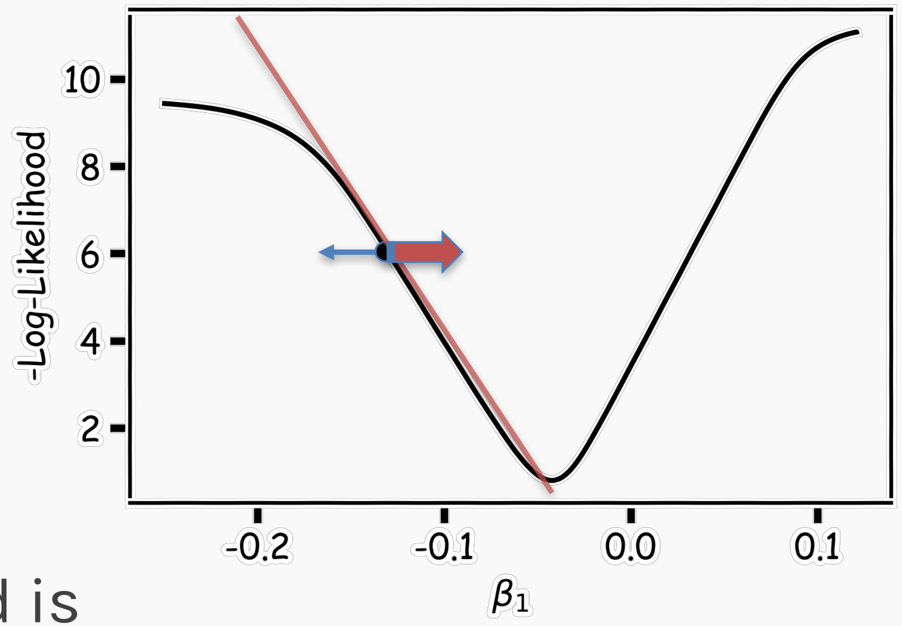


To find the optimal point of a function $\mathcal{L}(W)$

$$\frac{d\mathcal{L}(W)}{dW} = 0$$

And find the W that satisfies that equation. Sometimes there is no explicit solution for that.

Minimizing the Loss function



A more flexible method is

- Start from any point
  - Determine which direction to go to reduce the loss (left or right)
  - Specifically, we can calculate the slope of the function at this point
  - Shift to the right if slope is negative or shift to the left if slope is positive
- Repeat

Minimization of the Loss Function

If the step is proportional to the slope then you avoid overshooting the minimum.

Question: What is the mathematical function that describes the slope?

Question: How do we generalize this to more than one predictor?

Question: What do you think it is a good approach for telling the model how to change (what is the step size) to become better?

Minimization of the Loss Function

If the step is proportional to the slope then you avoid overshooting the minimum.

Question: What is the mathematical function that describes the slope?

Derivative

Question: How do we generalize this to more than one predictor?

Take the derivative with respect to each coefficient and do the same sequentially

Question: What do you think it is a good approach for telling the model how to change (what is the step size) to become better?

More on this later

Let's play the Pavlos game

We know that we want to go in the opposite direction of the derivative and we know we want to be making a step proportionally to the derivative.

Making a step means:

$$w^{new} = w^{old} + step$$

Learning Rate

Opposite direction of the derivative means:

$$w^{new} = w^{old} - \lambda \frac{d\mathcal{L}}{dw}$$

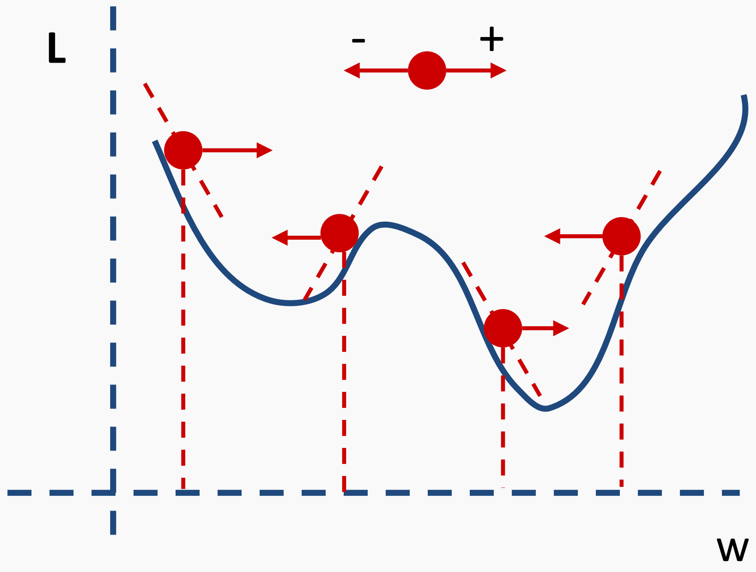
Change to more conventional notation:

$$w^{(i+1)} = w^{(i)} - \lambda \frac{d\mathcal{L}}{dw}$$

Gradient Descent

- Algorithm for optimization of first order to finding a minimum of a function.
- It is an iterative method.
- L is decreasing in the direction of the negative derivative.
- The learning rate is controlled by the magnitude of λ.

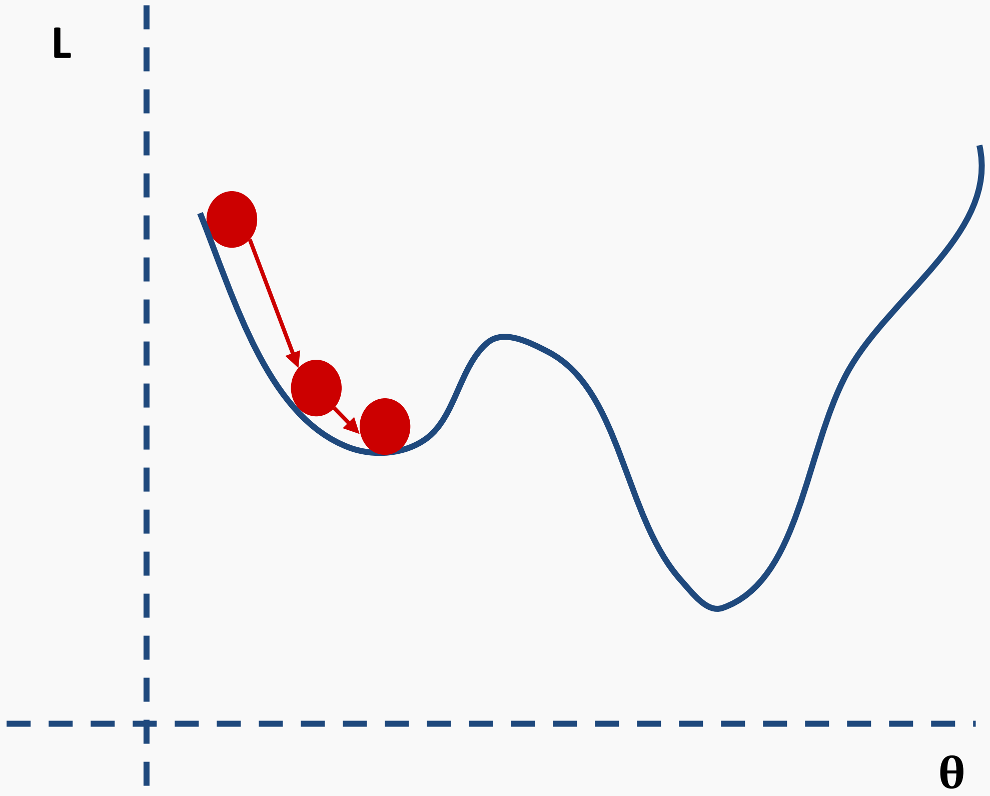
$$w^{(i+1)} = w^{(i)} - \lambda \frac{d\mathcal{L}}{dw}$$



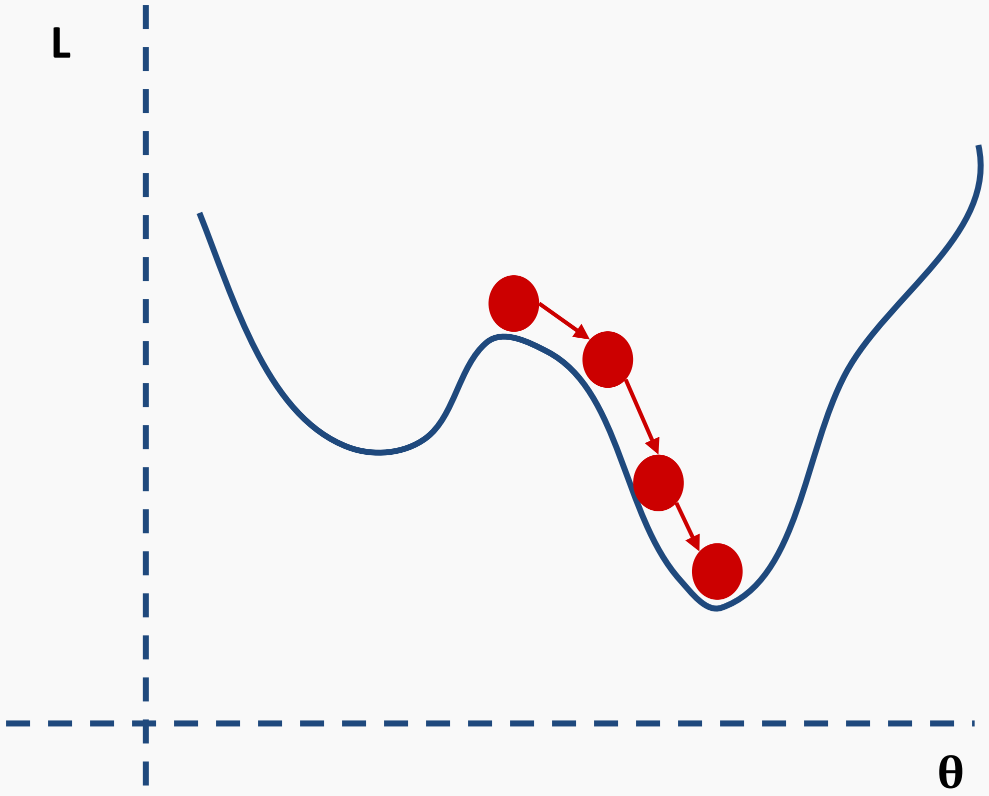
Considerations

- We still need to derive the derivatives.
- We need to know what is the learning rate or how to set it.
- We need to avoid local minima.
- Finally, the full likelihood function includes summing up all individual ‘*errors*’. Unless you are a statistician, this can be hundreds of thousands of examples.

Local vs Global Minima



Local vs Global Minima



Local vs Global Minima

No guarantee that we get the global minimum.

Question: What would be a good strategy?

Large data

Batch and Stochastic Gradient Descent

$$\mathcal{L} = -\sum_{i}[y_i \log p_i + (1 - y_i)\log(1 - p_i)]$$

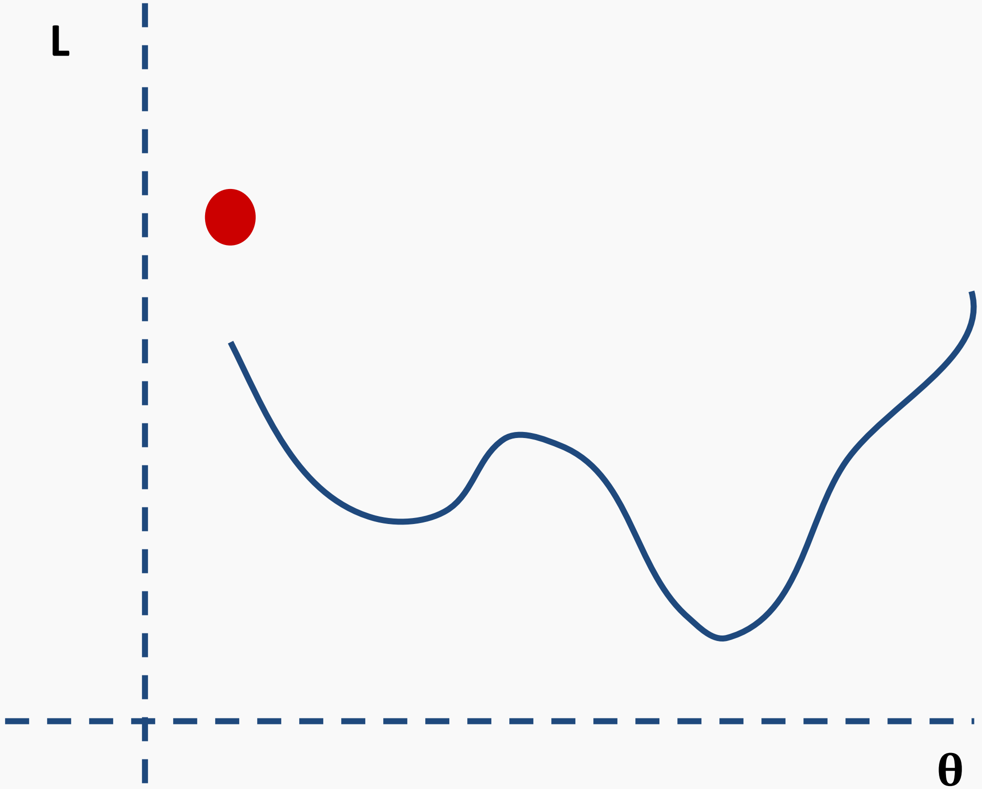
Instead of using all the examples for every step, use a subset of them (batch).

For each iteration k, use the following loss function to derive the derivatives:

$$\mathcal{L}^k = -\sum_{i \in b^k}[y_i \log p_i + (1 - y_i)\log(1 - p_i)]$$

which is an **approximation** to the full Loss function.

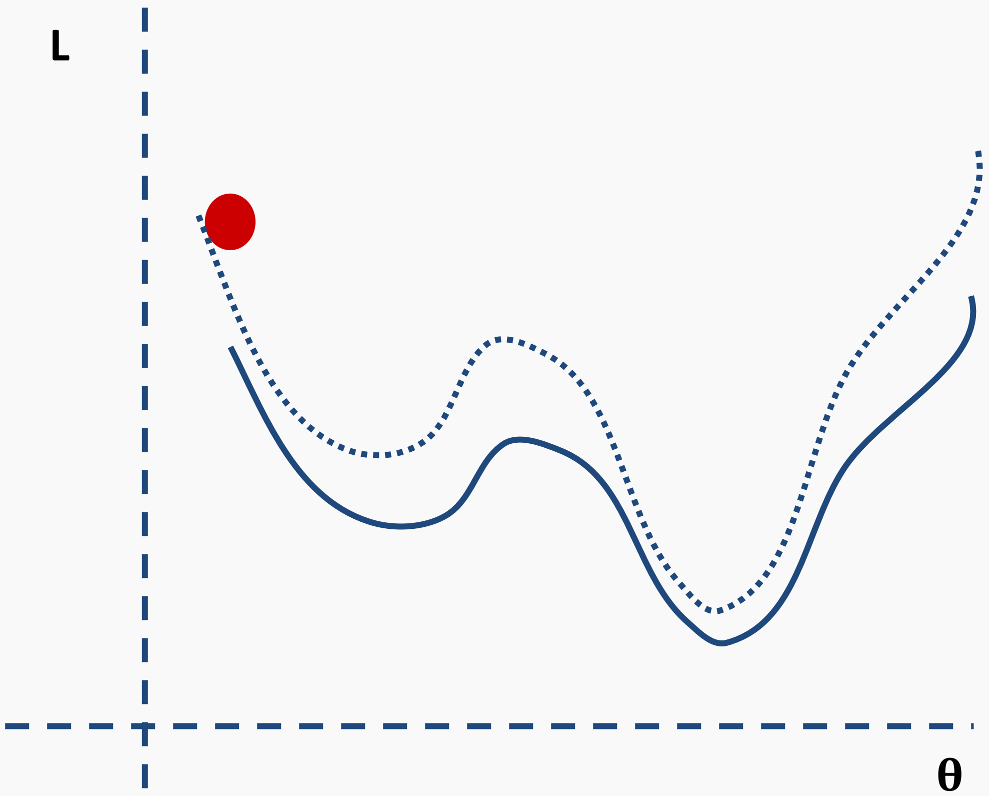
Batch and Stochastic Gradient Descent



Full Likelihood:

Batch Likelihood:

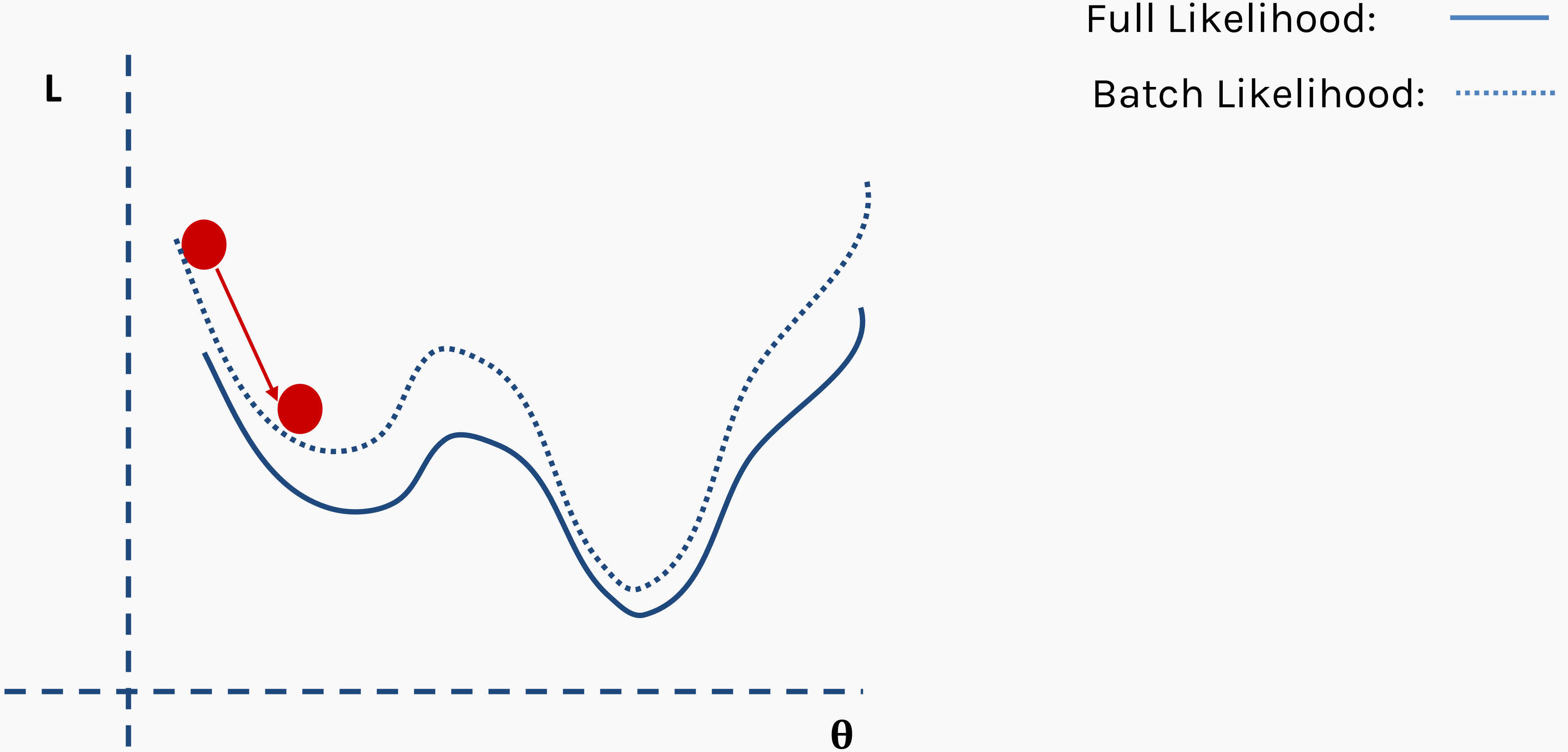
Batch and Stochastic Gradient Descent



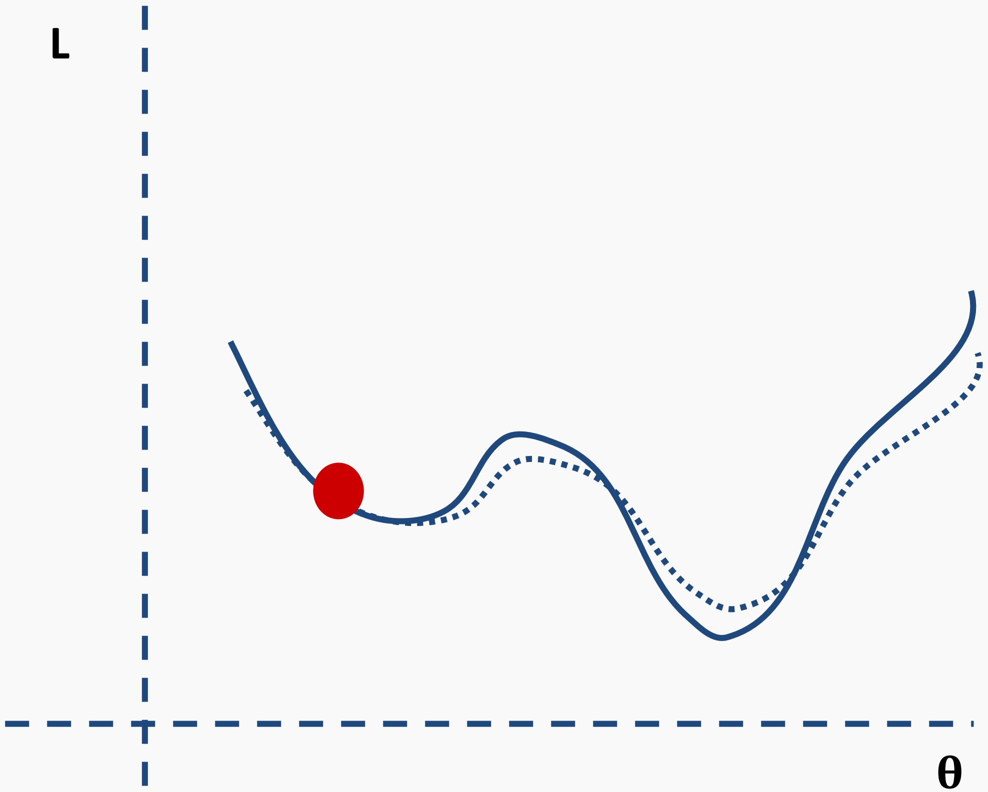
Full Likelihood:

Batch Likelihood:

Batch and Stochastic Gradient Descent



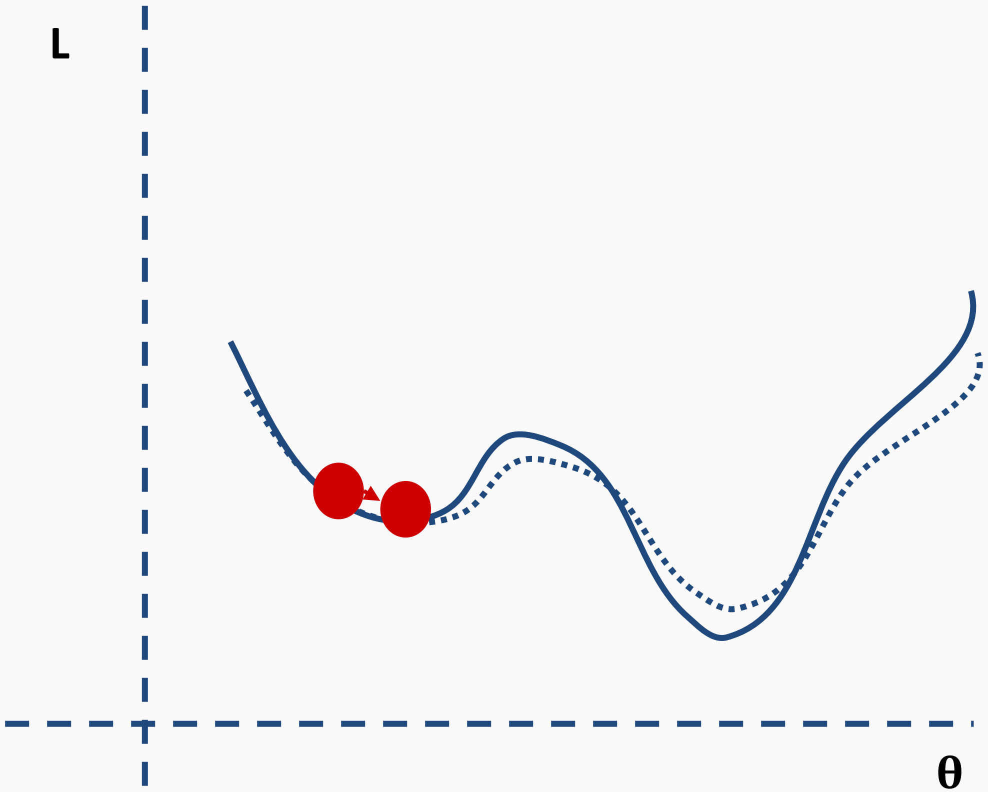
Batch and Stochastic Gradient Descent



Full Likelihood:

Batch Likelihood:

Batch and Stochastic Gradient Descent



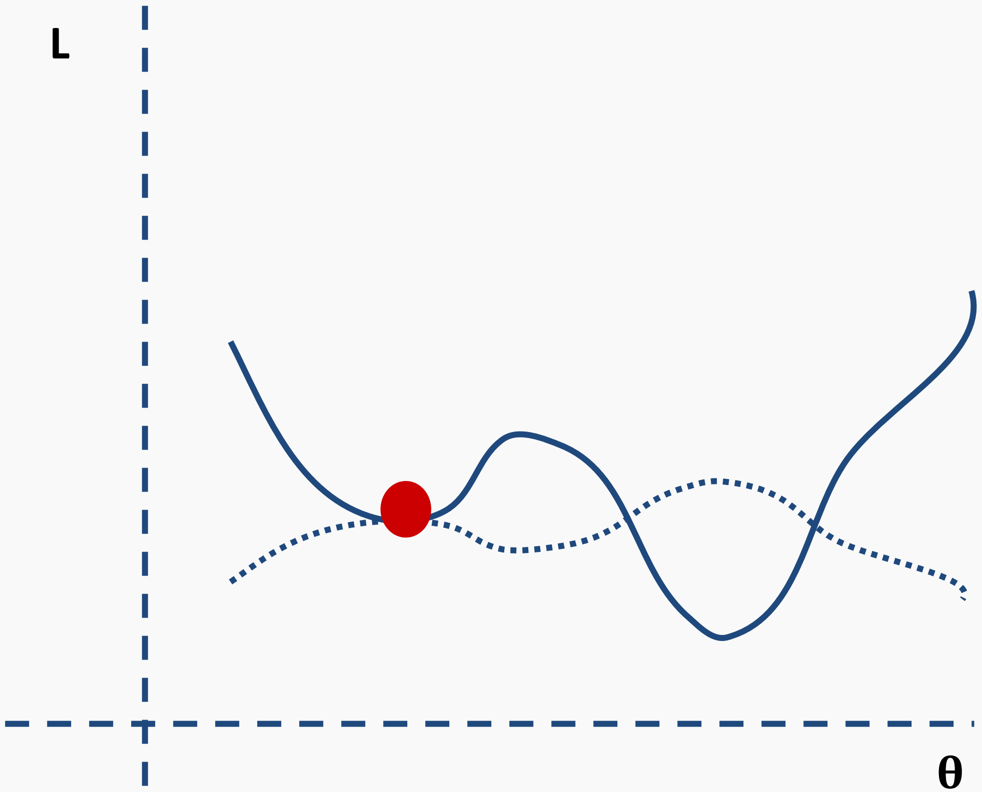
Full Likelihood:

Batch Likelihood:

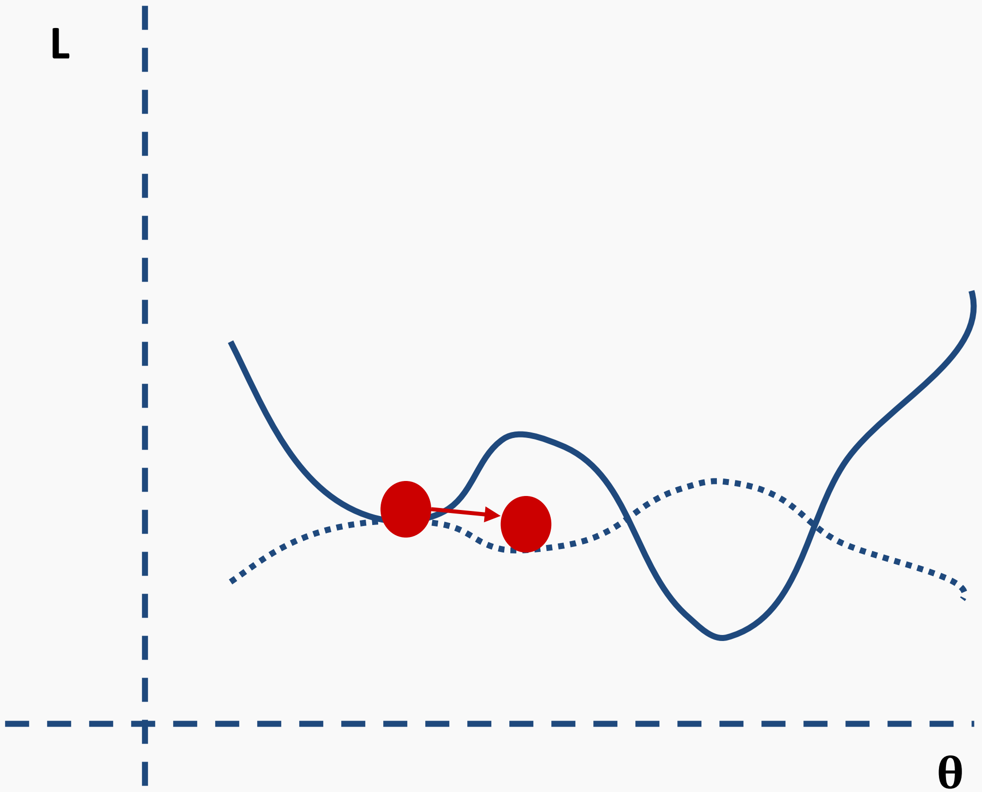
Batch and Stochastic Gradient Descent

Full Likelihood:

Batch Likelihood:



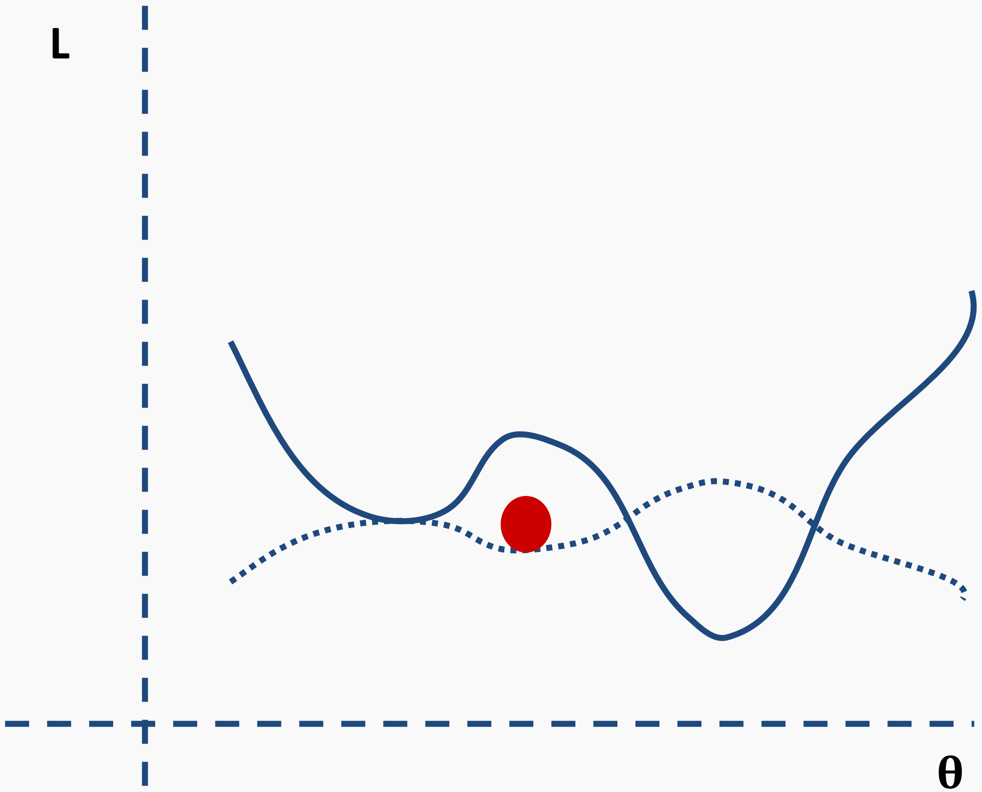
Batch and Stochastic Gradient Descent



Full Likelihood:

Batch Likelihood:

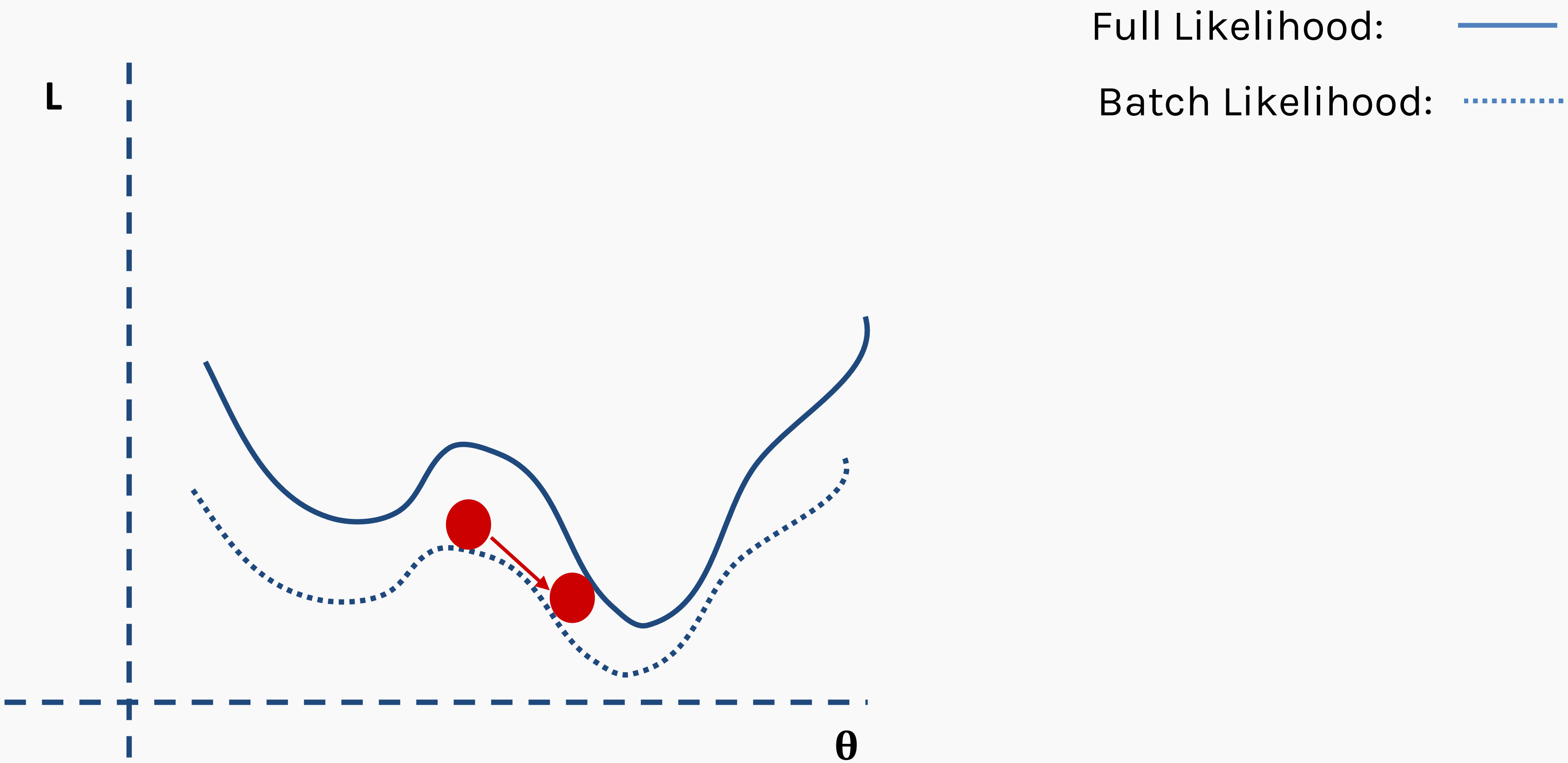
Batch and Stochastic Gradient Descent



Full Likelihood:

Batch Likelihood:

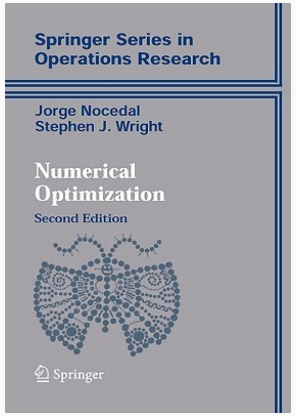
Batch and Stochastic Gradient Descent



Learning Rate

NEXT LECTURE

There are many alternative methods which address how to set or adjust the learning rate, using the derivative or second derivatives and or the momentum. To be discussed in the next lectures on NN.



* J. Nocedal y S. Wright, "Numerical optimization", Springer, 1999 🔗
* *TLDR*: J. Bullinaria, "Learning with Momentum, Conjugate Gradient Learning", 2015 🔗

Considerations

- We still need to derive the derivatives.
- ~~We need to know what is the learning rate or how to set it.~~
- ~~We need to avoid local minima.~~
- ~~Finally, the full likelihood function includes summing up all individual ‘*errors*’. Unless you are a statistician, this can be hundreds of thousands of examples.~~

Considerations

- **We still need to derive the derivatives.**
- ~~We need to know what is the learning rate or how to set it.~~
- ~~We need to avoid local minima.~~
- ~~Finally, the full likelihood function includes summing up all individual '*errors*'. Unless you are a statistician, this can be hundreds of thousands of examples.~~

Review of Feed Forward Artificial Neural Networks

Anatomy of a NN

Design choices

- Activation function
- Loss function
- Output units
- Architecture

Learning(more next lecture)

Basics ideas of optimizer

Backprop

Derivatives: Linear Regression

$$f = \sum_i (y_i - \beta_0 - \beta_1 x_i)^2$$

$$\frac{df}{d\beta_1} = 0 \Rightarrow 2\sum_i (y_i - \beta_0 - \beta_1 x_i)(-x_i)$$

$$-\sum_i x_i y_i + \beta_0 \sum_i x_i + \beta_1 \sum_i x_i^2 = 0$$

$$-\sum_i x_i y_i + (\bar{y} - \beta_1 \bar{x}) \sum_i x_i + \beta_1 \sum_i x_i^2 = 0$$

$$\beta_1 \left(\sum_i x_i^2 - n\bar{x}^2 \right) = \sum_i x_i y_i - n\bar{x}\bar{y}$$

$$\Rightarrow \beta_1 = \frac{\sum_i x_i y_i - n\bar{x}\bar{y}}{\sum_i x_i^2 - n\bar{x}^2}$$

$$\Rightarrow \beta_1 = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$$

$$\frac{df}{d\beta_0} = 0 \Rightarrow 2\sum_i (y_i - \beta_0 - \beta_1 x_i)$$

$$\sum_i y_i - \beta_0 n - \beta_1 \sum_i x_i = 0$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

Derivatives: Logistic Regression

Can we do it?

Wolfram Alpha can do it f

We need a general formalism to deal with these derivatives.

Backprop: Chain Rule

- Chain rule for computing gradients:

- $y = g(x) \qquad z = f(y) = f\big(g(x)\big)$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

$\boldsymbol{y} = g(\boldsymbol{x}) \qquad z = f(\boldsymbol{y}) = f\big(g(\boldsymbol{x})\big)$

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i}$$

- For longer chains $$\frac{\partial z}{\partial x_i} = \sum_{j_1} \cdots \sum_{j_m} \frac{\partial z}{\partial y_{j_1}} \cdots \frac{\partial y_{j_m}}{\partial x_i}$$

Logistic Regression derivatives

For logistic regression, the –ve log of the likelihood is:

$$\mathcal{L} = \sum_i \mathcal{L}_i = -\sum_i \log L_i \; = -\sum_i [y_i \log p_i + (1 - y_i) \log(1 - p_i)]$$

$$\mathcal{L}_i = -y_i \log \frac{1}{1 + e^{-W^T X}} - (1 - y_i) \log(1 - \frac{1}{1 + e^{-W^T X}})$$

To simplify the analysis let us split it into two parts,

$$\mathcal{L}_i = \mathcal{L}_i^A + \mathcal{L}_i^B$$

So the derivative with respect to W is:

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_i \frac{\partial \mathcal{L}_i}{\partial W} = \sum_i (\frac{\partial \mathcal{L}_i^A}{\partial W} + \frac{\partial \mathcal{L}_i^B}{\partial W})$$

$$\mathcal{L}_i^A = -y_i \log \frac{1}{1+e^{-W^T X}}$$

Variables	Partial derivatives	Partial derivatives
$\xi_1 = -W^T X$	$\frac{\partial \xi_1}{\partial W} = -X$	$\frac{\partial \xi_1}{\partial W} = -X$
$\xi_2 = e^{\xi_1} = e^{-W^T X}$	$\frac{\partial \xi_2}{\partial \xi_1} = e^{\xi_1}$	$\frac{\partial \xi_2}{\partial \xi_1} = e^{-W^T X}$
$\xi_3 = 1 + \xi_2 = 1 + e^{-W^T X}$	$\frac{\partial \xi_3}{\partial \xi_2} = 1$	$\frac{\partial \xi_3}{\partial \xi_2} = 1$
$\xi_4 = \frac{1}{\xi_3} = \frac{1}{1+e^{-W^T X}} = p$	$\frac{\partial \xi_4}{\partial \xi_3} = -\frac{1}{\xi_3^2}$	$\frac{\partial \xi_4}{\partial \xi_3} = -\frac{1}{\left(1+e^{-W^T X}\right)^2}$
$\xi_5 = \log \xi_4 = \log p = \log \frac{1}{1+e^{-W^T X}}$	$\frac{\partial \xi_5}{\partial \xi_4} = \frac{1}{\xi_4}$	$\frac{\partial \xi_5}{\partial \xi_4} = 1 + e^{-W^T X}$
$\mathcal{L}_i^A = -y\xi_5$	$\frac{\partial \mathcal{L}}{\partial \xi_5} = -y$	$\frac{\partial \mathcal{L}}{\partial \xi_5} = -y$
$\frac{\partial \mathcal{L}_i^A}{\partial W} = \frac{\partial \mathcal{L}_i}{\partial \xi_5}\frac{\partial \xi_5}{\partial \xi_4}\frac{\partial \xi_4}{\partial \xi_3}\frac{\partial \xi_3}{\partial \xi_2}\frac{\partial \xi_2}{\partial \xi_1}\frac{\partial \xi_1}{\partial W}$		$\frac{\partial \mathcal{L}_i^A}{\partial W} = -yXe^{-W^T X}\frac{1}{\left(1+e^{-W^T X}\right)}$

$$\mathcal{L}_i^B = -(1-y_i)\log[1-\frac{1}{1+e^{-W^TX}}]$$

Variables	derivatives	Partial derivatives wrt to X,W
$\xi_1 = -W^TX$	$\frac{\partial \xi_1}{\partial W} = -X$	$\frac{\partial \xi_1}{\partial W} = -X$
$\xi_2 = e^{\xi_1} = e^{-W^TX}$	$\frac{\partial \xi_2}{\partial \xi_1} = e^{\xi_1}$	$\frac{\partial \xi_2}{\partial \xi_1} = e^{-W^TX}$
$\xi_3 = 1+\xi_2 = 1+e^{-W^TX}$	$\frac{\partial \xi_3}{\partial \xi_2} = 1$	$\frac{\partial \xi_3}{\partial 2}$ =1
$\xi_4 = \frac{1}{\xi_3} = \frac{1}{1+e^{-W^TX}} = p$	$\frac{\partial \xi_4}{\partial \xi_3} = -\frac{1}{\xi_3^2}$	$\frac{\partial \xi_4}{\partial \xi_3} = -\frac{1}{\left(1+e^{-W^TX}\right)^2}$
$\xi_5 = 1-\xi_4 = 1-\frac{1}{1+e^{-W^TX}}$	$\frac{\partial \xi_5}{\partial \xi_4} = -1$	$\frac{\partial \xi_5}{\partial \xi_4}$ =-1
$\xi_6 = \log \xi_5 = \log(1-p) \; = \log \frac{1}{1+e^{-W^TX}}$	$\frac{\partial \xi_6}{\partial \xi_5} = \frac{1}{\xi_5}$	$\frac{\partial \xi_6}{\partial \xi_5} = \frac{1+e^{-W^TX}}{e^{-W^TX}}$
$\mathcal{L}_i^B = (1-y)\xi_6$	$\frac{\partial \mathcal{L}}{\partial \xi_6} = 1-y$	$\frac{\partial \mathcal{L}}{\partial \xi_6} = 1-y$
$\frac{\partial \mathcal{L}_i^B}{\partial W} = \frac{\partial \mathcal{L}_i^B}{\partial \xi_6}\frac{\partial \xi_6}{\partial \xi_5}\frac{\partial \xi_5}{\partial \xi_4}\frac{\partial \xi_4}{\partial \xi_3}\frac{\partial \xi_3}{\partial \xi_2}\frac{\partial \xi_2}{\partial \xi_1}\frac{\partial \xi_1}{\partial W}$		$\frac{\partial \mathcal{L}_i^B}{\partial W} = (1-y)X\frac{1}{\left(1+e^{-W^TX}\right)}$

Backpropagation: Logistic Regression Revisited

$$X \longrightarrow \boxed{\text{Affine}} \longrightarrow h = \beta_0 + \beta_1 X \longrightarrow \boxed{\text{Activation}} \longrightarrow p = \frac{1}{1+e^{-h}} \longrightarrow \boxed{\text{Loss Fun}} \longrightarrow \mathcal{L}(\beta) = \sum_i^n \mathcal{L}_i(\beta)$$

$$\frac{\partial \mathcal{L}}{\partial p}\frac{\partial p}{\partial h}\frac{\partial h}{\partial \beta} \longleftarrow \frac{\partial \mathcal{L}}{\partial p}\frac{\partial p}{\partial h} \longleftarrow \frac{\partial \mathcal{L}}{\partial p}$$

$$\frac{\partial h}{\partial \beta_1} = X, \frac{d\mathcal{L}}{d\beta_0} = 1$$

$$\frac{\partial p}{\partial h} = \sigma(h)(1-\sigma(h))$$

$$\frac{\partial \mathcal{L}}{\partial p} = -y\frac{1}{p} - (1-y)\frac{1}{1-p}$$

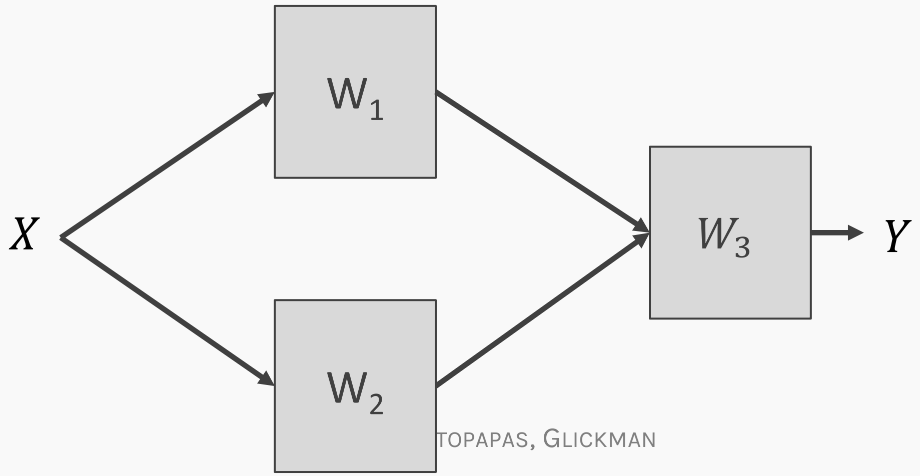
$$\frac{\partial \mathcal{L}}{\partial \beta_1} = -X\sigma(h)\big(1-\sigma(h)\big)[y\frac{1}{p} + (1-y)\frac{1}{1-p}]$$

$$\frac{\partial \mathcal{L}}{\partial \beta_0} = -\sigma(h)\big(1-\sigma(h)\big)[\,y\frac{1}{p} + (1-y)\frac{1}{1-p}]$$

Backpropagation

1. Derivatives need to be evaluated at some values of X, y and W.
2. But since we have an expression, we can build a function that takes as input X, y, W and returns the derivatives and then we can use gradient descent to update.
3. This approach works well but it does not generalize. For example if the network is changed, we need to write a new function to evaluate the derivatives.

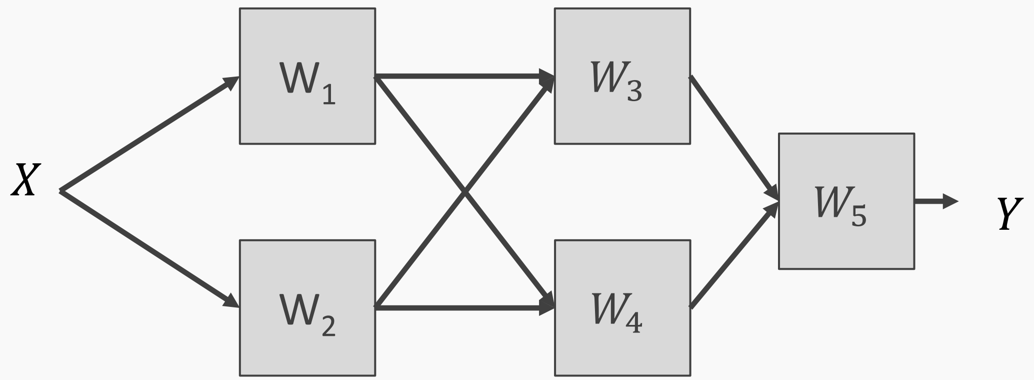
For example this network will need a different function for the derivatives,



Backpropagation

1. Derivatives need to be evaluated at some values of X, y and W.
2. But since we have an expression, we can build a function that takes as input X, y, W and returns the derivatives and then we can use gradient descent to update.
3. This approach works well but it does not generalize. For example if the network is changed, we need to write a new function to evaluate the derivatives.

Than this one:



Backpropagation. Pavlos game #456

Need to find a formalism to calculate the derivatives of the loss function wrt to weights that is:

1. Flexible enough that adding a node or a layer or changing something in the network won't require to re-derive the functional form from scratch.
2. It is exact.
3. It is computationally efficient.

Hints:

1. Remember we only need to evaluate the derivatives at X_i, y_i and $W^{(k)}$.
2. We should take advantage of the chain rule we learned before

Idea 1: Evaluate the derivative at: $X=\{3\}, y=1, W=3$

Variables	derivatives	Value of the variable	Value of the partial derivative	$\frac{d\xi_n}{dW}$
$\xi_1 = -W^T X$	$\frac{\partial \xi_1}{\partial W} = -X$	-9	-3	-3
$\xi_2 = e^{\xi_1} = e^{-W^T X}$	$\frac{\partial \xi_2}{\partial \xi_1} = e^{\xi_1}$	e^{-9}	e^{-9}	-3e^{-9}
$\xi_3 = 1 + \xi_2 = 1 + e^{-W^T X}$	$\frac{\partial \xi_3}{\partial \xi_2} = 1$	1+e^{-9}	1	-3e^{-9}
$\xi_4 = \frac{1}{\xi_3} = \frac{1}{1+e^{-W^T X}} = p$	$\frac{\partial \xi_4}{\partial \xi_3} = -\frac{1}{\xi_3^2}$	$\frac{1}{1+e^{-9}}$	$\left(\frac{1}{1+e^{-9}}\right)^2$	-3$e^{-9}\left(\frac{1}{1+e^{-9}}\right)^2$
$\xi_5 = \log \xi_4 = \log p = \log \frac{1}{1+e^{-W^T X}}$	$\frac{\partial \xi_5}{\partial \xi_4} = \frac{1}{\xi_4}$	$\log \frac{1}{1+e^{-9}}$	$1+e^{-9}$	-3$e^{-9}\left(\frac{1}{1+e^{-9}}\right)$
$\mathcal{L}_i^A = -y\xi_5$	$\frac{\partial \mathcal{L}}{\partial \xi_5} = -y$	$-\log \frac{1}{1+e^{-9}}$	-1	3$e^{-9}\left(\frac{1}{1+e^{-9}}\right)$
$\frac{\partial \mathcal{L}_i^A}{\partial W} = \frac{\partial \mathcal{L}_i}{\partial \xi_5}\frac{\partial \xi_5}{\partial \xi_4}\frac{\partial \xi_4}{\partial \xi_3}\frac{\partial \xi_3}{\partial \xi_2}\frac{\partial \xi_2}{\partial \xi_1}\frac{\partial \xi_1}{\partial W}$			-3	0.00037018372

Basic functions

We still need to derive **derivatives** ☹

Variables	derivatives	Value of the variable	Value of the partial derivative	$\frac{d\xi_n}{dW}$
$\xi_1 = -W^T X$	$\frac{\partial \xi_1}{\partial W} = -X$	-9	-3	-3
$\xi_2 = e^{\xi_1} = e^{-W^T X}$	$\frac{\partial \xi_2}{d\partial \xi_1} = e^{\xi_1}$	e^{-9}	e^{-9}	-3e^{-9}
$\xi_3 = 1 + \xi_2 = 1 + e^{-W^T X}$	$\frac{\partial \xi_3}{\partial \xi_2} = 1$	1+e^{-9}	1	-3e^{-9}
$\xi_4 = \frac{1}{\xi_3} = \frac{1}{1 + e^{-W^T X}} = p$	$\frac{\partial \xi_4}{\partial \xi_3} = -\frac{1}{\xi_3^2}$	$\frac{1}{1 + e^{-9}}$	$\left(\frac{1}{1 + e^{-9}}\right)^2$	-3$e^{-9}\left(\frac{1}{1+e^{-9}}\right)^2$
$\xi_5 = \log \xi_4 = \log p \; = \log \frac{1}{1 + e^{-W^T X}}$	$\frac{\partial \xi_5}{\partial \xi_4} = \frac{1}{\xi_4}$	$\log \frac{1}{1 + e^{-9}}$	$1 + e^{-9}$	-3$e^{-9}\left(\frac{1}{1+e^{-9}}\right)$
$\mathcal{L}_i^A = -y\xi_5$	$\frac{\partial \mathcal{L}}{\partial \xi_5} = -y$	$-\log \frac{1}{1 + e^{-9}}$	-1	3$e^{-9}\left(\frac{1}{1+e^{-9}}\right)$
$\frac{\partial \mathcal{L}_i^A}{\partial W} = \frac{\partial \mathcal{L}_i}{\partial \xi_5}\frac{\partial \xi_5}{\partial \xi_4}\frac{\partial \xi_4}{\partial \xi_3}\frac{\partial \xi_3}{\partial \xi_2}\frac{\partial \xi_2}{\partial \xi_1}\frac{\partial \xi_1}{\partial W}$			-3	0.00037018372

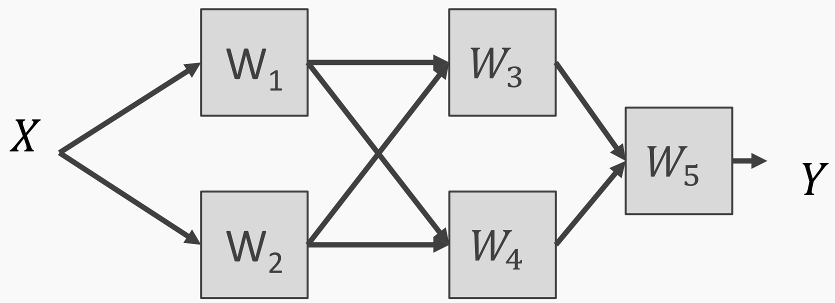
Basic functions

Notice though those are basic functions that my grandparent can do

$\xi_0 = X$	$\frac{\partial \xi_0}{\partial X} = 1$	`def x0(x):` `    return X`	`def derx0():` `    return 1`
$\xi_1 = -W^T \xi_0$	$\frac{\partial \xi_1}{\partial W} = -X$	`def x1(a,x):` `    return -a*X`	`def derx1(a,x):` `    return -a`
$\xi_2 = \mathrm{e}^{\xi_1}$	$\frac{\partial \xi_2}{\partial \xi_1} = e^{\xi_1}$	`def x2(x):` `    return np.exp(x)`	`def derx2(x):` `        return np.exp(x)`
$\xi_3 = 1 + \xi_2$	$\frac{\partial \xi_3}{\partial \xi_2} = 1$	`def x3(x):` `    return 1+x`	`def derx3(x):` `        return 1`
$\xi_4 = \frac{1}{\xi_3}$	$\frac{\partial \xi_4}{\partial \xi_3} = -\frac{1}{\xi_3^2}$	`def der1(x):` `    return 1/(x)`	`def derx4(x):` `        return  -(1/x)**(2)`
$\xi_5 = \log \xi_4$	$\frac{\partial \xi_5}{\partial \xi_4} = \frac{1}{\xi_4}$	`def der1(x):` `    return np.log(x)`	`def derx5(x)` `        return 1/x`
$\mathcal{L}_i^A = -y\xi_5$	$\frac{\partial \mathcal{L}}{\partial \xi_5} = -y$	`def der1(y,x):` `    return -y*x`	`def derL(y):` `    return -y`

Autograd: Auto-differentiation

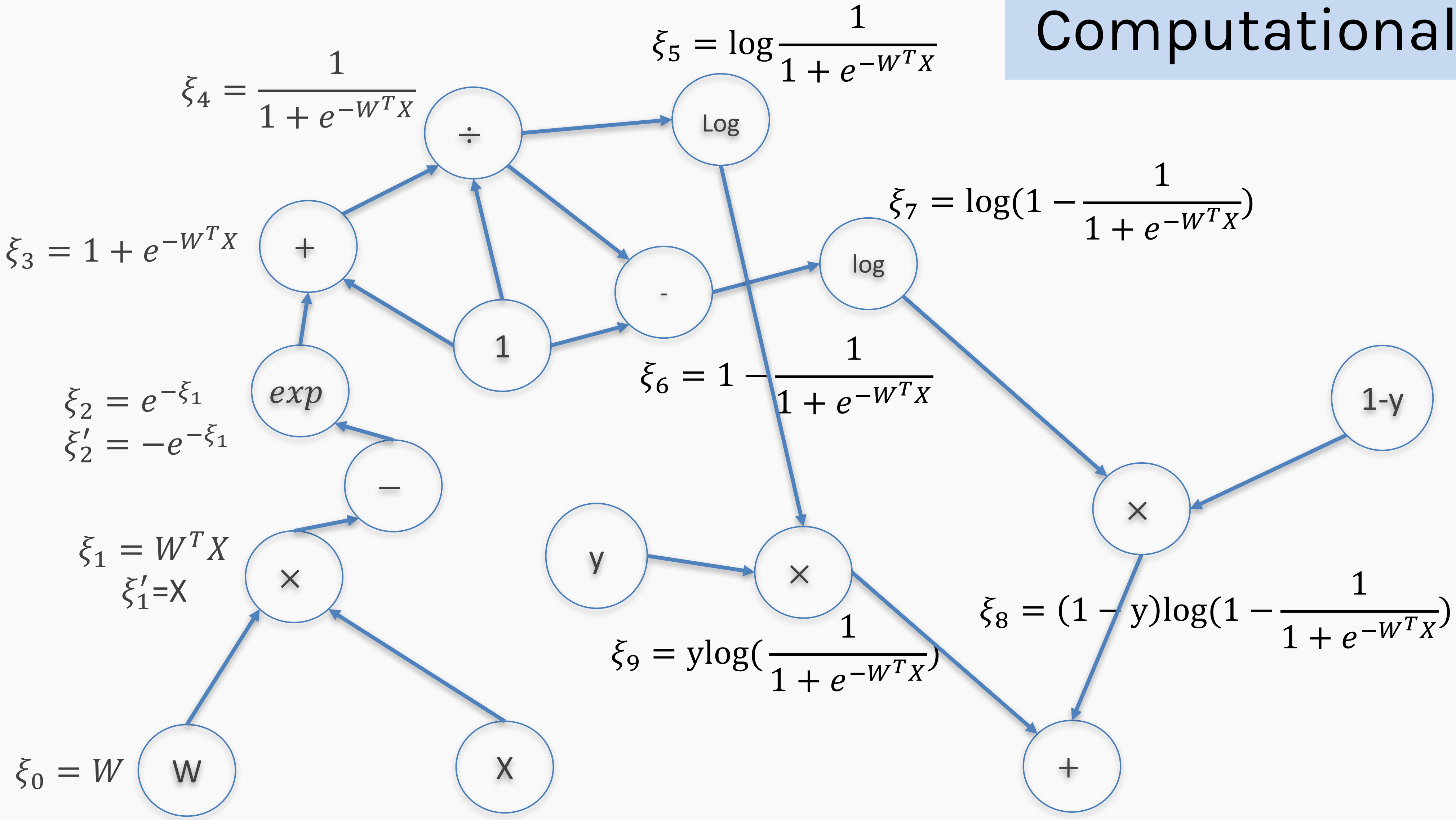
1. We specify the network structure



2. We create the computational graph …

What is computational graph?

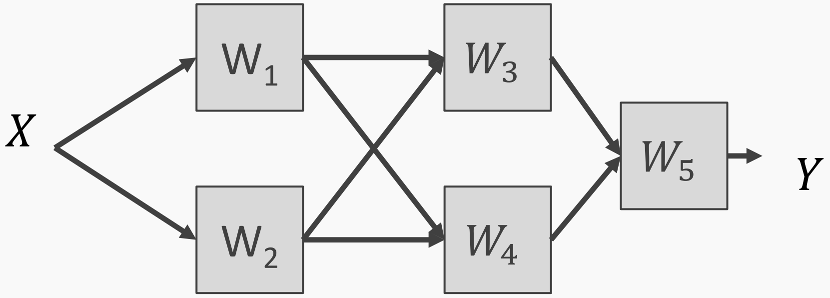
Computational Graph



$$-\mathcal{L} = \xi_9 = \text{y}\log(\frac{1}{1+e^{-W^TX}}) + (1-\text{y})\log(1-\frac{1}{1+e^{-W^TX}})$$

Autograd (cont)

1. We specify the network structure



- We create the computational graph.
- At each node of the graph we build two functions: the evaluation of the variable and its partial derivative with respect to the previous variable (as shown in the table few slides back)
- Now we can either go forward or backward depending on the situation. In general, forward is easier to implement and to understand. The difference is clearer when there are multiple nodes per layer.

Forward mode: Evaluate the derivative at: $X=\{3\}, y=1, W=3$

Variables	derivatives	Value of the variable	Value of the partial derivative	$\frac{d\mathcal{L}}{d\xi_n}$
$\xi_1 = -W^T X$	$\frac{\partial \xi_1}{\partial W} = -X$	-9	-3	-3
$\xi_2 = e^{\xi_1} = e^{-W^T X}$	$\frac{\partial \xi_2}{\partial \xi_1} = e^{\xi_1}$	e^{-9}	e^{-9}	-3e^{-9}
$\xi_3 = 1 + \xi_2 = 1 + e^{-W^T X}$	$\frac{\partial \xi_3}{\partial \xi_2} = 1$	1+e^{-9}	1	-3e^{-9}
$\xi_4 = \frac{1}{\xi_3} = \frac{1}{1+e^{-W^T X}} = p$	$\frac{\partial \xi_4}{\partial \xi_3} = -\frac{1}{\xi_3^2}$	$\frac{1}{1+e^{-9}}$	$\left(\frac{1}{1+e^{-9}}\right)^2$	-3$e^{-9}\left(\frac{1}{1+e^{-9}}\right)^2$
$\xi_5 = \log \xi_4 = \log p = \log \frac{1}{1+e^{-W^T X}}$	$\frac{\partial \xi_5}{\partial \xi_4} = \frac{1}{\xi_4}$	$\log \frac{1}{1+e^{-9}}$	$1 + e^{-9}$	-3$e^{-9}\left(\frac{1}{1+e^{-9}}\right)$
$\mathcal{L}_i^A = -y\xi_5$	$\frac{\partial \mathcal{L}}{\partial \xi_5} = -y$	$-\log \frac{1}{1+e^{-9}}$	-1	$3e^{-9}\left(\frac{1}{1+e^{-9}}\right)$
$\frac{\partial \mathcal{L}_i^A}{\partial W} = \frac{\partial \mathcal{L}_i}{\partial \xi_5}\frac{\partial \xi_5}{\partial \xi_4}\frac{\partial \xi_4}{\partial \xi_3}\frac{\partial \xi_3}{\partial \xi_2}\frac{\partial \xi_2}{\partial \xi_1}\frac{\partial \xi_1}{\partial W}$			-3	0.00037018372

Backward mode: Evaluate the derivative at: $X=\{3\}, y=1, W=3$

Variables	derivatives	Value of the variable	Value of the partial derivative
$\xi_1 = -W^T X$	$\frac{\partial \xi_1}{\partial W} = -X$	-9	-3
$\xi_2 = e^{\xi_1} = e^{-W^T X}$	$\frac{\partial \xi_2}{\partial \xi_1} = e^{\xi_1}$	e^{-9}	e^{-9}
$\xi_3 = 1 + \xi_2 = 1 + e^{-W^T X}$	$\frac{\partial \xi_3}{\partial \xi_2} = 1$	1+e^{-9}	1
$\xi_4 = \frac{1}{\xi_3} = \frac{1}{1 + e^{-W^T X}} = p$	$\frac{\partial \xi_4}{\partial \xi_3} = -\frac{1}{\xi_3^2}$	$\frac{1}{1+e^{-9}}$	$\left(\frac{1}{1+e^{-9}}\right)^2$
$\xi_5 = \log \xi_4 = \log p \ = \log \frac{1}{1 + e^{-W^T X}}$	$\frac{\partial \xi_5}{\partial \xi_4} = \frac{1}{\xi_4}$	$\log \frac{1}{1+e^{-9}}$	$1 + e^{-9}$
$\mathcal{L}_i^A = -y\xi_5$	$\frac{\partial \mathcal{L}}{\partial \xi_5} = -y$	$-\log \frac{1}{1+e^{-9}}$	-1
$\frac{\partial \mathcal{L}_i^A}{\partial W} = \frac{\partial \mathcal{L}_i}{\partial \xi_5}\frac{\partial \xi_5}{\partial \xi_4}\frac{\partial \xi_4}{\partial \xi_3}\frac{\partial \xi_3}{\partial \xi_2}\frac{\partial \xi_2}{\partial \xi_1}\frac{\partial \xi_1}{\partial W}$			Type equation here.

Store all these values