Lecture 12: Perceptron and Back Propagation

CS109A Introduction to Data Science

Pavlos Protopapas and Kevin Rader

Outline

1. Review of Classification and Logistic Regression
2. Introduction to Optimization
   - Gradient Descent
   - Stochastic Gradient Descent
3. Single Neuron Network (‘Perceptron’)
4. Multi-Layer Perceptron
5. Back Propagation

Outline

1. **Review of Classification and Logistic Regression**
2. Introduction to Optimization
   - Gradient Descent
   - Stochastic Gradient Descent
3. Single Neuron Network (‘Perceptron’)
4. Multi-Layer Perceptron
5. Back Propagation

Classification and Logistic Regression

Classification

Methods that are centered around modeling and prediction of a **quantitative** response variable (ex, number of taxi pickups, number of bike rentals, etc) are called **regressions** (and Ridge, LASSO, etc).

When the response variable is **categorical**, then the problem is no longer called a regression problem but is instead labeled as a **classification problem**.

The goal is to attempt to classify each observation into a category (aka, class or cluster) defined by Y, based on a set of predictor variables X.

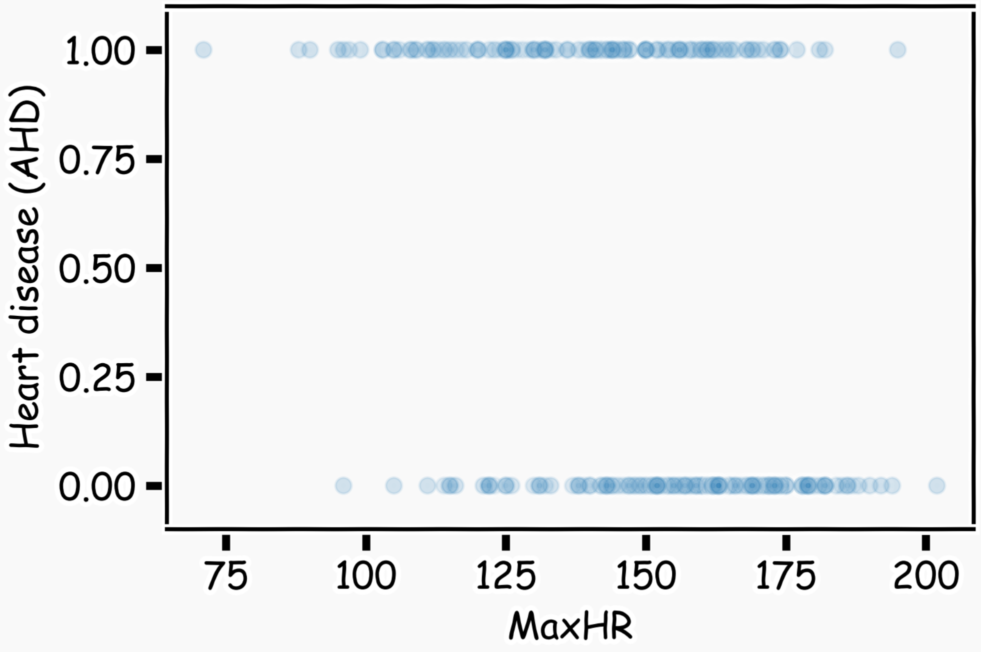
Heart Data

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

Heart Data: logistic estimation

We'd like to predict whether or not a person has a heart disease. And we'd like to make this prediction, for now, just based on the MaxHR.



Logistic Regression

Logistic Regression addresses the problem of estimating a probability, $P(y=1)$, given an input X. The logistic regression model uses a function, called the ***logistic*** function, to model $P(y=1)$:

$$P(Y=1) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}} = \frac{1}{1+e^{-(\beta_0+\beta_1 X)}}$$

Logistic Regression

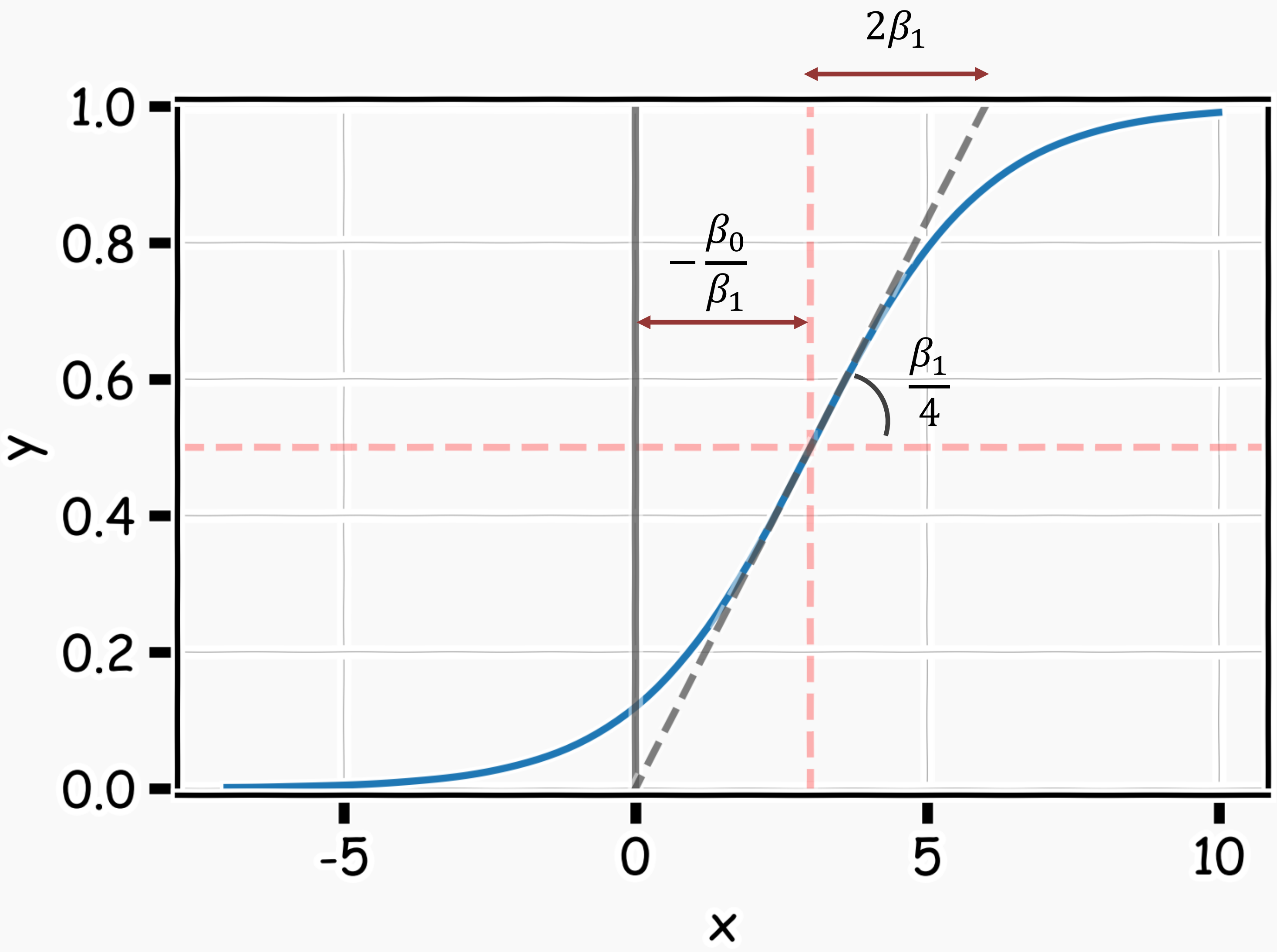
As a result the model will predict $P(y=1)$ with an S-shaped curve, which is the general shape of the logistic function.

β_0 shifts the curve right or left by $\mathrm{c} = -\frac{\beta_0}{\beta_1}$.

β_1 controls how steep the S-shaped curve is distance from ½ to ~1 or ½\ to ~0 to ½ is $\frac{2}{\beta_1}$

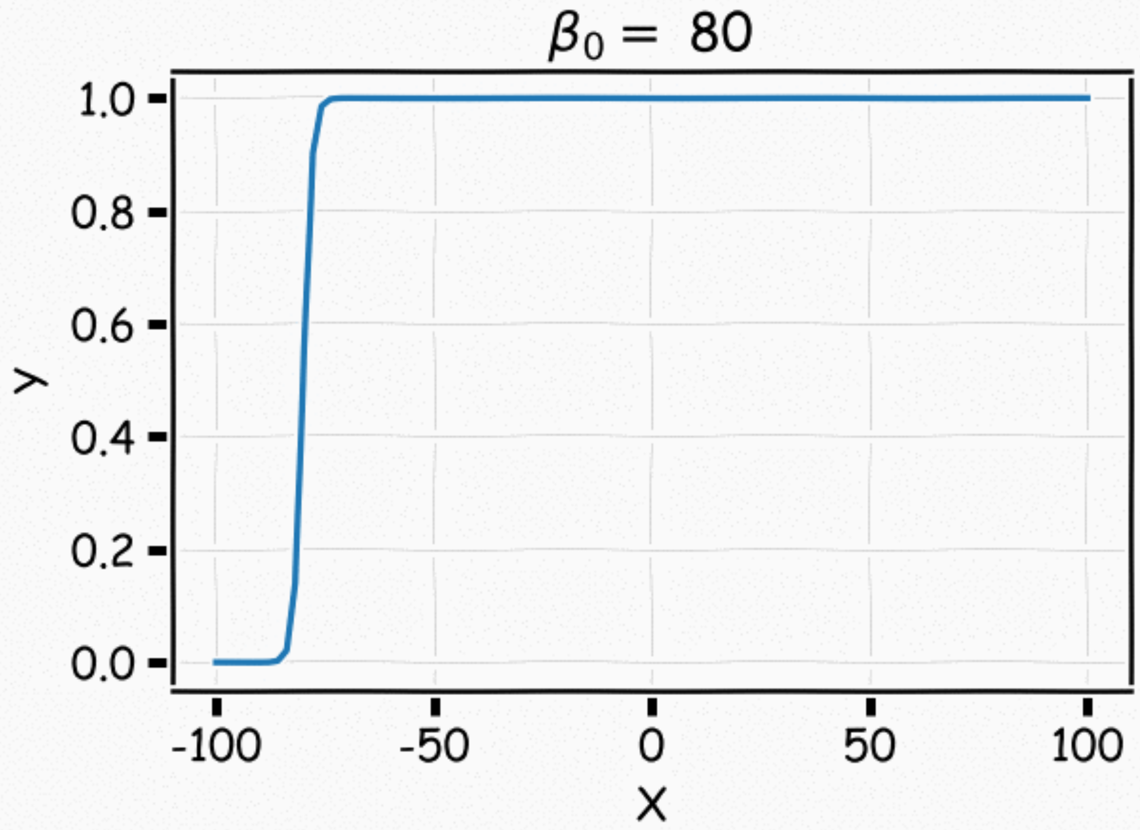
Note: if β_1 is positive, then the predicted $P(y=1)$ goes from zero for small values of X to one for large values of X and if β_1 is negative, then has the $P(y=1)$ opposite association.

Logistic Regression



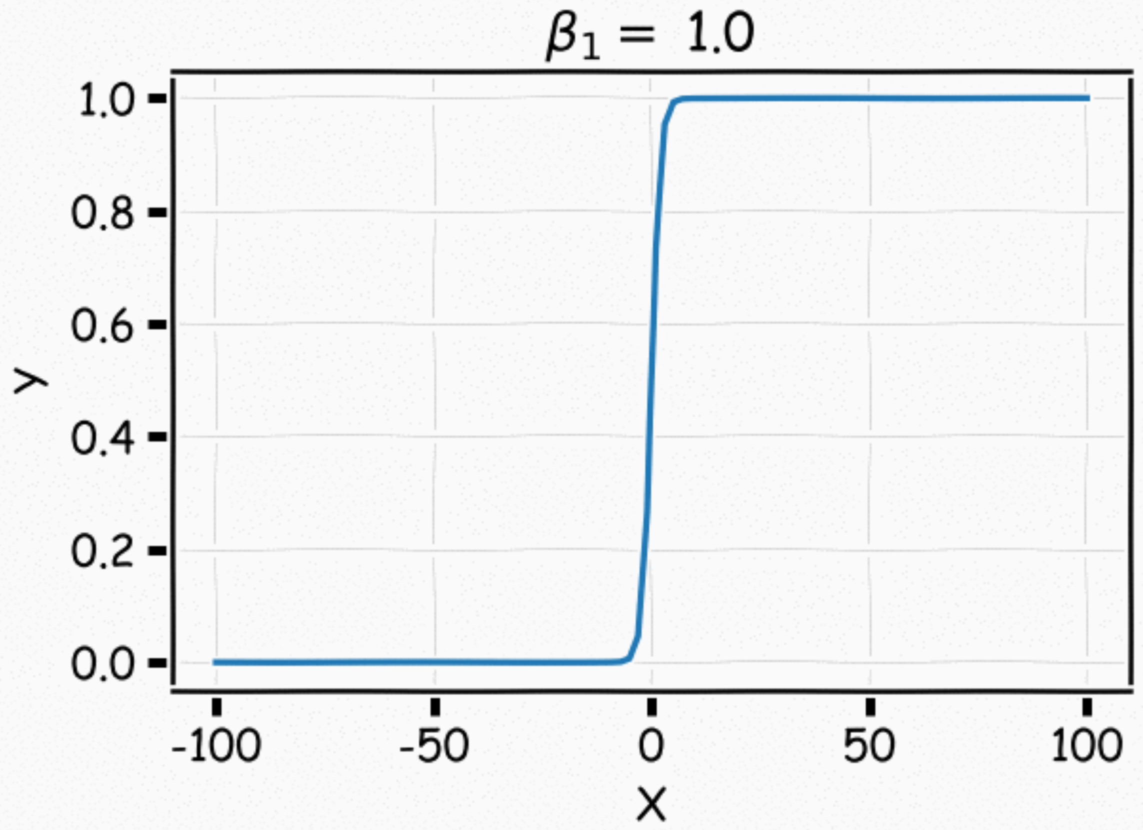
Logistic Regression

$$\boxed{P(Y=1) = \frac{1}{1+e^{-(\beta_0+\beta_1 X)}}}$$



Logistic Regression

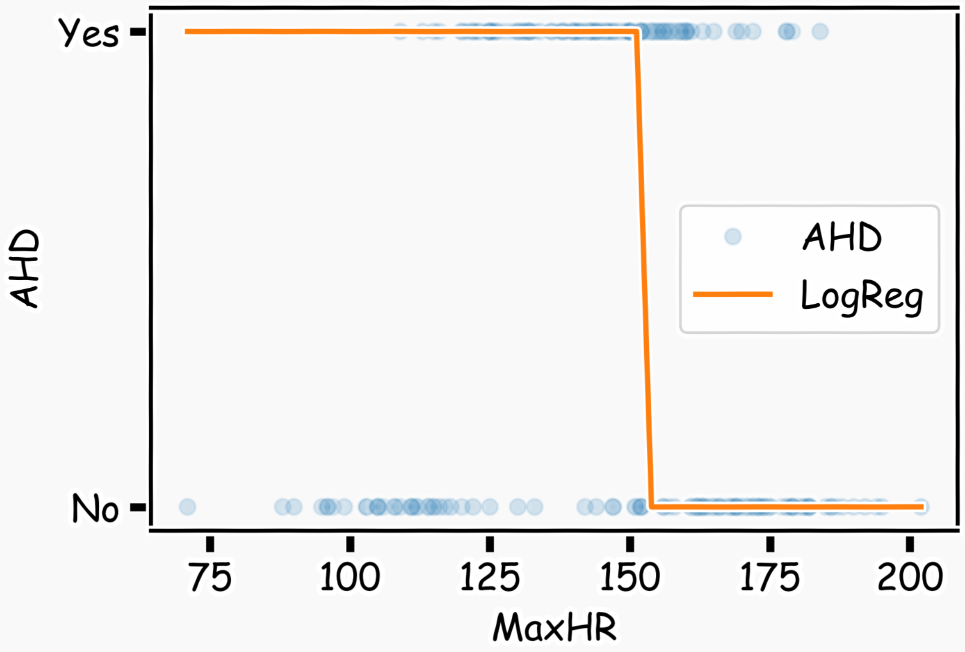
$$P(Y=1) = \frac{1}{1+e^{-(\beta_0+\beta_1 X)}}$$



Estimating the coefficients for Logistic Regression

Find the coefficients that minimize the loss function

$$\mathcal{L}(\beta_0, \beta_1) = -\sum_i [y_i \log p_i + (1 - y_i)\log(1 - p_i)]$$



But what is the idea?

Start with Regression or Logistic Regression

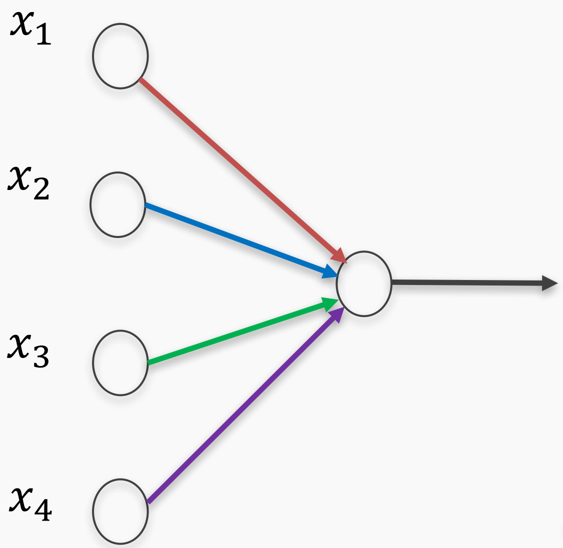
Classification

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

Regression

$$f(X) = W^T X$$

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



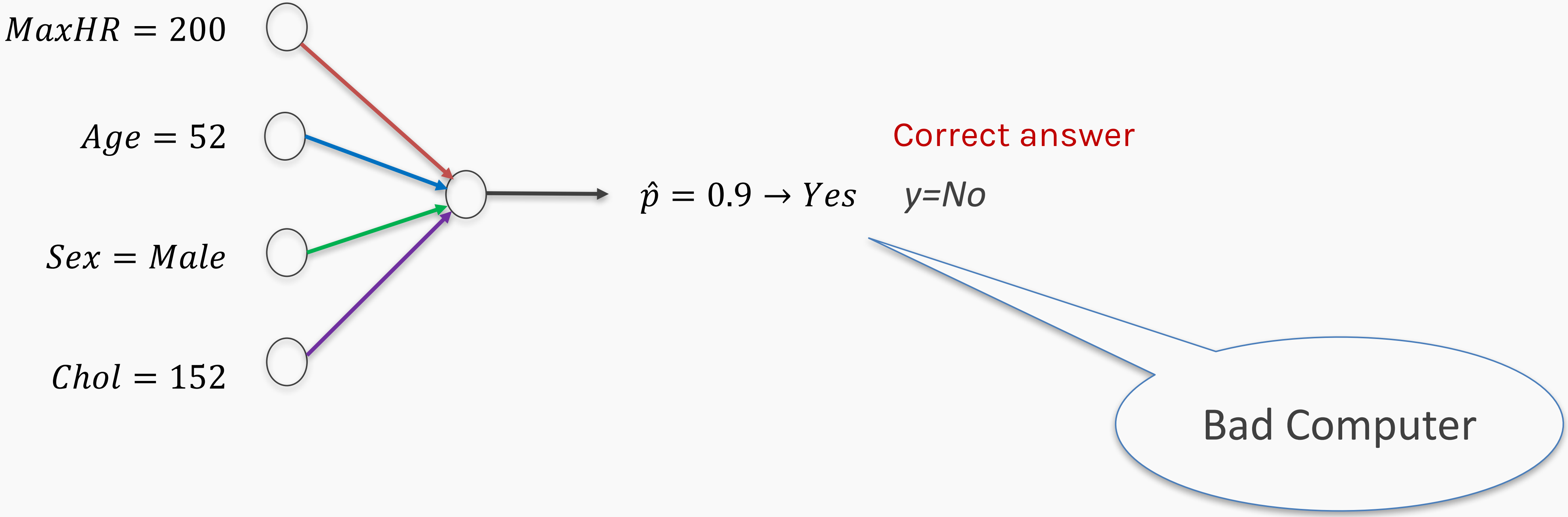
$$Y = f(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4)$$

Intercept or **Bias**

Coefficients or **Weights**

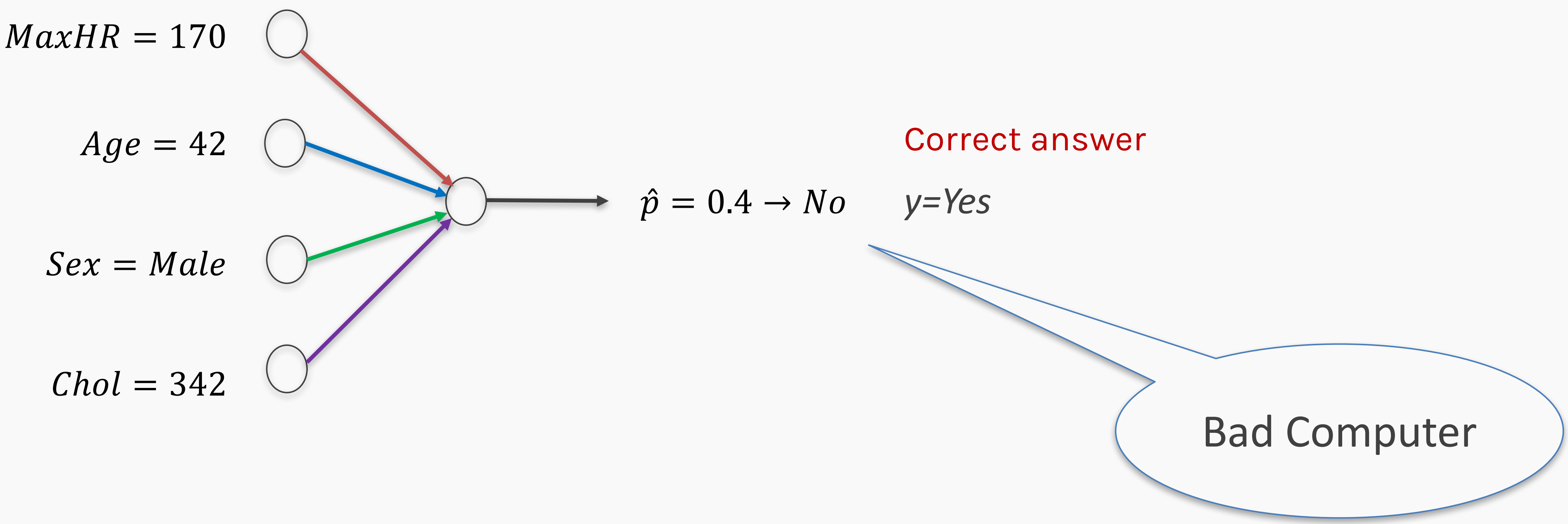
But what is the idea?

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?

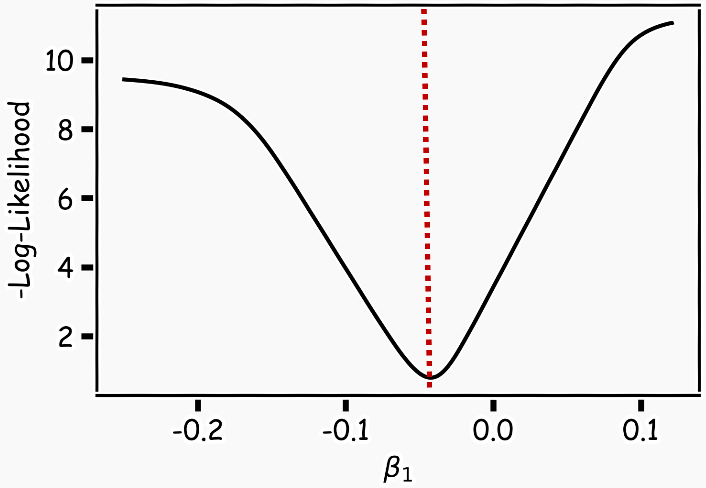
- **Loss Function:** Takes all of these results and averages them and tells us how bad or good the computer or those weights are.
- Telling the computer how **bad** or **good** is, does not help.
- You want to tell it how to change those weights so it gets better.

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

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

Minimizing the Loss function

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

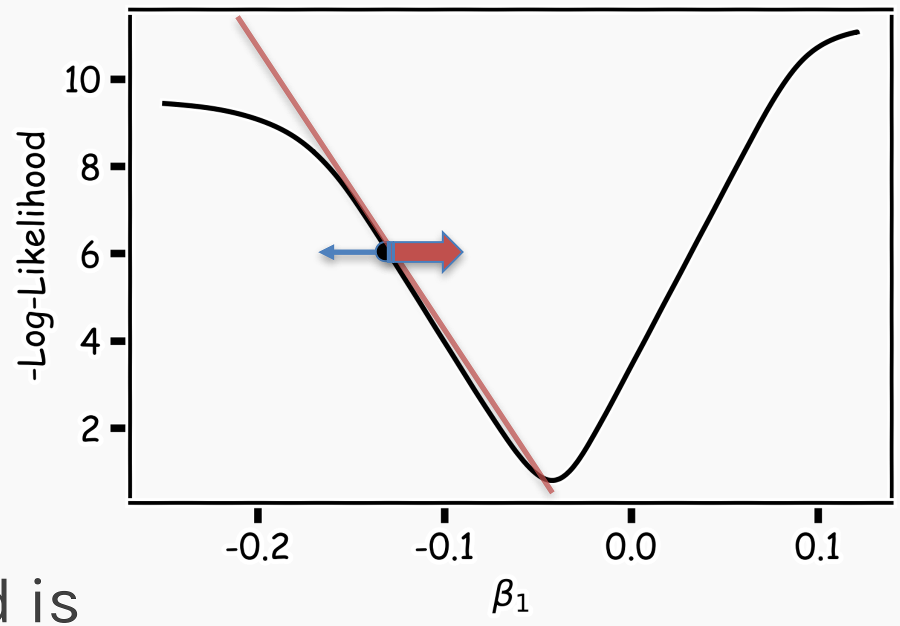


To find the optimal point of a function $\mathcal{L}(W)$

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

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:

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

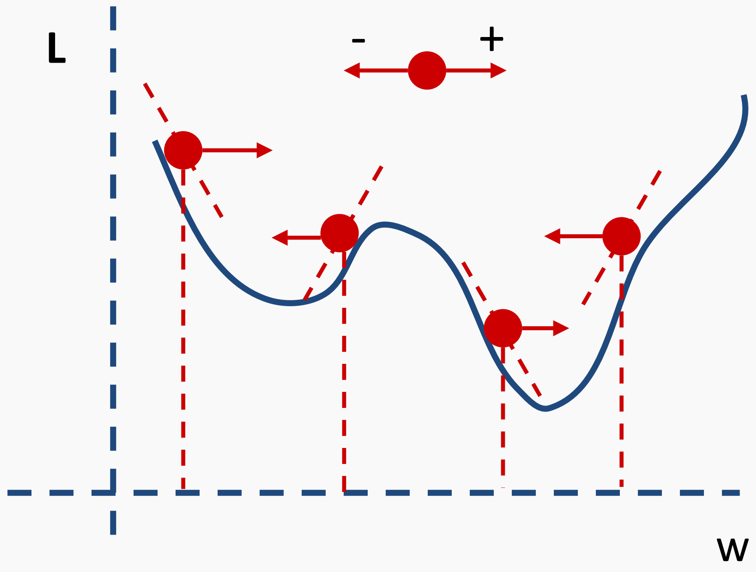
Change to more conventional notation:

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

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.

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.

Derivatives: Memories from middle school

Linear Regression

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

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

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

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

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

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

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

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

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

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

Logistic Regression Derivatives

Can we do it?

Wolfram Alpha can do it f 

We need a formalism to deal with these derivatives.

Chain Rule

- Chain rule for computing gradients:

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

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

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

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

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

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

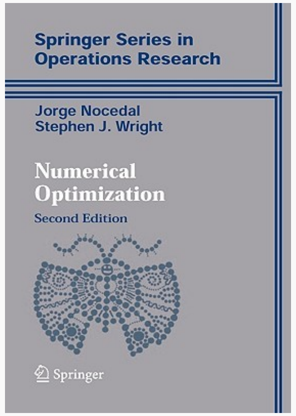
Variables	derivatives	Partial derivatives wrt to X,W
$\xi_1 = -W^TX$	$\frac{\partial \xi_1}{\partial W} = -X$	$\frac{\partial \xi_1}{\partial W} = -X$
$\xi_2 = e^{\xi_1} = e^{-W^TX}$	$\frac{\partial \xi_2}{\partial \xi_1} = e^{\xi_1}$	$\frac{\partial \xi_2}{\partial \xi_1} = e^{-W^TX}$
$\xi_3 = 1+\xi_2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

Learning Rate

Learning Rate

Trial and Error.

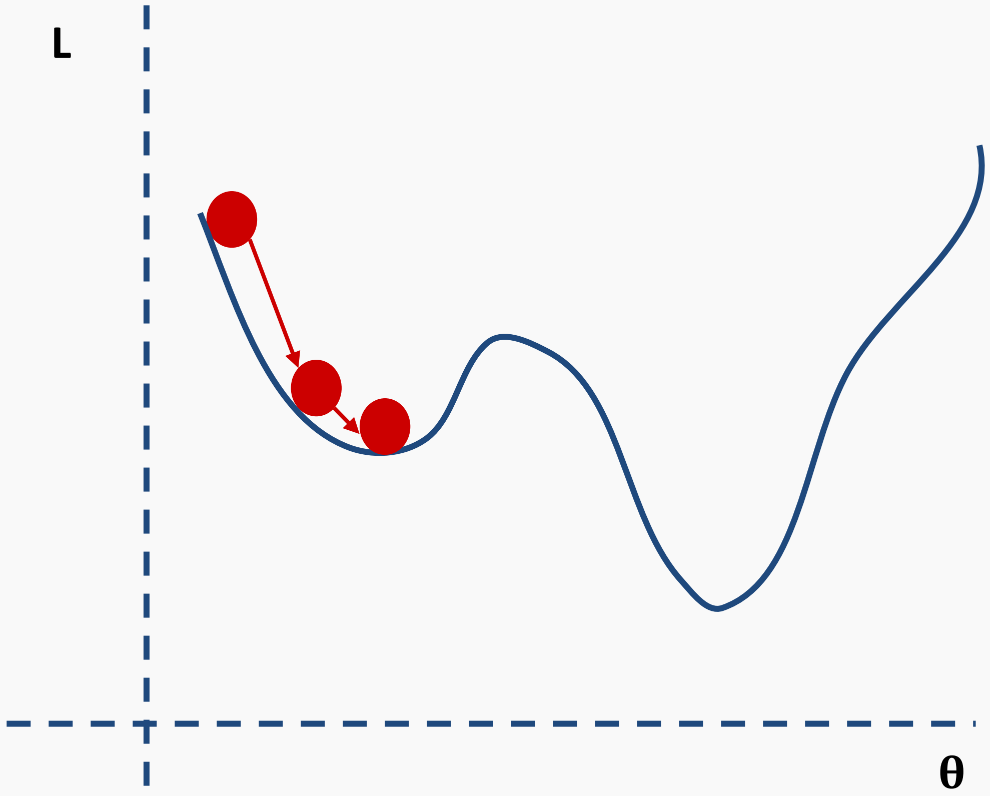
There are many alternative methods which address how to set or adjust the learning rate, using the derivative or second derivatives and or the momentum. To be discussed in the next lectures on NN.



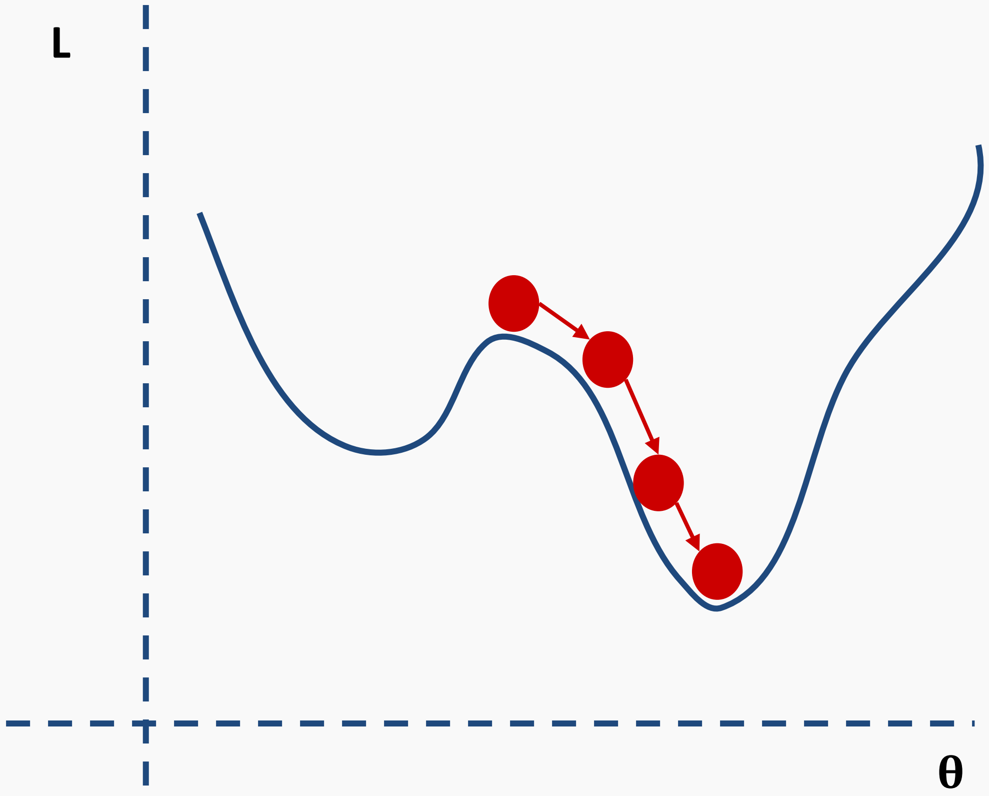
* J. Nocedal y S. Wright, "Numerical optimization", Springer, 1999 🔗
* *TLDR*: J. Bullinaria, "Learning with Momentum, Conjugate Gradient Learning", 2015 🔗

Local and Global minima

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

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

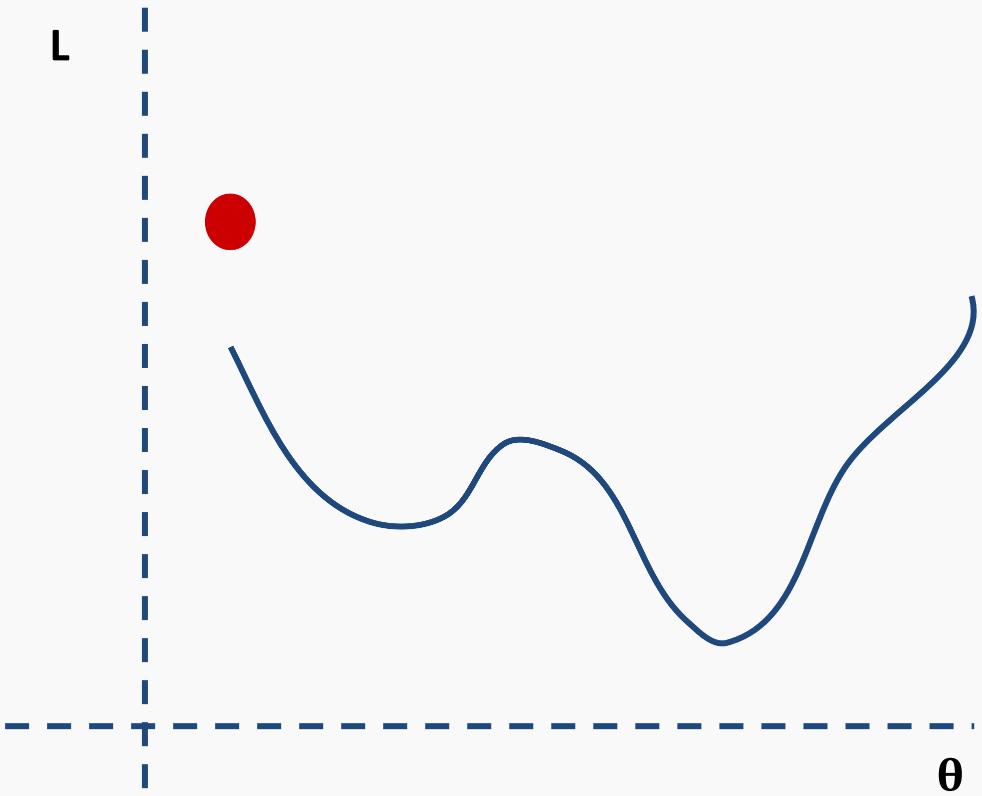
Instead of using all the examples for every step, use a subset of them (batch).

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.

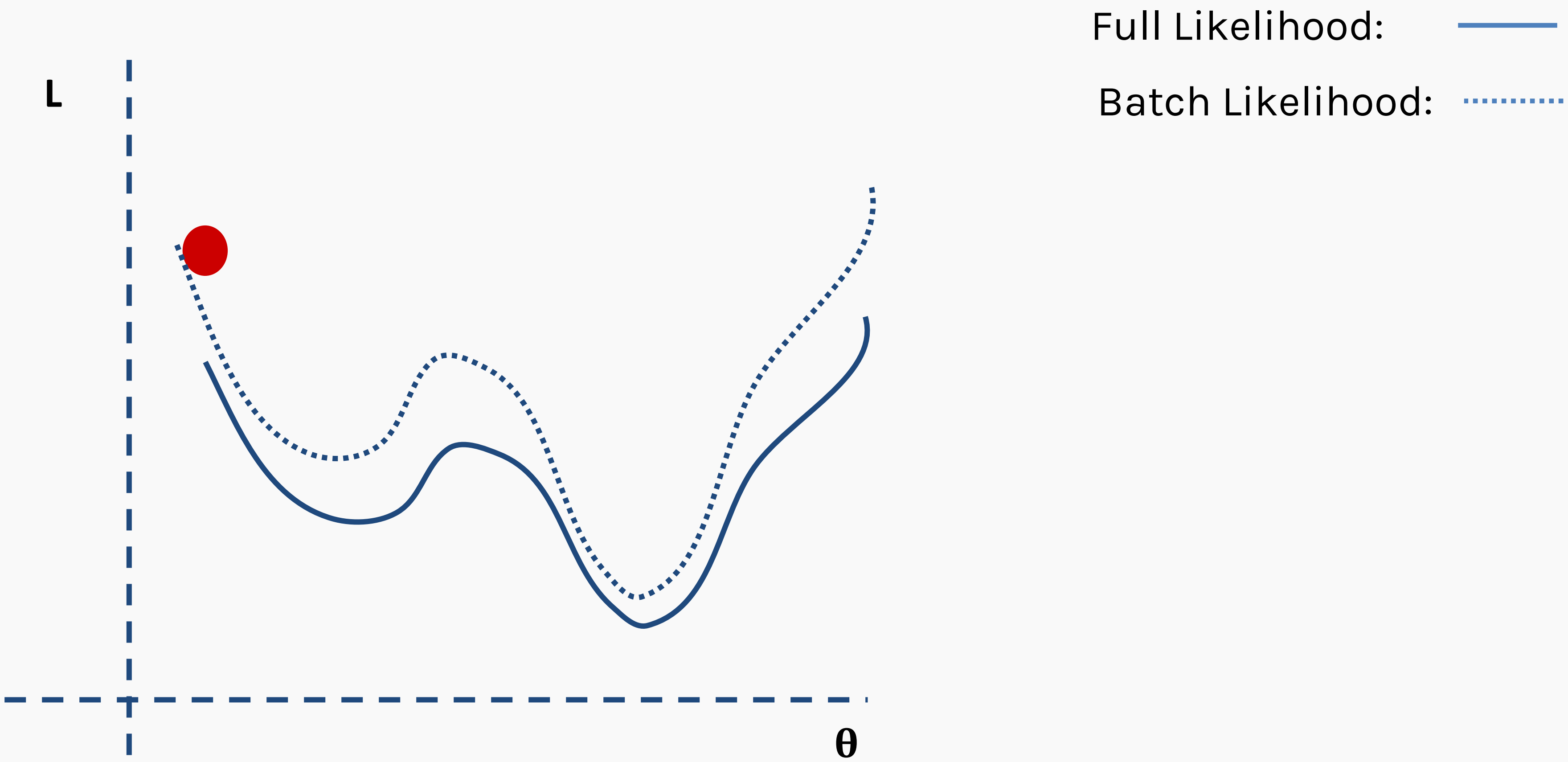
Batch and Stochastic Gradient Descent



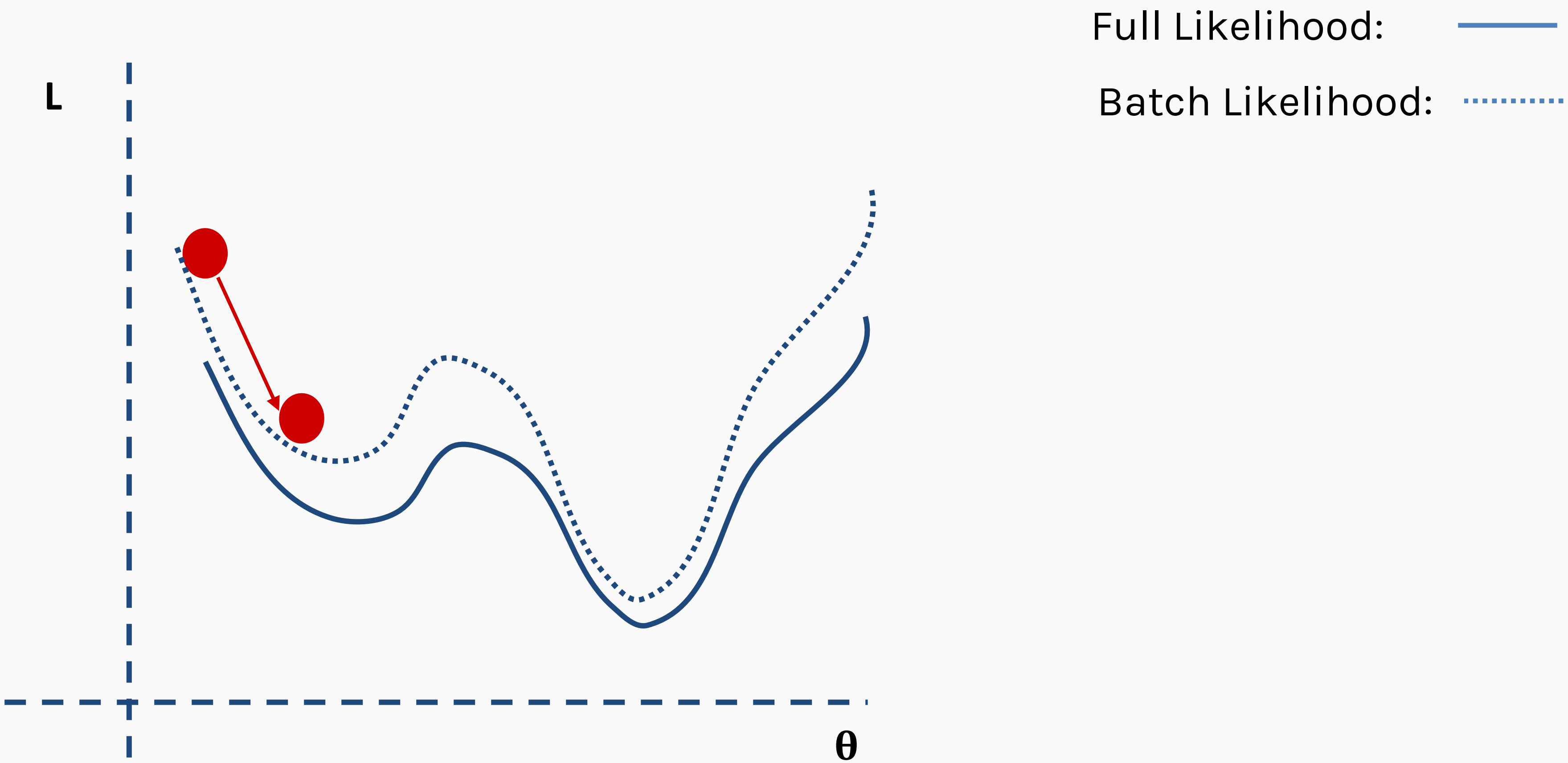
Full Likelihood:

Batch Likelihood:

Batch and Stochastic Gradient Descent



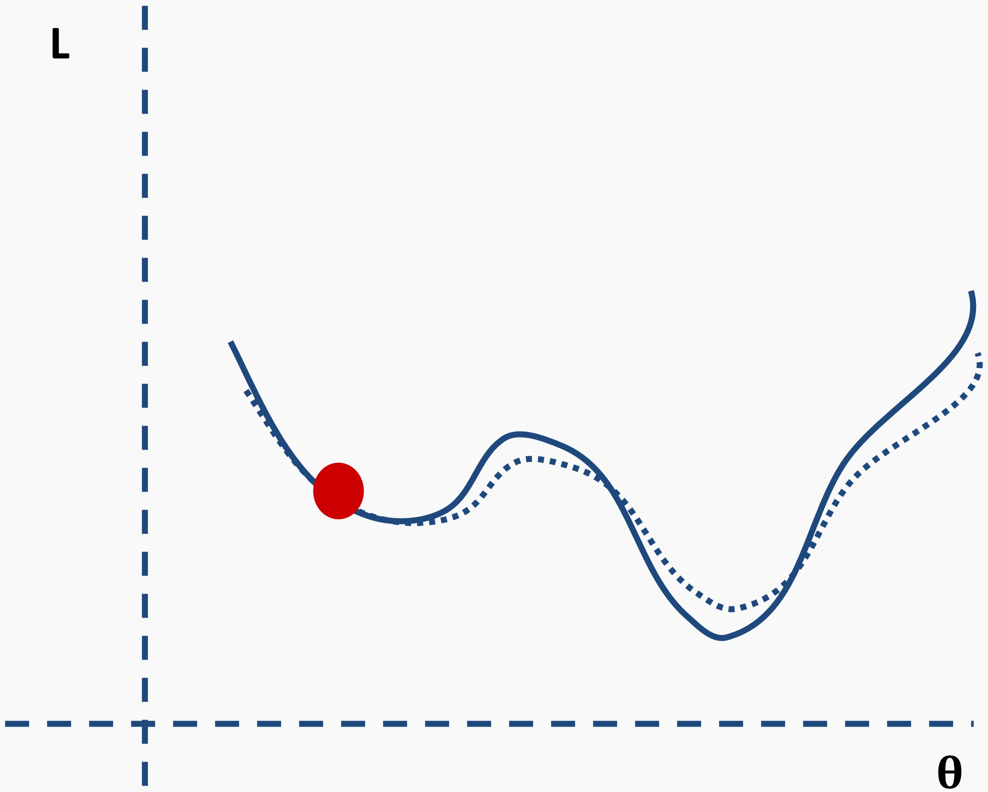
Batch and Stochastic Gradient Descent



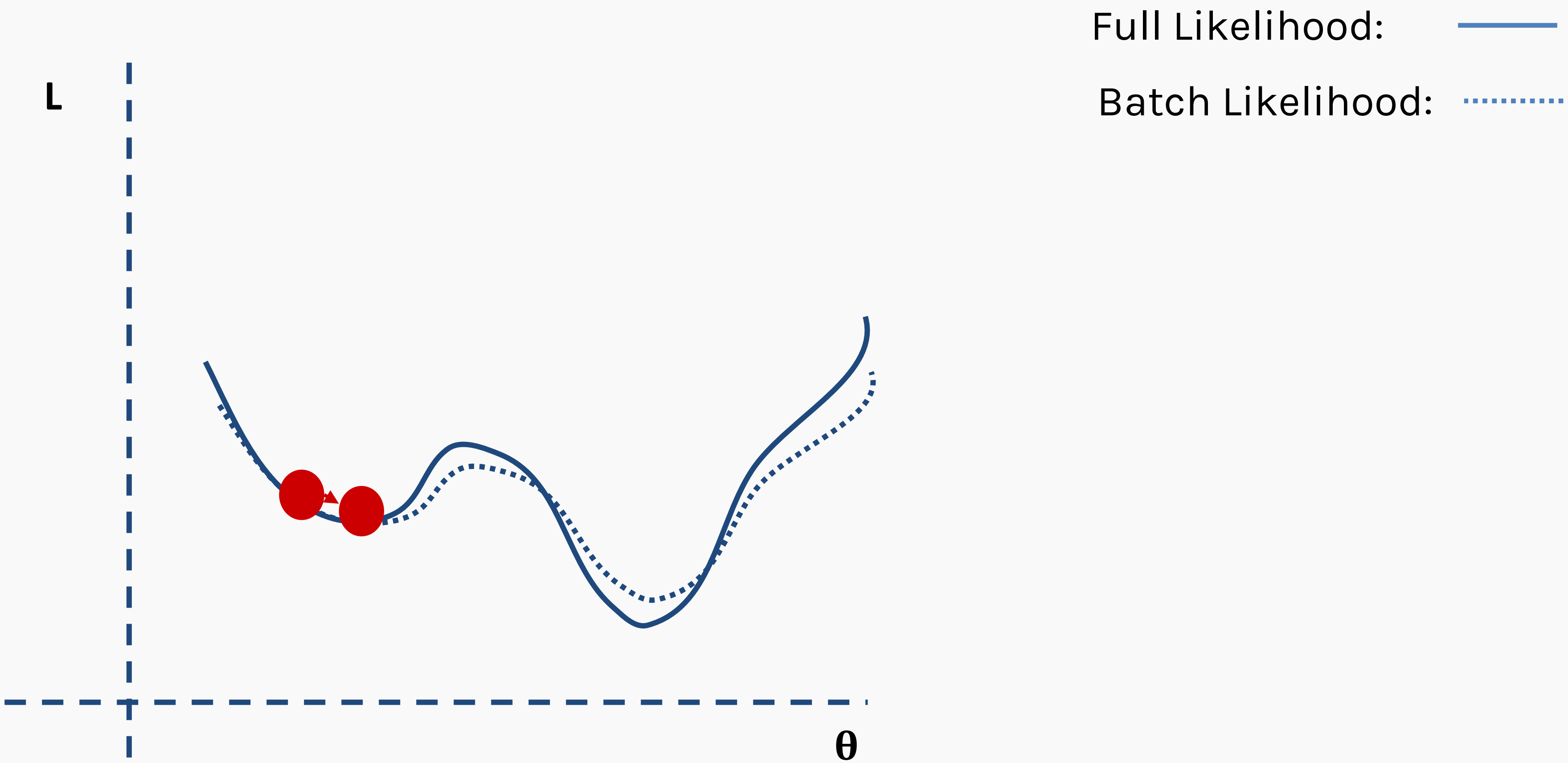
Batch and Stochastic Gradient Descent

Full Likelihood:

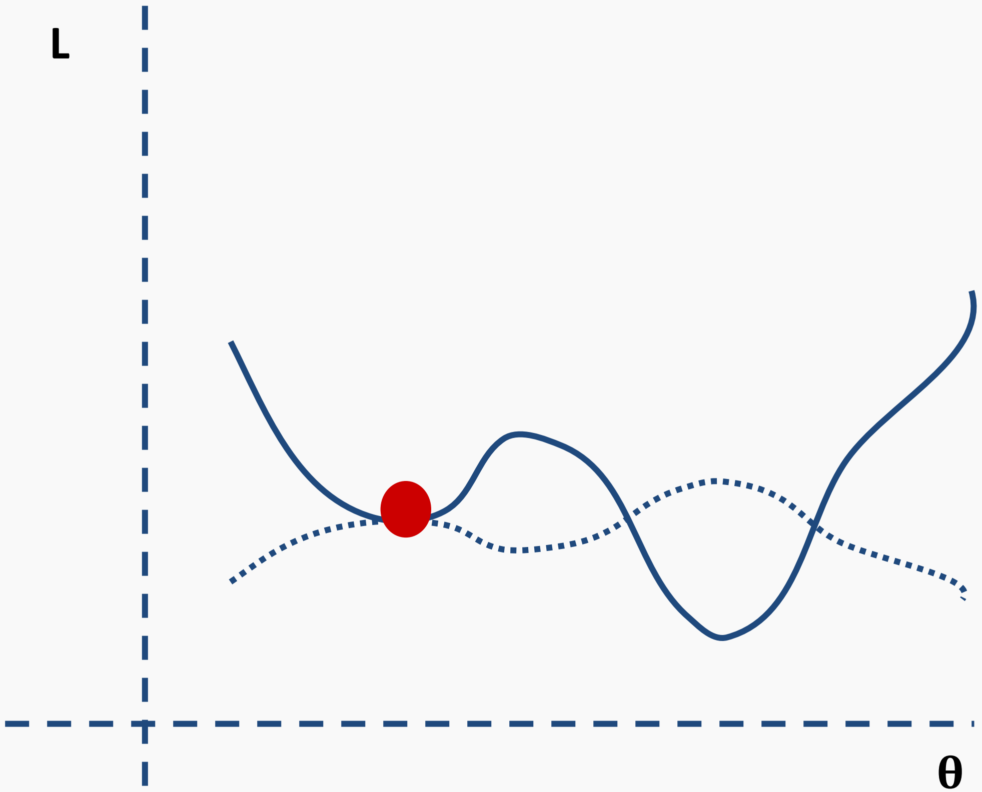
Batch Likelihood:



Batch and Stochastic Gradient Descent



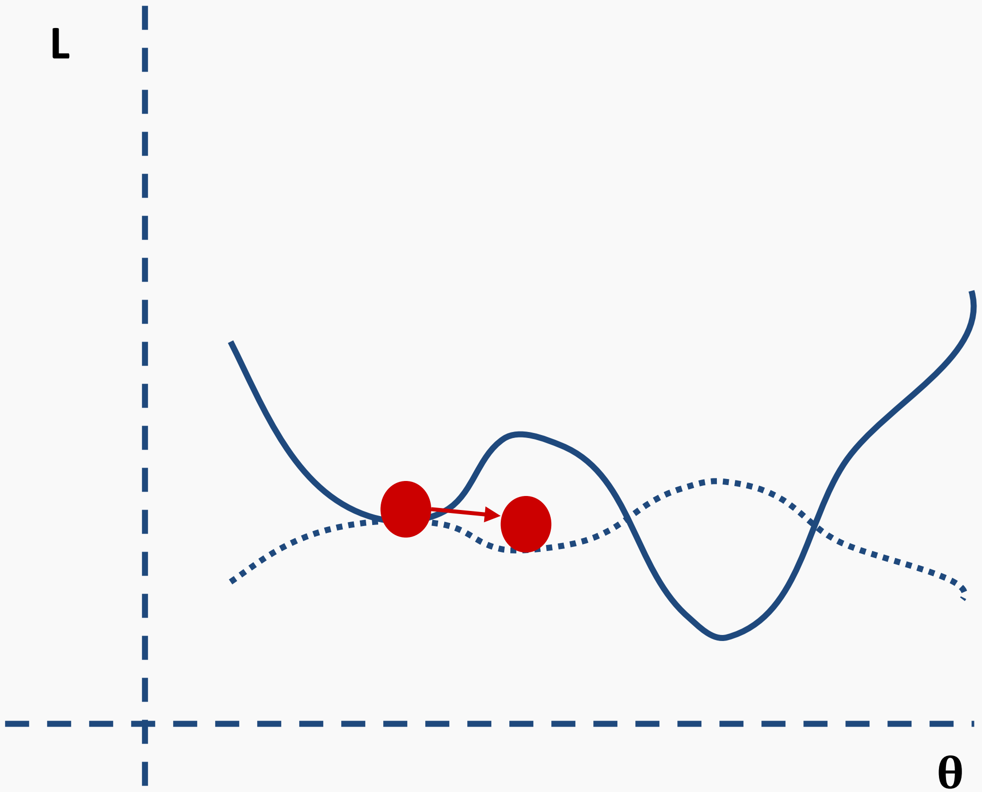
Batch and Stochastic Gradient Descent



Full Likelihood:

Batch Likelihood:

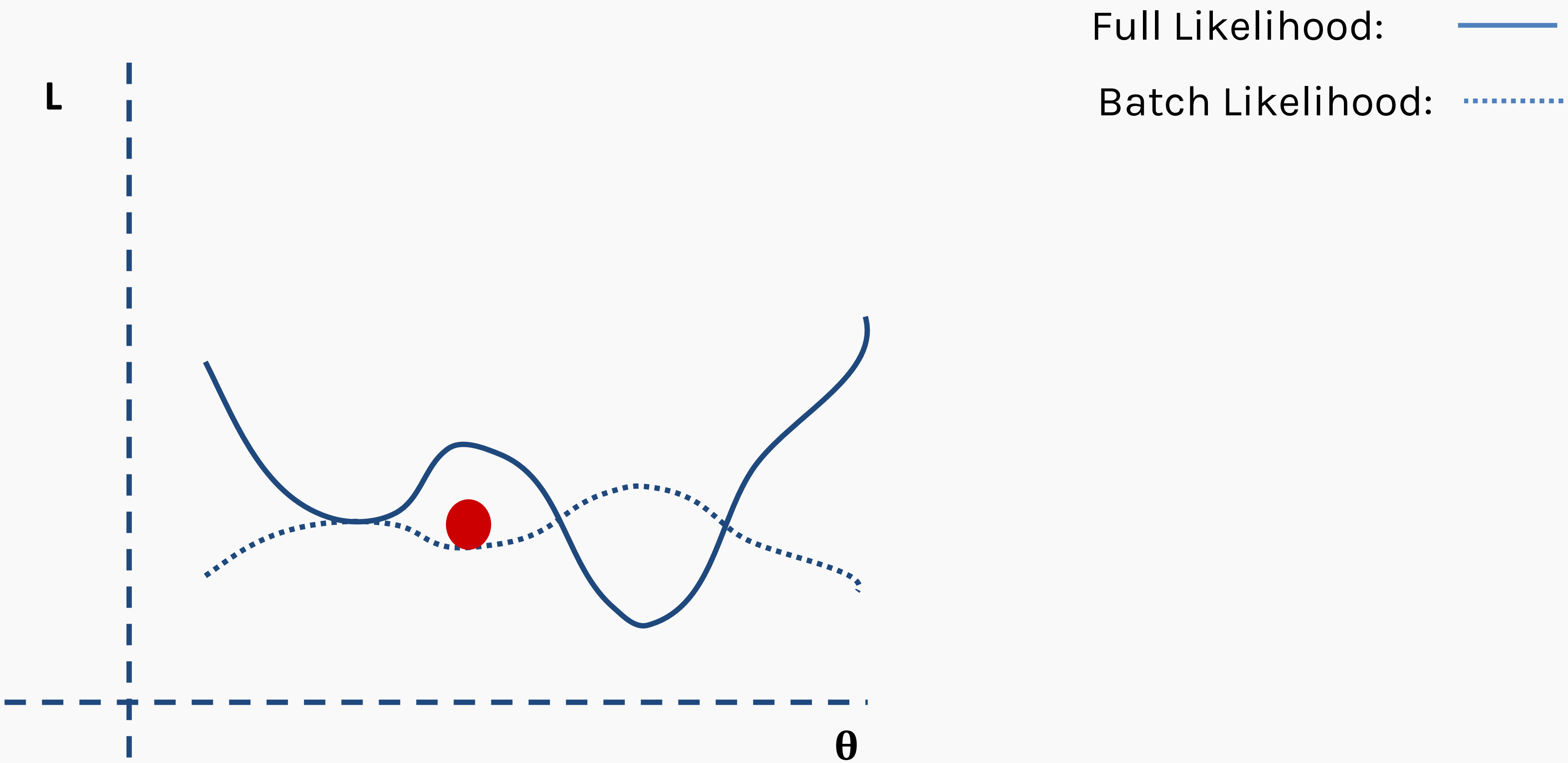
Batch and Stochastic Gradient Descent



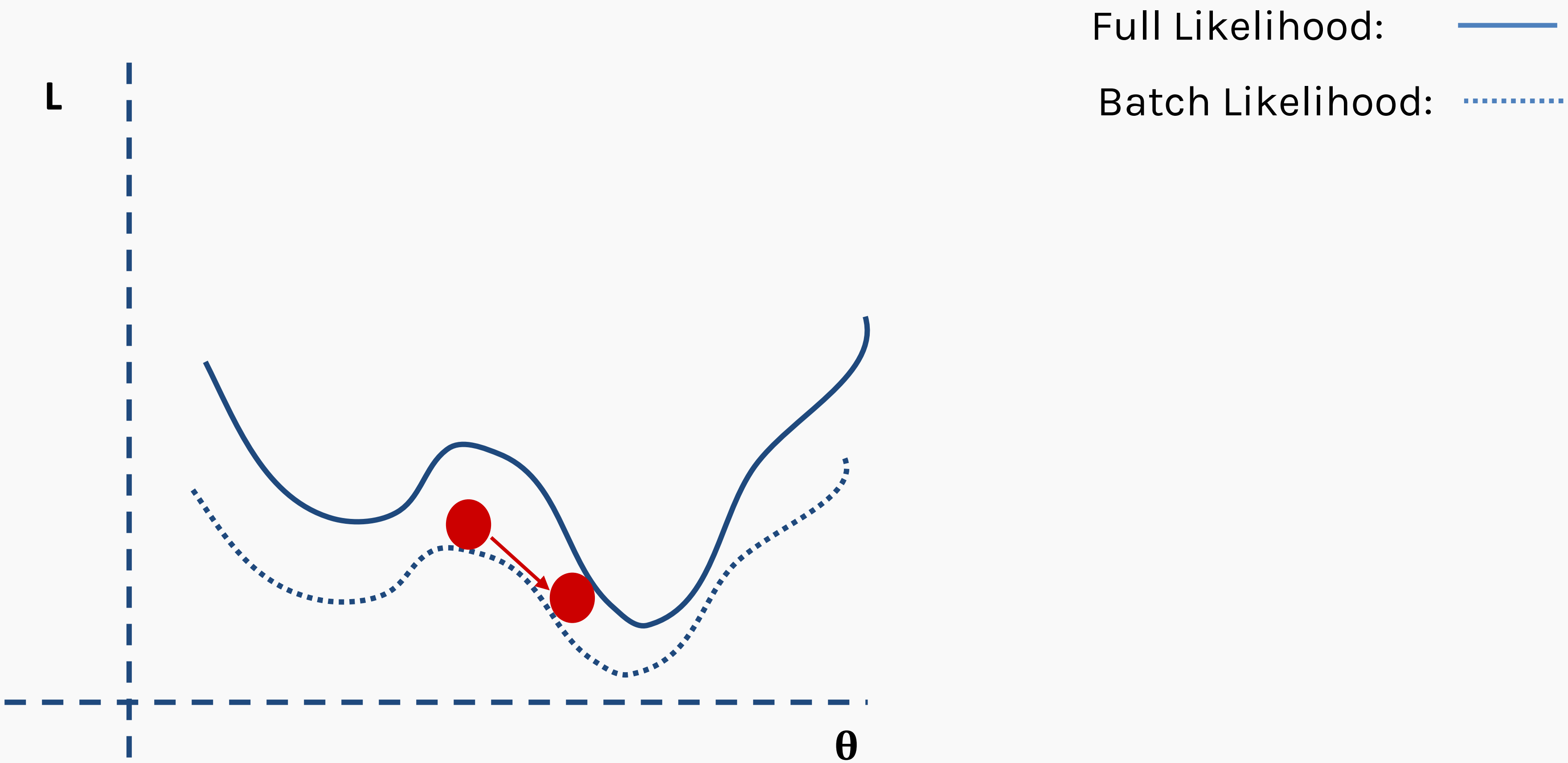
Full Likelihood:

Batch Likelihood:

Batch and Stochastic Gradient Descent



Batch and Stochastic Gradient Descent



Artificial Neural Networks (ANN)

Logistic Regression Revisited

$x_i \rightarrow$ Affine $\rightarrow h_i = \beta_0 + \beta_1 x_i \rightarrow$ Activation $\rightarrow p_i = \dfrac{1}{1+e^{-h_i}} \rightarrow$ Loss Fun $\rightarrow \mathcal{L}_i(\beta) = -y_i \ln(p_i) - (1-y_i)\ln(1-p_i)$

$x_k \rightarrow$ Affine $\rightarrow h_k = \beta_0 + \beta_1 x_k \rightarrow$ Activation $\rightarrow p_i = \dfrac{1}{1+e^{-h_k}} \rightarrow$ Loss Fun $\rightarrow \mathcal{L}_k(\beta) = -y_k \ln(p_k) - (1-y_k)\ln(1-p_k)$

$\vdots$

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

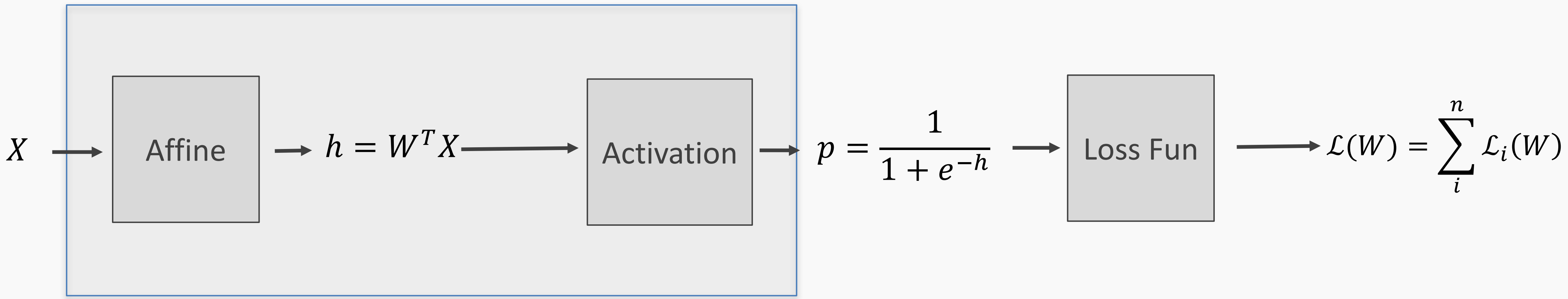
Build our first ANN

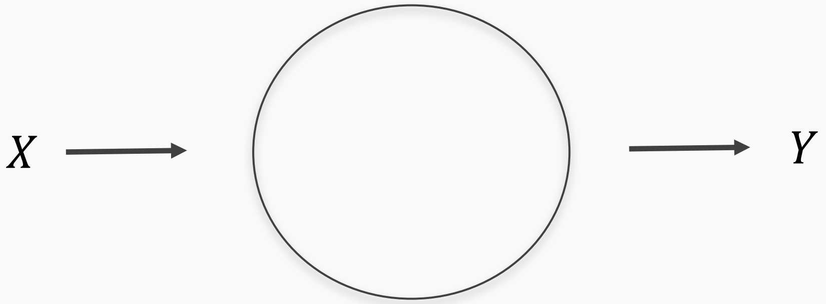
$X \longrightarrow$ Affine $\longrightarrow h = \beta_0 + \beta_1 X \longrightarrow$ Activation $\longrightarrow p = \dfrac{1}{1 + e^{-h}} \longrightarrow$ Loss Fun $\longrightarrow \mathcal{L}(\beta) = \sum_{i}^{n} \mathcal{L}_i(\beta)$

$X \longrightarrow$ Affine $\longrightarrow h = \beta^T X \longrightarrow$ Activation $\longrightarrow p = \dfrac{1}{1 + e^{-h}} \longrightarrow$ Loss Fun $\longrightarrow \mathcal{L}(\beta) = \sum_{i}^{n} \mathcal{L}_i(\beta)$

$X \longrightarrow$ Affine $\longrightarrow h = W^T X \longrightarrow$ Activation $\longrightarrow y = \dfrac{1}{1 + e^{-h}} \longrightarrow$ Loss Fun $\longrightarrow \mathcal{L}(W) = \sum_{i}^{n} \mathcal{L}_i(W)$

Build our first ANN



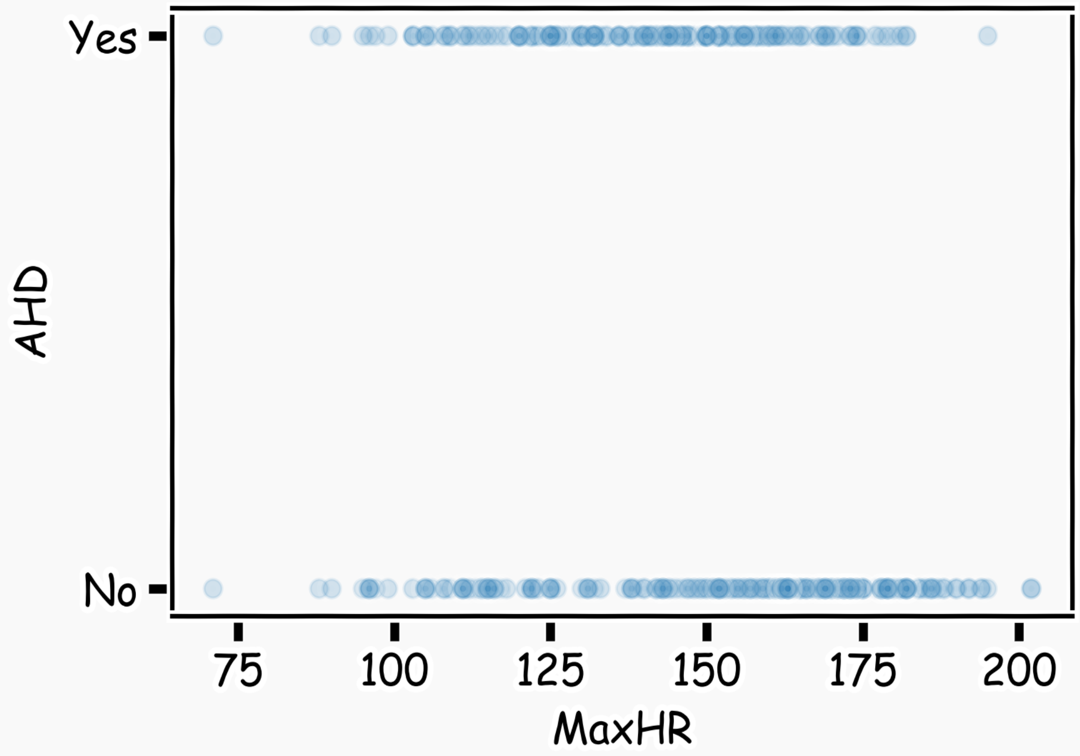


Single Neuron Network

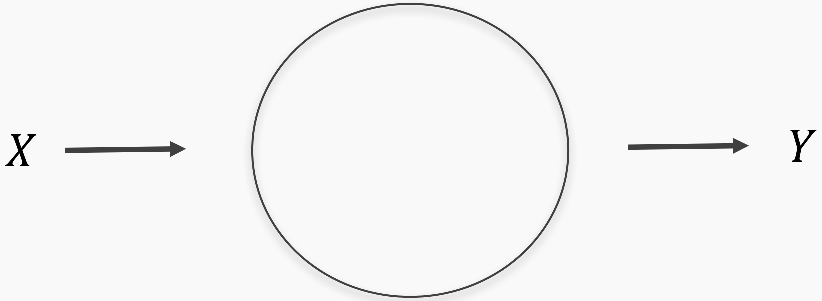
Very similar to Perceptron

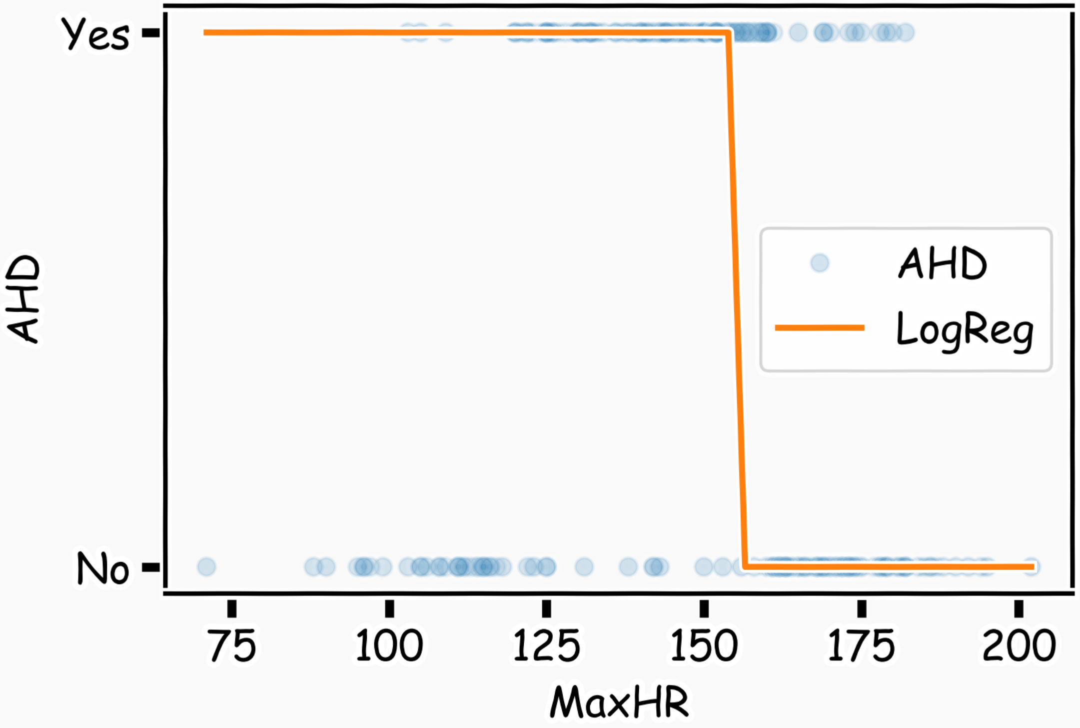
Example Using Heart Data

Slightly modified data to illustrate a concept.

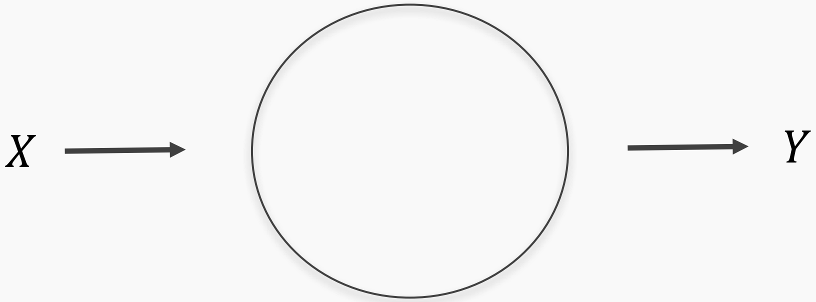


Example Using Heart Data

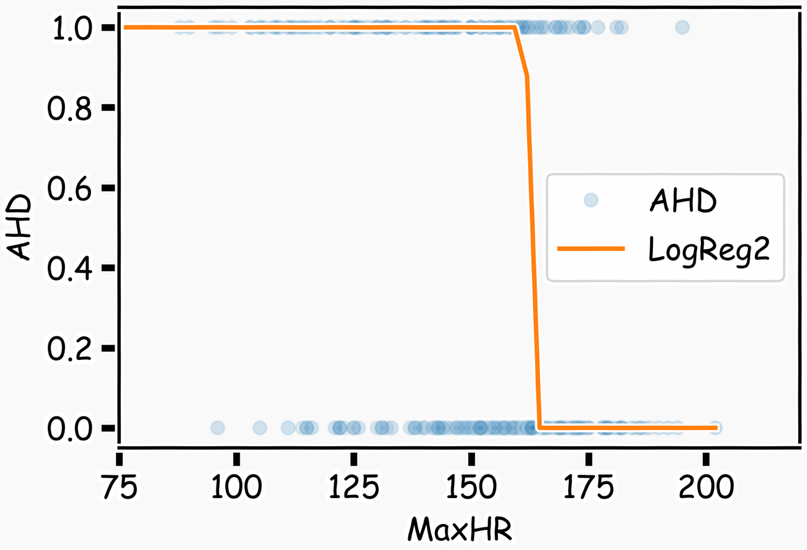




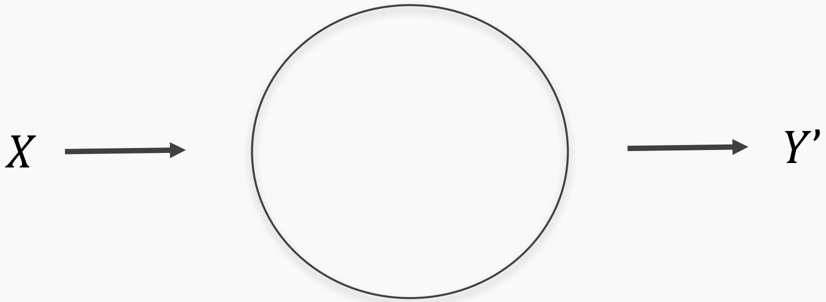
Example



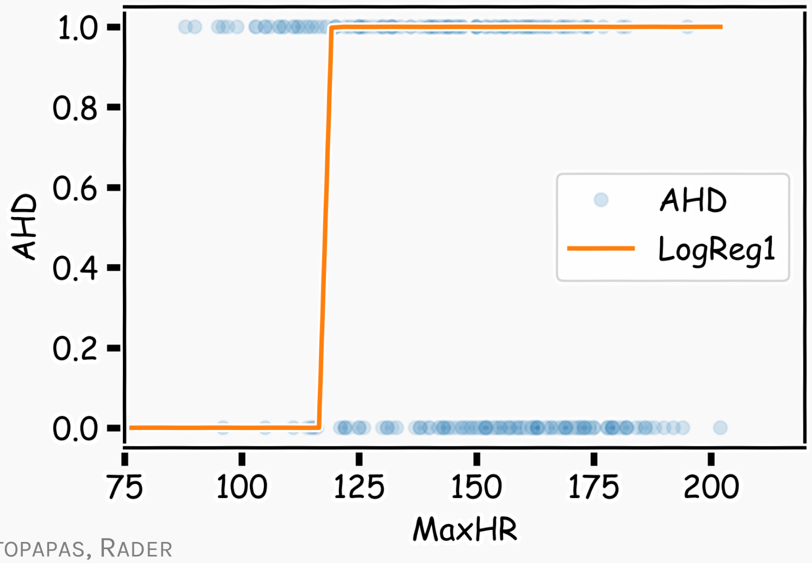
X
Y



1.0
0.8
0.6
0.4
0.2
0.0
AHD
75
100
125
150
175
200
MaxHR
AHD
LogReg2

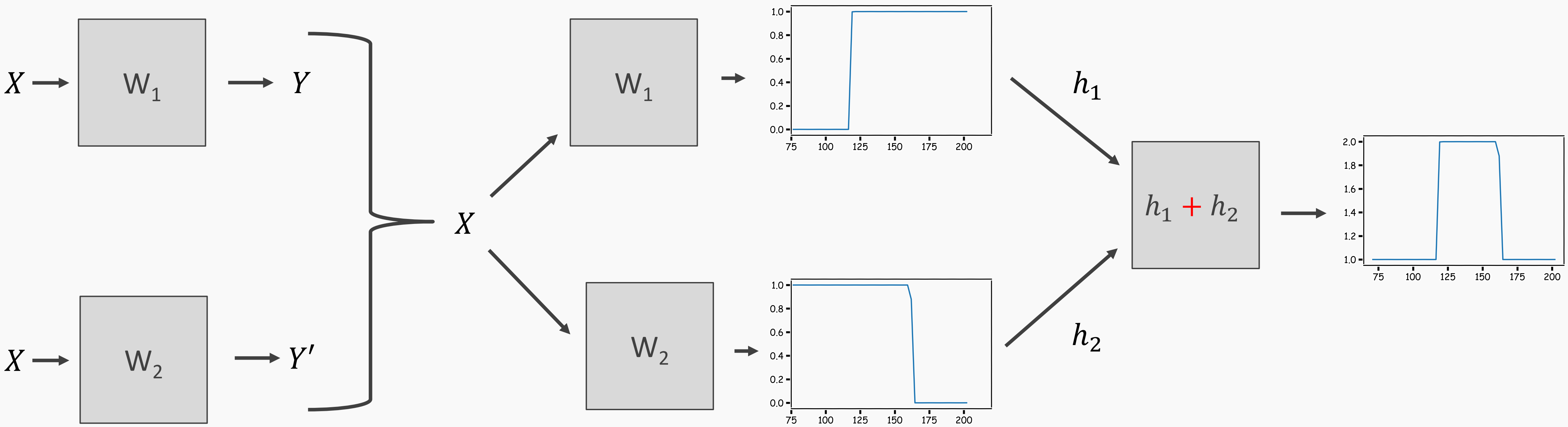


X
Y’



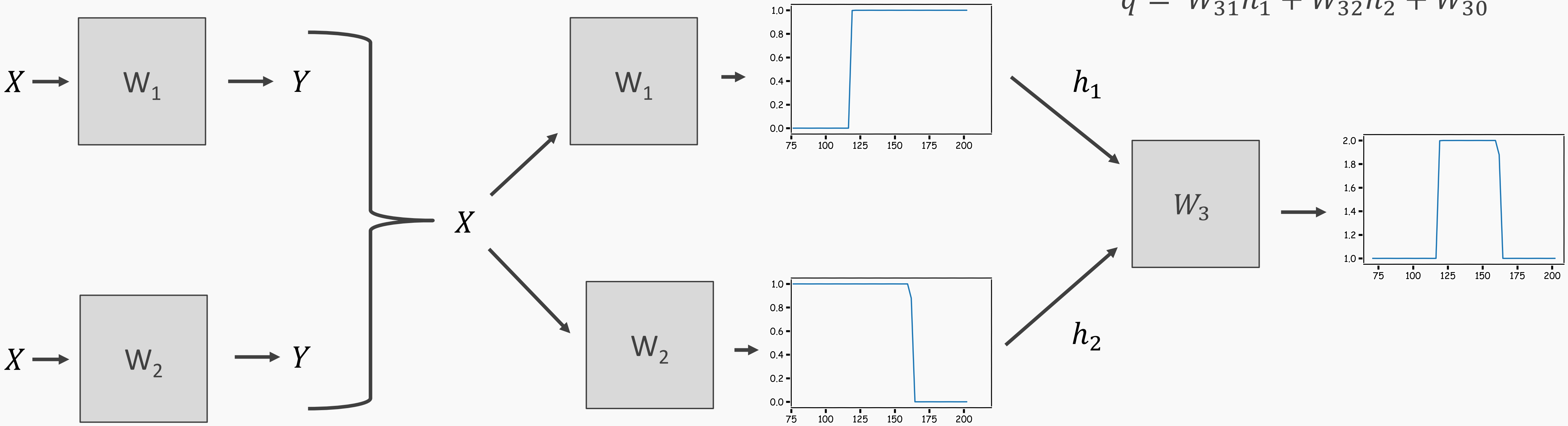
1.0
0.8
0.6
0.4
0.2
0.0
AHD
75
100
125
150
175
200
MaxHR
AHD
LogReg1

Pavlos game #232

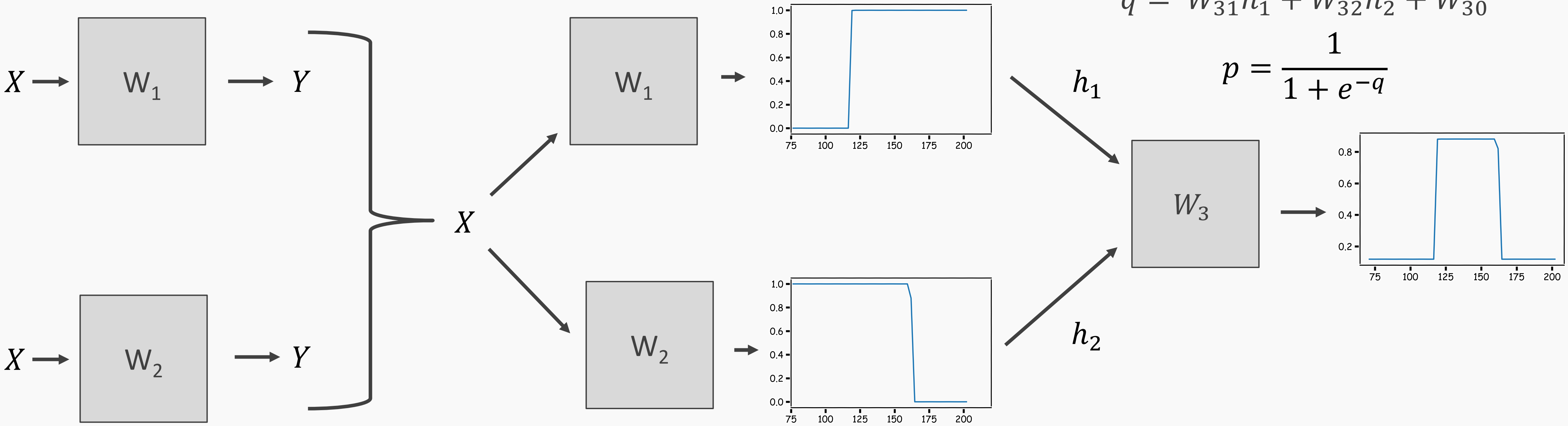


Pavlos game #232

$$q = W_{31}h_1 + W_{32}h_2 + W_{30}$$



Pavlos game #232



$$q = W_{31}h_1 + W_{32}h_2 + W_{30}$$

$$p = \frac{1}{1 + e^{-q}}$$

$$L = -y\ \ln(\mathrm{p}) - (1 - \mathrm{y})\ln(1 - \mathrm{p})$$

Need to learn W1, W2 and W3.

Backpropagation

Backpropagation: Logistic Regression Revisited

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

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

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

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

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

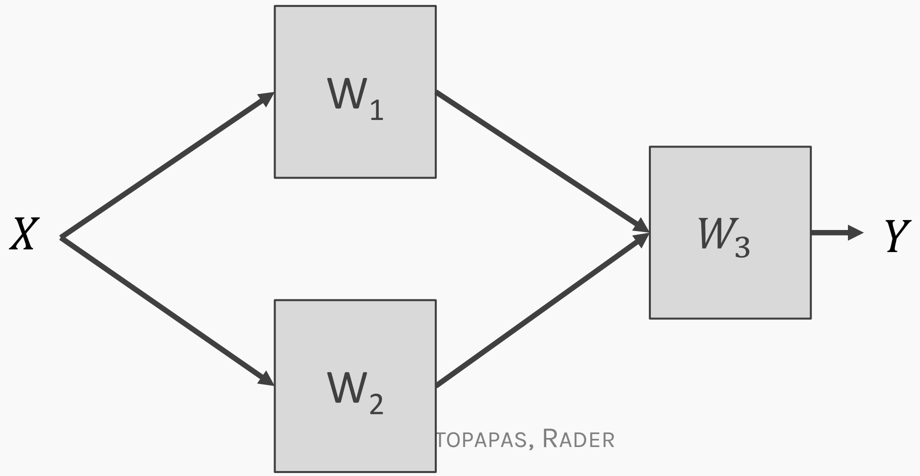
$$\frac{\partial \mathcal{L}}{\partial \beta_1} = -X\sigma(h)\big(1-\sigma(h)\big)[y\frac{1}{p} + (1-y)\frac{1}{1-p}]$$

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

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.

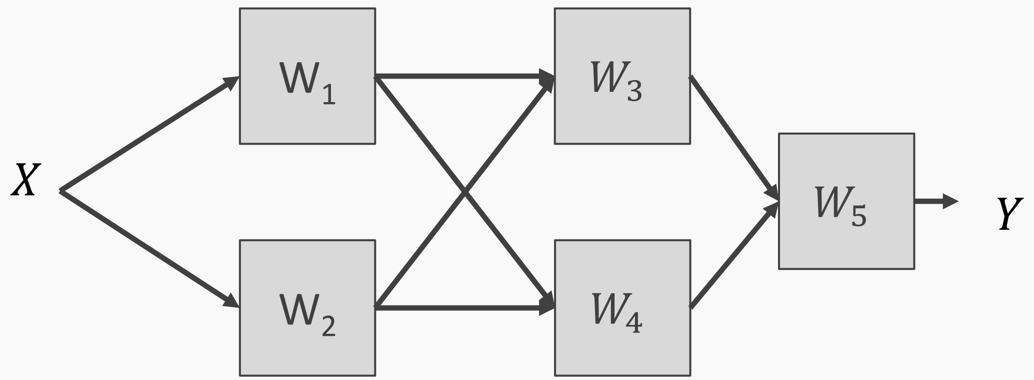
For example this network will need a different function for the derivatives



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.

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



Backpropagation. Pavlos game #456

Need to find a formalism to calculate the derivatives of the loss 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

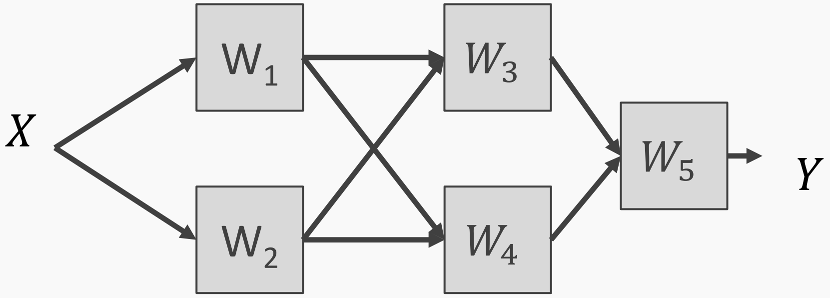
Basic functions

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`

Putting it altogether

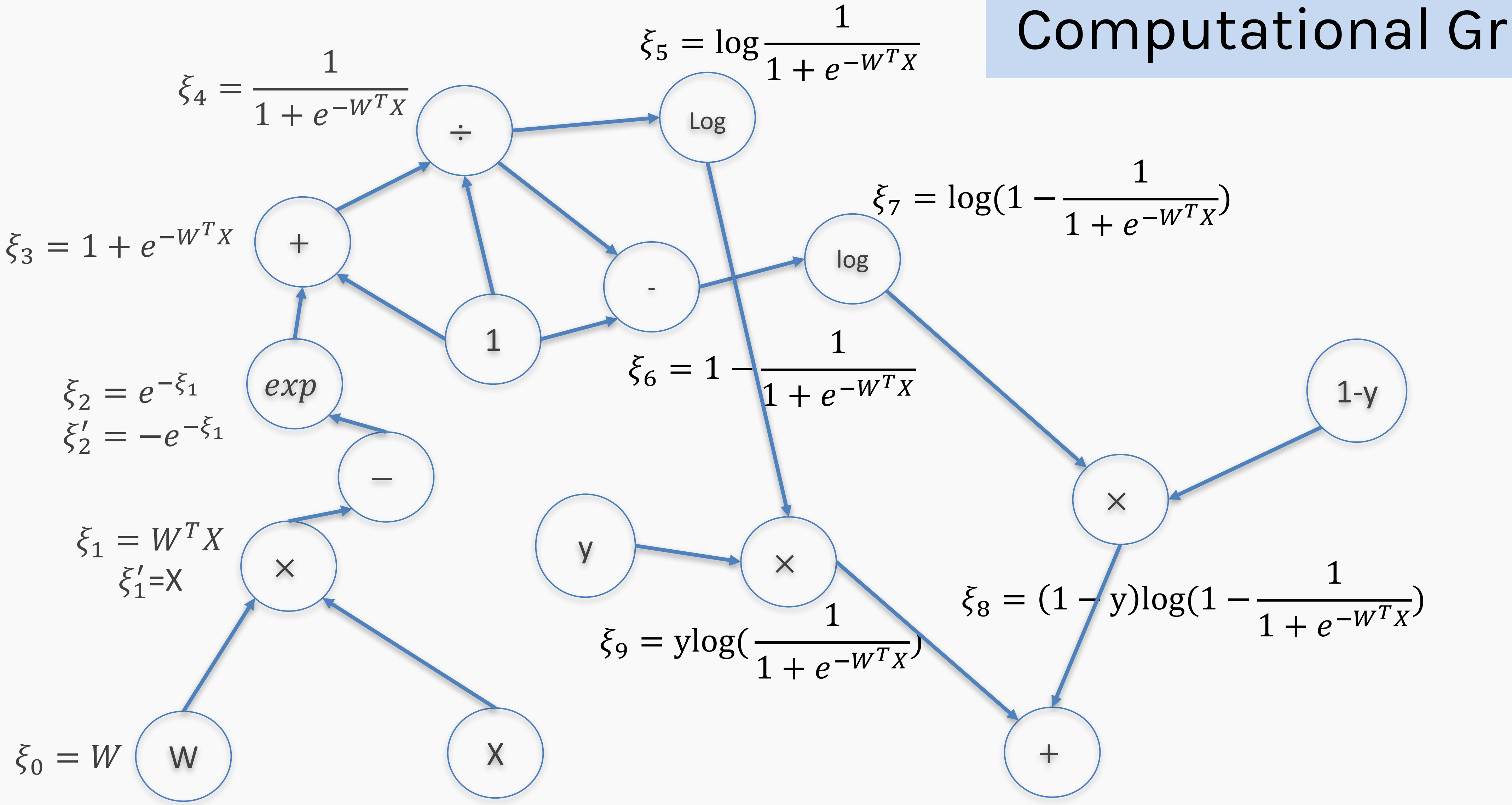
1. We specify the network structure



2. We create the computational graph …

What is computational graph?

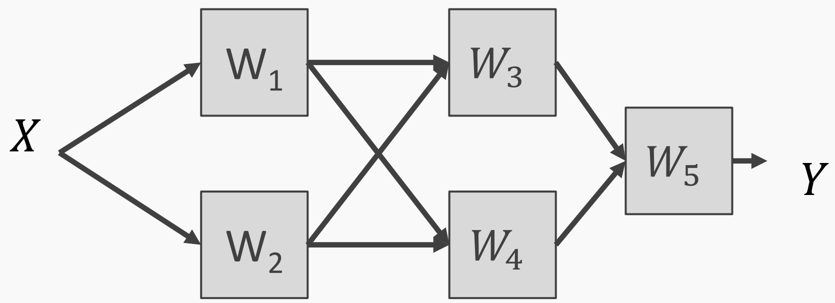
Computational Graph



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

Putting it altogether

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 3 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	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

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