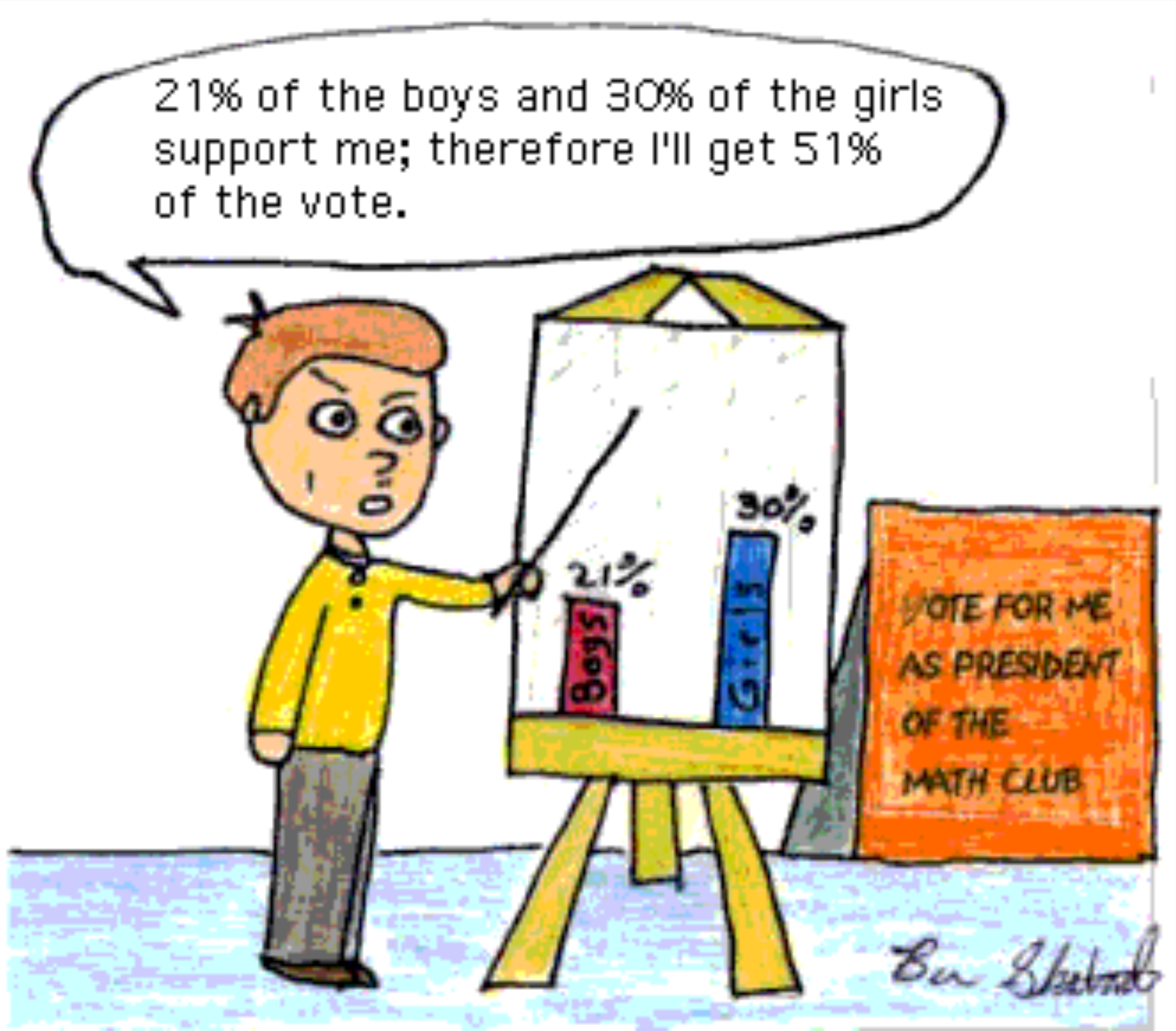
Perceptron and **Multilayer Perceptron**

CS109A Introduction to Data Science

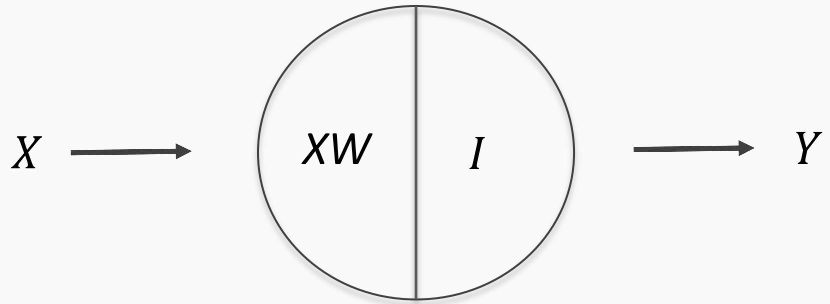
Pavlos Protopapas, Kevin Rader and Chris Tanner


21% of the boys and 30% of the girls support me; therefore I'll get 51% of the vote.
30%
21%
Boys
Girls
VOTE FOR ME AS PRESIDENT OF THE MATH CLUB

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

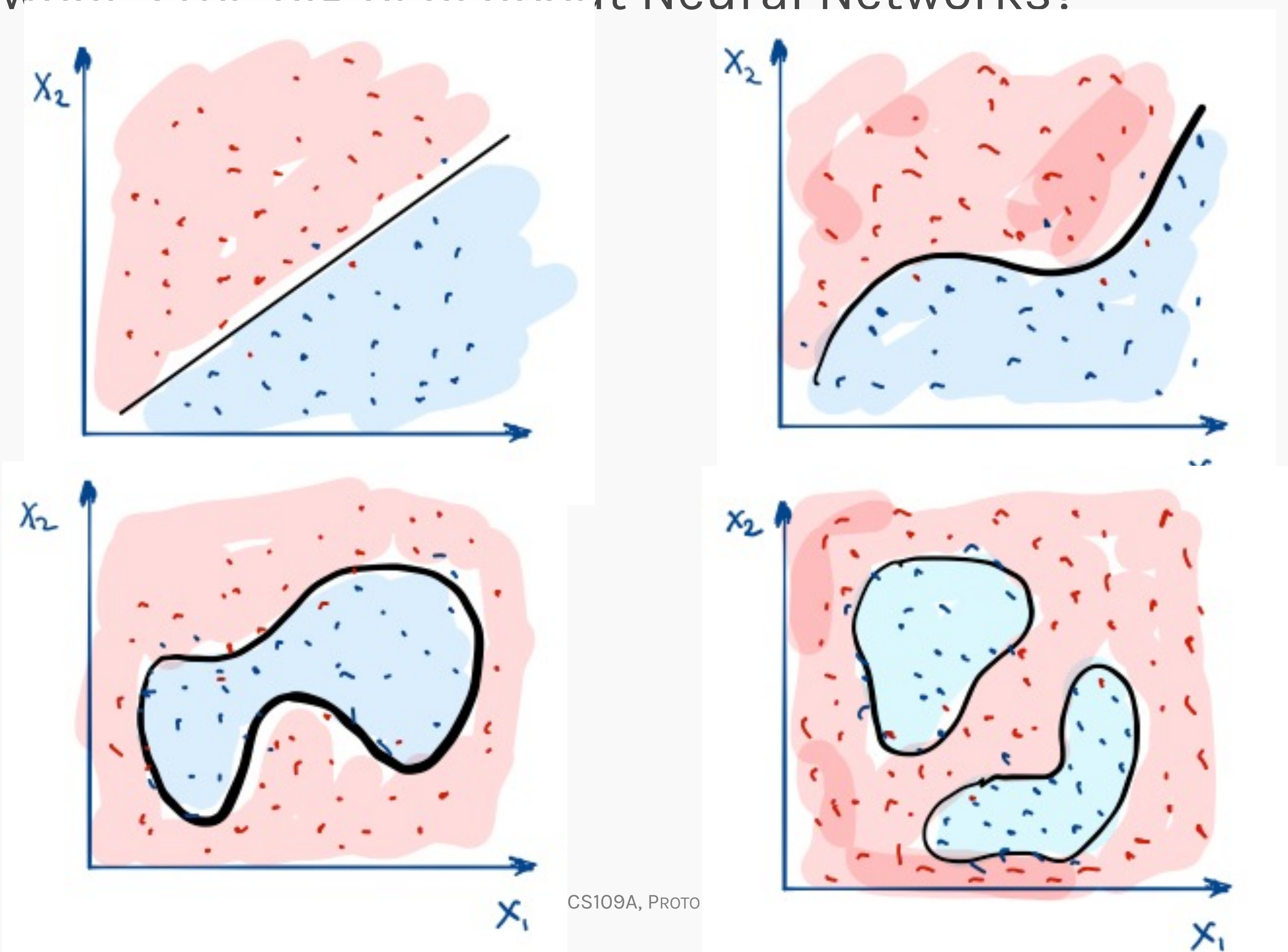
How about regression?

X → Affine → $h = \beta_0 + \beta_1 X$ → Activation Linear → Y= h → Loss Fun → $\mathcal{L}(\beta) = \sum_i^n \mathcal{L}_i(\beta)$

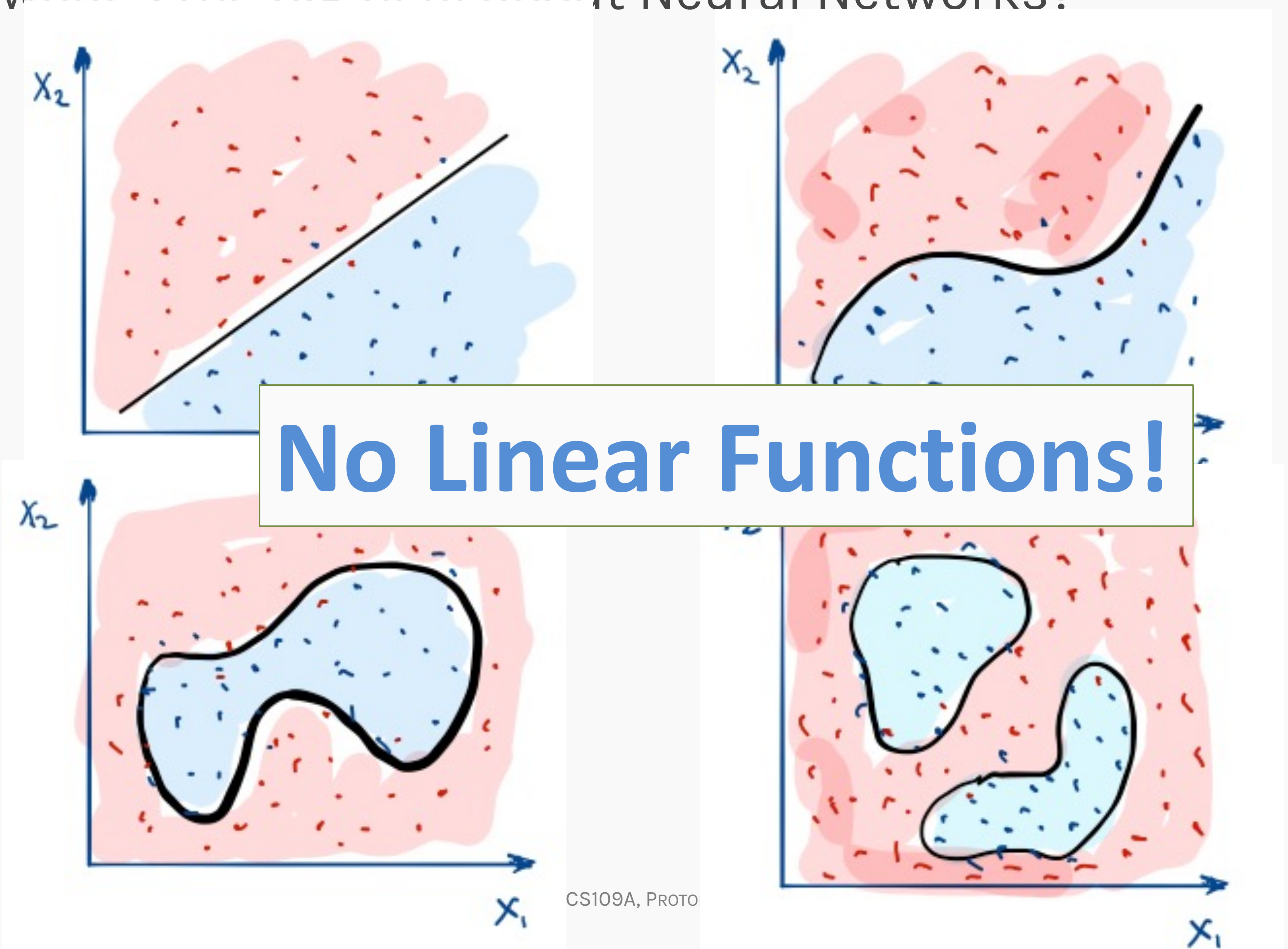


Where I is the identity function

So what's the big deal about Neural Networks?

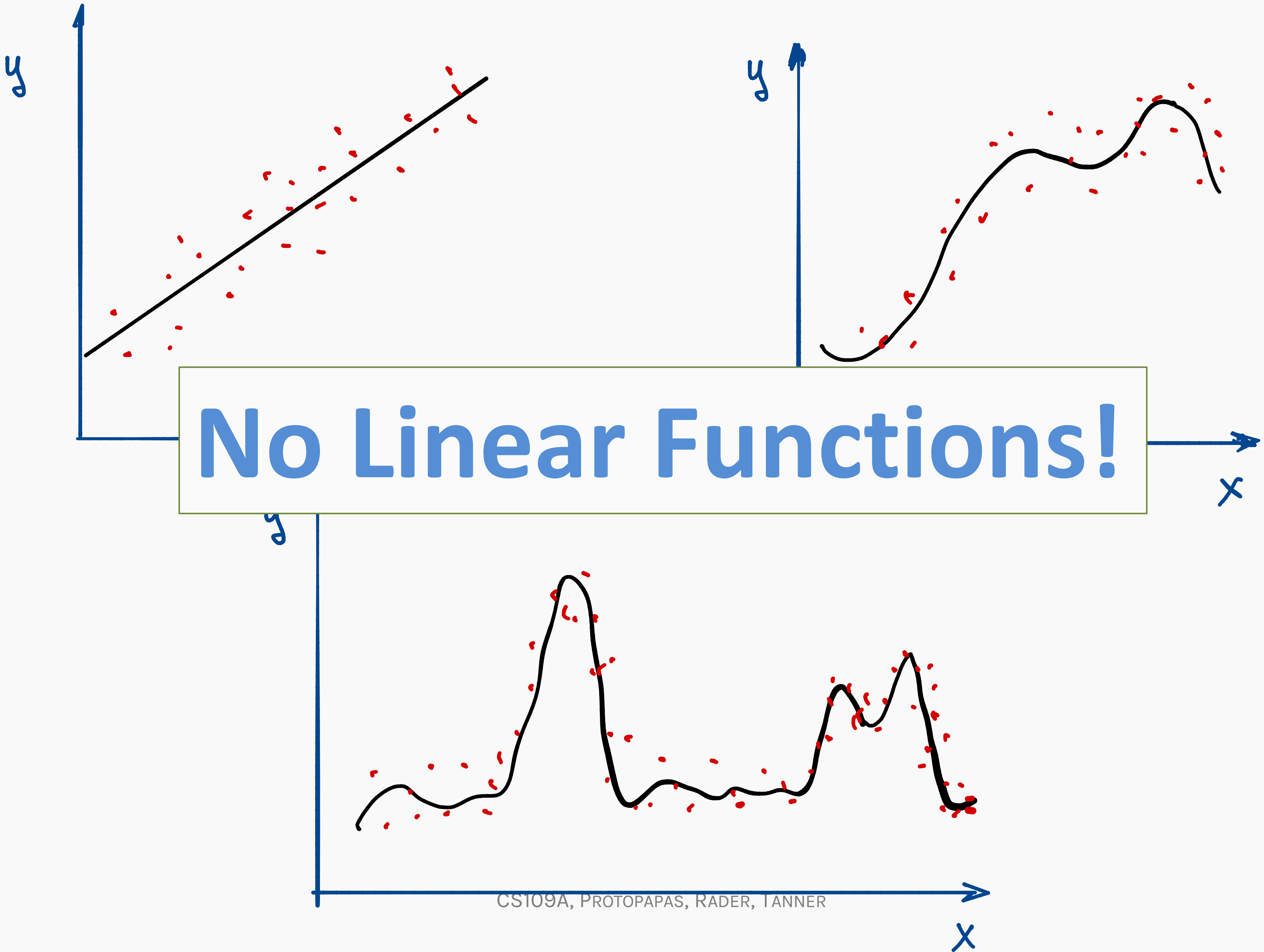


So what's the big deal about Neural Networks?



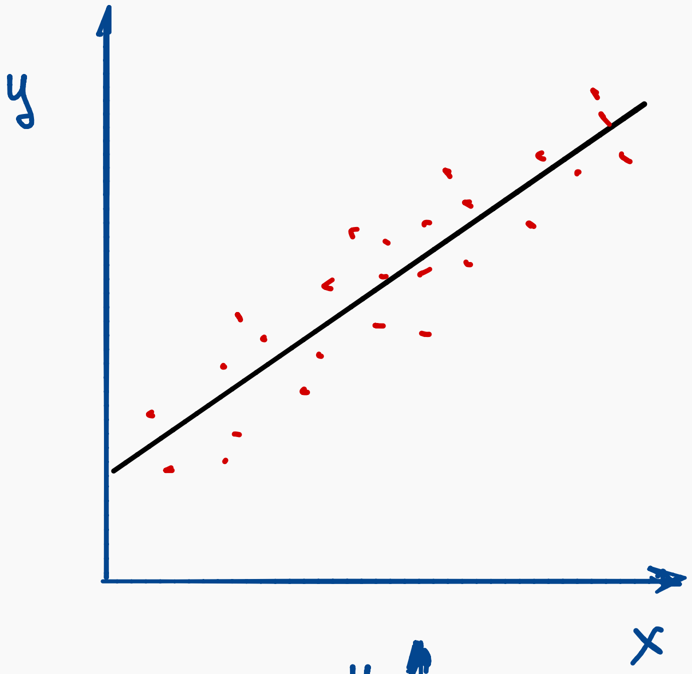
No Linear Functions!

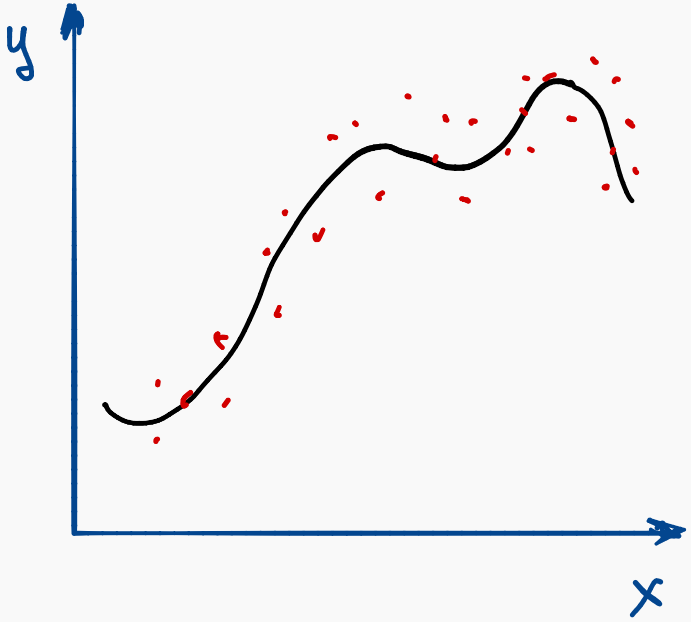
For regression?

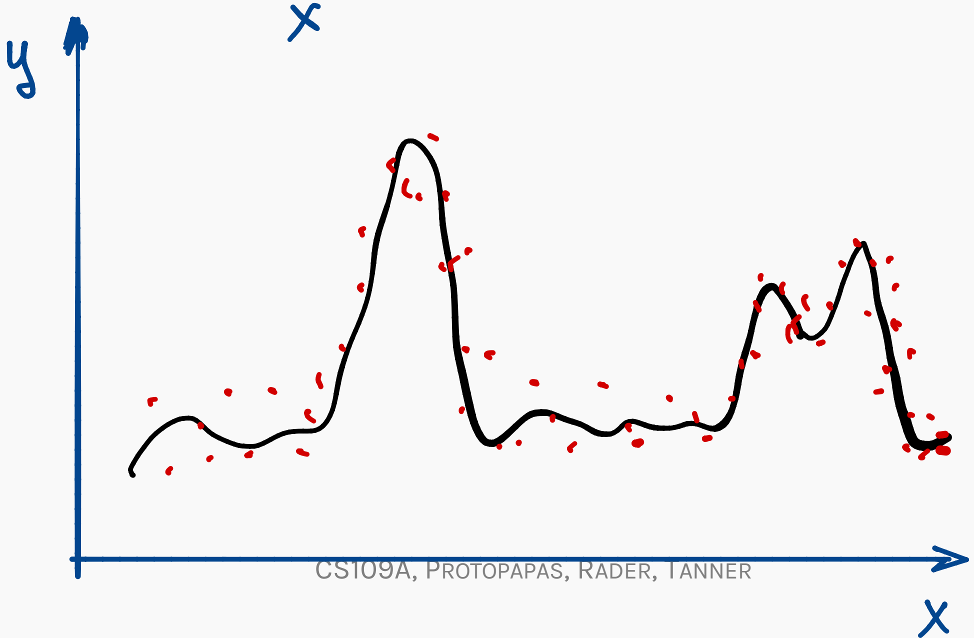


No Linear Functions!

For regression?





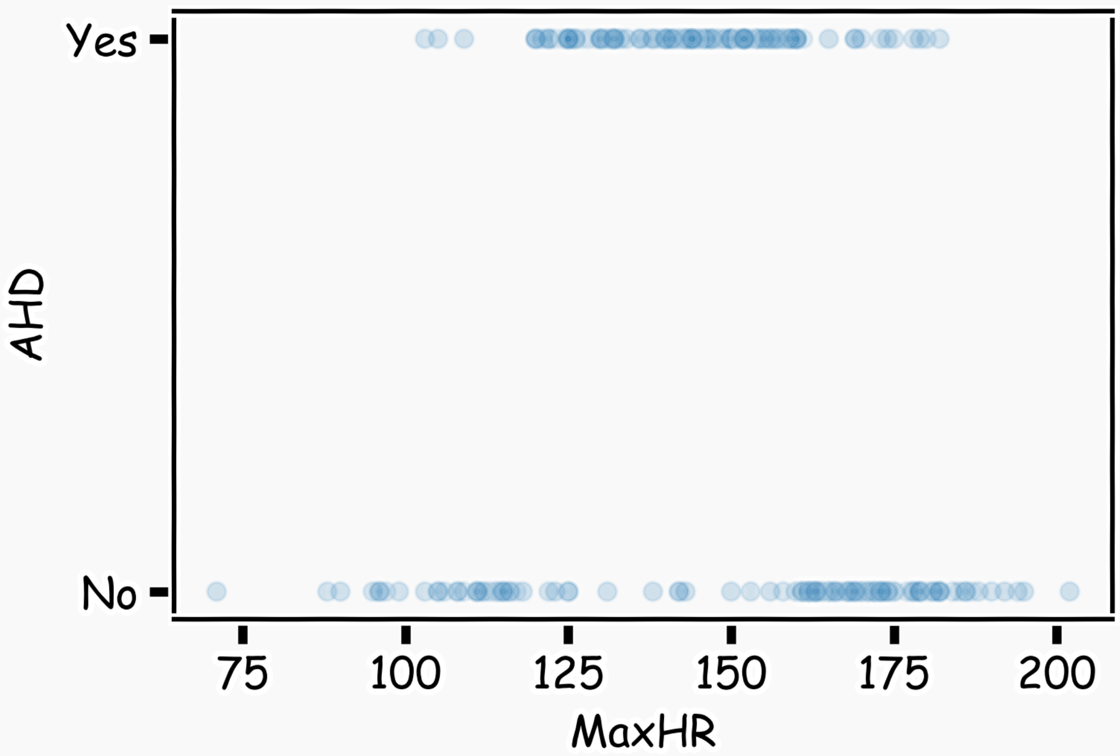


Outline

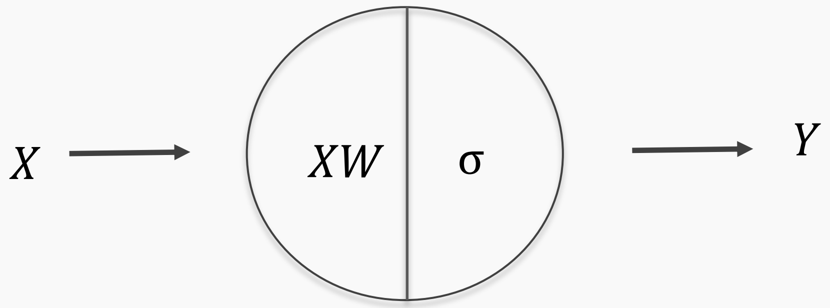
1. Introduction to Artificial Neural Networks
2. Review of Classification and Logistic Regression
3. Single Neuron Network ('Perceptron')
4. **Multi-Layer Perceptron (MLP)**

Example Using Heart Data

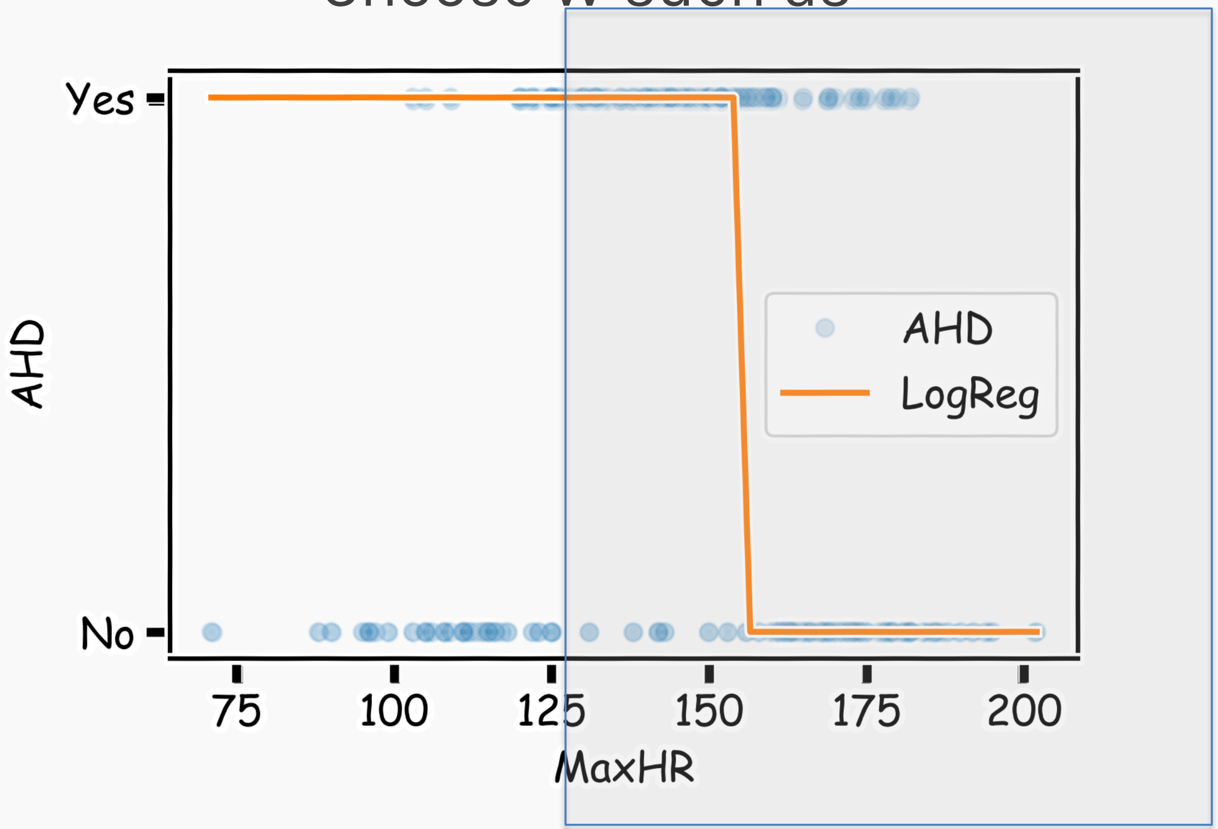
Slightly modified data to illustrate concepts.



Example Using Heart Data

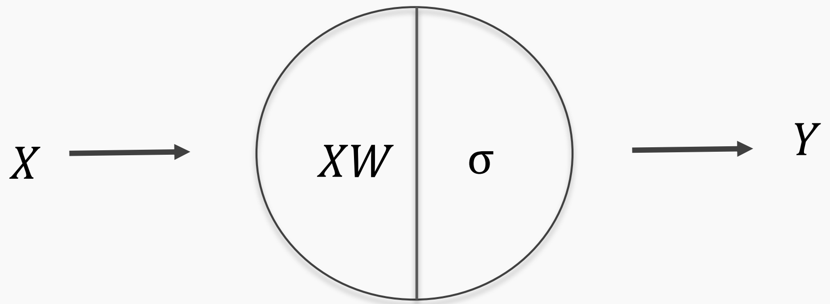


Choose W such as

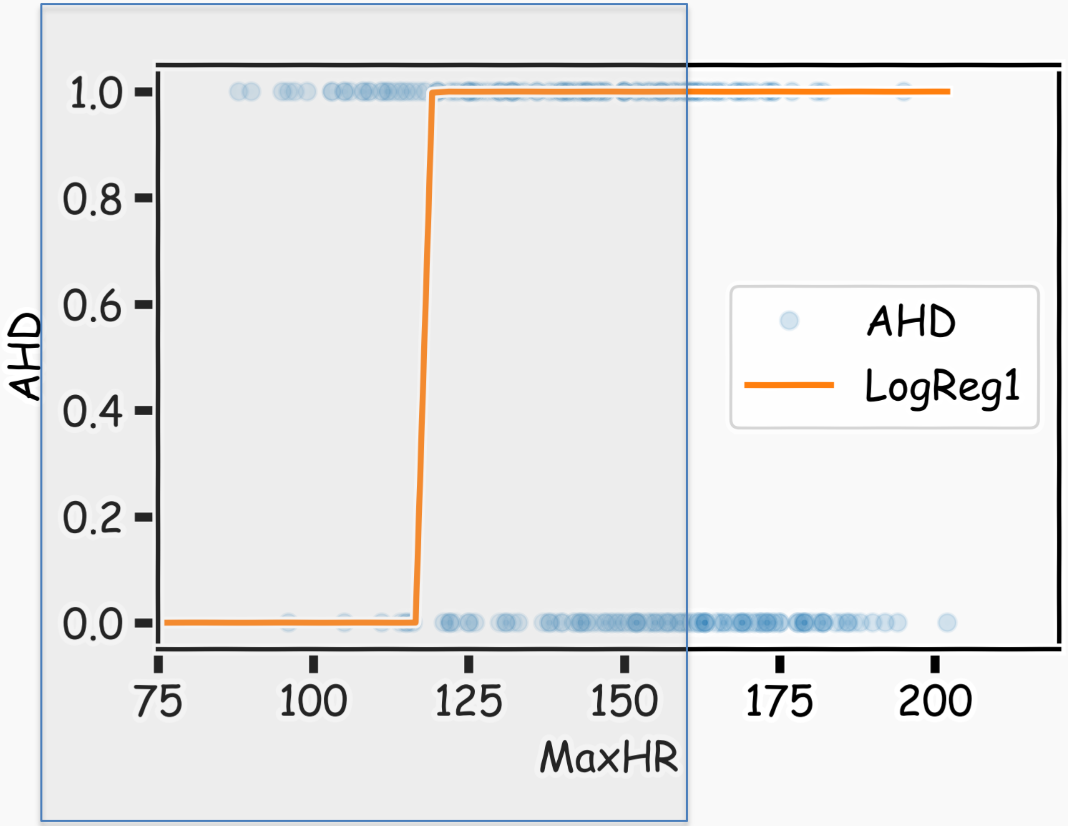


Right part of data are fitted well

Example Using Heart Data



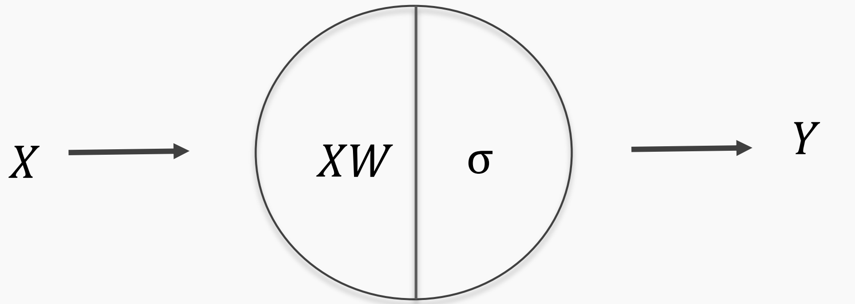
Choose W such as

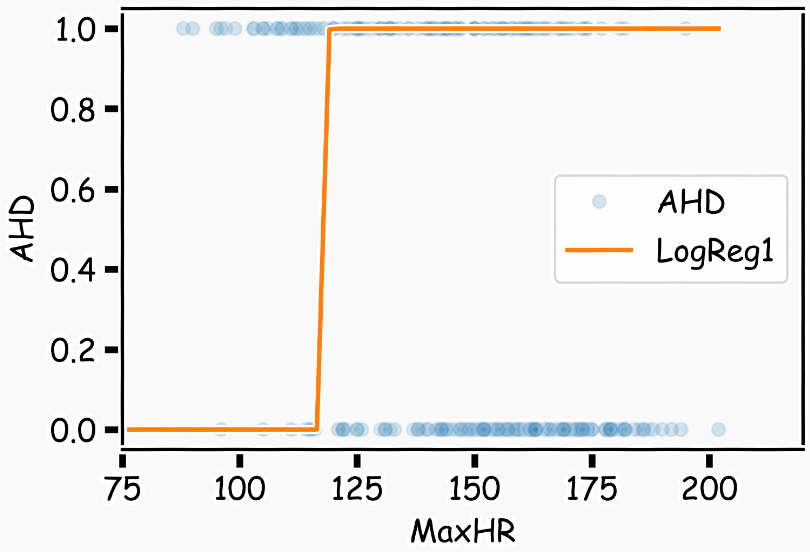


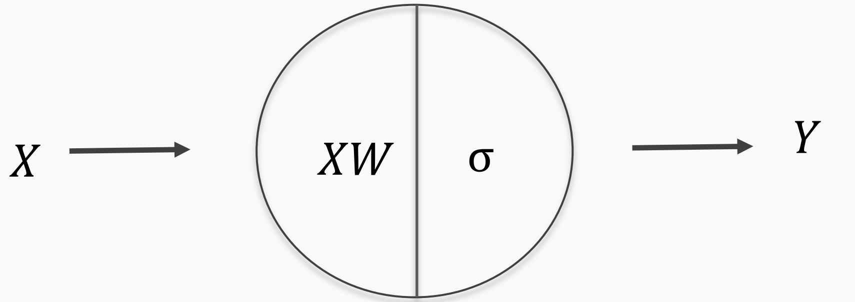
Left part of data are fitted well

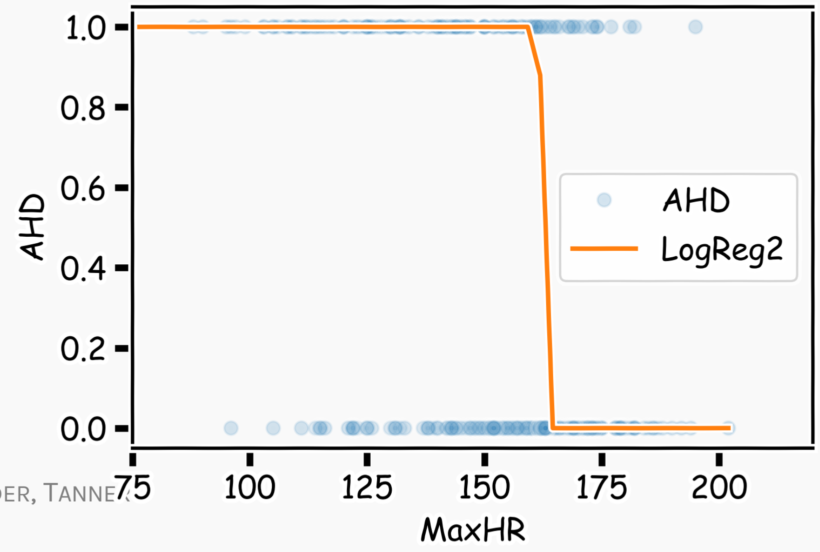
Example Using Heart Data

Two regions, two nodes

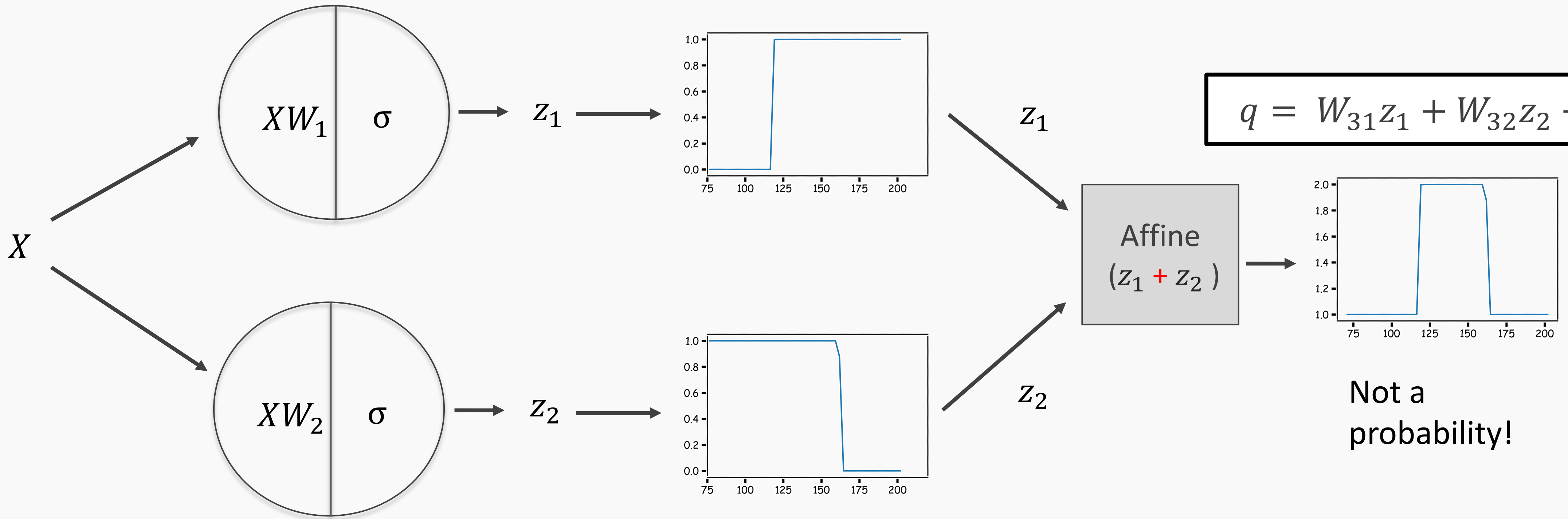








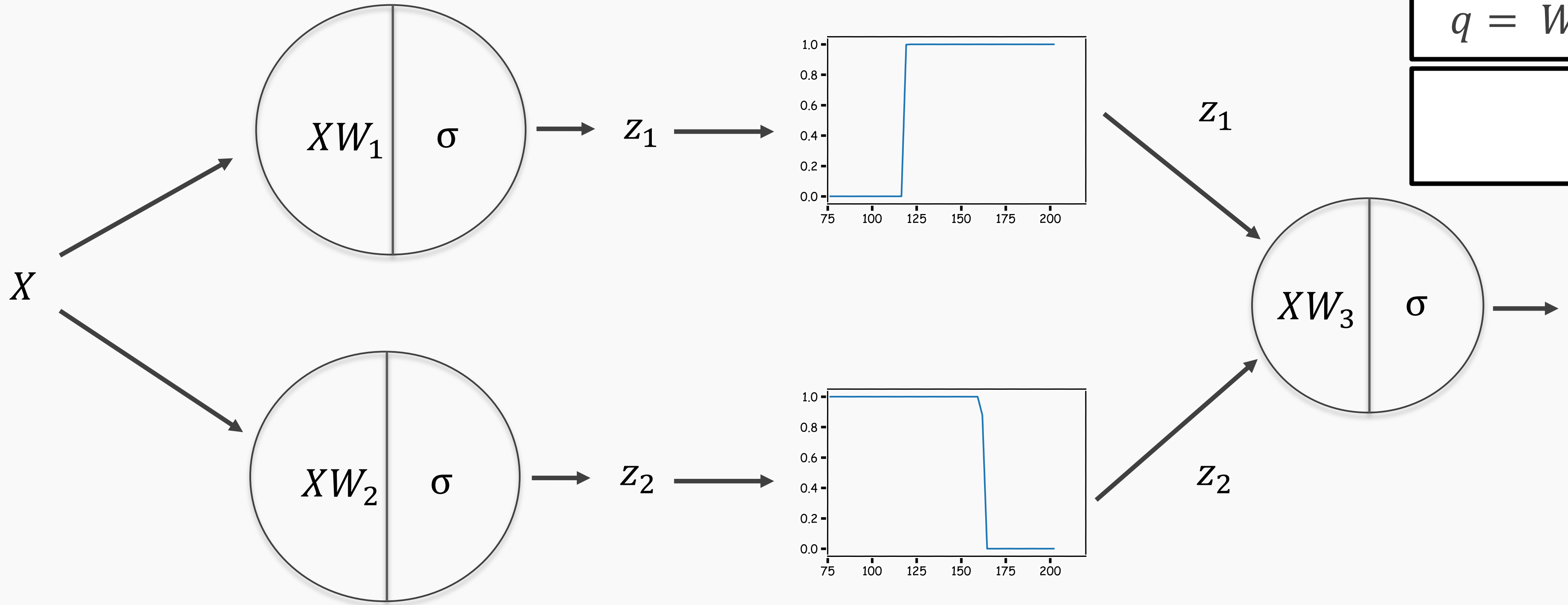
Combining Neurons



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

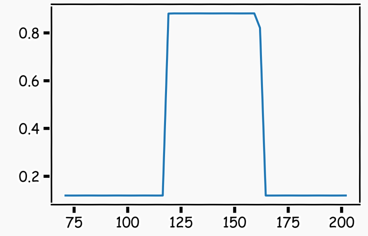
Not a probability!

Combining Neurons ...



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

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

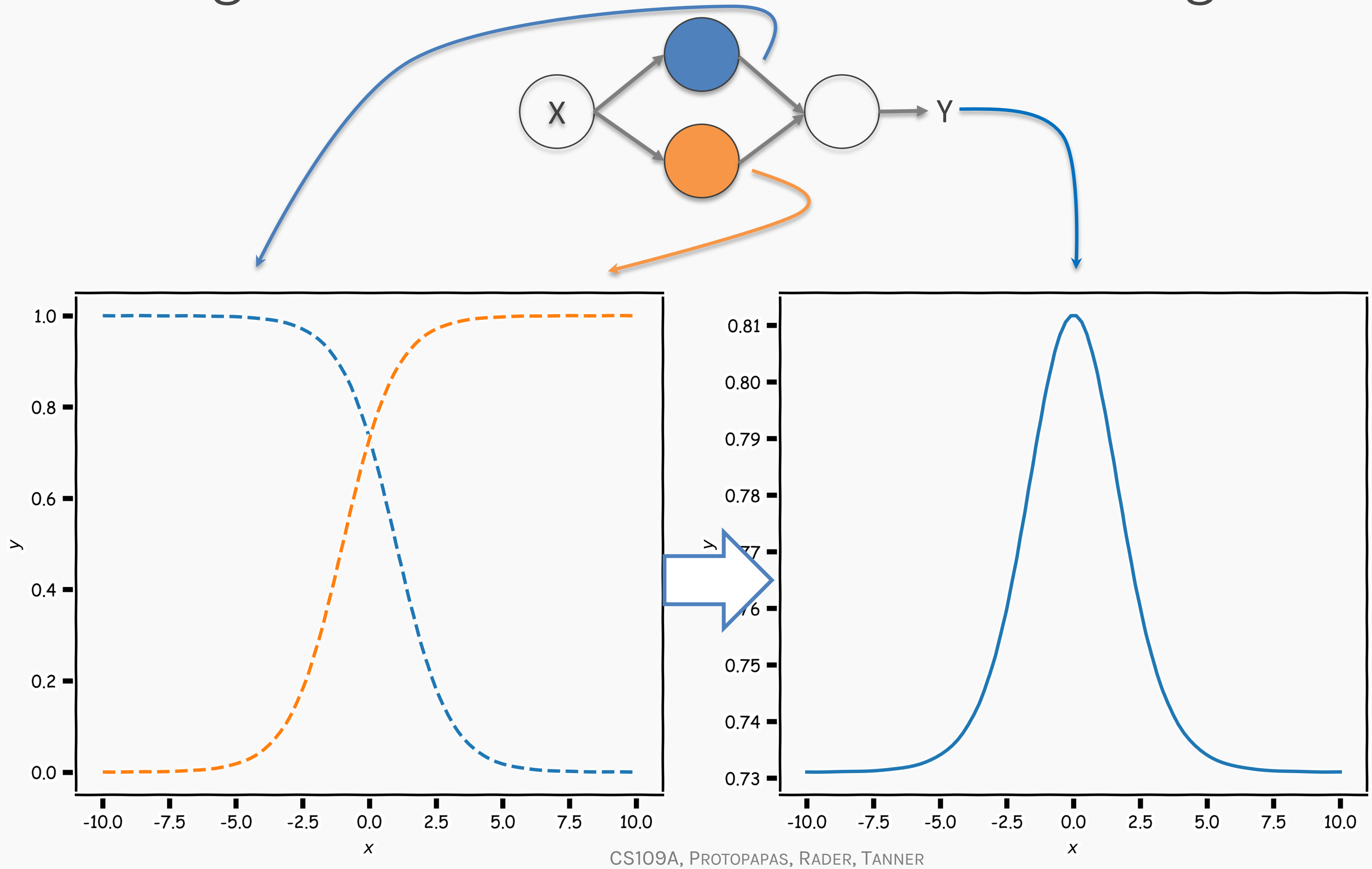


Passing through sigmoid yields probability

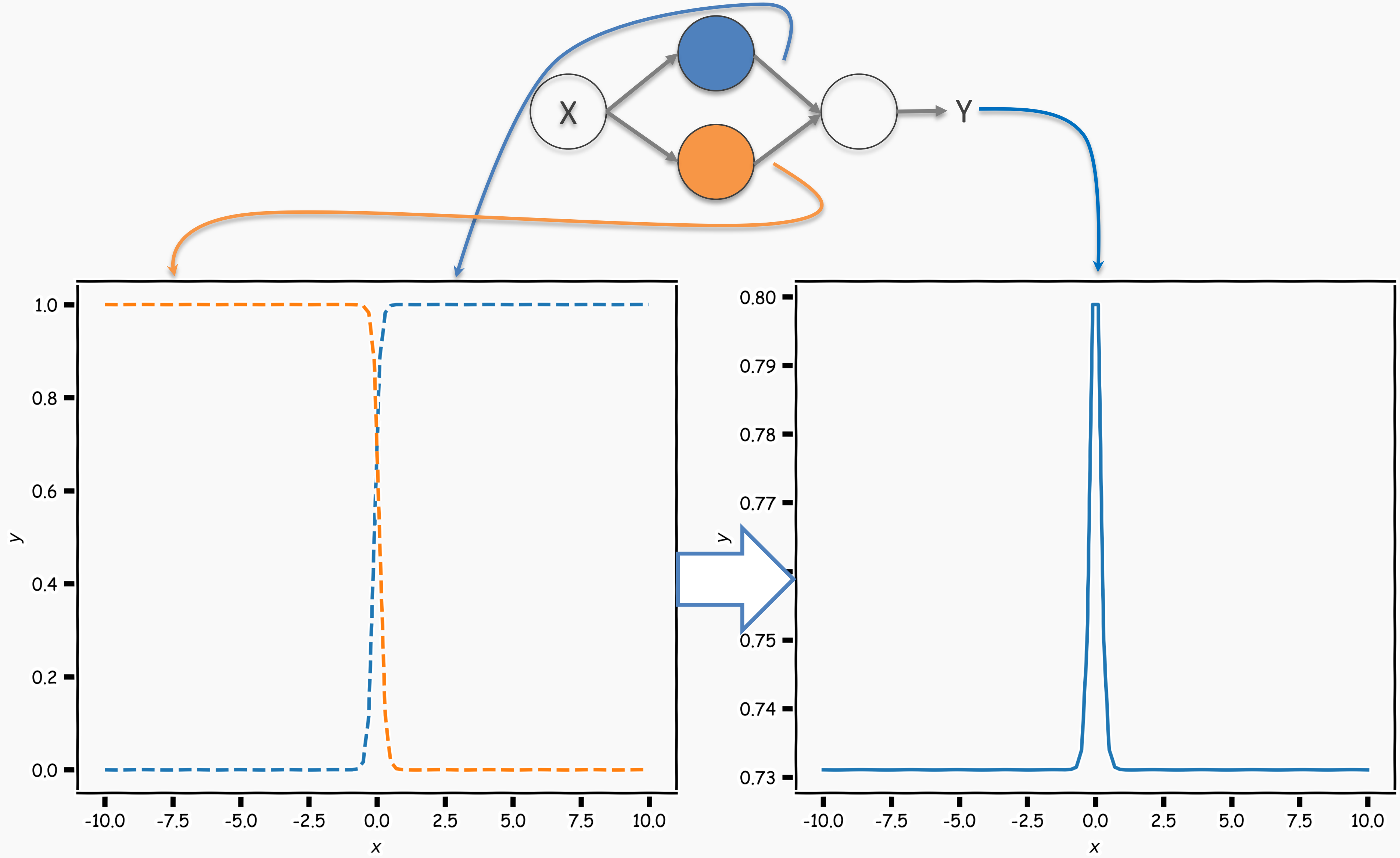
$$L = -y\ln(\mathrm{p}) - (1 - \mathrm{y})\ln(1 - \mathrm{p})$$

Need to learn W_1, W_2 and W_3

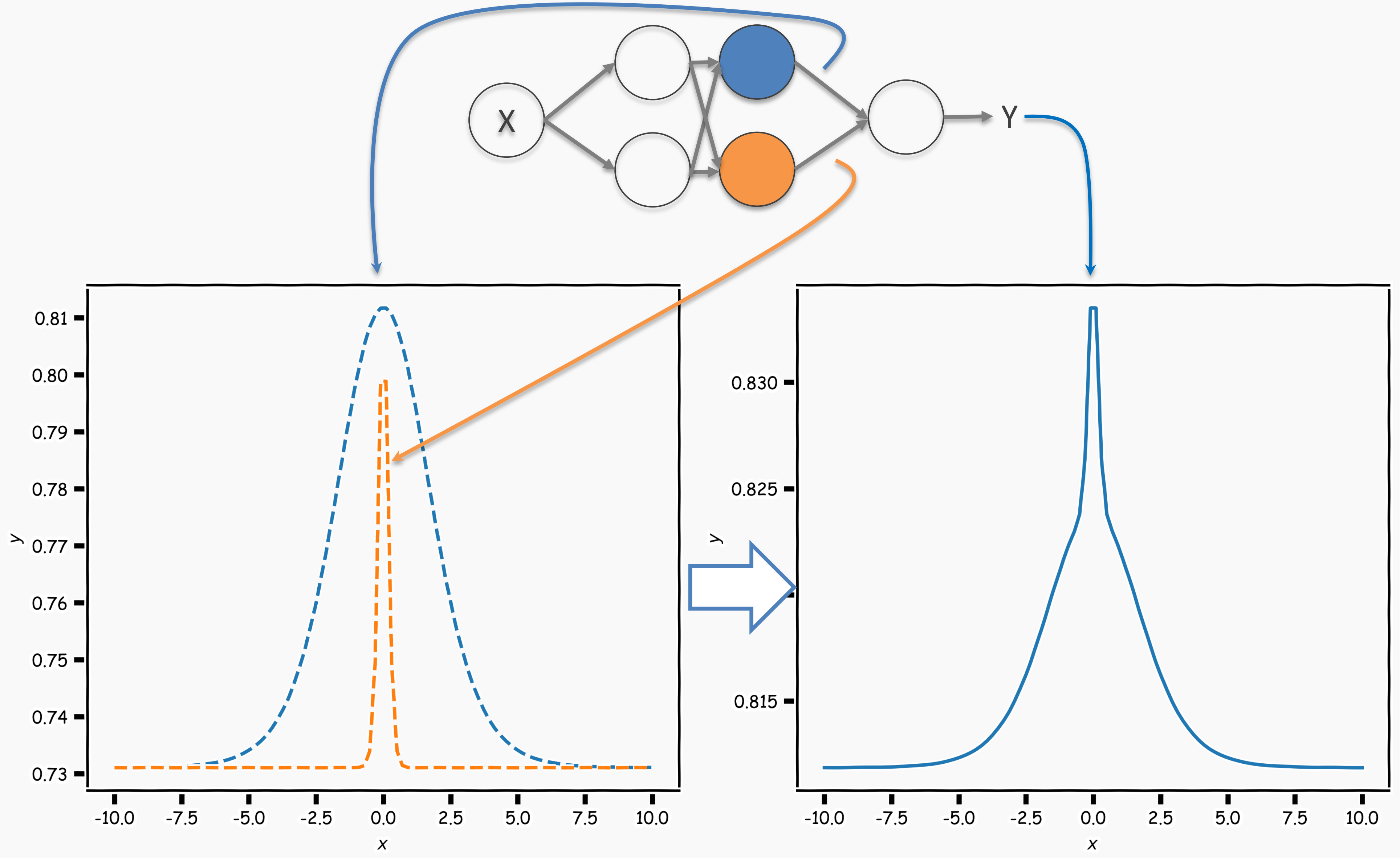
Combining neurons allows us to model interesting functions



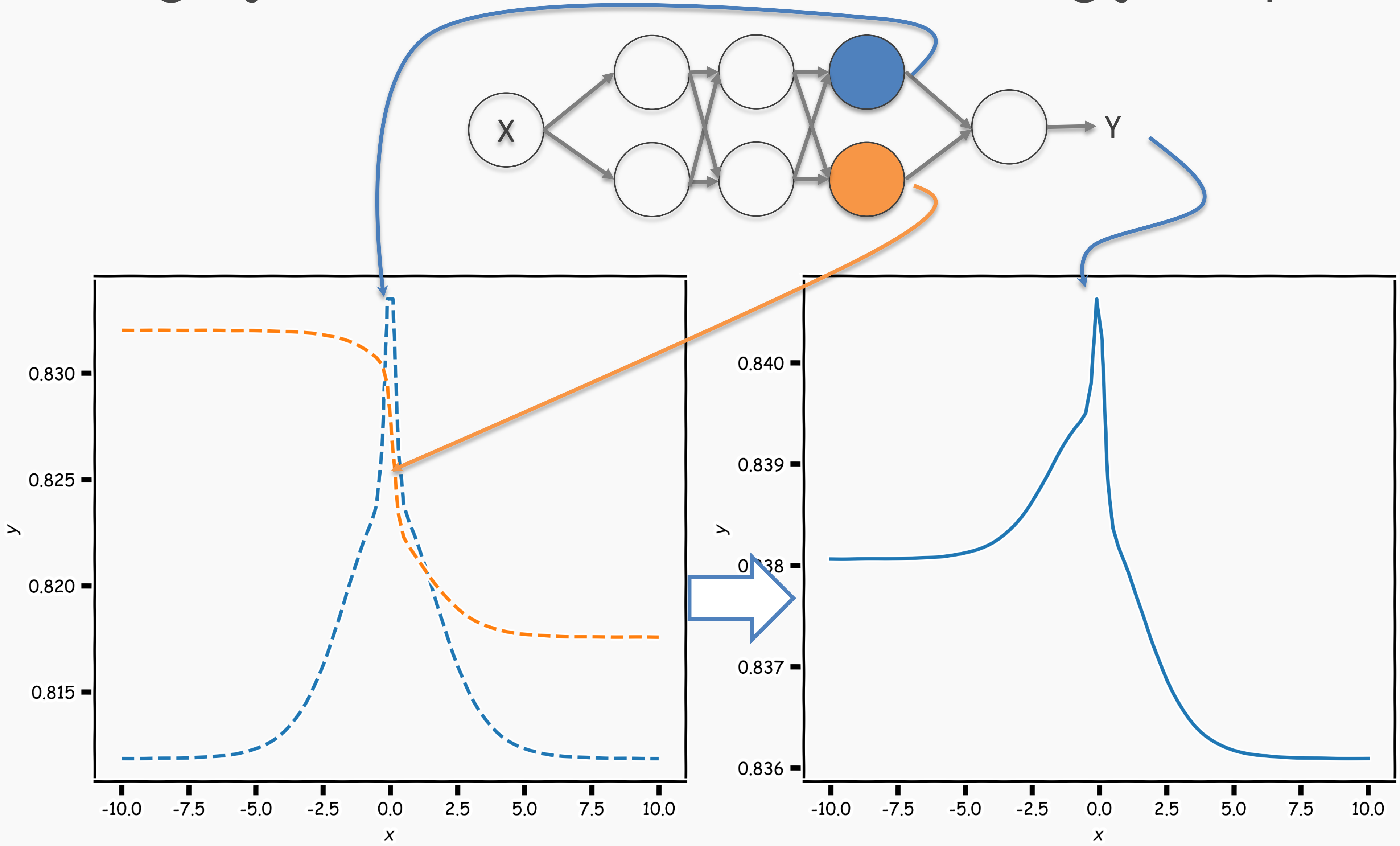
Different weights change the shape and position



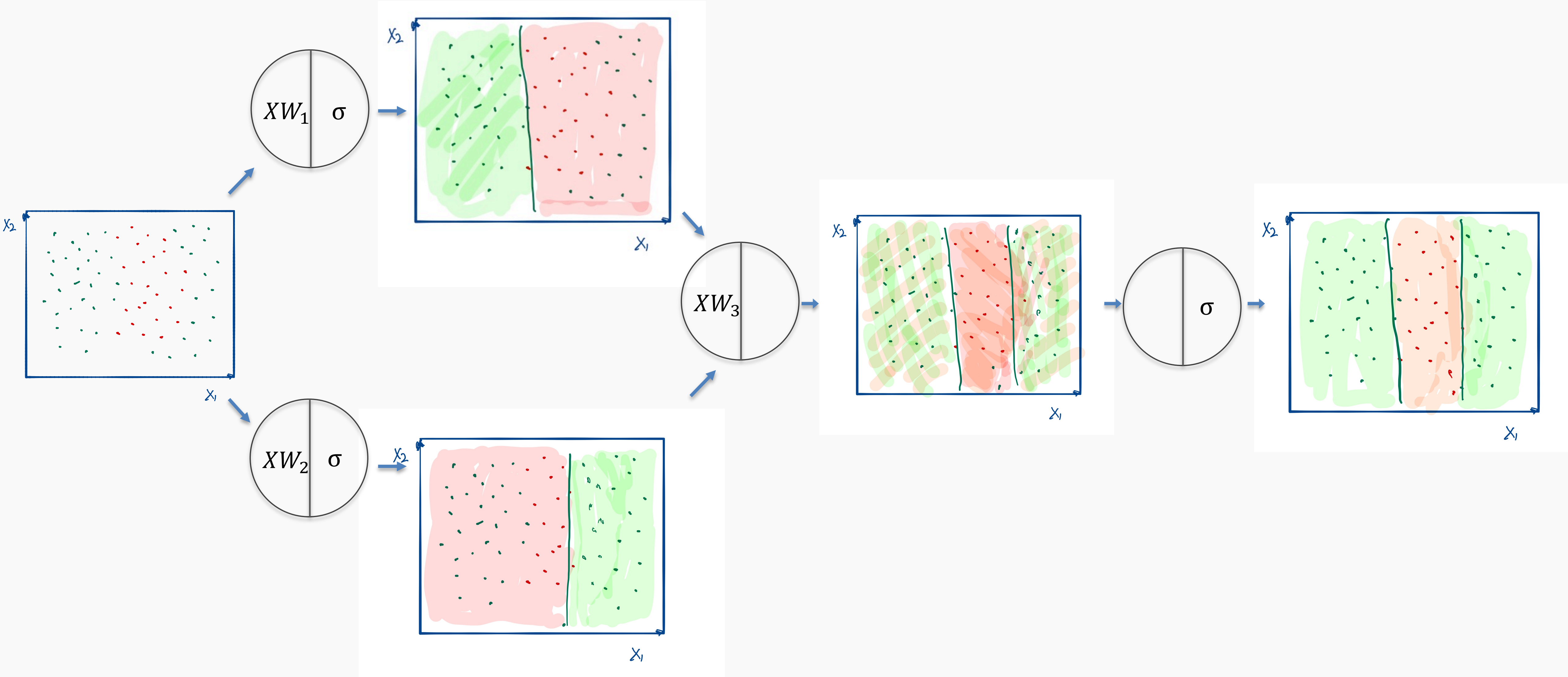
Neural networks can model *any* reasonable function



Adding layers allows us to model increasingly complex functions



For 2-D input the same idea applies.



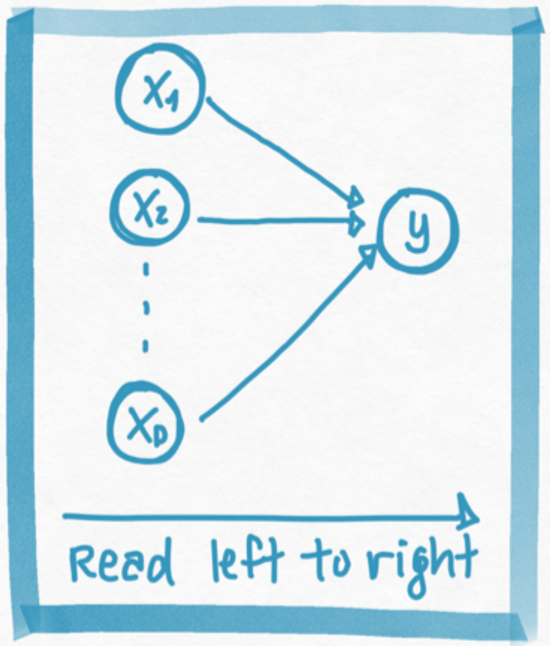
Summary

We build these complex functions by composing simple functions of the form:

$$h_w(x) = f(XW + b)$$

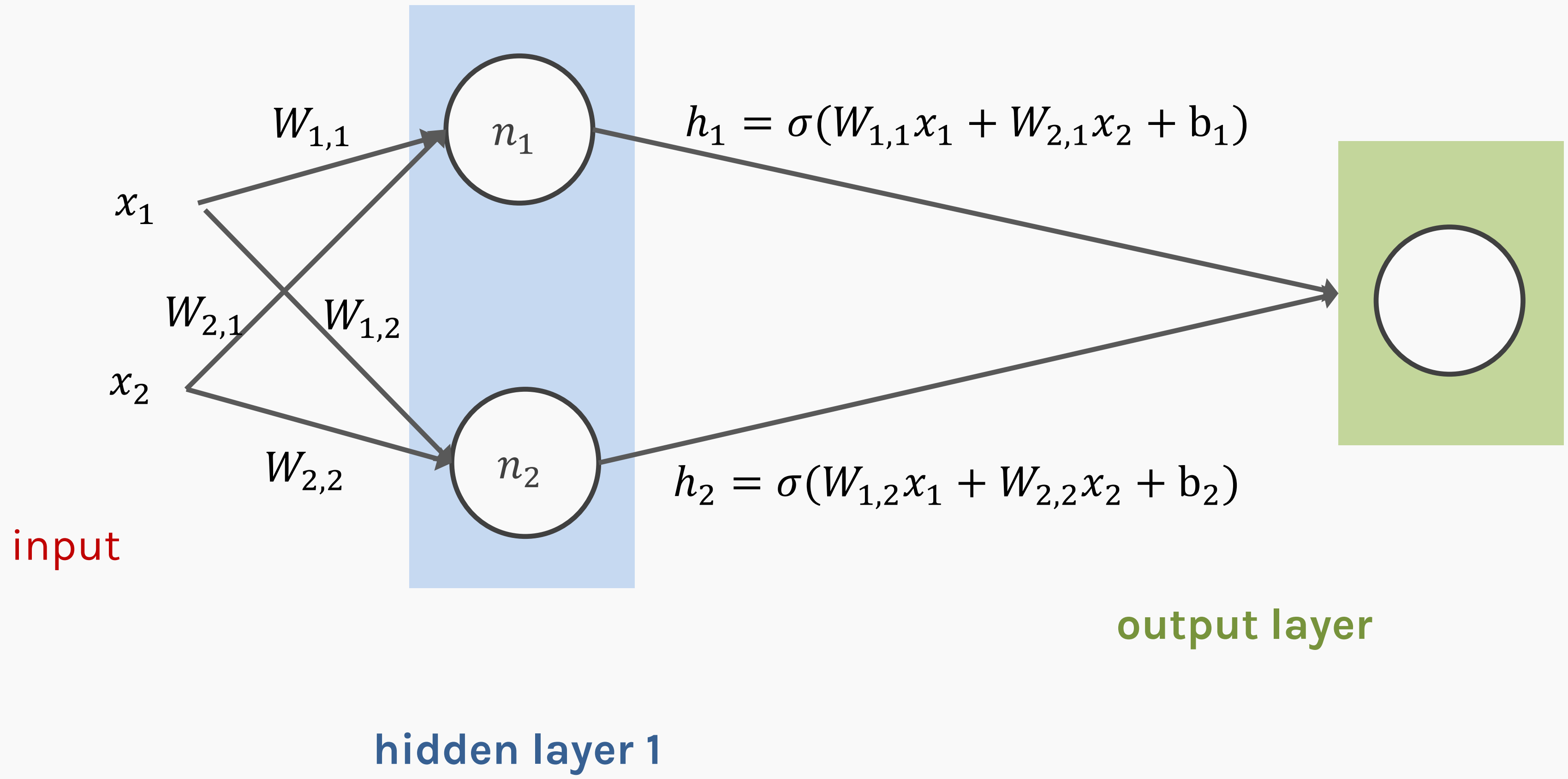
where f is the activation function.

We represent our simple function as a **graph**



Each edge in this graph represents multiplication by a different weight, w_i.

Flow in NN



$W_{1,1}$
n_1
$h_1 = \sigma(W_{1,1}x_1 + W_{2,1}x_2 + b_1)$
x_1
$W_{2,1}$
$W_{1,2}$
x_2
$W_{2,2}$
n_2
$h_2 = \sigma(W_{1,2}x_1 + W_{2,2}x_2 + b_2)$
input
output layer
hidden layer 1

Summary

So far:

- A single neuron can be a logistic regression or linear unit. We will soon see other choices of activation function.
- 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

Next:

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

Following lectures on NN:

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

Next

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?