Perceptron and Multilayer Perceptron

CS109B Data Science 2
Pavlos Protopapas, Mark Glickman

- Π Lecture 8: Neural Networks I
- α Lecture 9: Neural Networks: II
- ύ Lecture 10: Optimizers
- λ Lecture 11: Regularization of NN
- ο Lecture 11 - CNNs basics
- ς Lecture 12 - CNNs - Pooling and CNNs Structure
- Π Lecture 13 - Backprop max pooling, Receptive Fields and Feature Map viz
- ρ Lecture 15 - Saliency maps
- ω🇬🇷 Lecture 16: Intro to Language Model and Traditional Language Modeling
- τ 🏴󠁧󠁢󠁥󠁮󠁧󠁿 Lecture 17: RNNs
- ο🇫🇷 Lecture 18: GRUs/LSTMs
- π 🇨🇳 Lecture 19: Language models with RNNs; ELMO
- α 🇮🇹 Lecture 20: Seq2Seq and Attention
- π 🇯🇵 Lecture 21: Tranformers
- α Lecture 22: GANs
- ς Lecture 23 - GANs DOS

Advanced Sections

- A-Sec 3: Solvers
- A-Sec 4: Semantic Segmentation and Object Detection
- A-Sec 5: SOTA and transfer learning
- A-Sec 6: Autoencoders
- ✓ A-Sec 7: Word Embeddings
- A-Sec 8: BERT
- ✓ A-Sec 9: Modern GANs

Outline

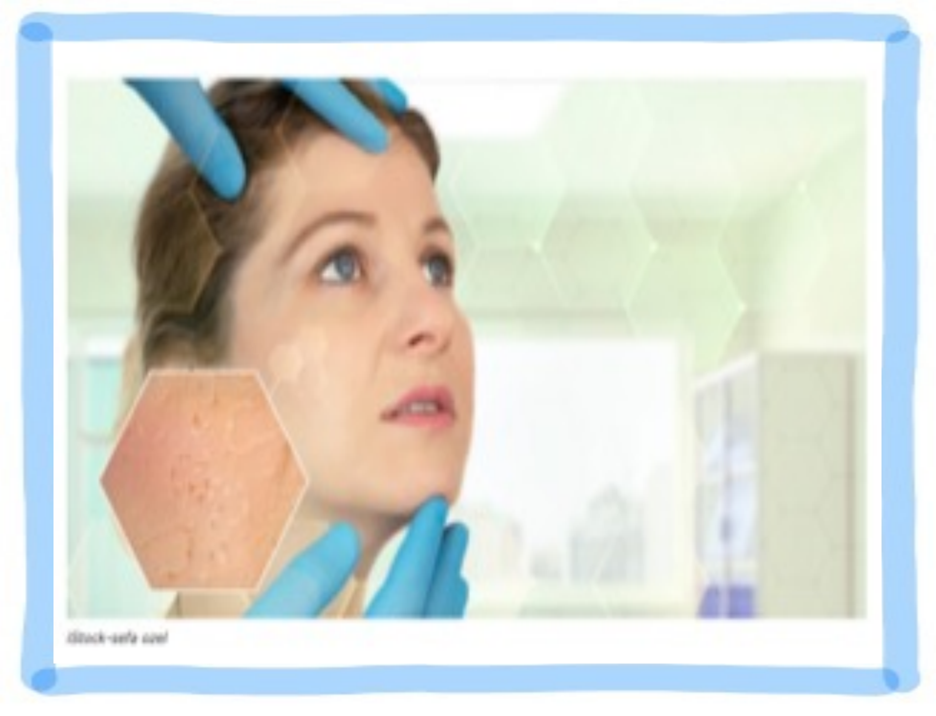
1. Introduction to Artificial Neural Networks
2. Review of basic concepts
3. Single Neuron Network (‘Perceptron’)
4. Multi-Layer Perceptron (MLP)

Outline

1. **Introduction to Artificial Neural Networks**
2. Review of basic concepts
3. Single Neuron Network (‘Perceptron’)
4. Multi-Layer Perceptron (MLP)

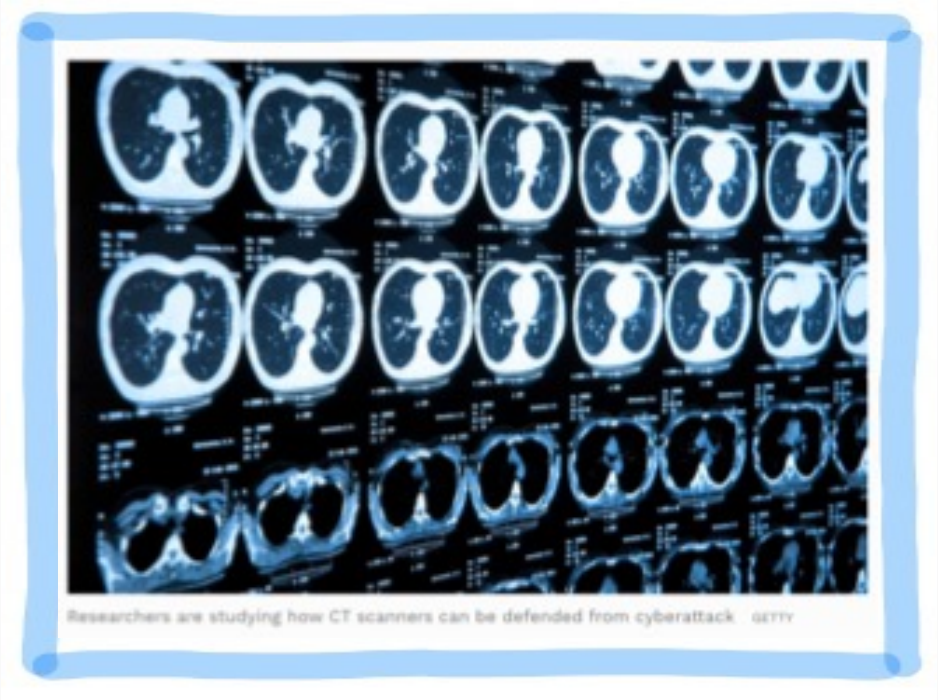
Today’s news

Skin Conditions



Using Deep Learning in diagnosing skin conditions

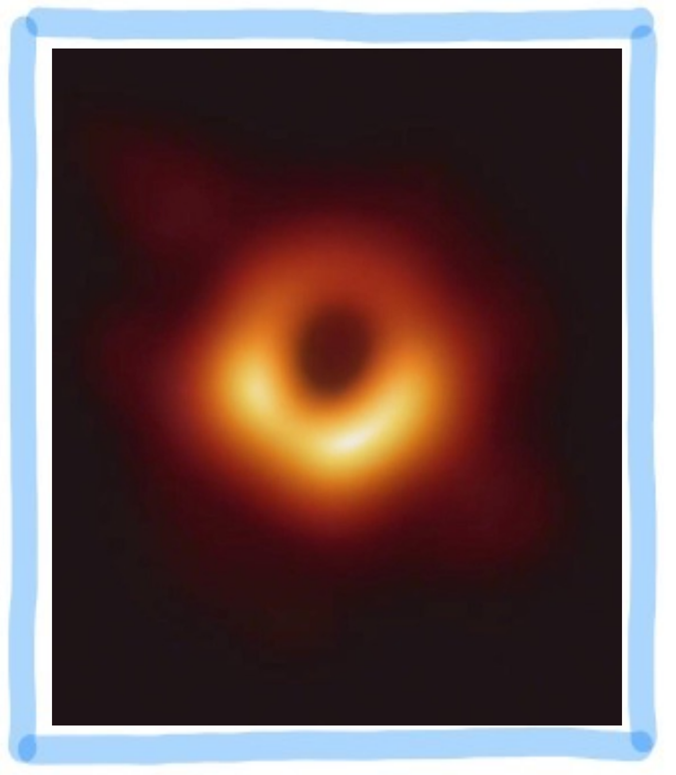
Stopping Cyberattacks



Detecting tampering with the diagnostic images, or quietly upped the radiation levels.

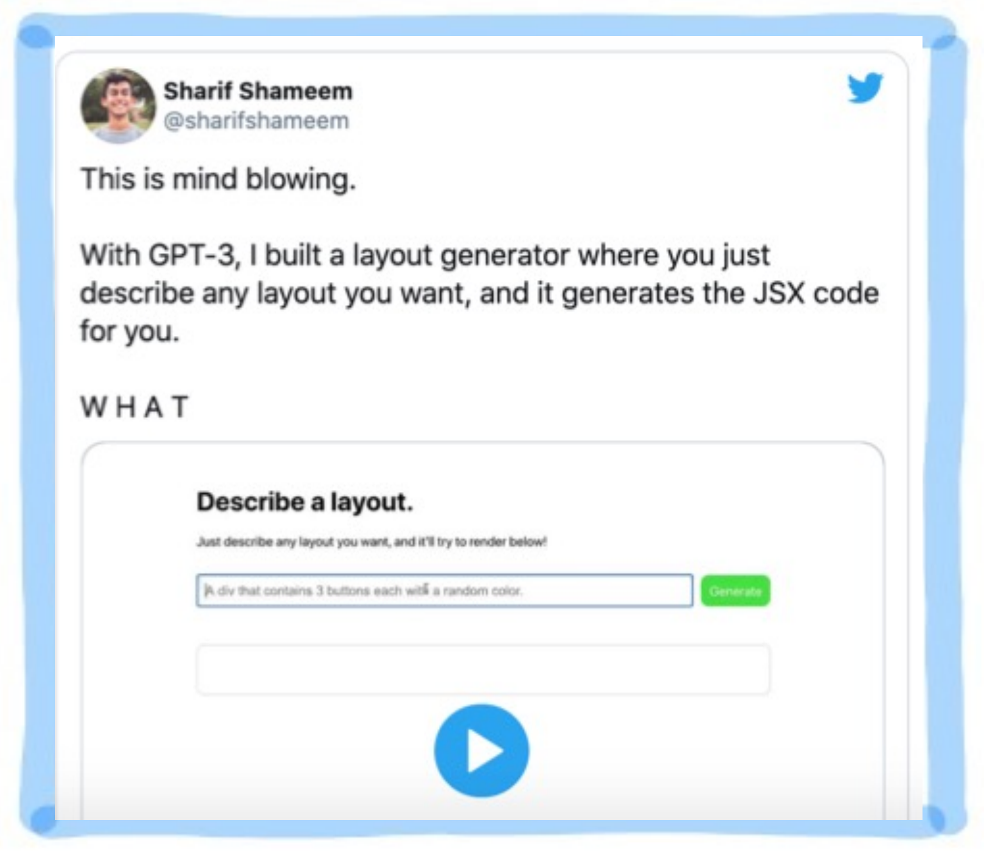
Today’s news

Image generation



Katie Bouman’s CHIRP produces the first-ever image of a black hole.

Computer Code Generation



The Potential of Data Science

Gender Bias

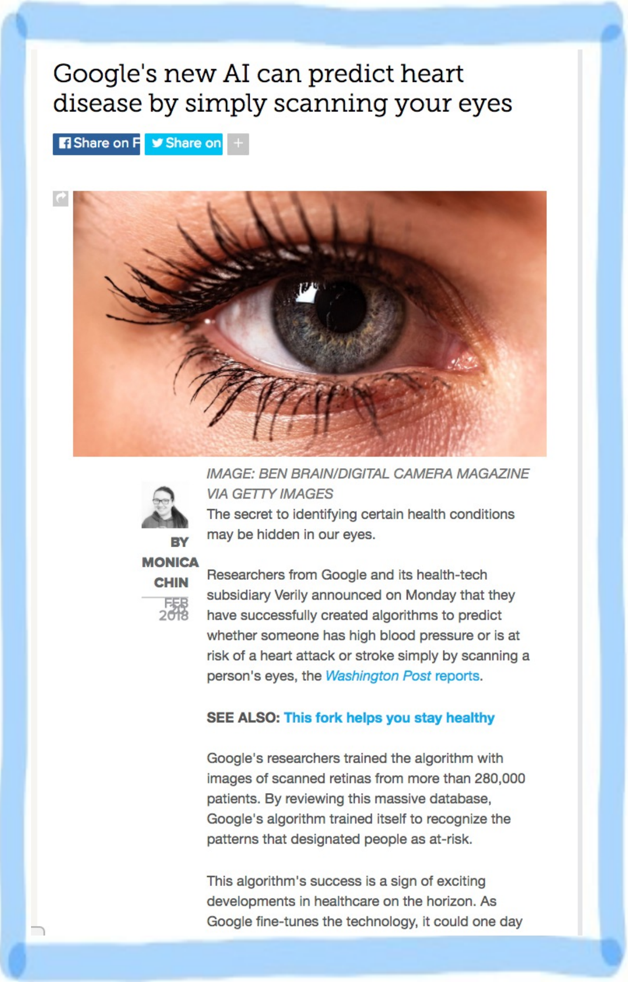
Some DS models for evaluate job applications show bias in favor of male candidate

Racial Bias

Risk models used in US courts have shown to be biased against non-white defendants

Historical Trends

Disease prediction



Google's new AI can predict heart disease by simply scanning your eyes

Share on F Share on +

IMAGE: BEN BRAIN/DIGITAL CAMERA MAGAZINE VIA GETTY IMAGES

BY MONICA CHIN

FEB 2018

The secret to identifying certain health conditions may be hidden in our eyes.

Researchers from Google and its health-tech subsidiary Verily announced on Monday that they have successfully created algorithms to predict whether someone has high blood pressure or is at risk of a heart attack or stroke simply by scanning a person's eyes, the *Washington Post* reports.

SEE ALSO: This fork helps you stay healthy

Google's researchers trained the algorithm with images of scanned retinas from more than 280,000 patients. By reviewing this massive database, Google's algorithm trained itself to recognize the patterns that designated people as at-risk.

This algorithm's success is a sign of exciting developments in healthcare on the horizon. As Google fine-tunes the technology, it could one day

Game strategy

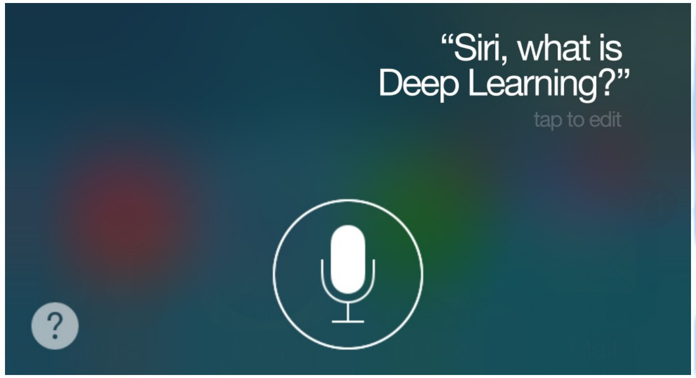
DeepMind

AlphaZero AI beats champion chess program after teaching itself in four hours

Google's artificial intelligence sibling DeepMind repurposes Go-playing AI to conquer chess and shogi without aid of human knowledge

2018

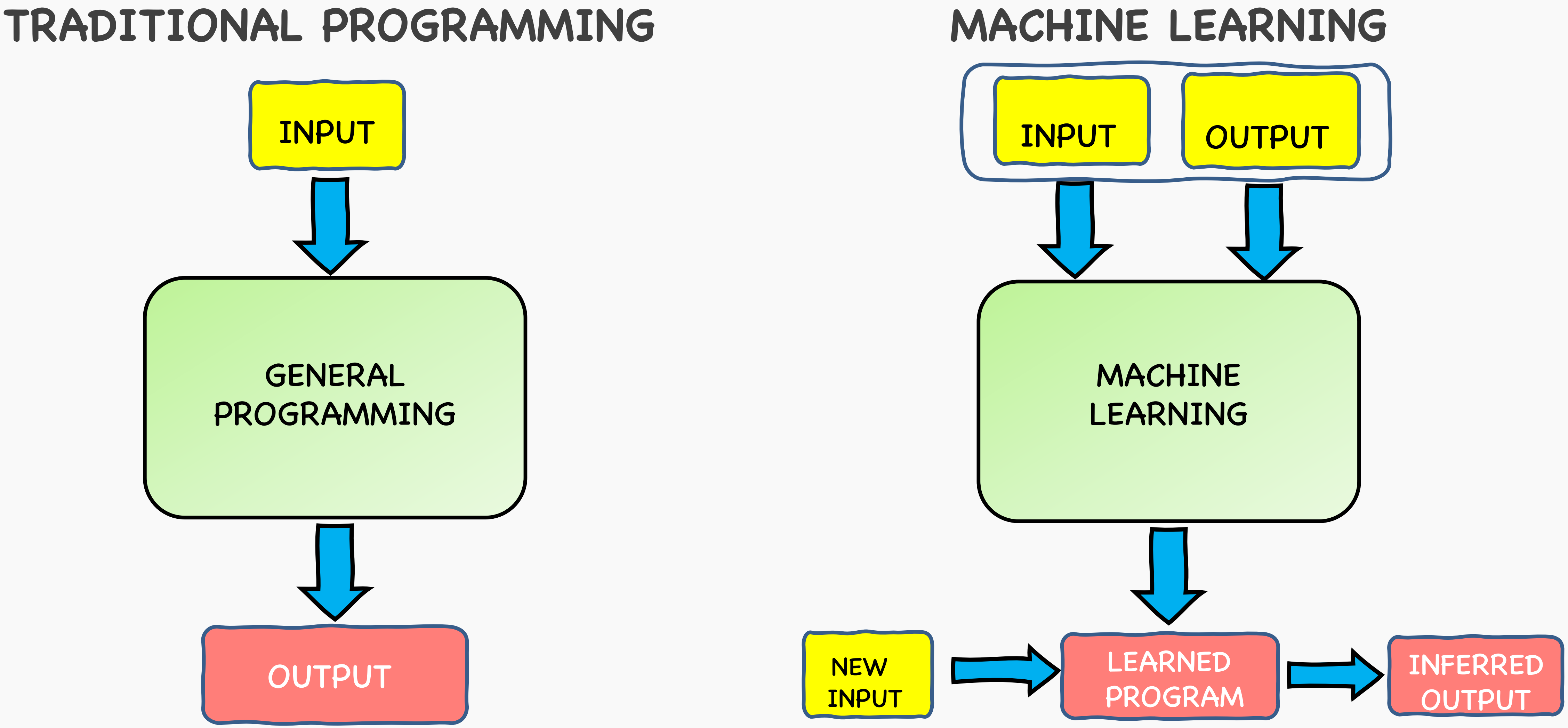
Natural Language Processing



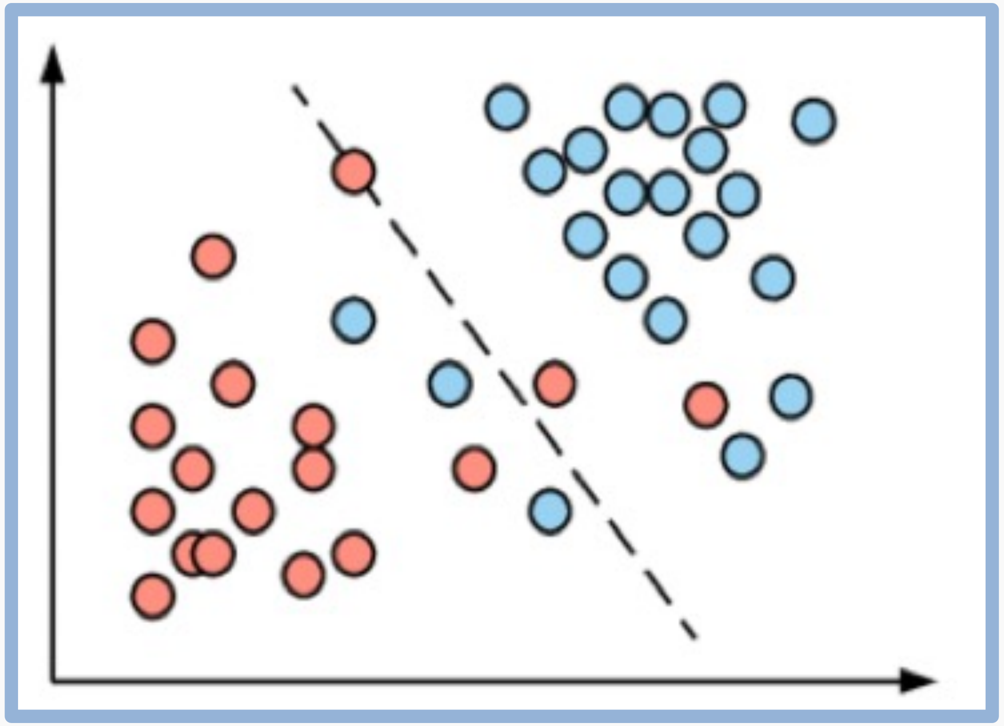
Outline

1. Introduction to Artificial Neural Networks
2. **Review of basic concepts**
3. Single Neuron Network (‘Perceptron’)
4. Multi-Layer Perceptron (MLP)

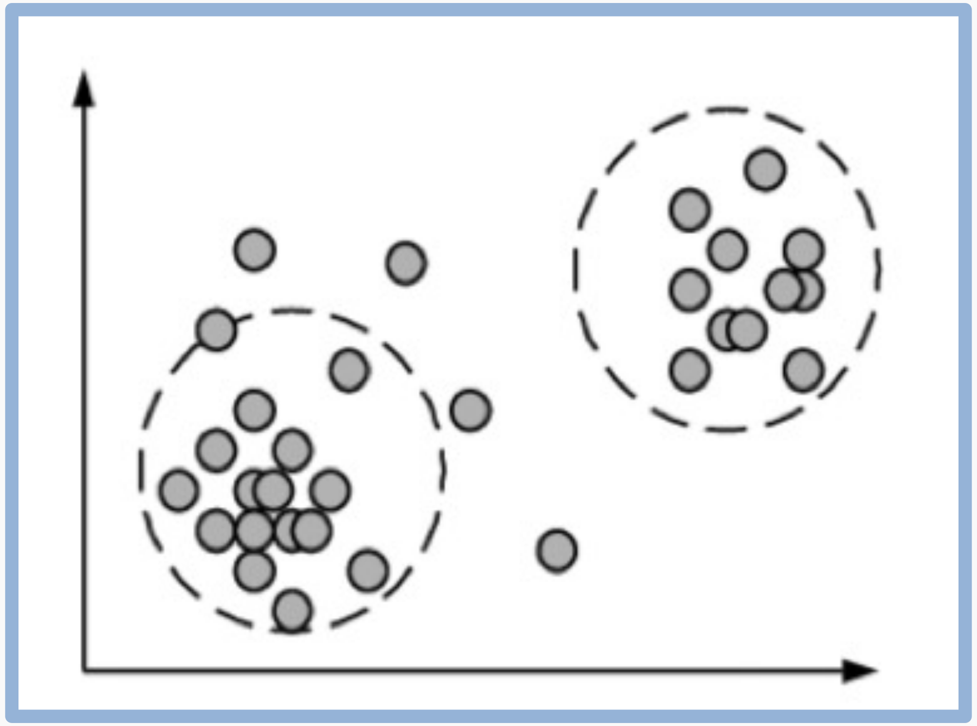
What is Machine Learning?



Supervised v/s Unsupervised Machine Learning

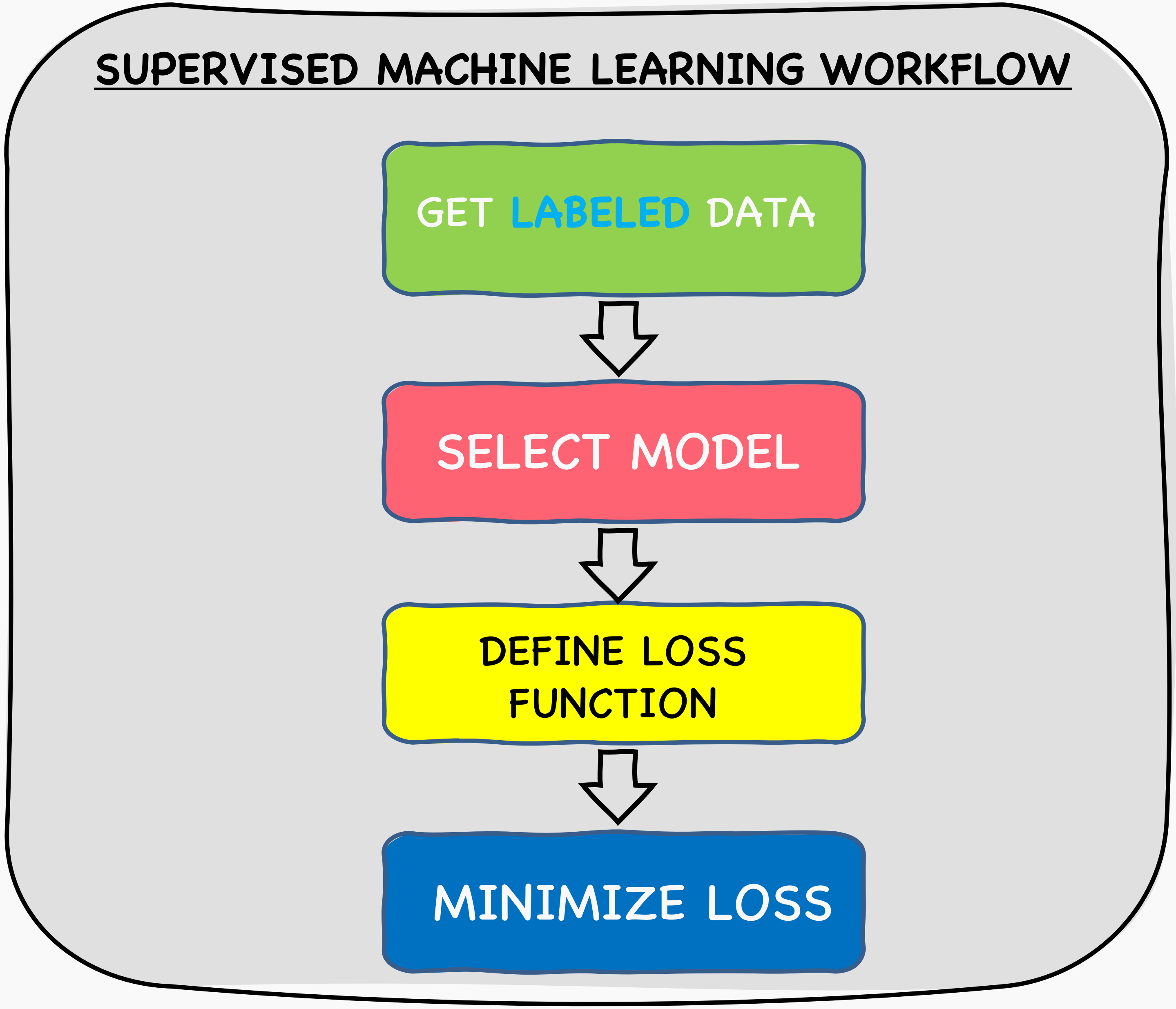


Supervised Learning: Learns with “labeled” data



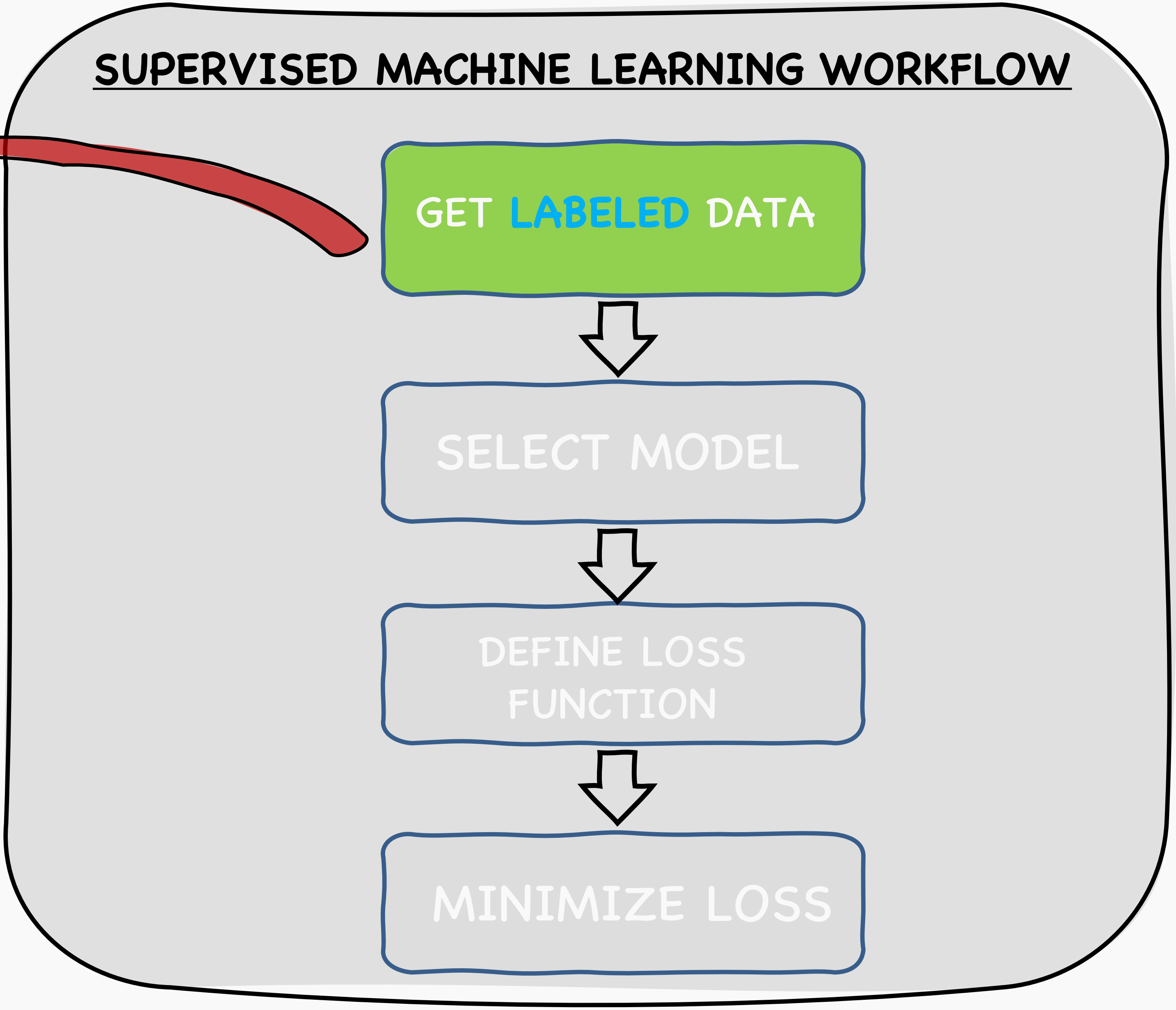
Unsupervised Learning: Learns by clustering or association

Building blocks of **supervised** machine learning



Building blocks of **supervised** machine learning

Sales vs Advertising budget,
Patient history vs heart disease
(Response & Predictor variables)



Response vs. Predictor Variables

$X = X_1, \dots, X_p$

$X_j = x_{1j}, \dots, x_{ij}, \dots, x_{nj}$

predictors

features

covariates

response variable $\boldsymbol{Y}$

is continuous

$Y = y_1, \dots, y_n$

outcome

response variable

dependent variable

TV	radio	newspaper	sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	58.5	18.5
180.8	10.8	58.4	12.9

$\underbrace{\text{TV} \quad \text{radio} \quad \text{newspaper}}_{\boldsymbol{p}\ \text{predictors}}$

n **observations**

p **predictors**

Heart Data

These data contain a binary outcome AHD for 303 patients who presented with chest pain.

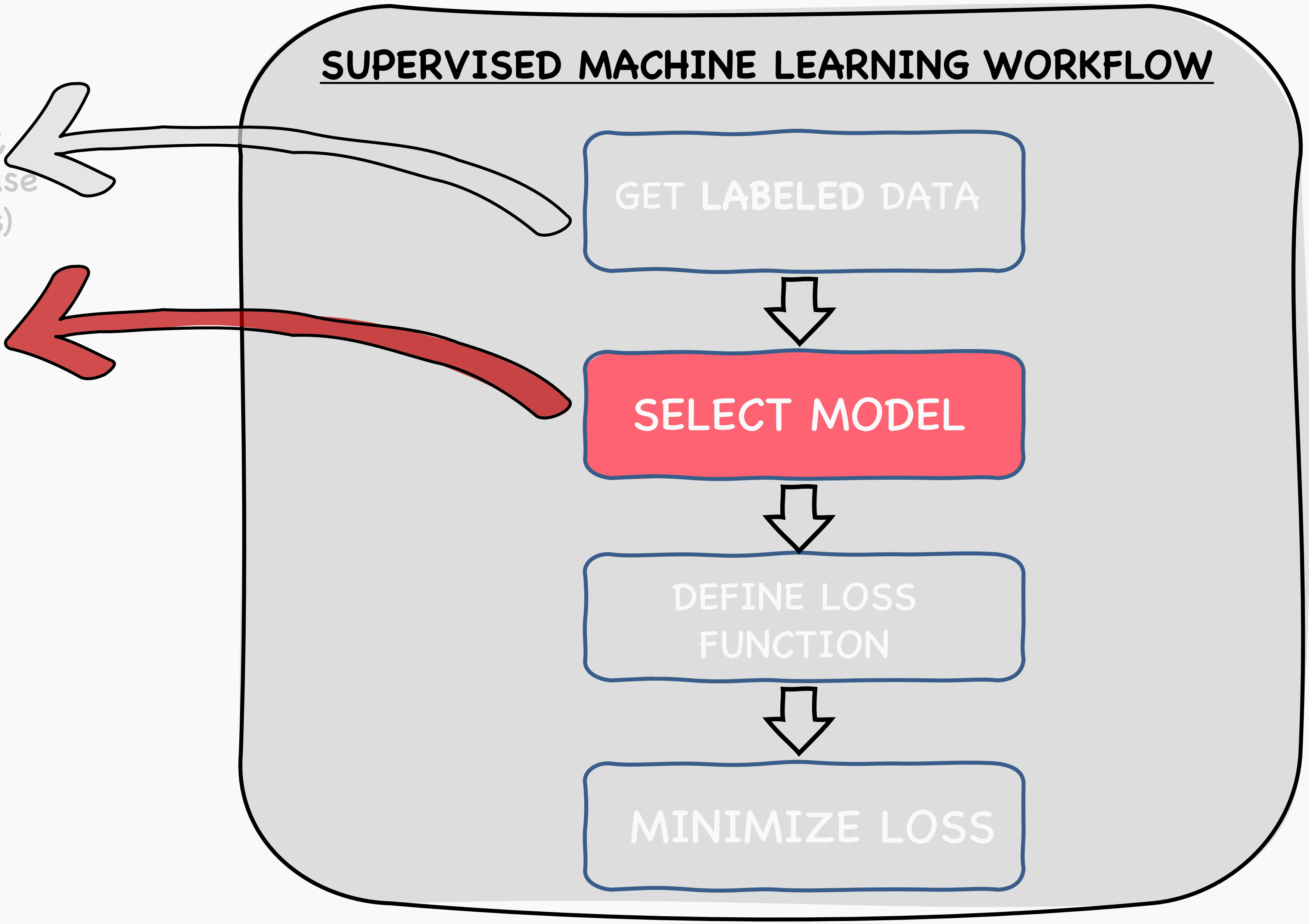
response variable ***Y*** is Yes/No

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

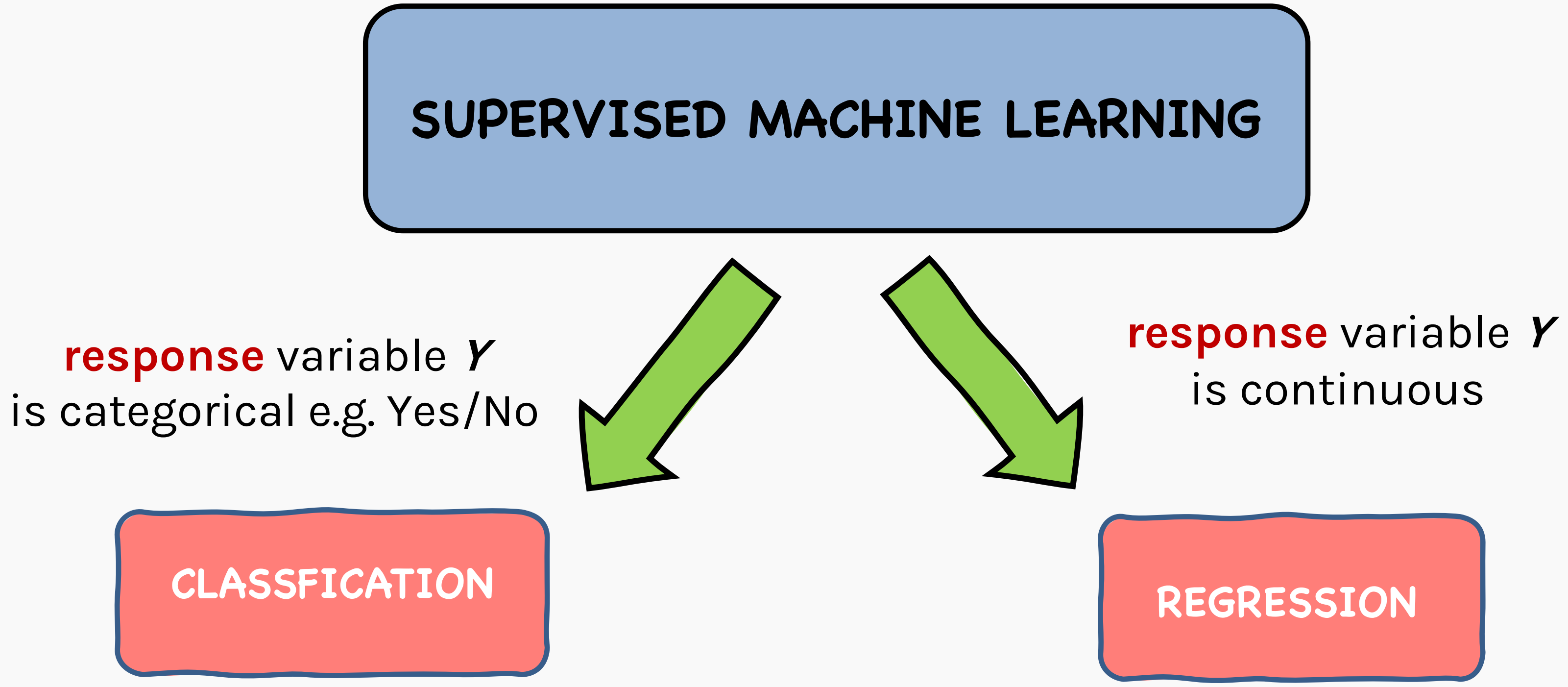
Building blocks of **supervised** machine learning

Sales vs Advertising budget,
Patient history vs heart disease
(Response & Predictor variables)

Regression,
Classification, Object
Detection



Supervised Machine Learning examples



- FRAUD DETECTION
- EMAIL SPAM DETECTION
- IMAGE CLASSIFICATION

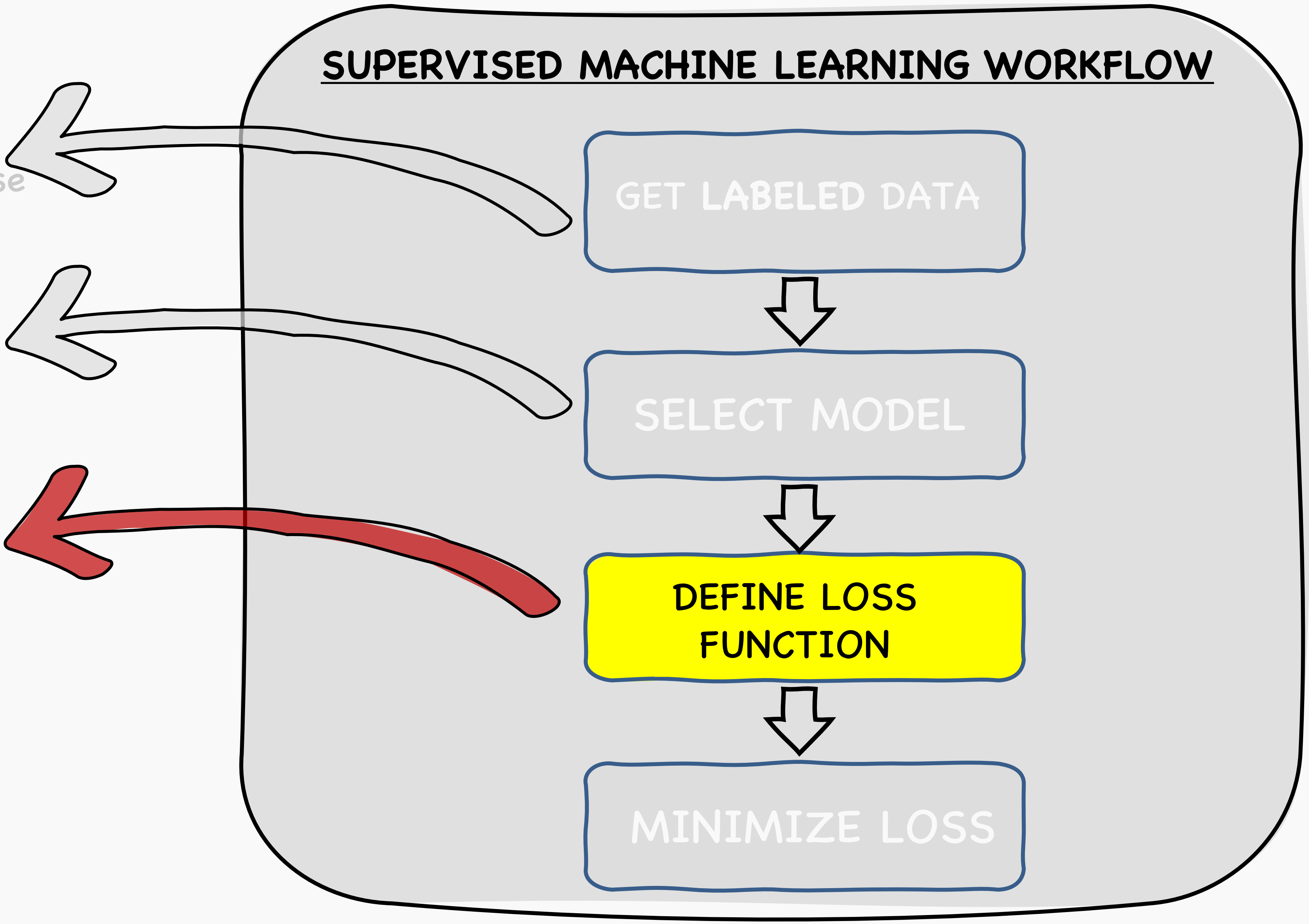
- RISK ASSESSMENT
- SCORE PREDICTION
- HOUSING PRICES

Building blocks of **supervised** machine learning

Sales vs Advertising budget,
Patient history vs heart disease
(Response & Predictor variables)

Regression,
Classification, Object
Detection

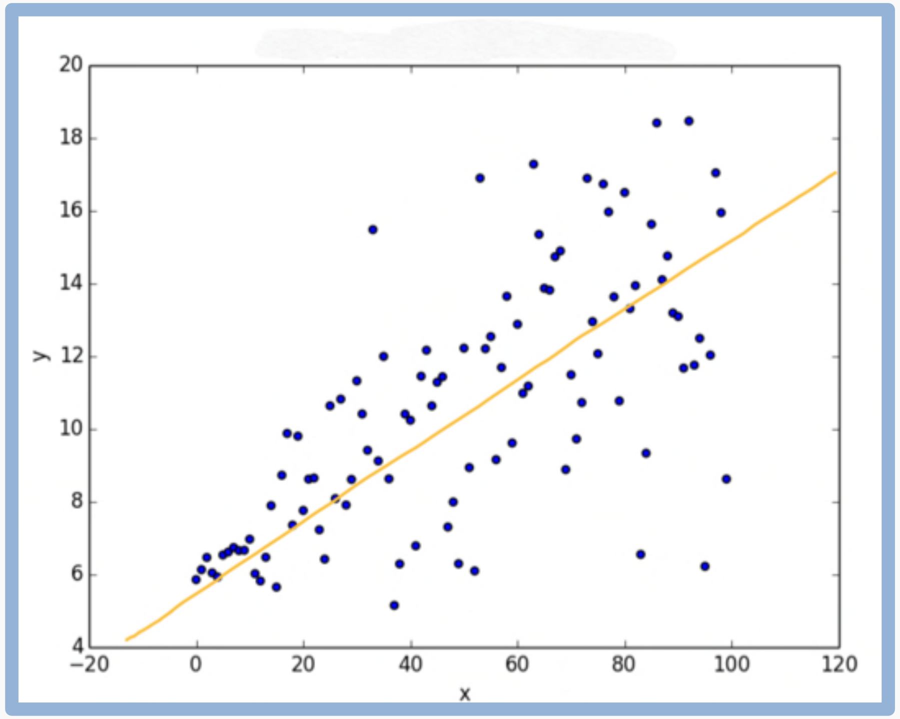
Mean Squared Error,
Binary Cross Entropy,
Categorical Cross
Entropy, Hinge Loss



Loss for linear regression

MSE as the **loss function**,

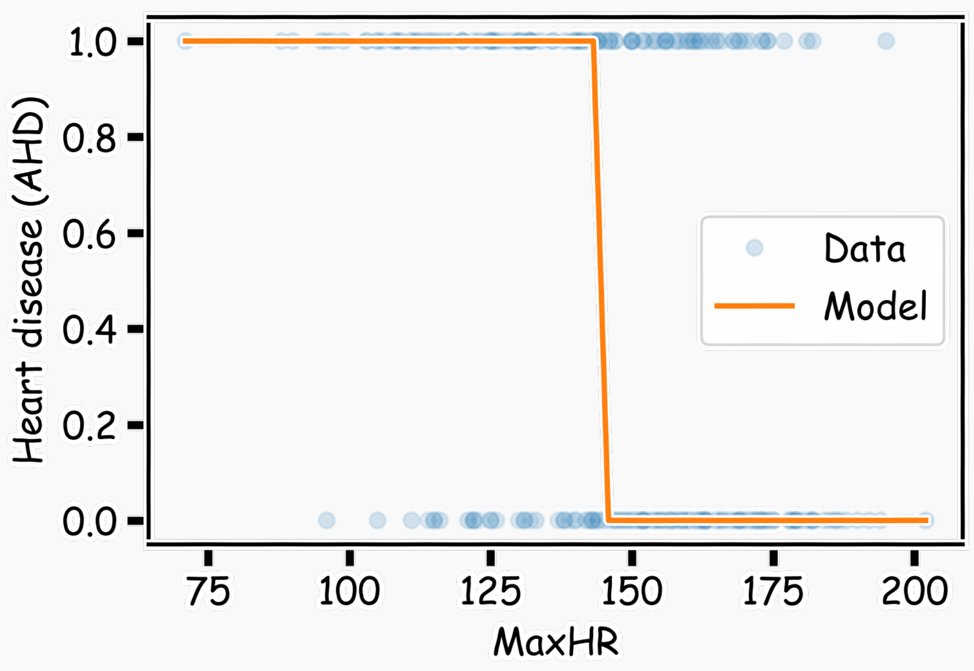
$$L(\beta_0, \beta_1) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}\left[y_i - (\beta_1 X + \beta_0)\right]^2 .$$



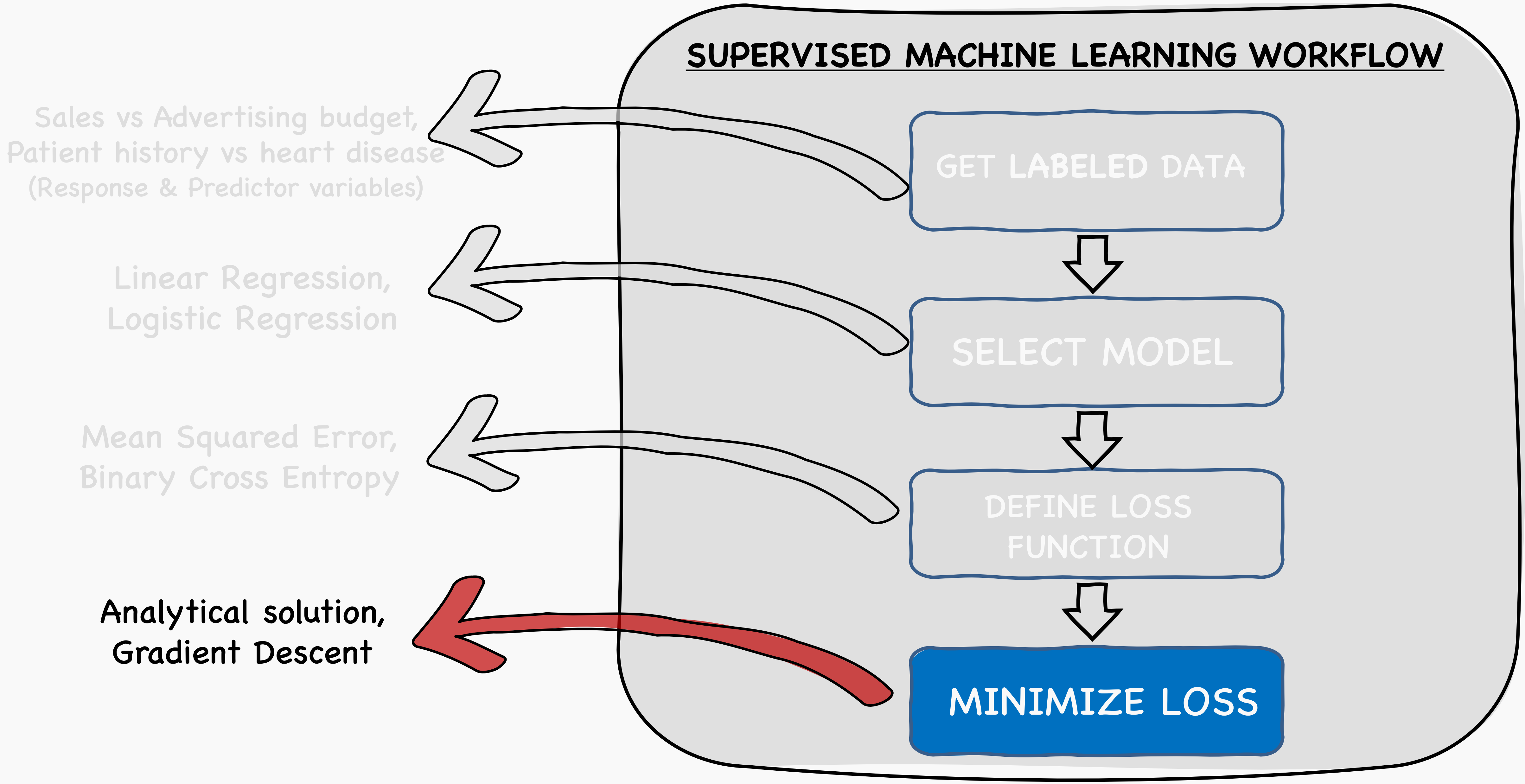
Loss function for Logistic Regression

Cross Entropy as a loss function

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

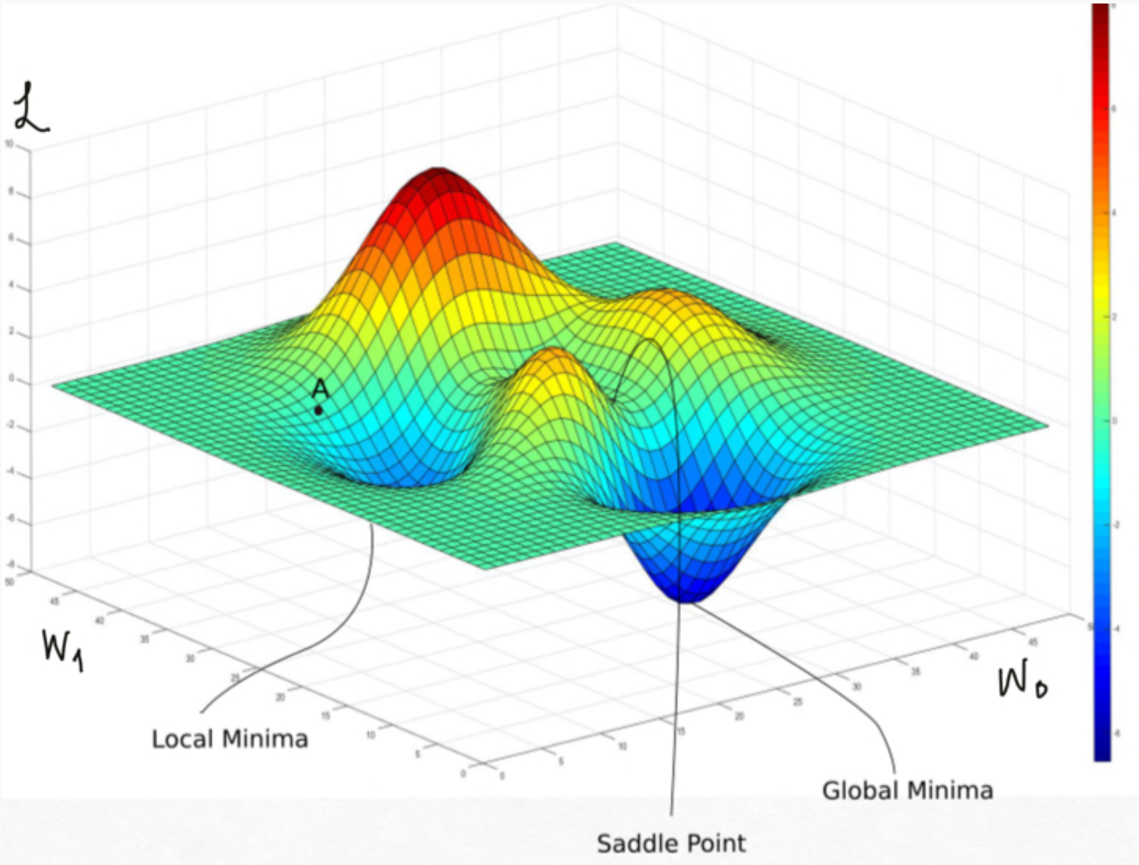


Building blocks of **supervised** machine learning



Optimization

How does one minimize a loss function?



Minima or maxima of $L(\beta_0, \beta_1)$ must occur at points where the gradient (slope)

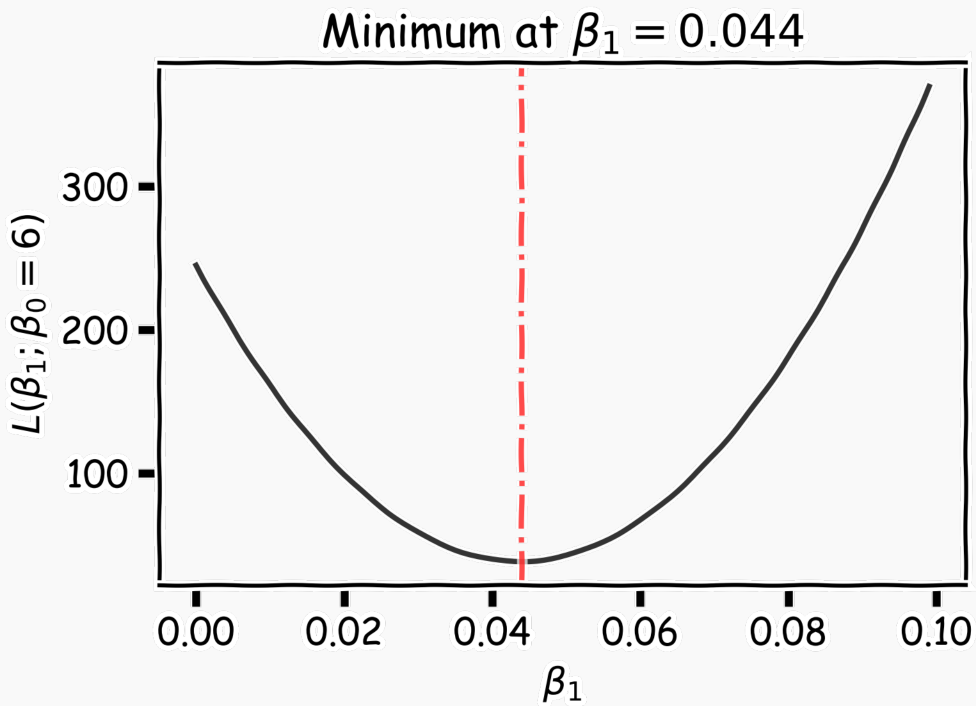
$$\nabla L = \left[\frac{\partial L}{\partial \beta_0}, \frac{\partial L}{\partial \beta_1}\right] = 0$$

- Brute Force: Try every combination
- Exact: Solve the above equations
- Greedy Algorithm: Gradient Descent

Optimization: Brute force

A way to estimate $\operatorname{argmin}_{\beta_0,\beta_1} L$ is to calculate the loss function for every possible β_0 and β_1. Then select the β_0 and β_1 where the loss function is minimum.

E.g. the loss function for different β_1 when β_0 is fixed to be 6:



Very **computationally expensive** with many coefficients

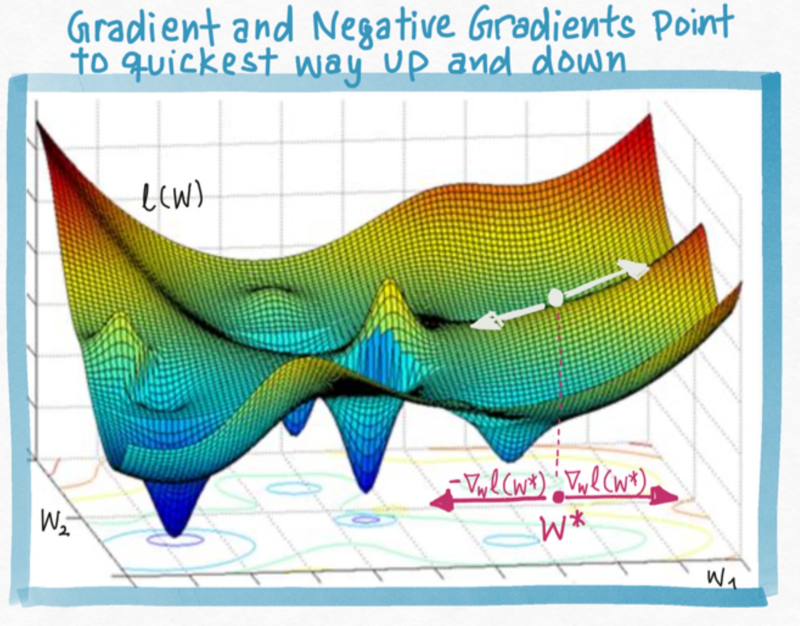
Gradient Descent

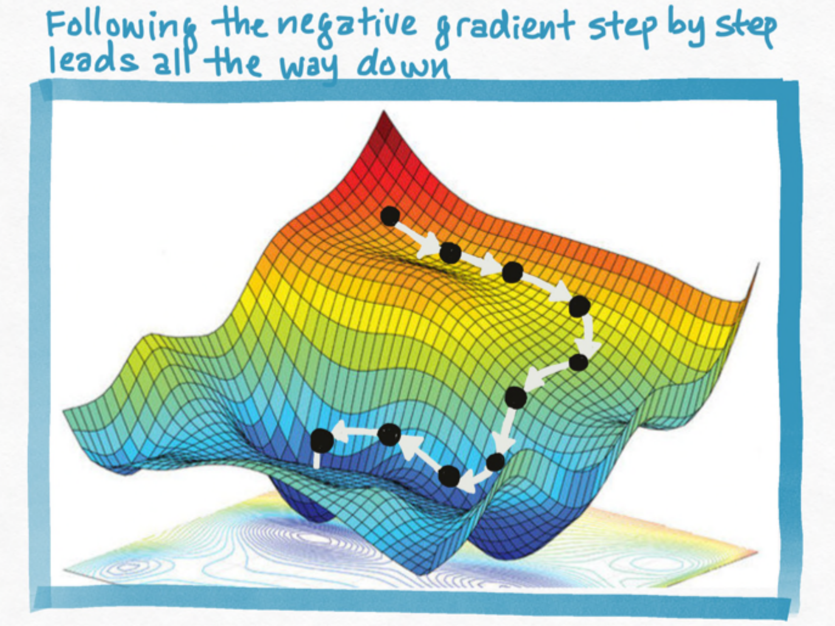
When we can’t analytically solve for the stationary points of the gradient, we can still exploit the information in the gradient.

The gradient ∇L at any point is the direction of the steepest increase. The negative gradient is the direction of steepest decrease.

By following the –ve gradient, we can eventually find the lowest point.

This method is called Gradient Descent





Outline

1. Introduction to Artificial Neural Networks
2. Review of basic concepts
3. **Single Neuron Network (‘Perceptron’)**
4. Multi-Layer Perceptron (MLP)

Logistic Regression Revisited

True y

$x_1 \rightarrow$ Affine $\rightarrow h_1 = \beta_0 + \beta_1 x_1 \rightarrow$ Activation $\rightarrow p_1 = \dfrac{1}{1 + e^{-h_1}} \rightarrow$ Loss Fun $\rightarrow \mathcal{L}_1(\beta) = -y_1 \ln(p_1) - (1 - y_1)\ln(1 - p_1)$

Logistic Regression Revisited

$$x_1 \rightarrow \boxed{\text{Affine}} \rightarrow h_1 = \beta_0 + \beta_1 x_1 \rightarrow \boxed{\text{Activation}} \rightarrow p_1 = \frac{1}{1+e^{-h_1}} \rightarrow \boxed{\text{Loss Fun}} \rightarrow \mathcal{L}_1(\beta) = -y_1 \ln(p_1) - (1-y_1)\ln(1-p_1)$$

$$x_2 \rightarrow \boxed{\text{Affine}} \rightarrow h_2 = \beta_0 + \beta_1 x_2 \rightarrow \boxed{\text{Activation}} \rightarrow p_2 = \frac{1}{1+e^{-h_2}} \rightarrow \boxed{\text{Loss Fun}} \rightarrow \mathcal{L}_2(\beta) = -y_2 \ln(p_1) - (1-y_2)\ln(1-p_2)$$

Logistic Regression Revisited

$$x_1 \longrightarrow \boxed{\text{Affine}} \longrightarrow h_1 = \beta_0 + \beta_1 x_1 \longrightarrow \boxed{\text{Activation}} \longrightarrow p_1 = \frac{1}{1+e^{-h_1}} \longrightarrow \boxed{\text{Loss Fun}} \longrightarrow \mathcal{L}_1(\beta) = -y_1 \ln(p_1) - (1-y_1)\ln(1-p_1)$$

$$x_2 \longrightarrow \boxed{\text{Affine}} \longrightarrow h_2 = \beta_0 + \beta_1 x_2 \longrightarrow \boxed{\text{Activation}} \longrightarrow p_2 = \frac{1}{1+e^{-h_2}} \longrightarrow \boxed{\text{Loss Fun}} \longrightarrow \mathcal{L}_2(\beta) = -y_2 \ln(p_1) - (1-y_2)\ln(1-p_2)$$

$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$$

Logistic Regression Revisited

$x_1 \longrightarrow$ Affine $\longrightarrow h_1 = \beta_0 + \beta_1 x_1 \longrightarrow$ Activation $\longrightarrow p_1 = \dfrac{1}{1+e^{-h_1}} \longrightarrow$ Loss Fun $\longrightarrow \mathcal{L}_1(\beta) = -y_1 \ln(p_1) - (1 - y_1)\ln(1 - p_1)$

$x_2 \longrightarrow$ Affine $\longrightarrow h_2 = \beta_0 + \beta_1 x_2 \longrightarrow$ Activation $\longrightarrow p_2 = \dfrac{1}{1+e^{-h_2}} \longrightarrow$ Loss Fun $\longrightarrow \mathcal{L}_2(\beta) = -y_2 \ln(p_1) - (1 - y_2)\ln(1 - p_2)$

$\vdots \qquad \vdots \qquad \vdots$

$x_n \longrightarrow$ Affine $\longrightarrow h_n = \beta_0 + \beta_1 x_n \longrightarrow$ Activation $\longrightarrow p_n = \dfrac{1}{1+e^{-h_n}} \longrightarrow$ Loss Fun $\longrightarrow \mathcal{L}_n(\beta) = -y_n \ln(p_n) - (1 - y_n)\ln(1 - p_{1n})$

Logistic Regression Revisited

$$x_1 \longrightarrow \boxed{\text{Affine}} \longrightarrow h_1 = \beta_0 + \beta_1 x_1 \longrightarrow \boxed{\text{Activation}} \longrightarrow p_1 = \frac{1}{1+e^{-h_1}} \longrightarrow \boxed{\text{Loss Fun}} \longrightarrow \mathcal{L}_1(\beta) = -y_1 \ln(p_1) - (1-y_1)\ln(1-p_1)$$

$$x_2 \longrightarrow \boxed{\text{Affine}} \longrightarrow h_2 = \beta_0 + \beta_1 x_2 \longrightarrow \boxed{\text{Activation}} \longrightarrow p_2 = \frac{1}{1+e^{-h_2}} \longrightarrow \boxed{\text{Loss Fun}} \longrightarrow \mathcal{L}_2(\beta) = -y_2 \ln(p_1) - (1-y_2)\ln(1-p_2)$$

$$\vdots \qquad \vdots \qquad \vdots$$

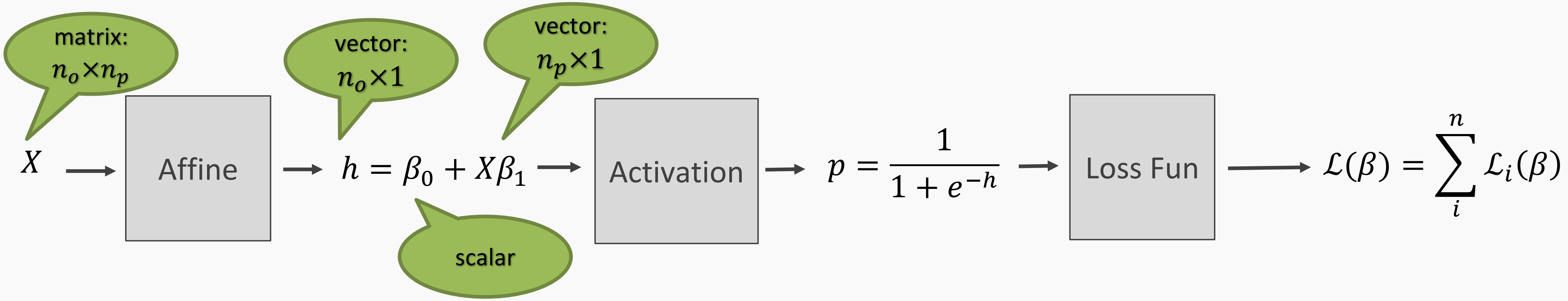
$$x_n \longrightarrow \boxed{\text{Affine}} \longrightarrow h_n = \beta_0 + \beta_1 x_n \longrightarrow \boxed{\text{Activation}} \longrightarrow p_n = \frac{1}{1+e^{-h_n}} \longrightarrow \boxed{\text{Loss Fun}} \longrightarrow \mathcal{L}_n(\beta) = -y_n \ln(p_n) - (1-y_n)\ln(1-p_{1n})$$

$$\boxed{\mathcal{L}(\beta) = \sum_i^n \mathcal{L}_i(\beta)}$$

Build our first ANN

n_p: number of predictors

n_o: number of observations



Build our first ANN

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

$$X \longrightarrow \boxed{\text{Affine}} \longrightarrow h = XW + b \longrightarrow \boxed{\text{Activation}} \longrightarrow \hat{y} = \frac{1}{1 + e^{-h}} \longrightarrow \boxed{\text{Loss Fun}} \longrightarrow \mathcal{L}(\beta) = \sum_{i}^{n} \mathcal{L}_i(\beta)$$

Build our first ANN

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

$$X \longrightarrow \boxed{\text{Affine}} \longrightarrow h = X{\color{blue}W} + b \longrightarrow \boxed{\text{Activation}} \longrightarrow {\color{blue}\hat{y}} = \frac{1}{1+e^{-h}} \longrightarrow \boxed{\text{Loss Fun}} \longrightarrow \mathcal{L}(\beta) = \sum_{i}^{n} \mathcal{L}_i(\beta)$$

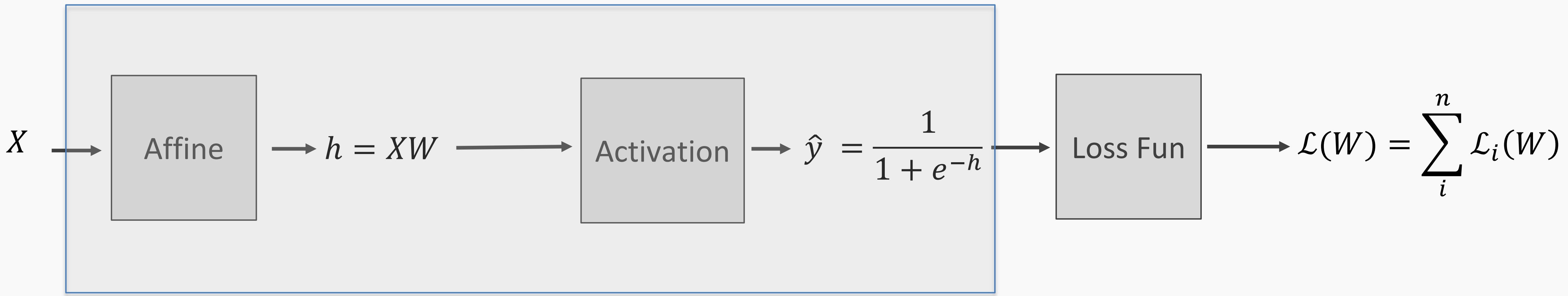
$$X \longrightarrow \boxed{\text{Affine}} \longrightarrow h = XW \longrightarrow \boxed{\text{Activation}} \longrightarrow \hat{y} = \frac{1}{1+e^{-h}} \longrightarrow \boxed{\text{Loss Fun}} \longrightarrow \mathcal{L}(\beta) = \sum_{i}^{n} \mathcal{L}_i(\beta)$$

$$X = \begin{bmatrix} 1 & X_{11} & \dots & X_{1p} \\ 1 & \vdots & \dots & \vdots \\ 1 & X_{o1} & \dots & X_{op} \end{bmatrix} \quad \mathrm{W} = \begin{bmatrix} b \\ W_1 \\ \vdots \\ W_p \end{bmatrix}$$

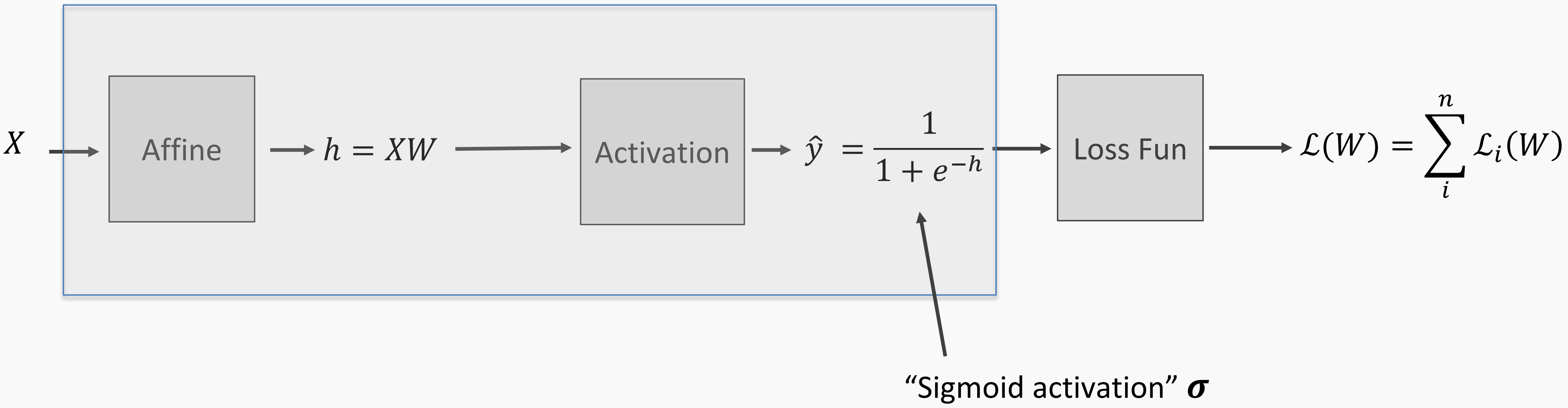
Build our first ANN

$$X \longrightarrow \boxed{\text{Affine}} \longrightarrow h = XW \longrightarrow \boxed{\text{Activation}} \longrightarrow \hat{y} = \frac{1}{1 + e^{-h}} \longrightarrow \boxed{\text{Loss Fun}} \longrightarrow \mathcal{L}(W) = \sum_{i}^{n} \mathcal{L}_i(W)$$

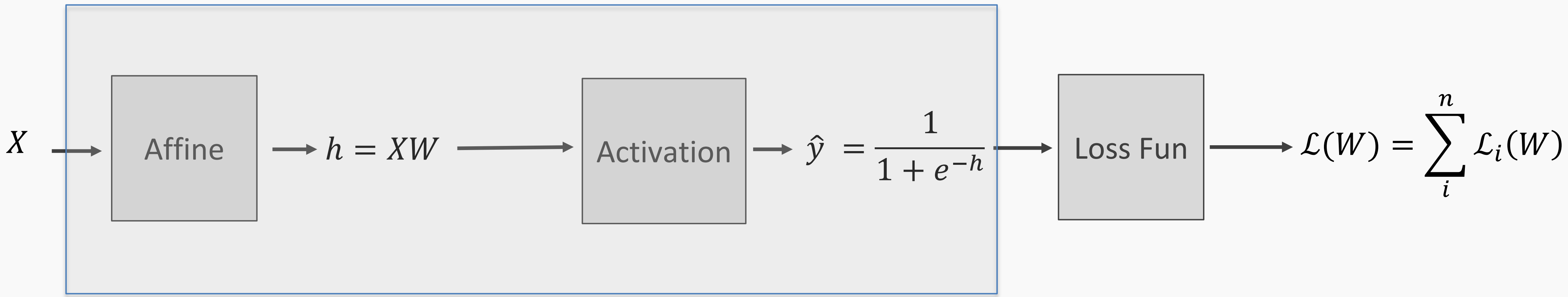
Build our first ANN

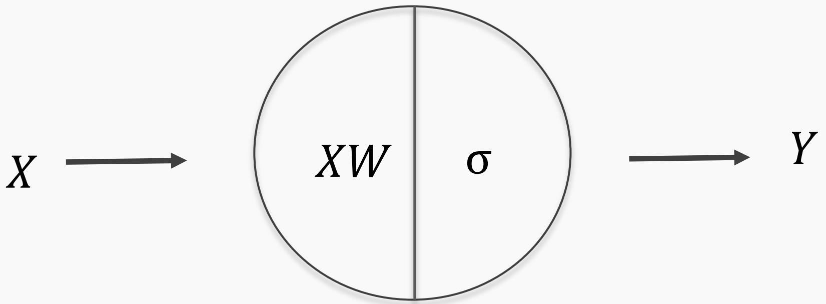


Build our first ANN

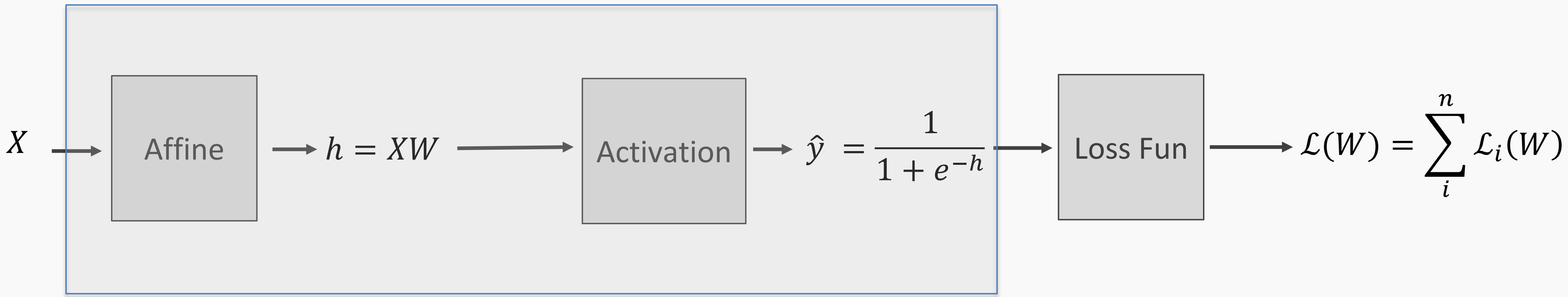


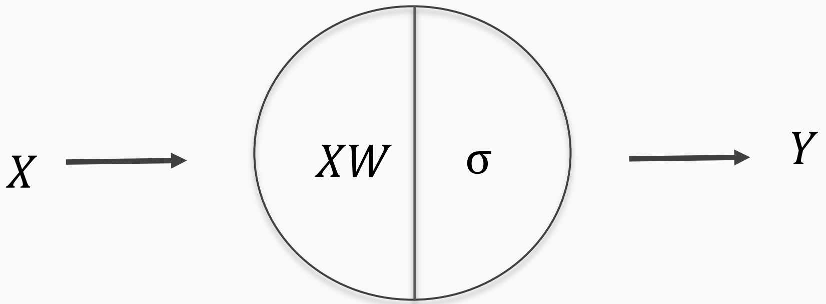
Build our first ANN





Build our first ANN

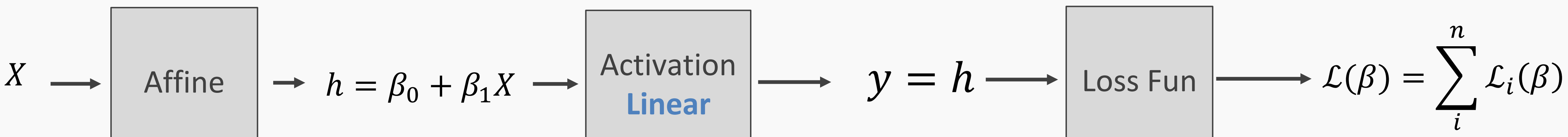


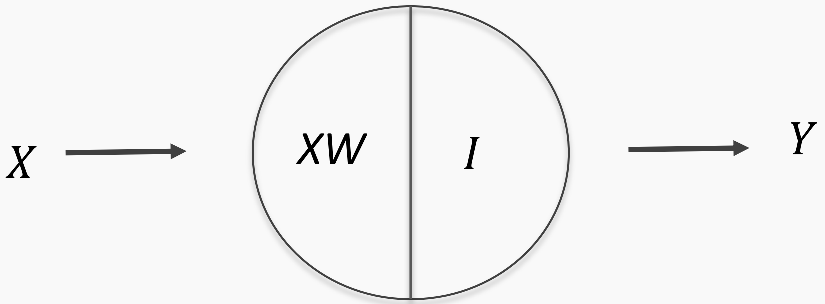


Single Neuron Neural "Network"

Up to this point we just re-branded logistic regression to look like a neuron.

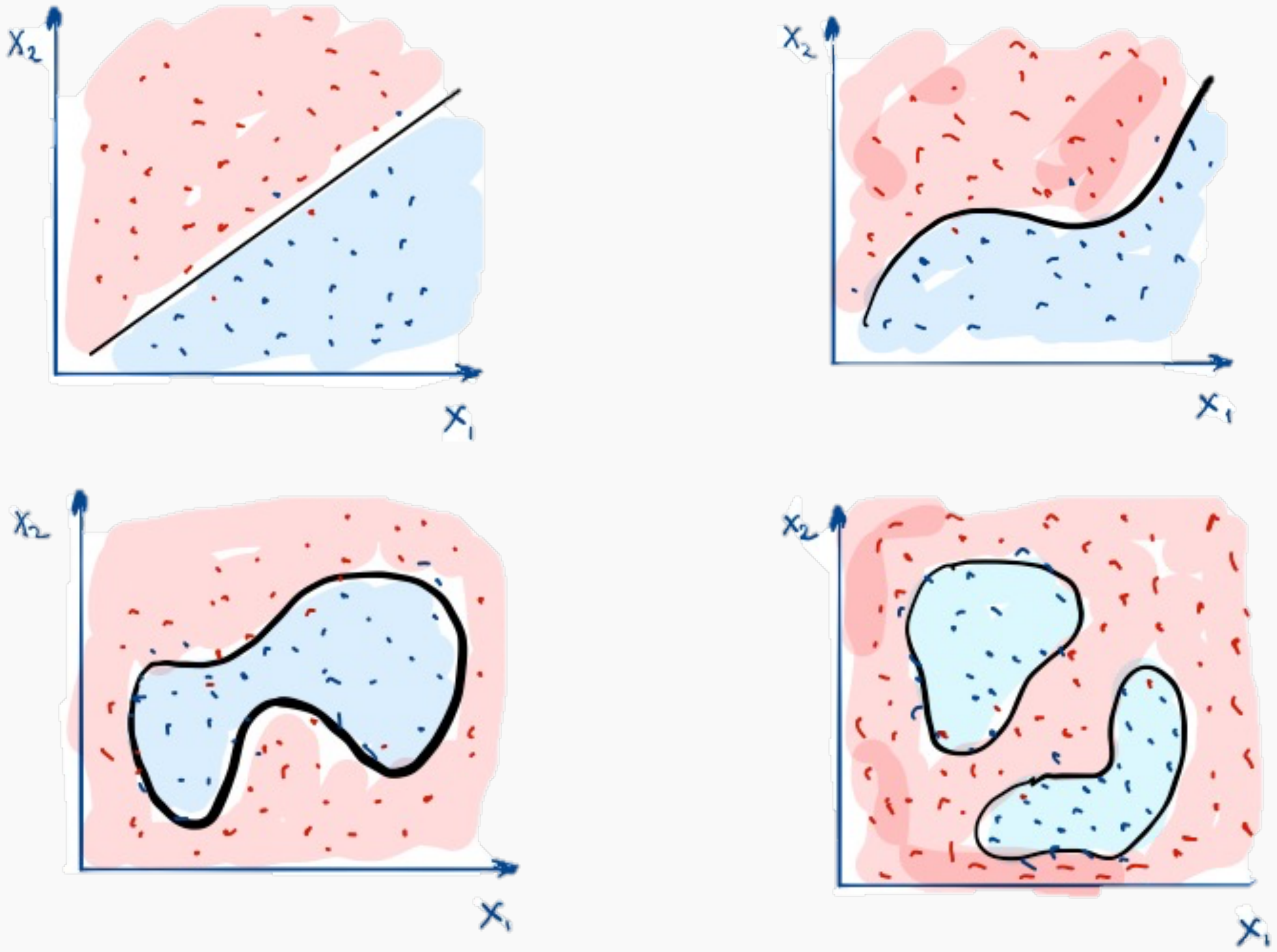
How about linear regression?



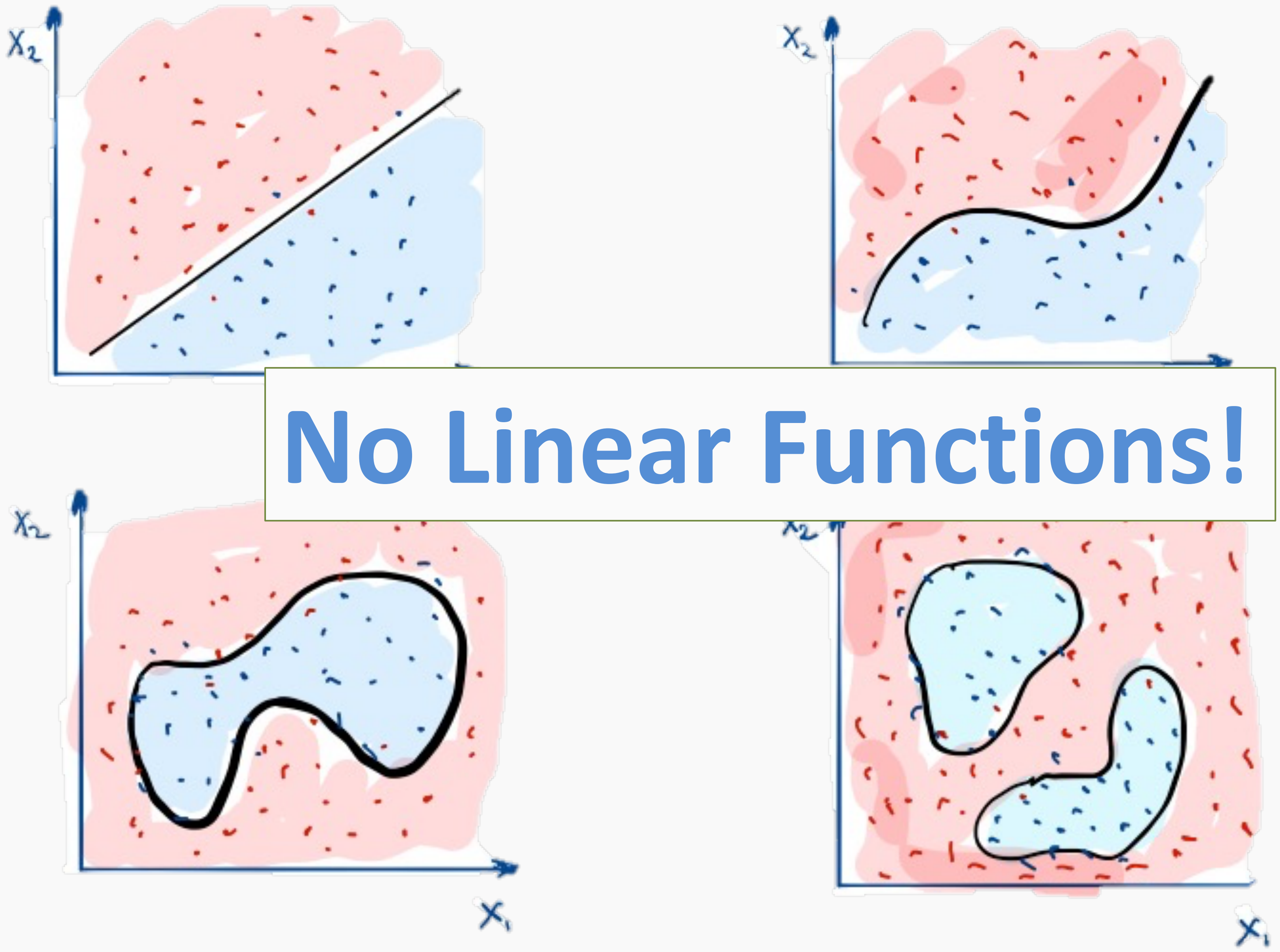


Where I is the identity function

So what's the big deal about Neural Networks?

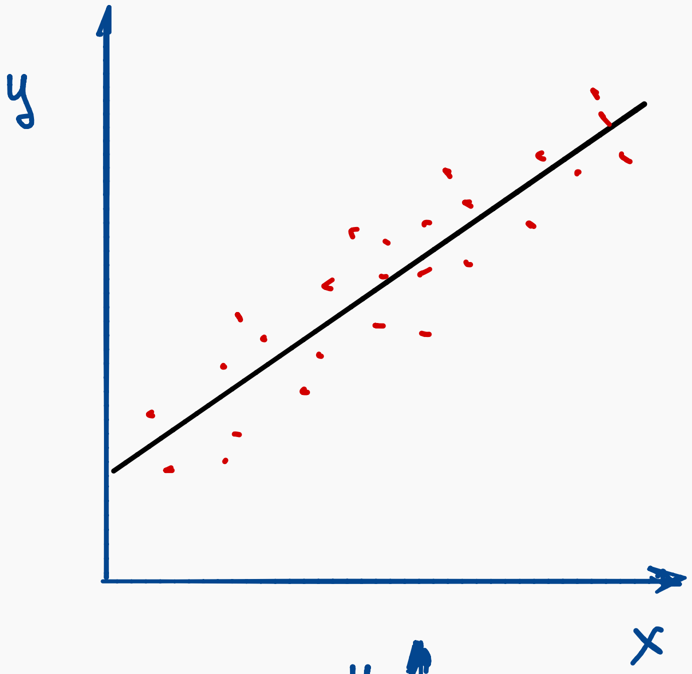


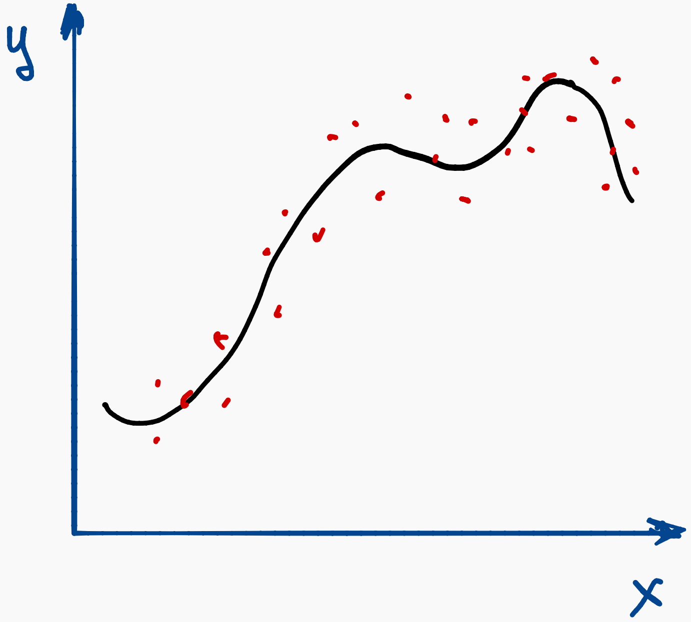
So what's the big deal about Neural Networks?

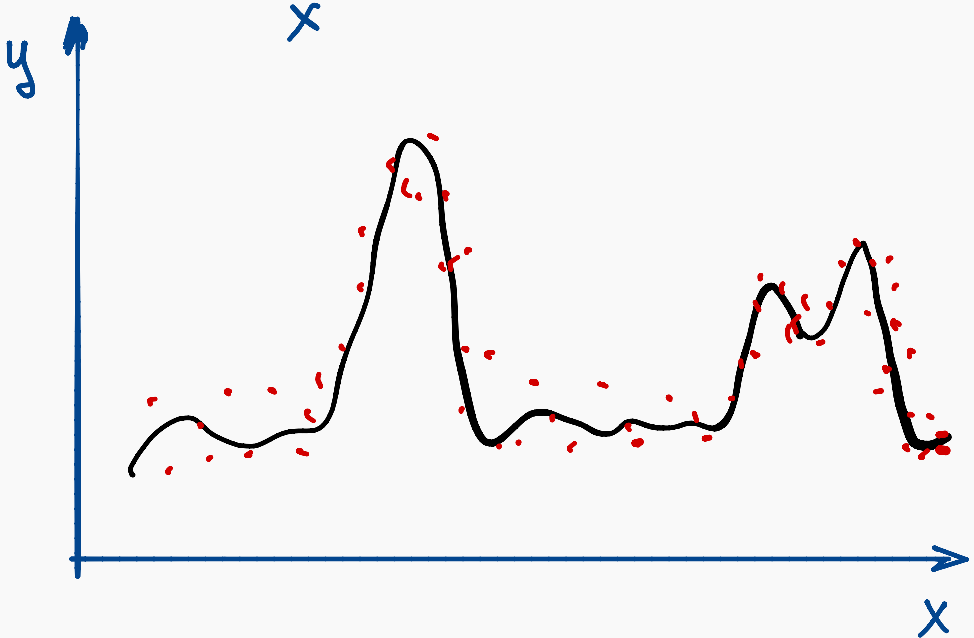


No Linear Functions!

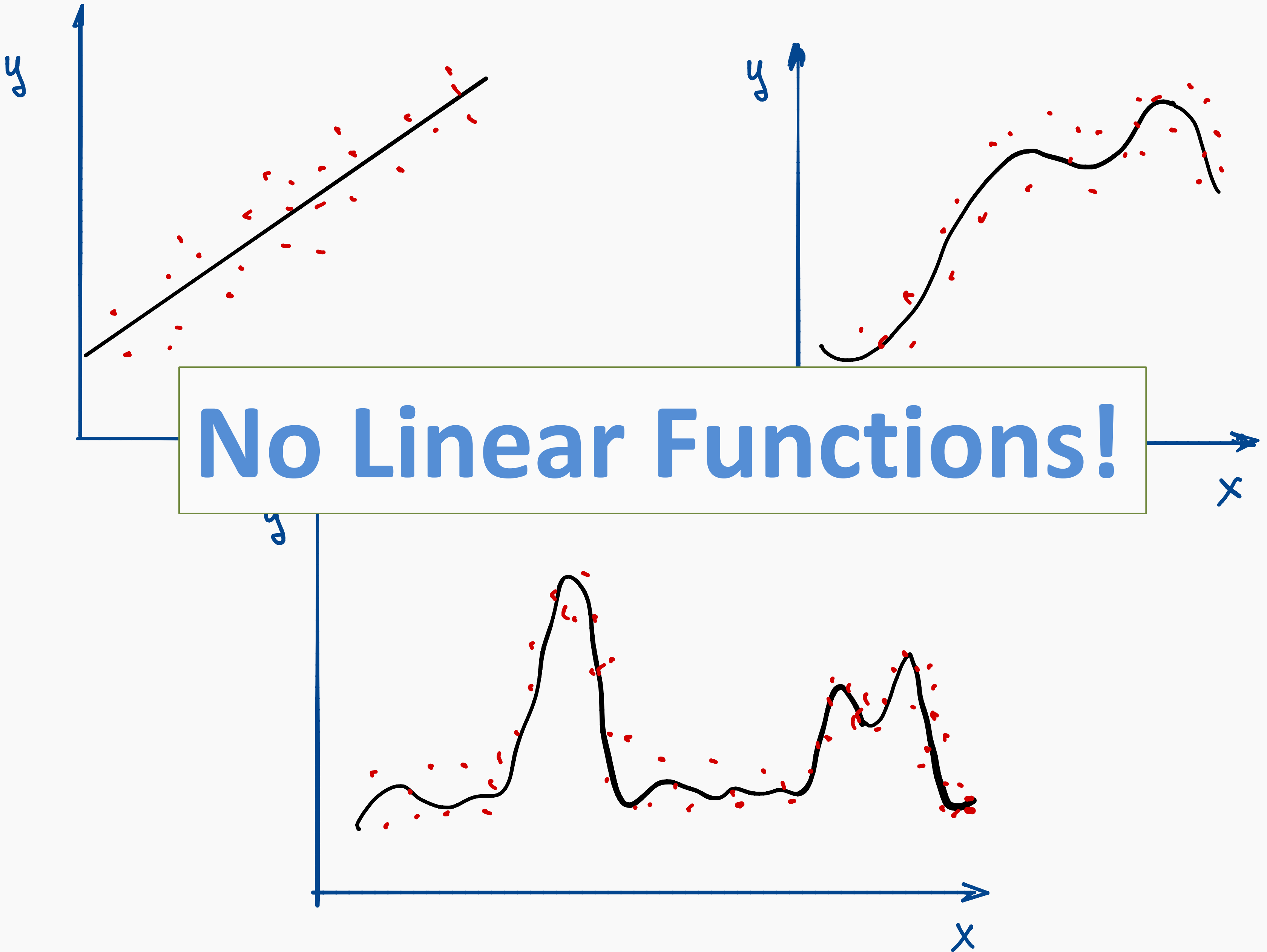
For regression?







For regression?



No Linear Functions!

Outline

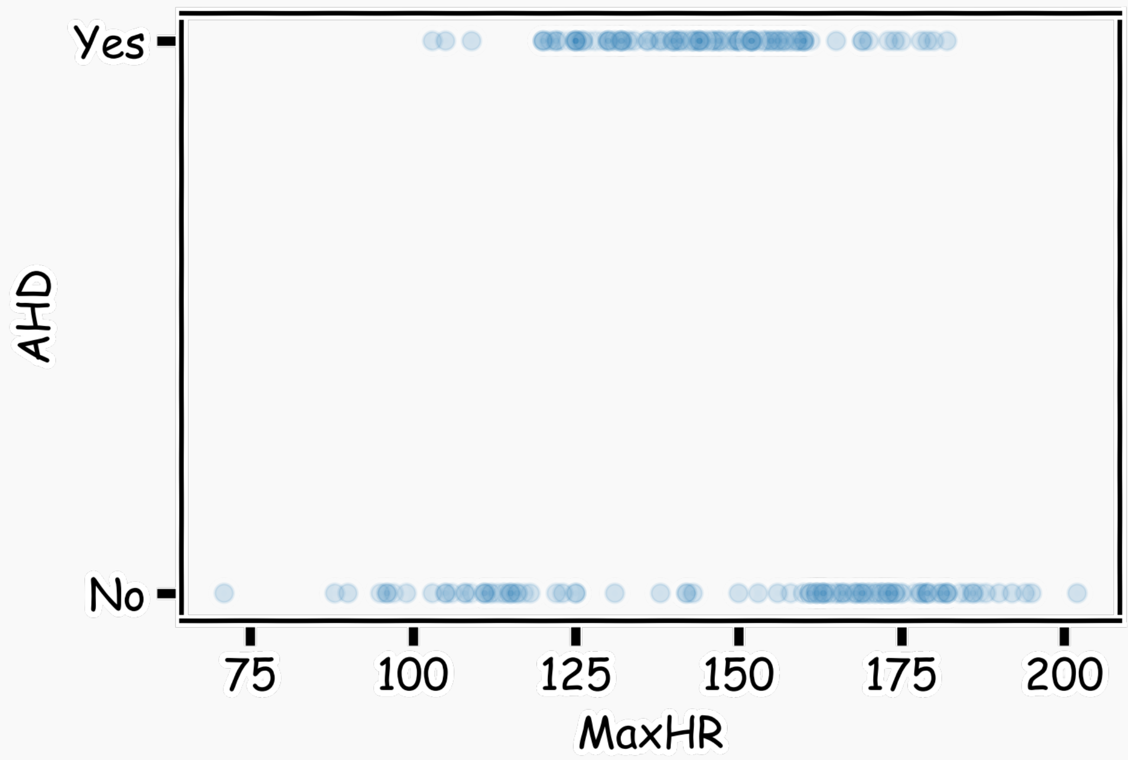
1. Introduction to Artificial Neural Networks
2. Review of Classification and Logistic Regression
3. Single Neuron Network (‘Perceptron’)
4. **Multi-Layer Perceptron (MLP)**

Example Using Heart Data

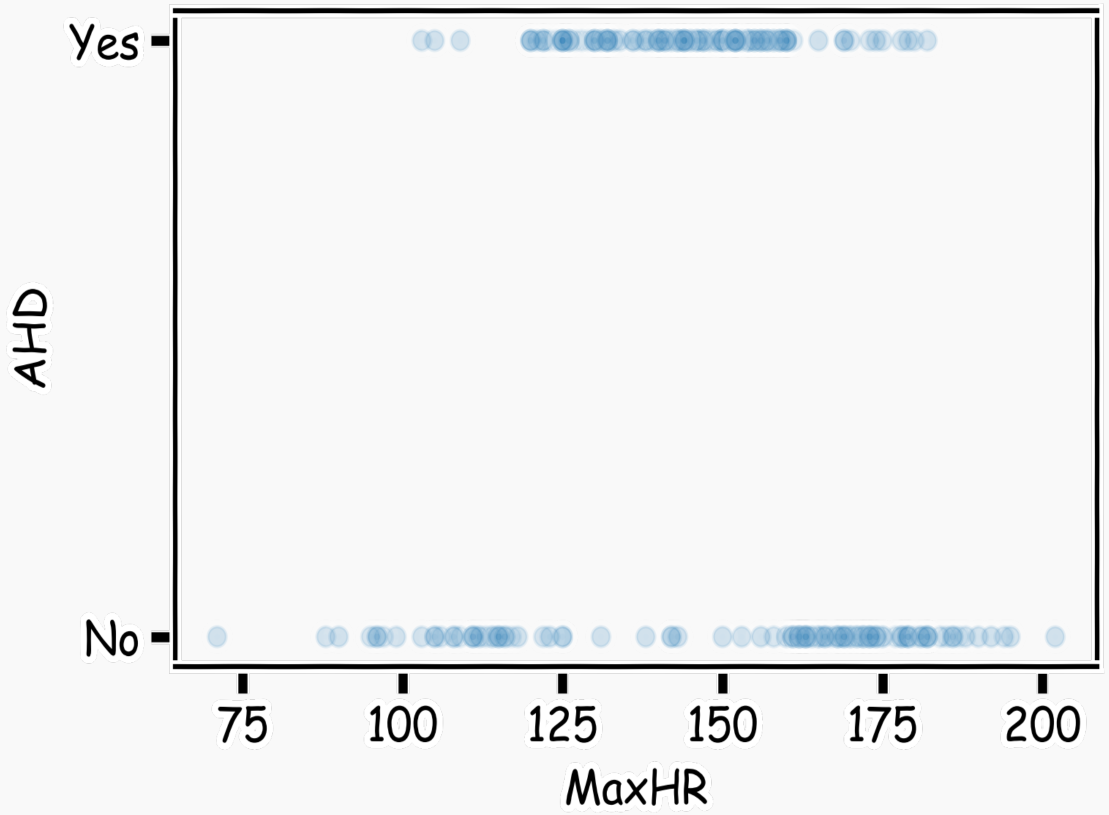
Slightly modified data to illustrate concepts.

Example Using Heart Data

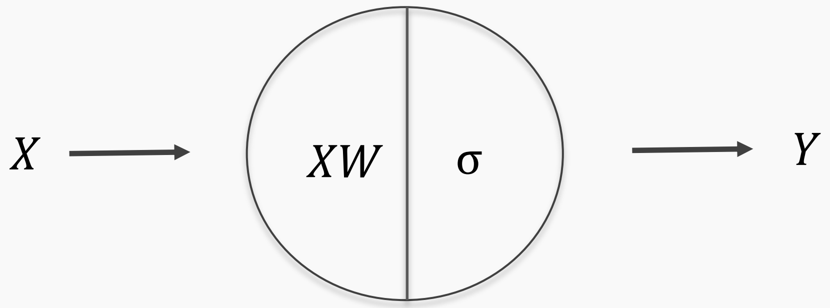
Slightly modified data to illustrate a point.



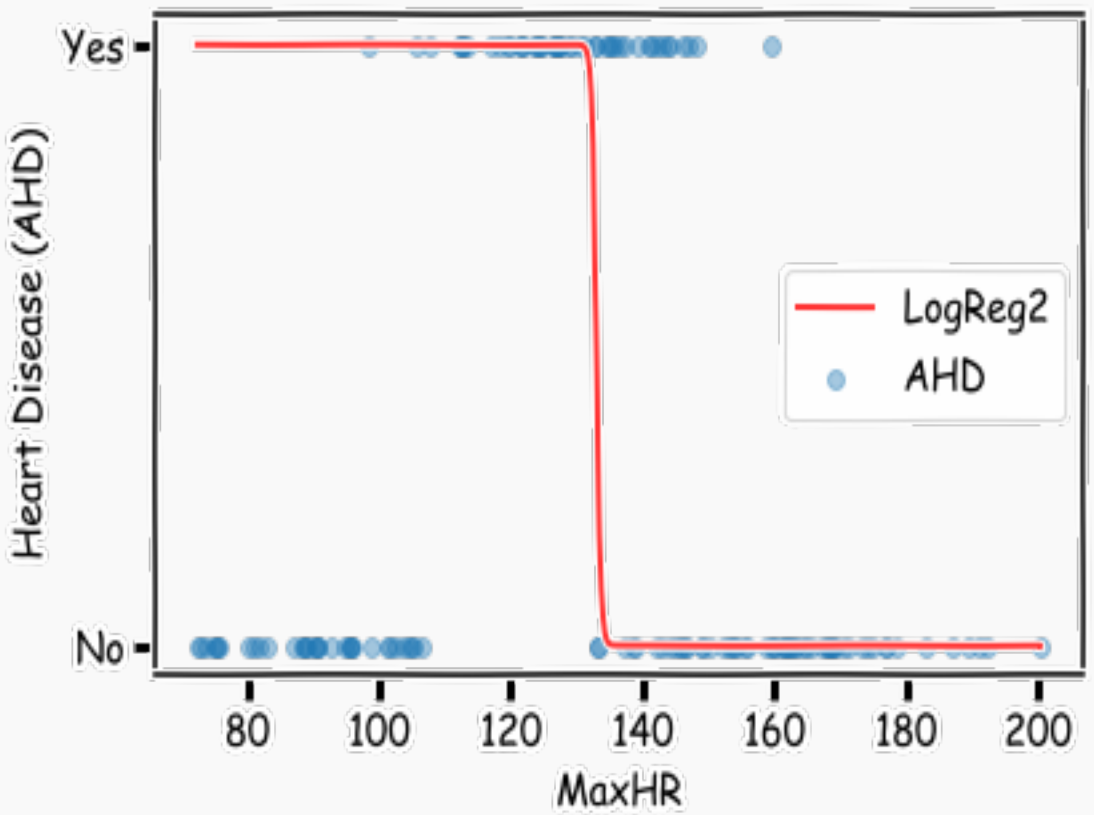
Example Using Heart Data



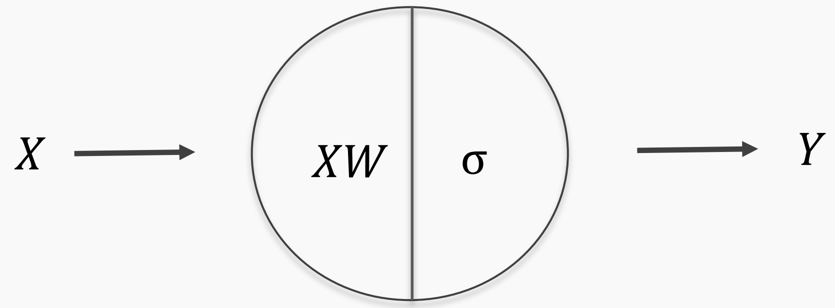
Example Using Heart Data



Choose W such as

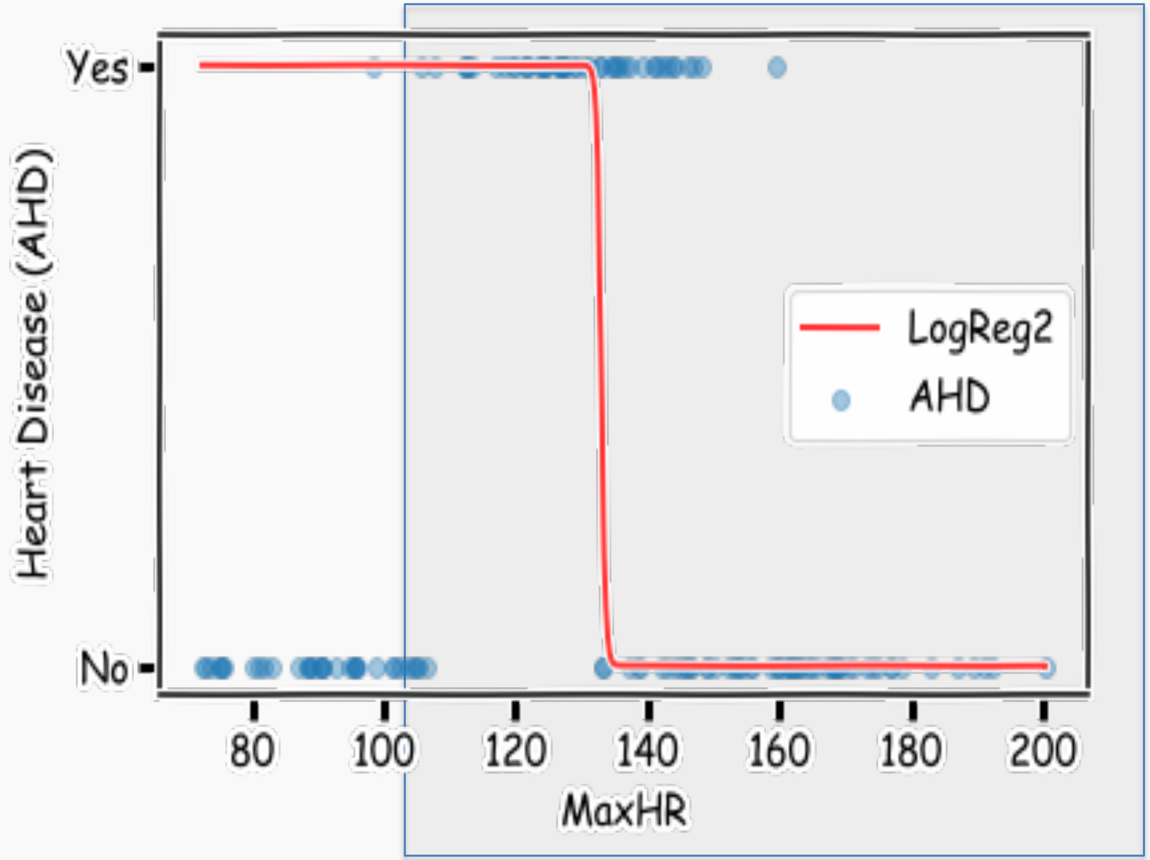


Example Using Heart Data

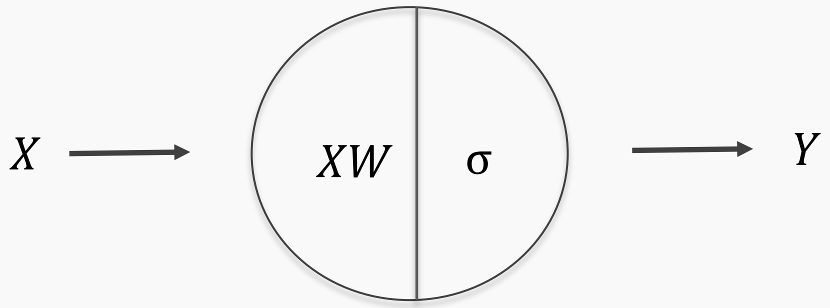


Right part of data is fitted well

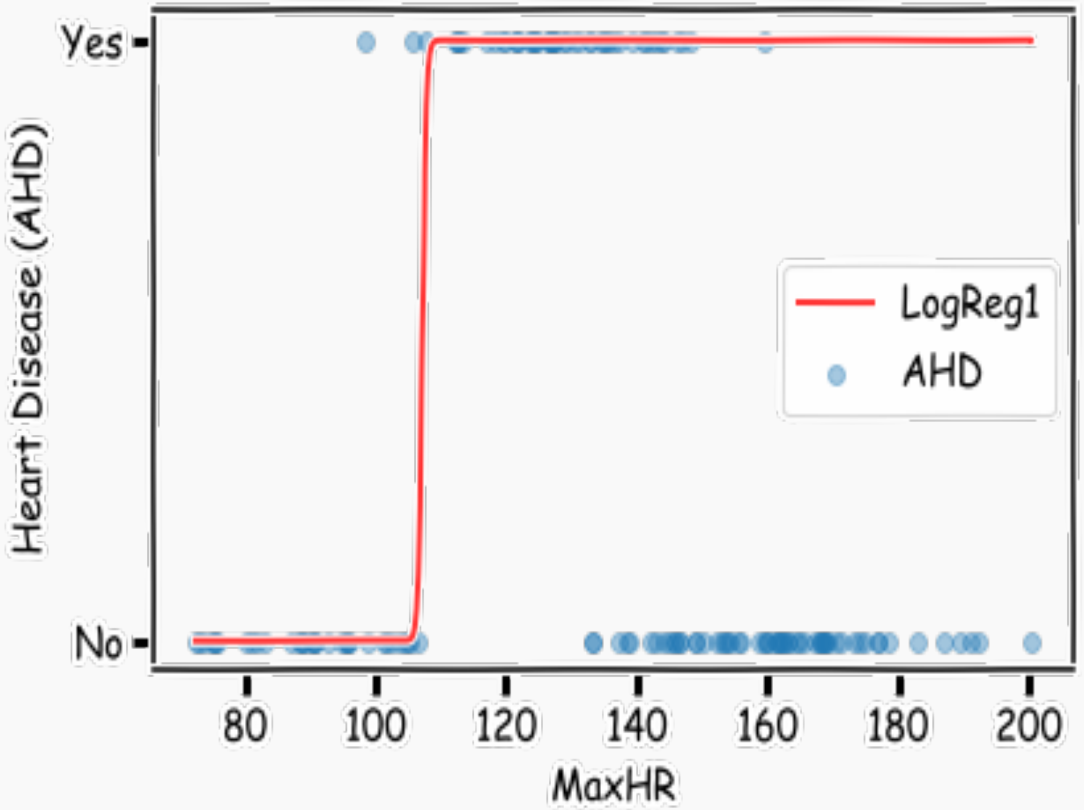
Choose W such as



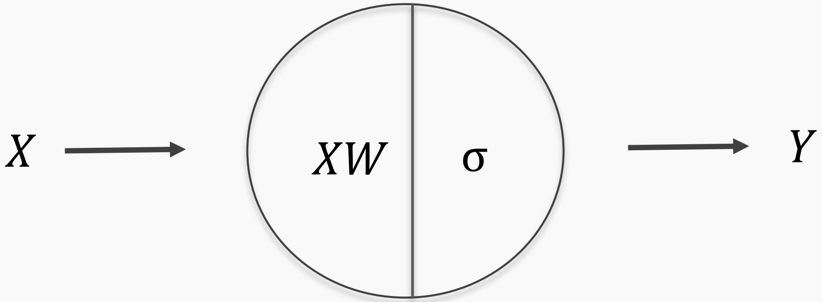
Example Using Heart Data



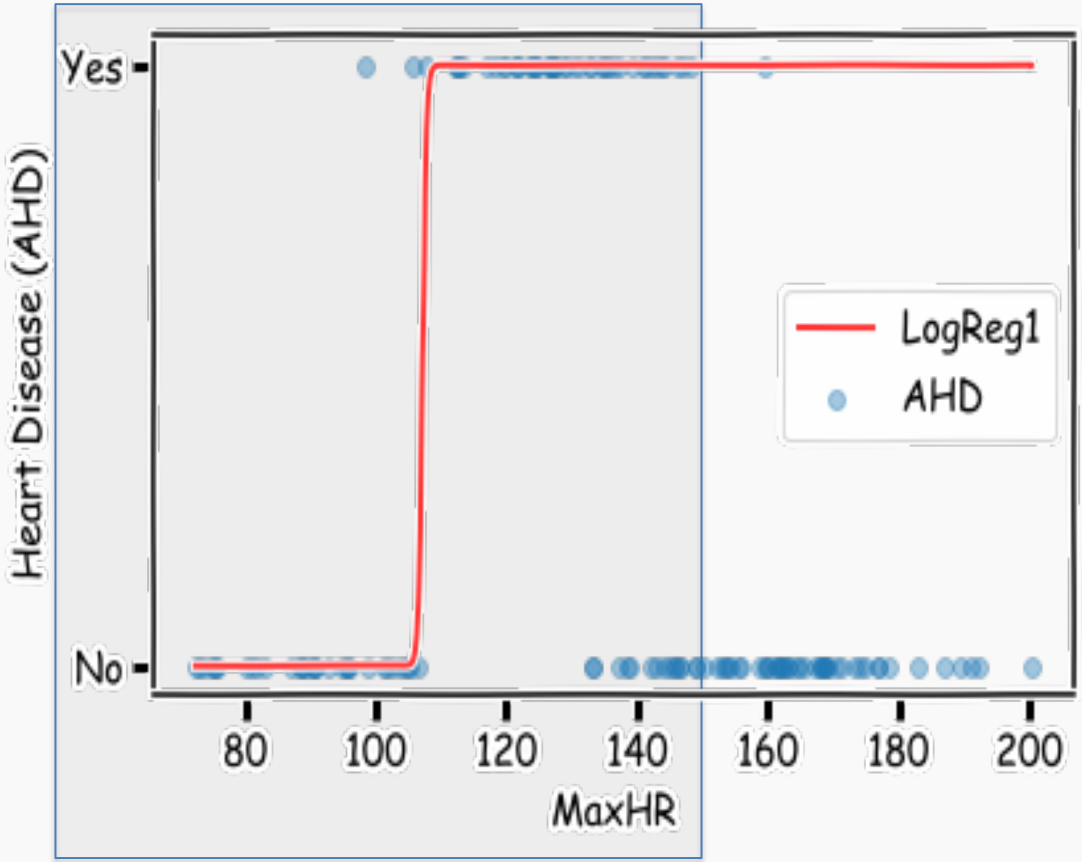
Choose W such as



Example Using Heart Data



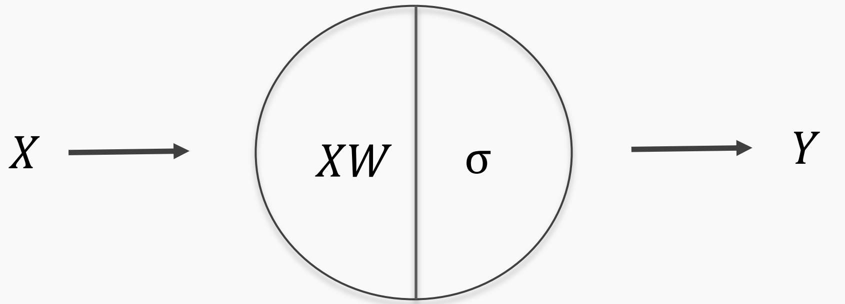
Choose W such as

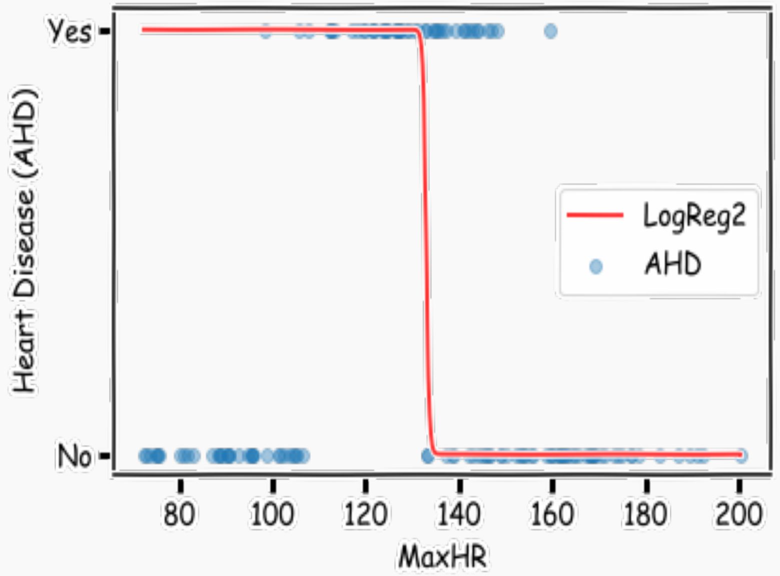


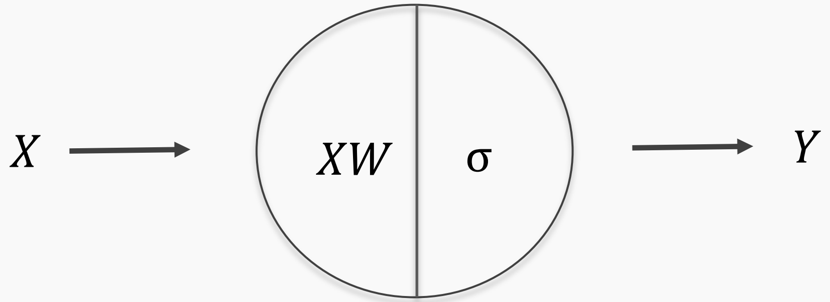
Left part of data is fitted well

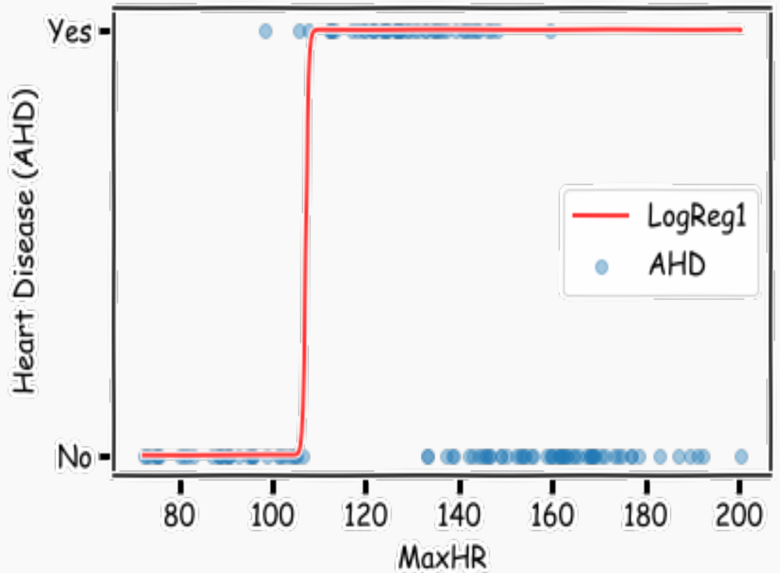
Example Using Heart Data

Two regions, two nodes

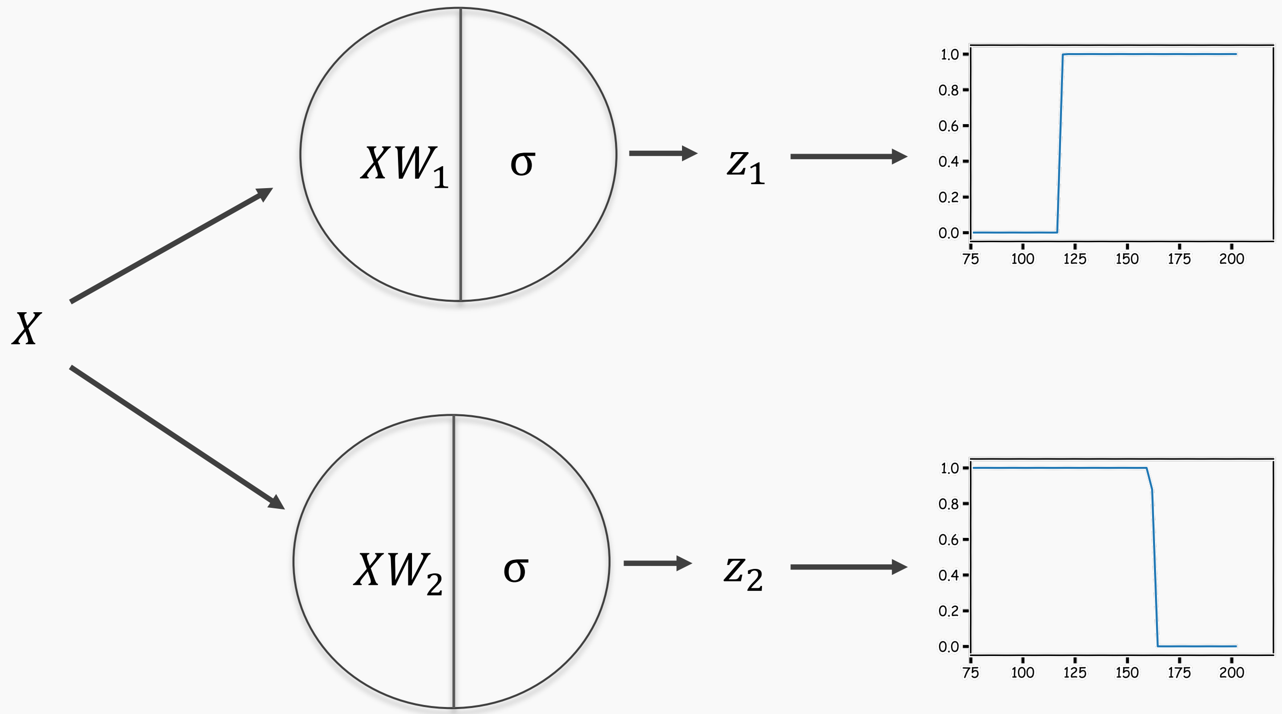




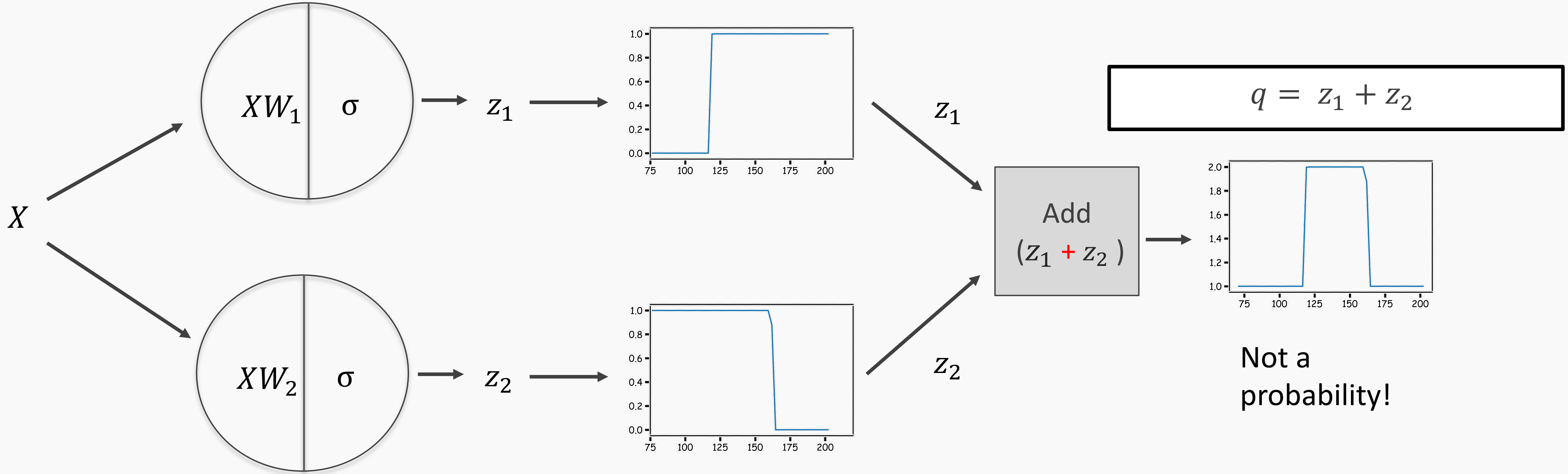




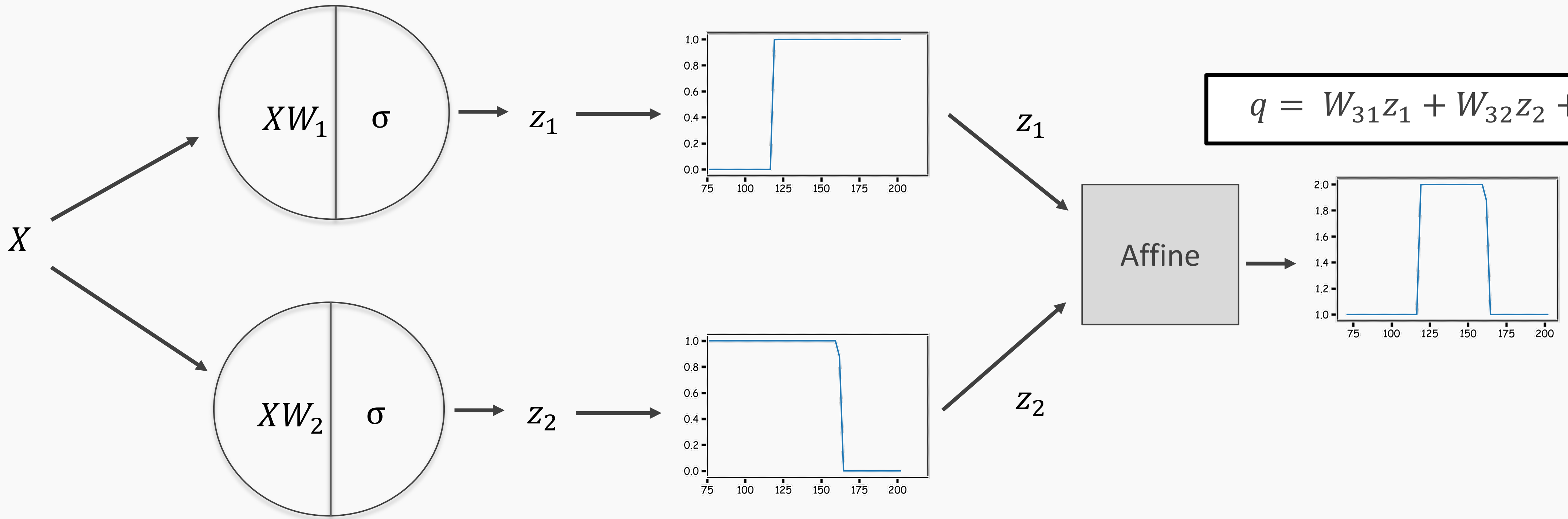
Combining Neurons



Combining Neurons

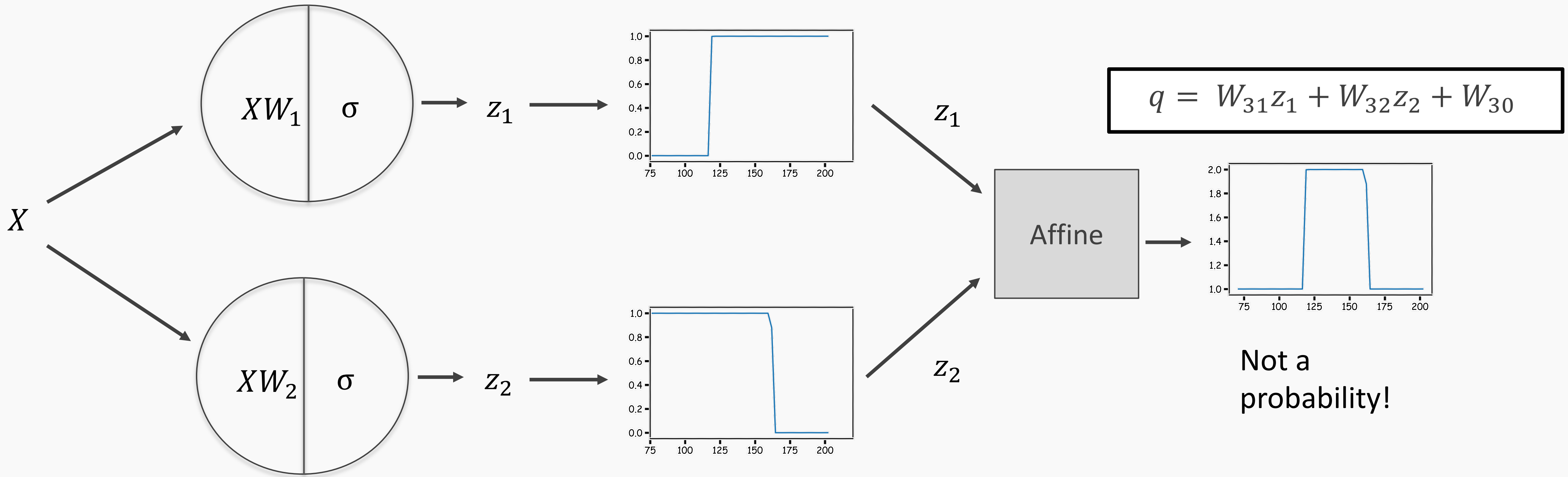


Combining Neurons



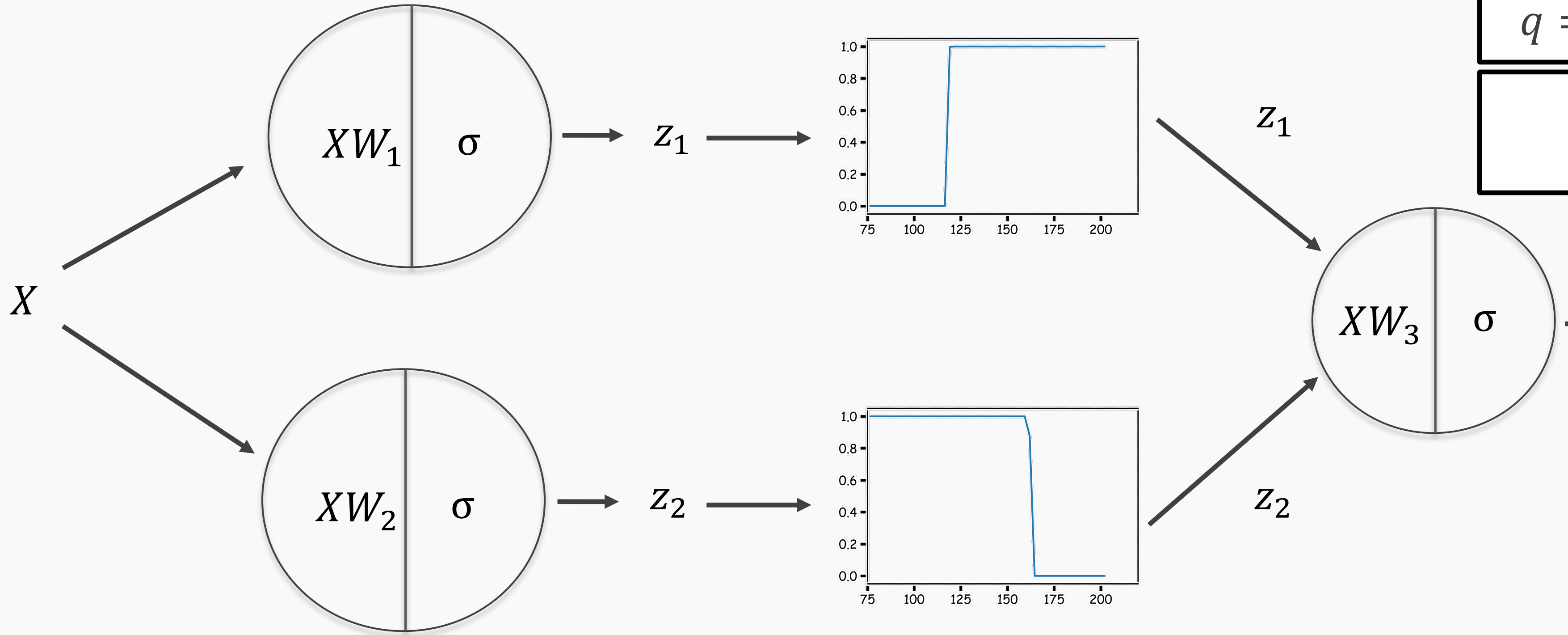
$$q = W_{31}z_1 + W_{32}z_2 + W_{30}$$

Combining Neurons



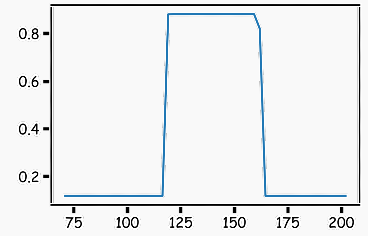
Not a probability!

Combining Neurons



$$q = W_{31}z_1 + W_{32}z_2 + W_{30}$$

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



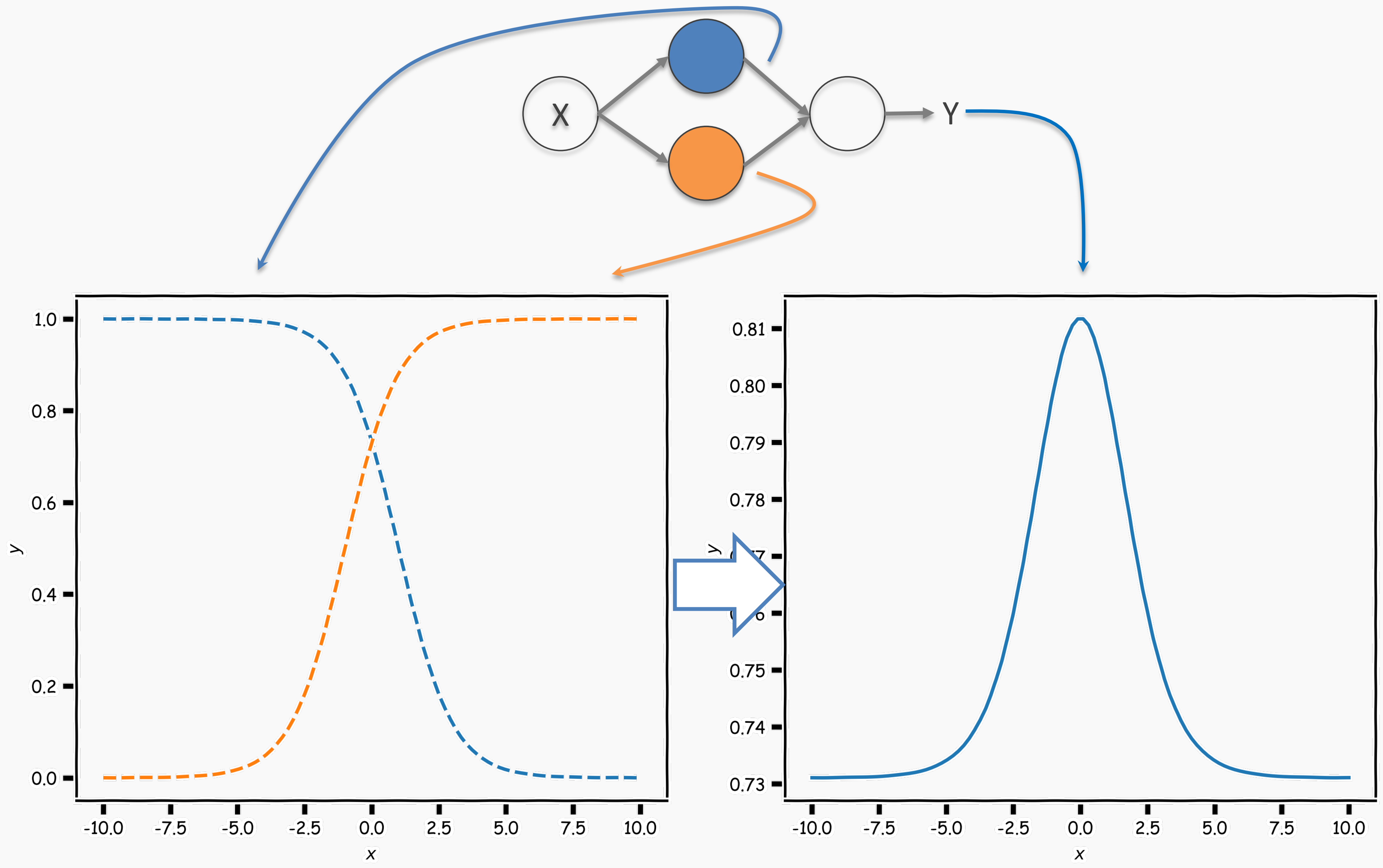
Passing through sigmoid yields probabilities

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

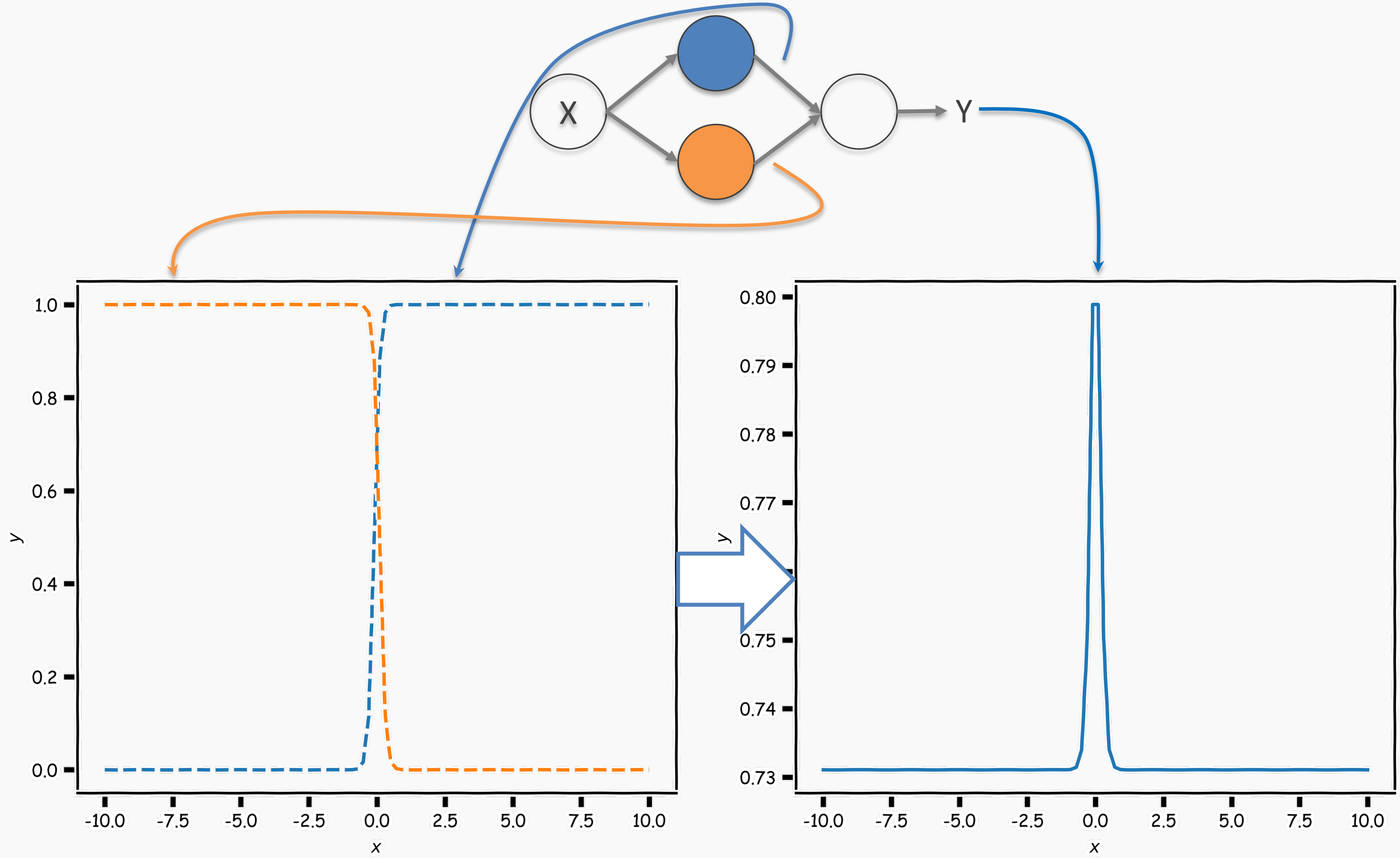
We will talk about about this in the coming lectures

Need to learn W_1, W_2 and W_3

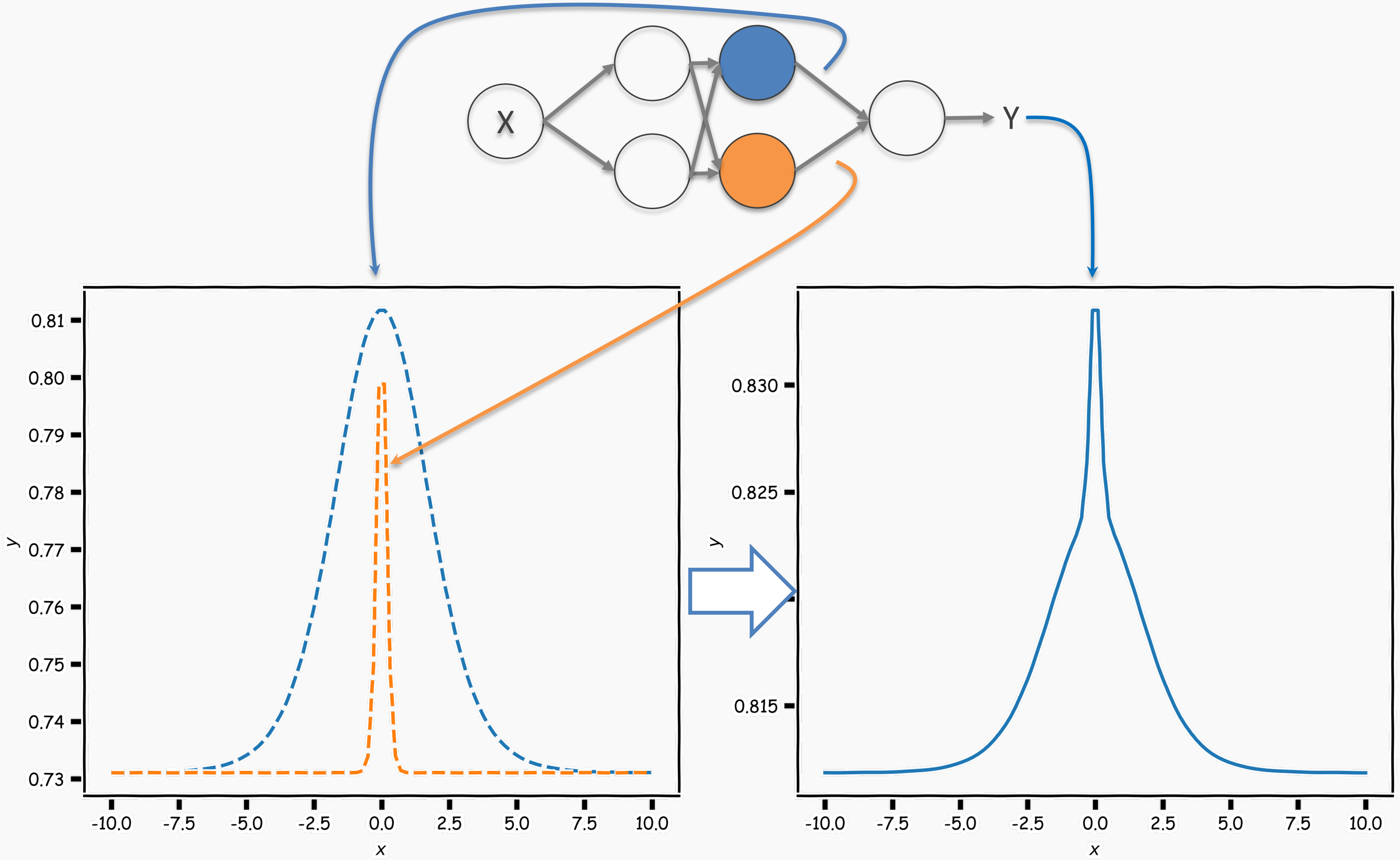
Combining neurons allows us to model interesting functions



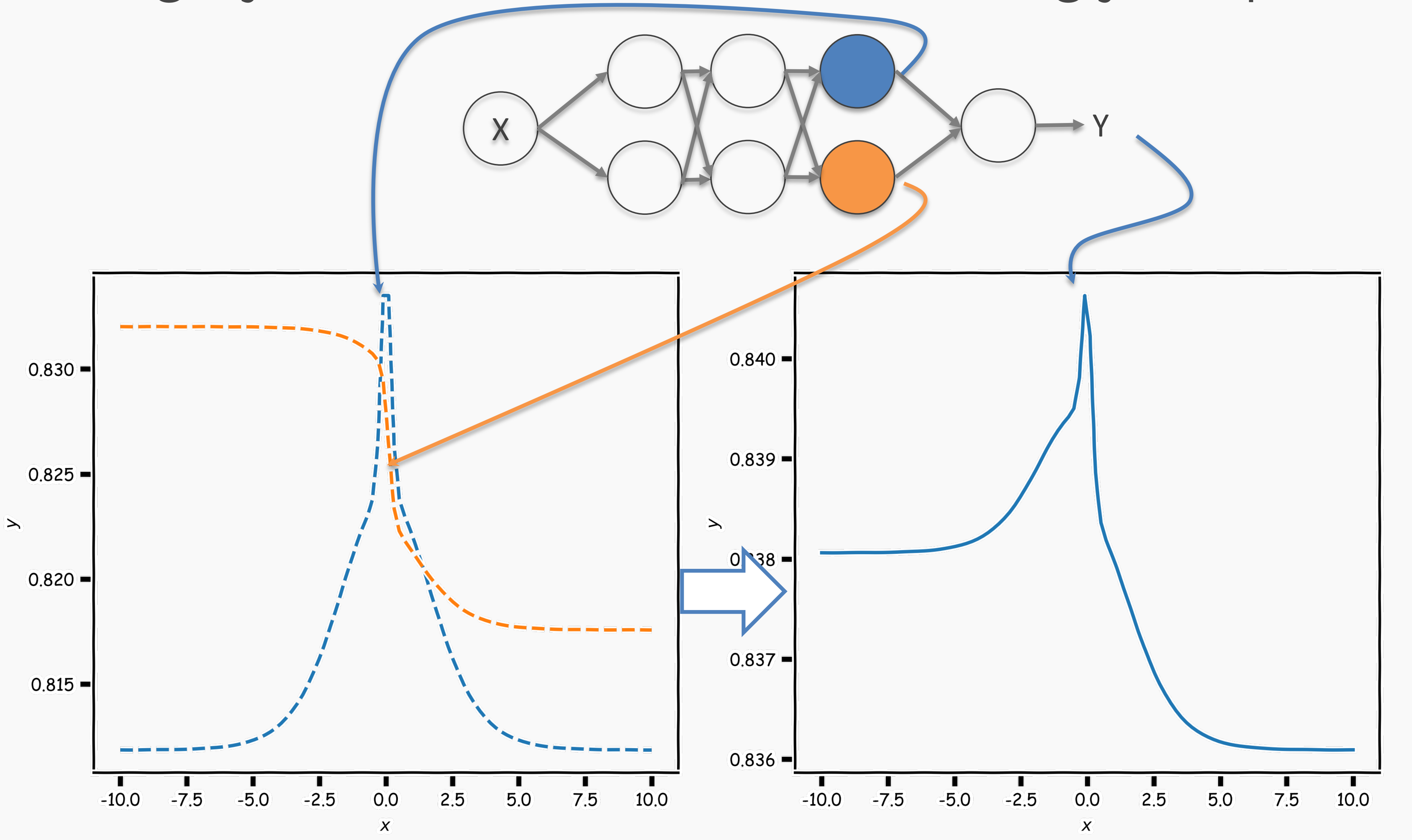
Different weights change the shape and position



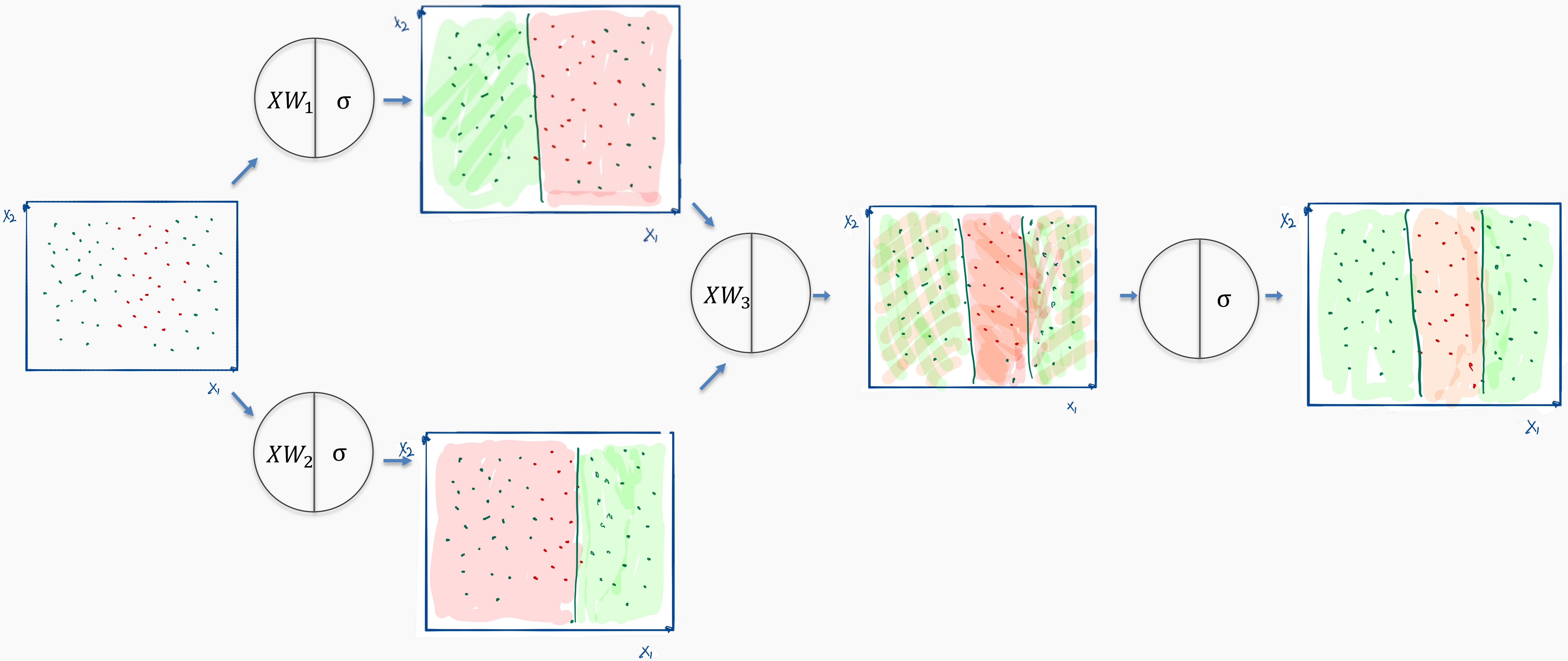
Neural networks can model *any* reasonable function



Adding layers allows us to model increasingly complex functions



For 2-D input the same idea applies.



Summary

So far:

- A single neuron can be a logistic regression or linear unit. We will soon see other choices of activation functions.
- A neural network is a combination of logistic regression (or other types) units.
- A neural network can approximate non-linear functions either for regression or classification.

Summary

So far:

- A single neuron can be a logistic regression or linear unit. We will soon see other choices of activation functions.
- A neural network is a combination of logistic regression (or other types) units.
- A neural network can approximate non-linear functions either for regression or classification.

Next:

- What kind of activations, how many neurons, how many layers, how to construct the output unit and what loss functions are appropriate?

Summary

So far:

- A single neuron can be a logistic regression or linear unit. We will soon see other choices.
- A neural network is a combination of logistic regression (or other types) units.
- A neural network can approximate non-linear functions either for regression or classification.

Next:

- What kind of activations, how many neurons, how many layers, how to construct the output unit and what loss functions are appropriate?

Following two lectures on NN:

- How do we estimate the weights and biases?
- How to regularize Neural Networks?