Topics in Social Data Science

Week 2

Artificial Neural Networks 1

Multilayer Perceptron, Gradient Descent

Overview of today + tomorrow

- Watch 3BLUE1BROWN's chapter 1+2 on Neural Networks
- Read (or at least familiarize yourself with) Michael Nielsen's book up to and including Chapter 1
- My lecture (MLP and Gradient Descent)
- Exercises in Python

Multilayer Perceptron

The simplest Neural Network I can think of…

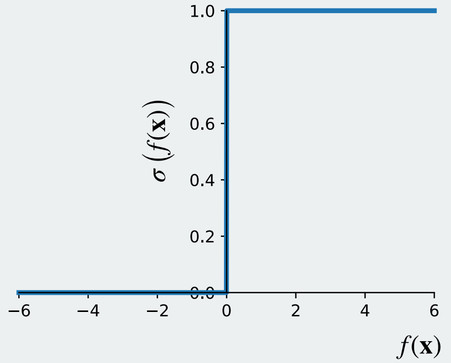
Linear regression

$$w_0 + x_0 w_1 + x_1 w_2 + x_2 w_3 = f(\mathbf{x})$$

Linear regression classifier

$$w_0 + x_0 w_1 + x_1 w_2 + x_2 w_3 = f(\mathbf{x})$$

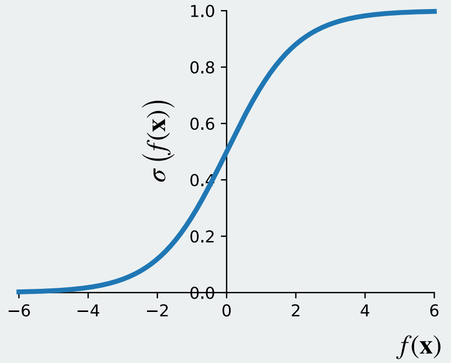
$$\sigma\left(f(\mathbf{x})\right) = \begin{cases} 1, & \textbf{if } f(\mathbf{x}) \geq 0 \\ 0, & \textbf{otherwise} \end{cases}$$



Linear regression classifier

$$w_0 + x_0 w_1 + x_1 w_2 + x_2 w_3 = f(\mathbf{x})$$

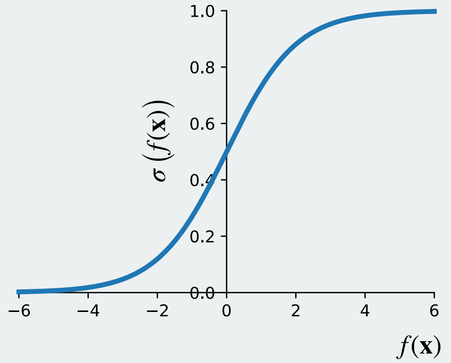
$$\sigma\left(f(\mathbf{x})\right) = \frac{1}{1 + exp\left(-f(x)\right)}$$



¡Logistic regression!
classifier

$$w_0 + x_0 w_1 + x_1 w_2 + x_2 w_3 = f(\mathbf{x})$$

$$\sigma\left(f(\mathbf{x})\right) = \frac{1}{1 + exp\left(-f(x)\right)}$$



Logistic regression classifier

(Input)

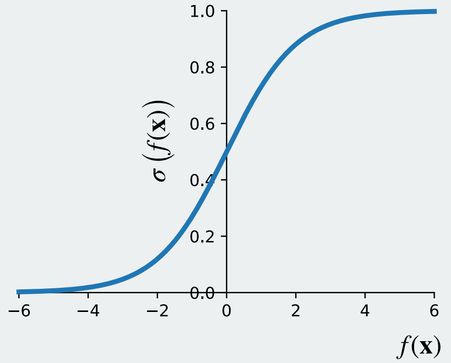
Variables **Weights**

$$w_0 + x_0 w_1 + x_1 w_2 + x_2 w_3 = f(\mathbf{x})$$

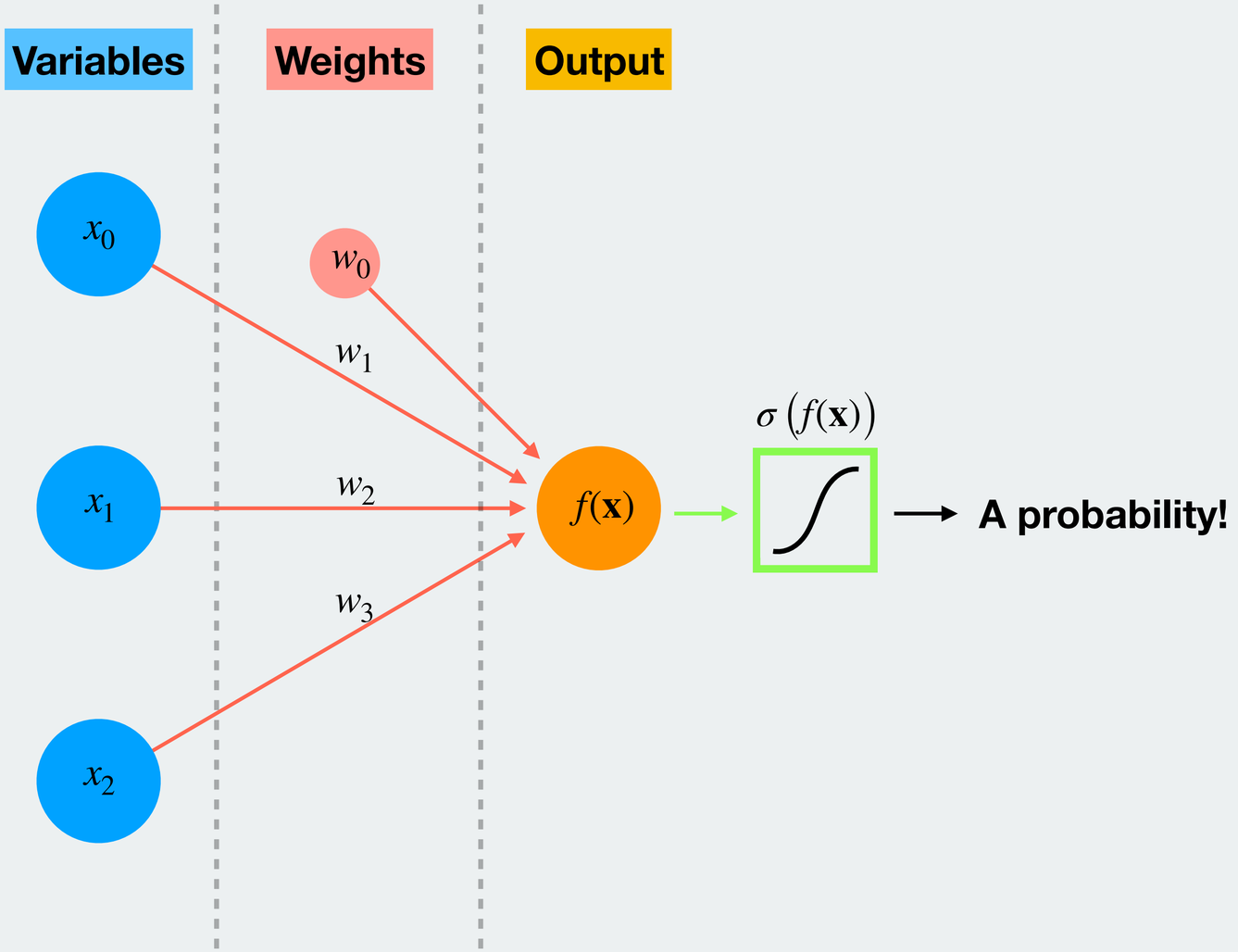
Output

$$\sigma\left(f(\mathbf{x})\right) = \frac{1}{1 + exp\left(-f(x)\right)}$$

Activation function

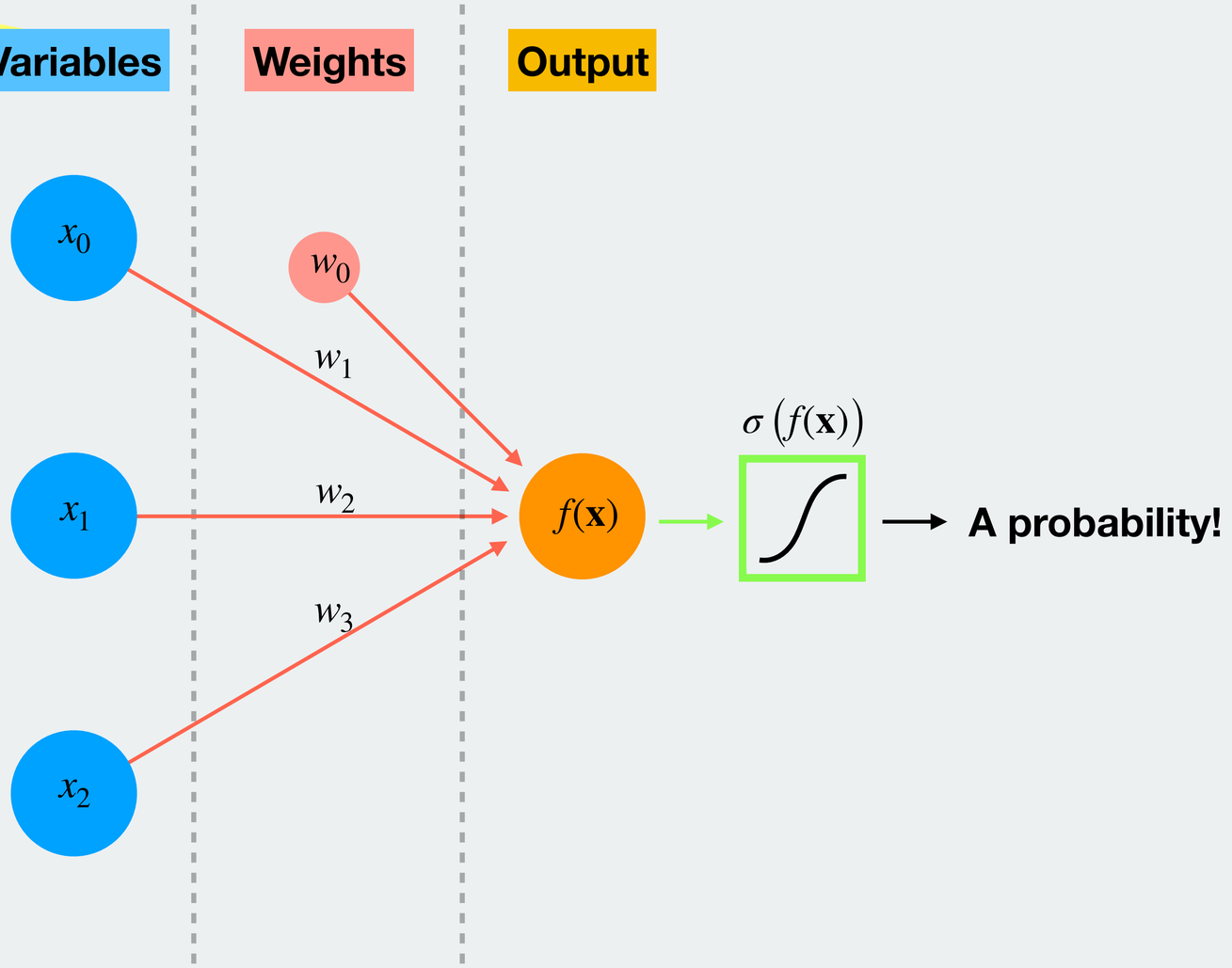


Logistic regression schematic illustration



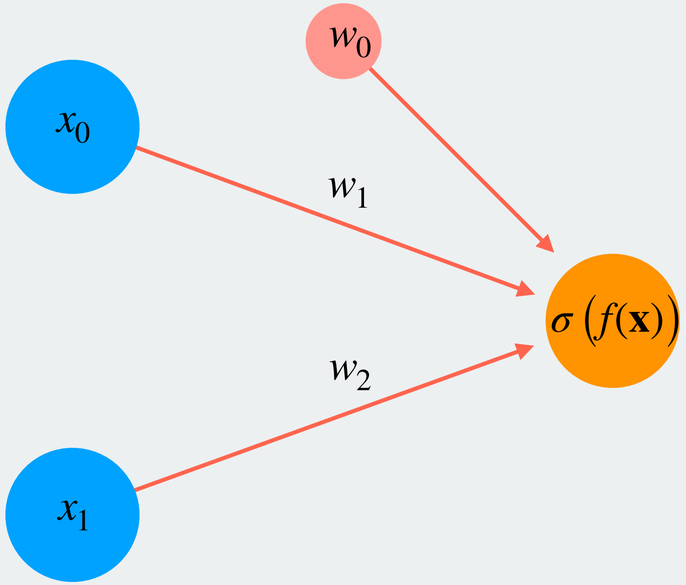
¡Perceptron!

Invented by **Frank Rosenblat** in 1957



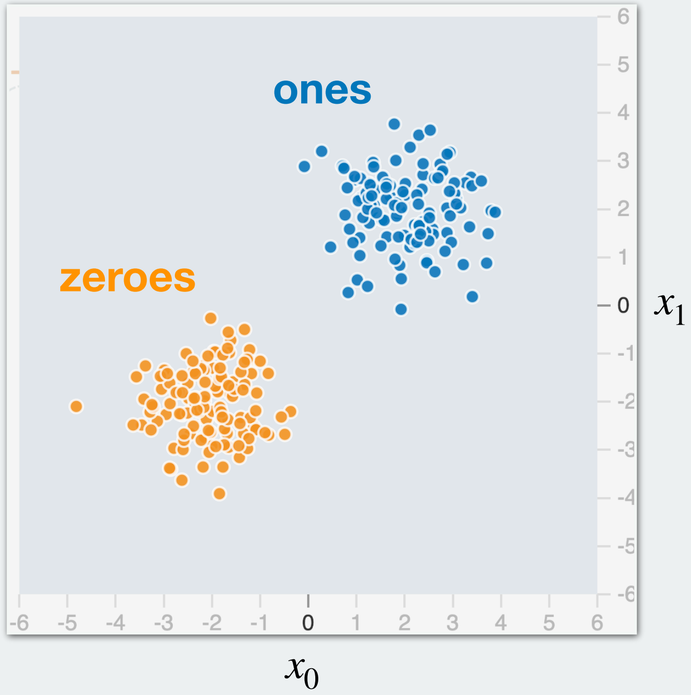
Logistic regression simple problem

> Find values of $\{w_0, w_1, w_2\}$ that minimizes $\sum_n (\tilde{y}_n - y_n)^2$



$$\mathbf{X} = \begin{array}{|c|c|}\hline x_0 & x_1 \\ \hline 1.2 & 2.5 \\ 2.1 & 3.4 \\ \dots & \dots \\ -3.3 & -2.0 \\ -1.9 & -2.5 \\ \hline \end{array} \quad , \quad \mathbf{y} = \begin{array}{|c|}\hline y \\ \hline 1 \\ 1 \\ \dots \\ 0 \\ 0 \\ \hline \end{array}$$

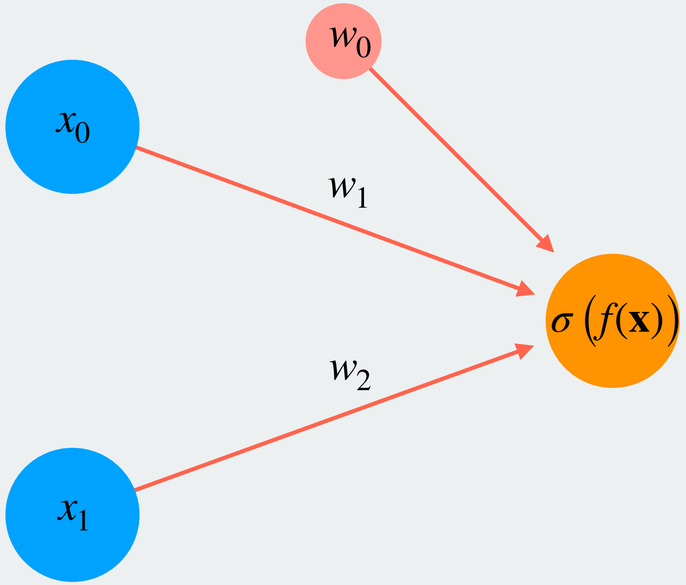
Colors are not related



Reminder: $w_0 + x_0 w_1 + x_1 w_2 = f(\mathbf{x})$

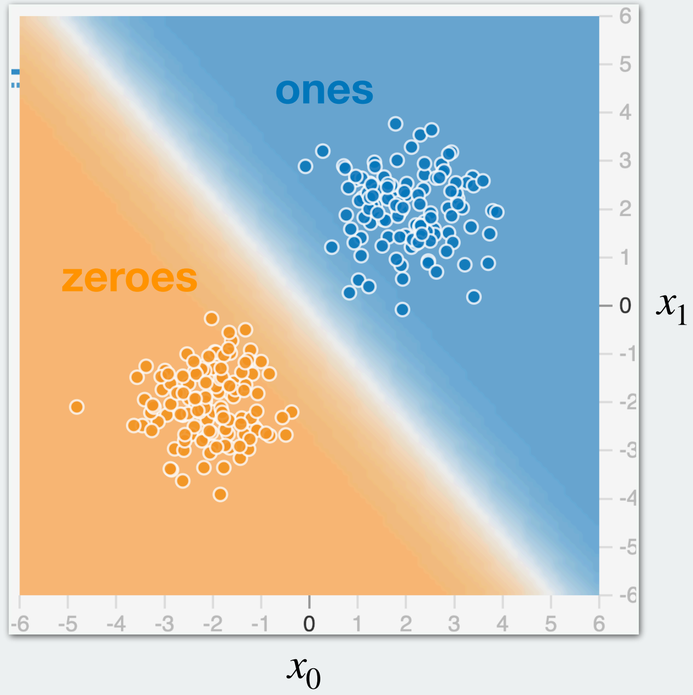
Logistic regression simple problem

> Find values of $\{w_0, w_1, w_2\}$ that minimizes $\sum_n (\tilde{y}_n - y_n)^2$



f has amazing weights gives **perfect prediction**

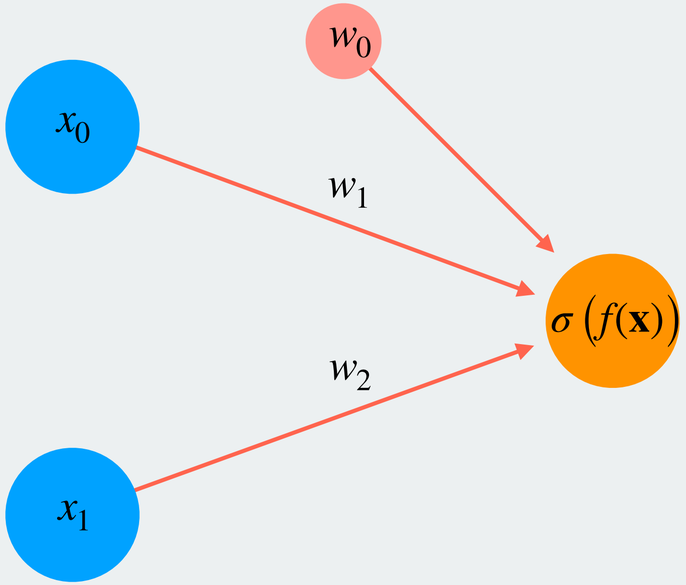
$$\mathbf{X} = \begin{array}{|c|c|}\hline x_0 & x_1 \\ \hline 1.2 & 2.5 \\ 2.1 & 3.4 \\ \dots & \dots \\ -3.3 & -2.0 \\ -1.9 & -2.5 \\ \hline \end{array}, \quad \sigma\left(f(\mathbf{X})\right) = \begin{array}{|c|}\hline \tilde{y} \\ \hline 1 \\ 1 \\ \dots \\ 0 \\ 0 \\ \hline \end{array}, \quad \mathbf{y} = \begin{array}{|c|}\hline y \\ \hline 1 \\ 1 \\ \dots \\ 0 \\ 0 \\ \hline \end{array}$$

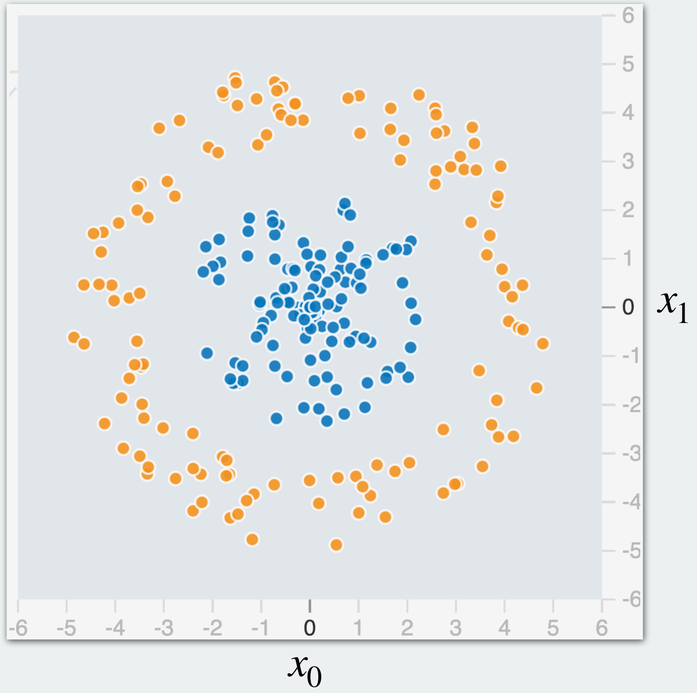


Reminder: $w_0 + x_0 w_1 + x_1 w_2 = f(\mathbf{x})$

Logistic regression not so simple problem

> Find values of $\{w_0, w_1, w_2\}$ that minimizes $\sum_n (\tilde{y}_n - y_n)^2$



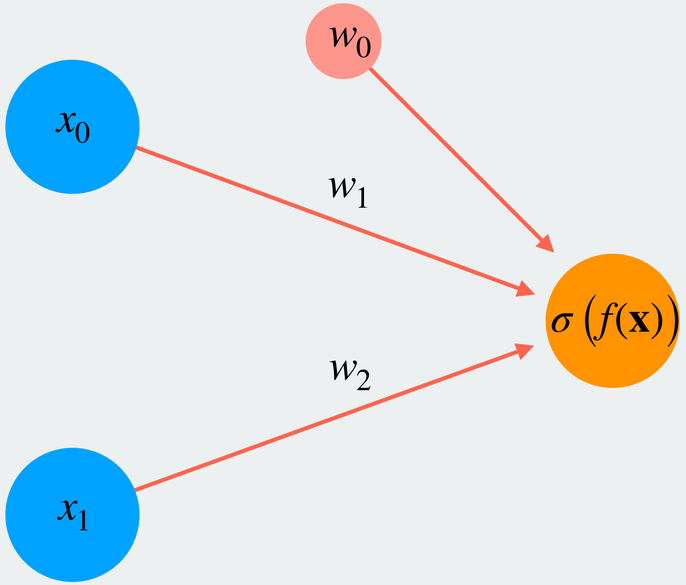


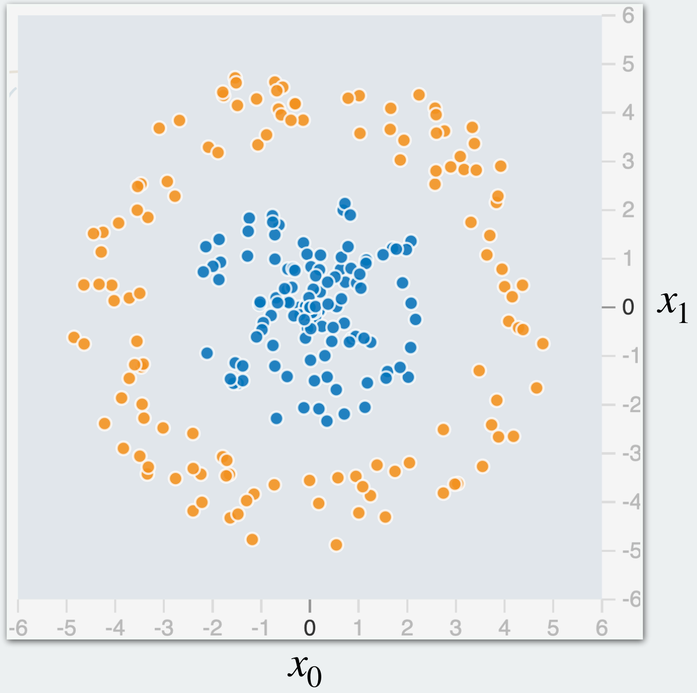
Reminder: $w_0 + x_0 w_1 + x_1 w_2 = f(\mathbf{x})$

Logistic regression not so simple problem

> Find values of $\{w_0, w_1, w_2\}$ that minimizes $\sum_n (\tilde{y}_n - y_n)^2$

Model too simple

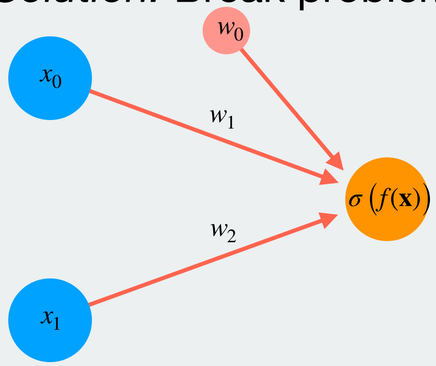




Reminder: $w_0 + x_0 w_1 + x_1 w_2 = f(\mathbf{x})$

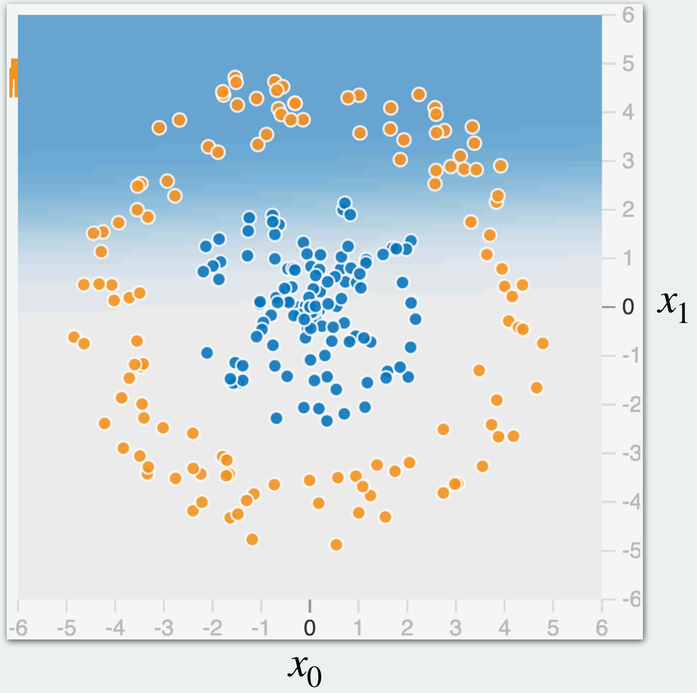
Logistic regression not so simple problem

> *Solution:* Break problem into subproblems



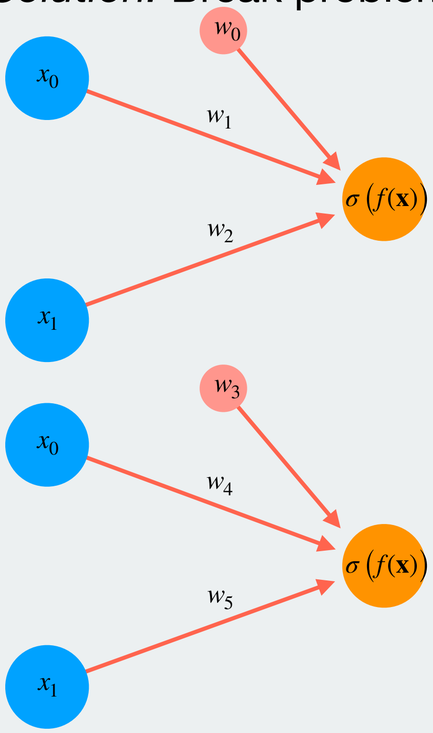
$$\sigma(w_0 + x_0 w_1 + x_1 w_2) = z_0(\mathbf{x})$$

$$\mathbf{X} = \begin{array}{|c|c|}\hline x_0 & x_1 \\ \hline 0.5 & 1.5 \\ \hline 2.3 & -1.7 \\ \hline \ldots & \ldots \\ \hline 4.2 & -0.2 \\ \hline -1.9 & 2.3 \\ \hline \end{array}, \quad z_0(\mathbf{X}) = \begin{array}{|c|}\hline z_0 \\ \hline 0.3 \\ \hline 0.25 \\ \hline \ldots \\ \hline 0.79 \\ \hline 0.34 \\ \hline \end{array}$$



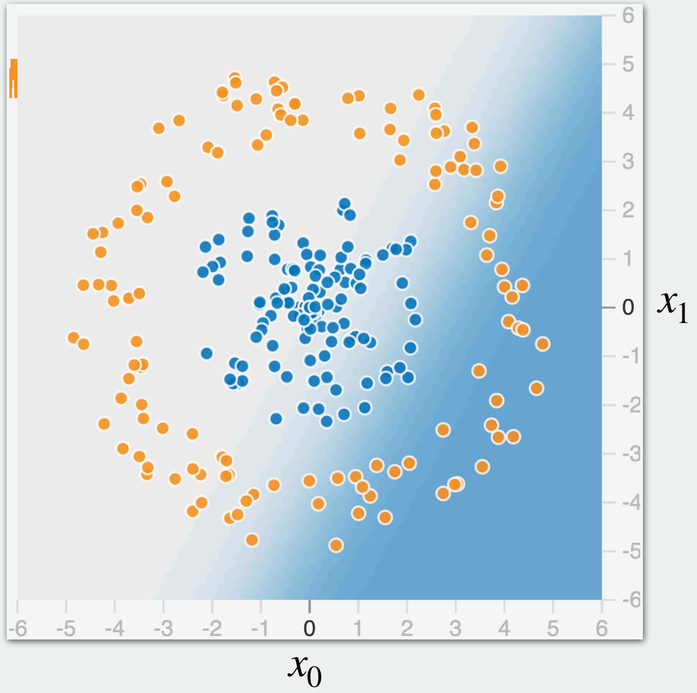
Logistic regression not so simple problem

> *Solution:* Break problem into subproblems



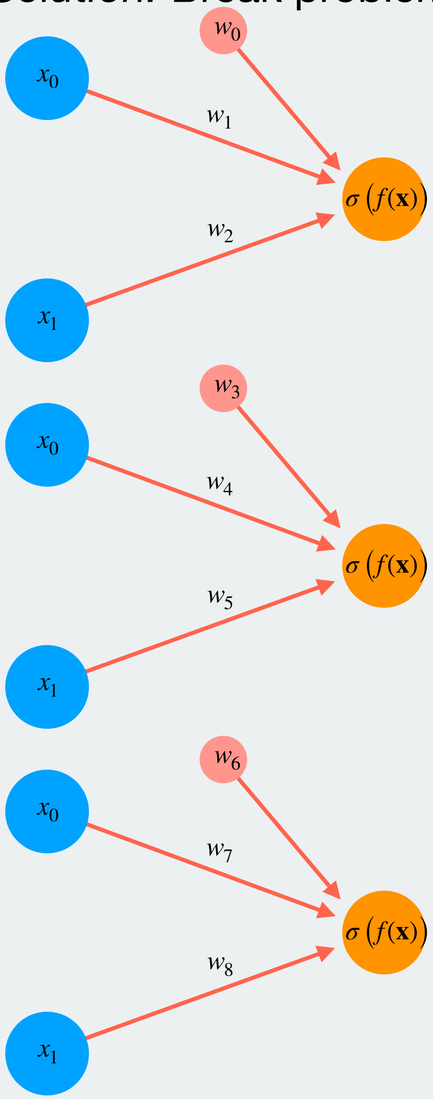
$$\sigma(w_0 + x_0 w_1 + x_1 w_2) = z_0(\mathbf{x})$$

$$\sigma(w_3 + x_0 w_4 + x_1 w_5) = z_1(\mathbf{x})$$



Logistic regression not so simple problem

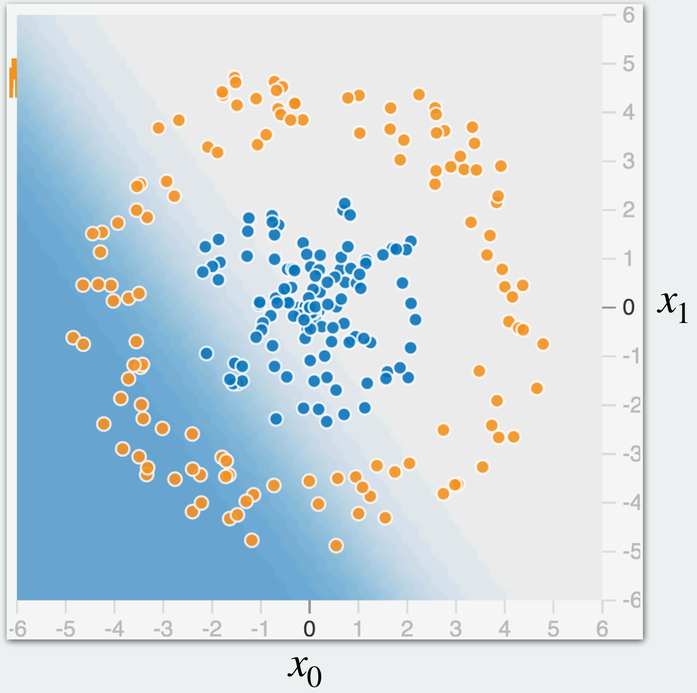
> *Solution:* Break problem into subproblems



$$\sigma(w_0 + x_0 w_1 + x_1 w_2) = z_0(\mathbf{x})$$

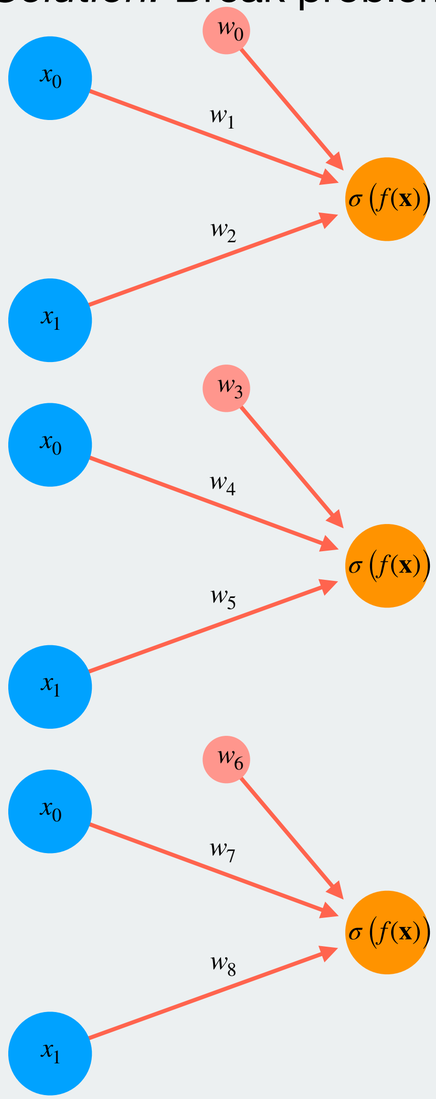
$$\sigma(w_3 + x_0 w_4 + x_1 w_5) = z_1(\mathbf{x})$$

$$\sigma(w_6 + x_0 w_7 + x_1 w_8) = z_2(\mathbf{x})$$



Logistic regression not so simple problem

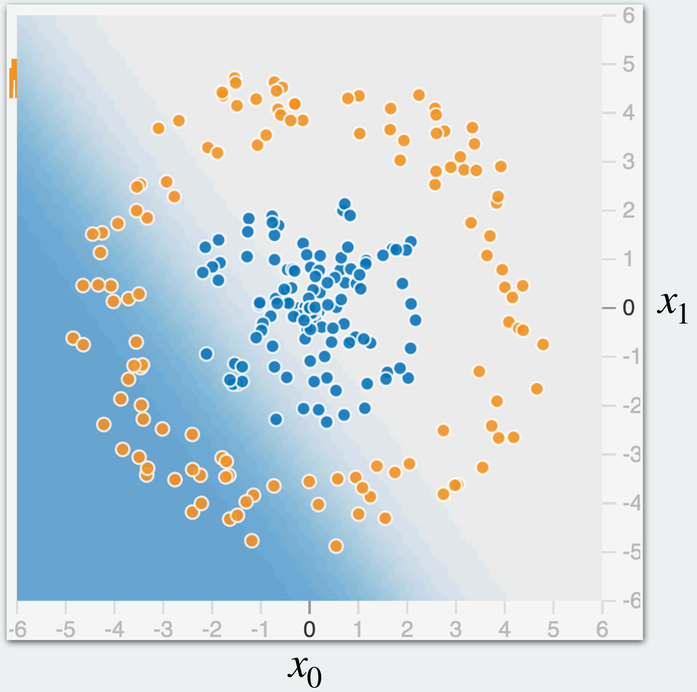
> *Solution:* Break problem into subproblems



$$\sigma(w_0 + x_0 w_1 + x_1 w_2) = z_0(\mathbf{x})$$

$$\sigma(w_3 + x_0 w_4 + x_1 w_5) = z_1(\mathbf{x})$$

$$\sigma(w_6 + x_0 w_7 + x_1 w_8) = z_2(\mathbf{x})$$

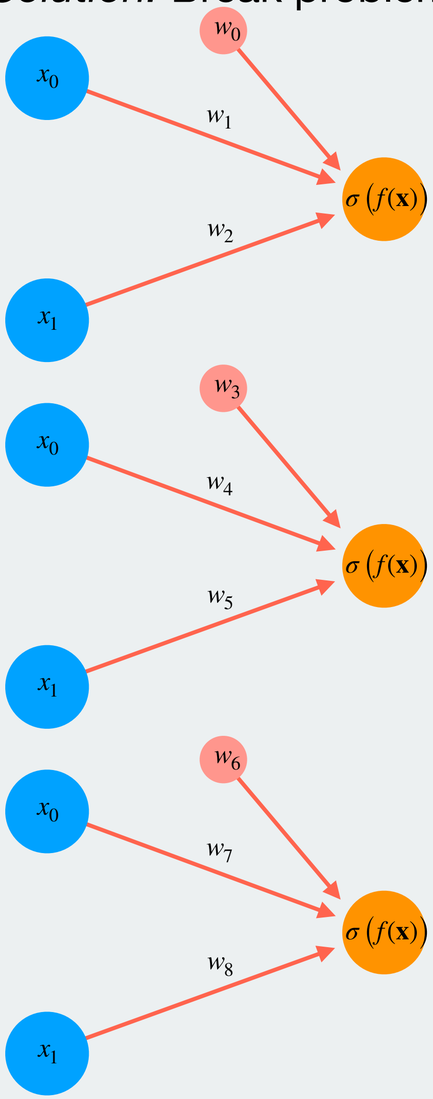


New problem: Given **Z**, predict **y**

$$\mathbf{Z} = \begin{array}{|c|c|c|} \hline z_0 & z_1 & z_2 \\ \hline 0.3 & 0.75 & 0.78 \\ \hline 0.25 & 0.1 & 0.95 \\ \hline \ldots & \ldots & \\ \hline 0.79 & 0.99 & 0.3 \\ \hline 0.34 & 0.6 & 0.1 \\ \hline \end{array} \qquad \mathbf{y} = \begin{array}{|c|} \hline y \\ \hline 0 \\ \hline 1 \\ \hline \ldots \\ \hline 1 \\ 0 \end{array}$$

Logistic regression not so simple problem

> *Solution:* Break problem into subproblems



$$\sigma(w_0 + x_0 w_1 + x_1 w_2) = z_0(\mathbf{x})$$

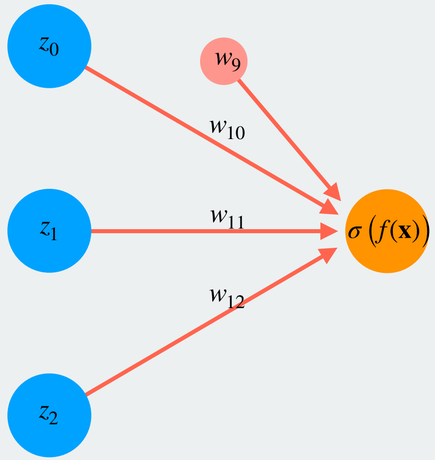
$$\sigma(w_3 + x_0 w_4 + x_1 w_5) = z_1(\mathbf{x})$$

$$\sigma(w_6 + x_0 w_7 + x_1 w_8) = z_2(\mathbf{x})$$

New problem: Given **Z**, predict **y**

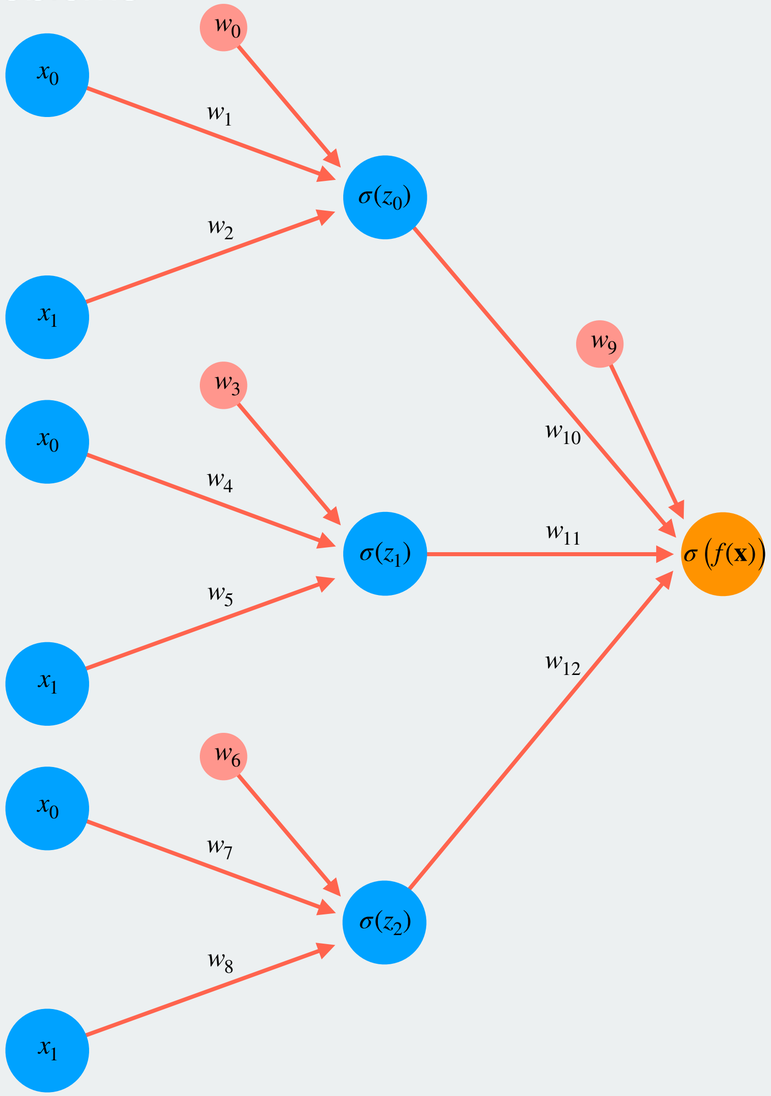
$$\mathbf{Z} = \begin{array}{|c|c|c|}\hline z_0 & z_1 & z_2 \\ \hline 0.3 & 0.75 & 0.78 \\ 0.25 & 0.1 & 0.95 \\ \ldots & \ldots & \\ 0.79 & 0.99 & 0.3 \\ 0.34 & 0.6 & 0.1 \\ \hline \end{array} \qquad \mathbf{y} = \begin{array}{|c|}\hline y \\ \hline 0 \\ 1 \\ \ldots \\ 1 \\ 0 \\ \hline \end{array}$$

Solution: Why not use a logistic regression?



Logistic regression not so simple problem

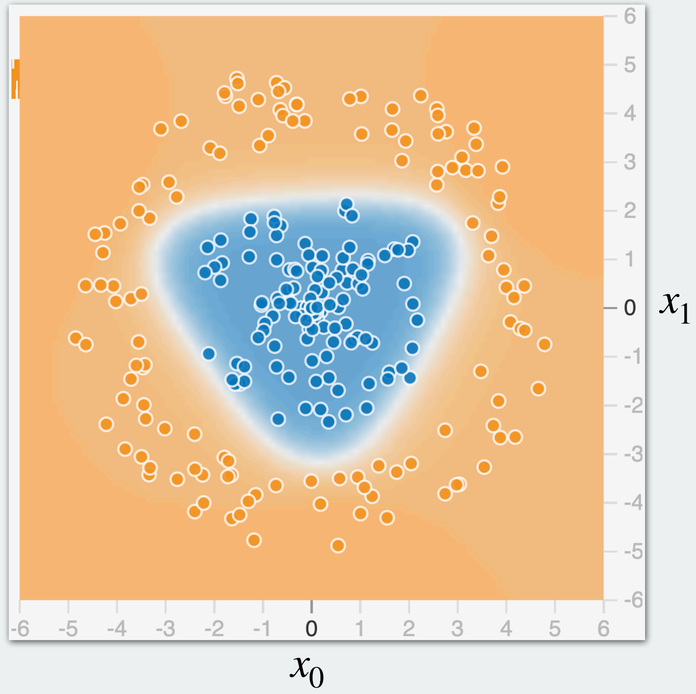
> *Solution:* Connect subproblems



Mathematical form

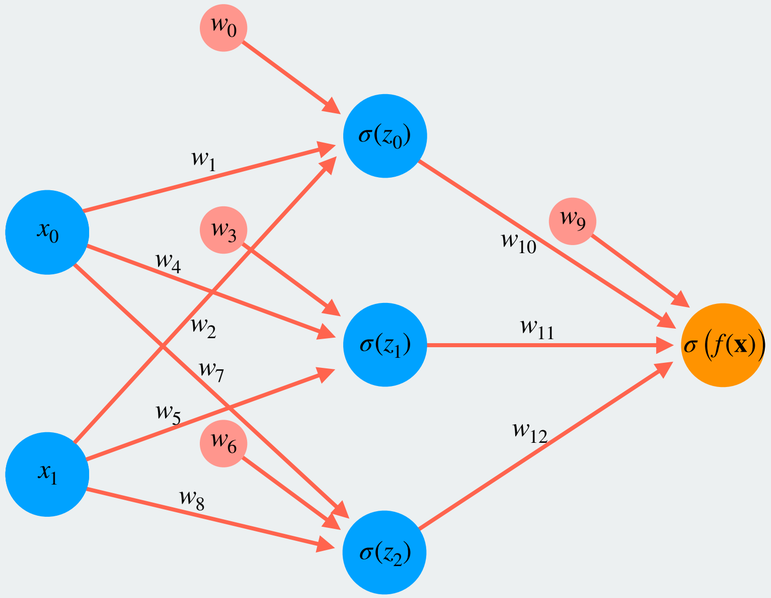
$$\sigma\Big(w_9 + \sigma(w_0 + x_0 w_1 + x_1 w_2) w_{10} \quad + \\ \sigma(w_3 + x_0 w_4 + x_1 w_5) w_{11} \quad + \\ \sigma(w_6 + x_0 w_7 + x_1 w_8) w_{12}\Big) = \sigma\Big(f(x)\Big)$$

Solution



Logistic regression not so simple problem

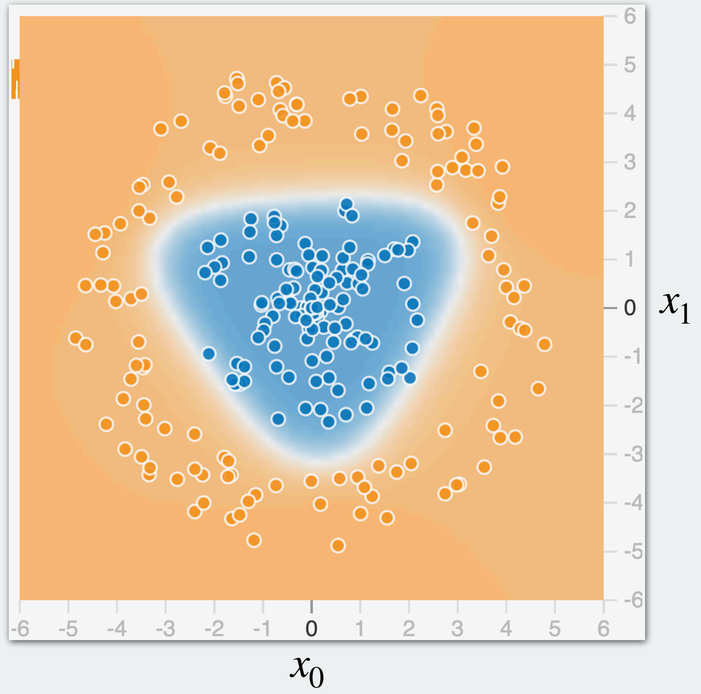
> *Solution:* Connect subproblems



Mathematical form

$$\sigma\Big(w_9 + \sigma(w_0 + x_0 w_1 + x_1 w_2) w_{10} \quad + \\ \sigma(w_3 + x_0 w_4 + x_1 w_5) w_{11} \quad + \\ \sigma(w_6 + x_0 w_7 + x_1 w_8) w_{12}\Big) = \sigma\Big(f(x)\Big)$$

Solution

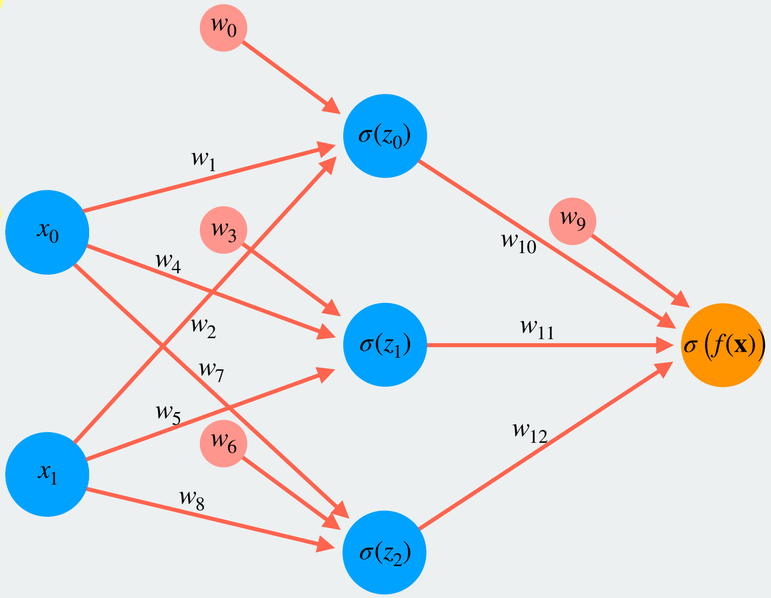


Logistic regression not so simple problem

problems

¡Multilayer perceptron!

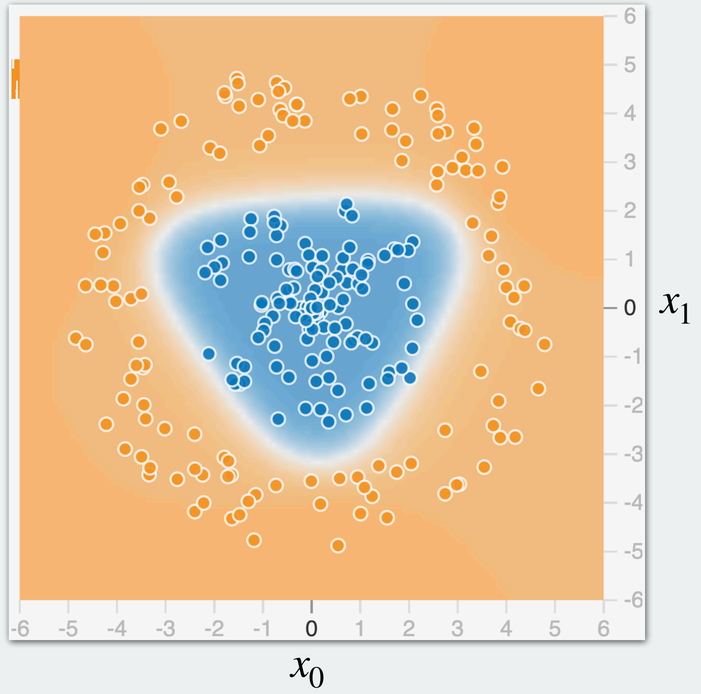
classifier



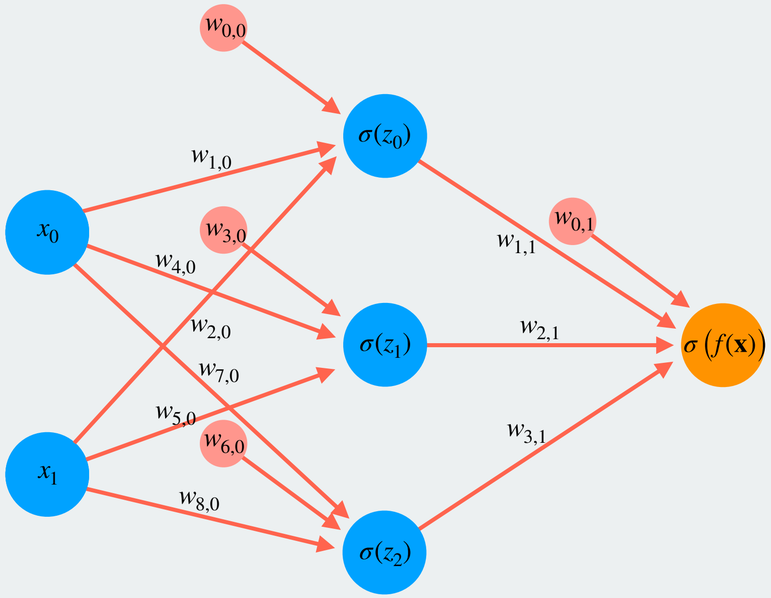
Mathematical form

$$\sigma\Big(w_9 + \sigma(w_0 + x_0 w_1 + x_1 w_2) w_{10} \quad + \\ \sigma(w_3 + x_0 w_4 + x_1 w_5) w_{11} \quad + \\ \sigma(w_6 + x_0 w_7 + x_1 w_8) w_{12}\Big) = \sigma\Big(f(x)\Big)$$

Solution



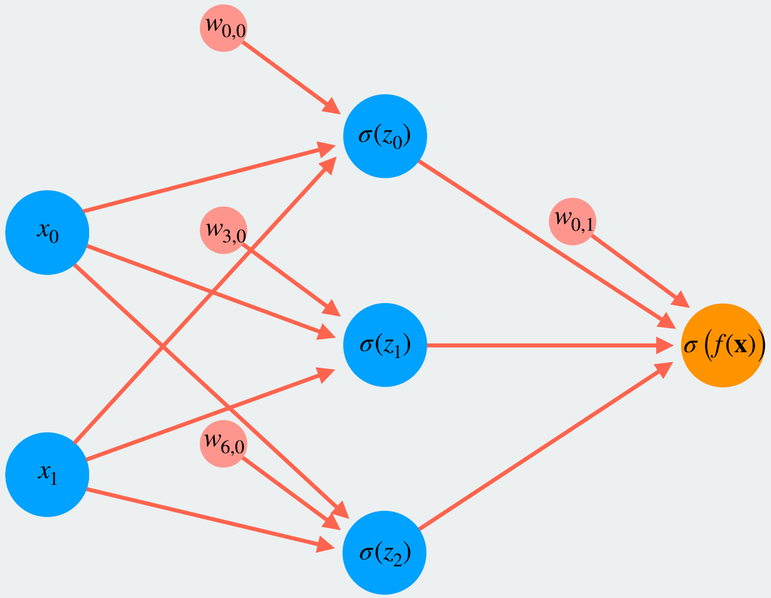
Multilayer perceptron – the math



Mathematical form

$$\sigma\Big(w_{0,1} + \sigma(w_{0,0} + x_0 w_{1,0} + x_1 w_{2,0}) w_{1,1} \quad + \\ \sigma(w_{3,0} + x_0 w_{4,0} + x_1 w_{5,0}) w_{2,1} \quad + \\ \sigma(w_{6,0} + x_0 w_{7,0} + x_1 w_{8,0}) w_{3,1}\Big) = \sigma\Big(f(x)\Big)$$

Multilayer perceptron – the math



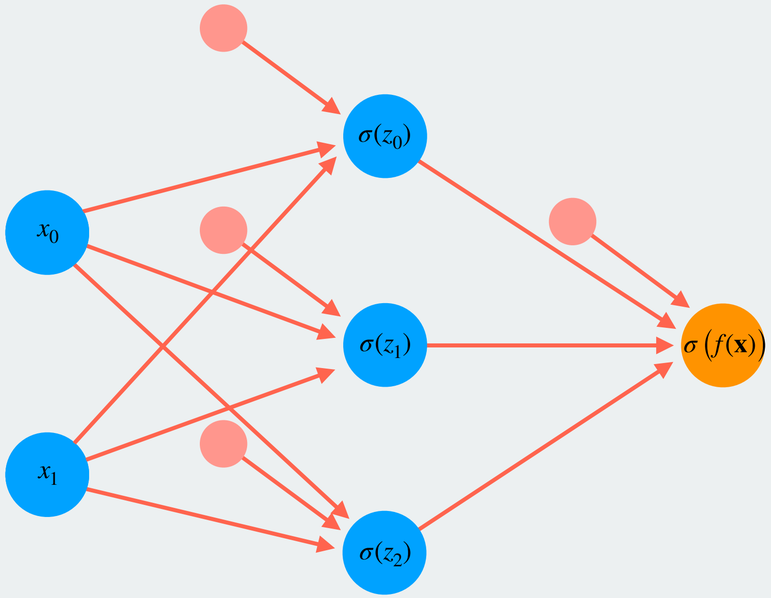
Mathematical form

$$\sigma\Big(w_{0,1} + \sigma(w_{0,0} + x_0 w_{1,0} + x_1 w_{2,0}) w_{1,1} \quad + \\ \sigma(w_{3,0} + x_0 w_{4,0} + x_1 w_{5,0}) w_{2,1} \quad + \\ \sigma(w_{6,0} + x_0 w_{7,0} + x_1 w_{8,0}) w_{3,1}\Big) = \sigma\Big(f(x)\Big)$$

$$\mathbf{W_0} = \begin{bmatrix} w_{1,0} & w_{2,0} \\ w_{4,0} & w_{5,0} \\ w_{7,0} & w_{8,0} \end{bmatrix}$$

$$\mathbf{W_1} = \begin{bmatrix} w_{1,1} & w_{2,1} & w_{3,1} \end{bmatrix}$$

Multilayer perceptron – the math

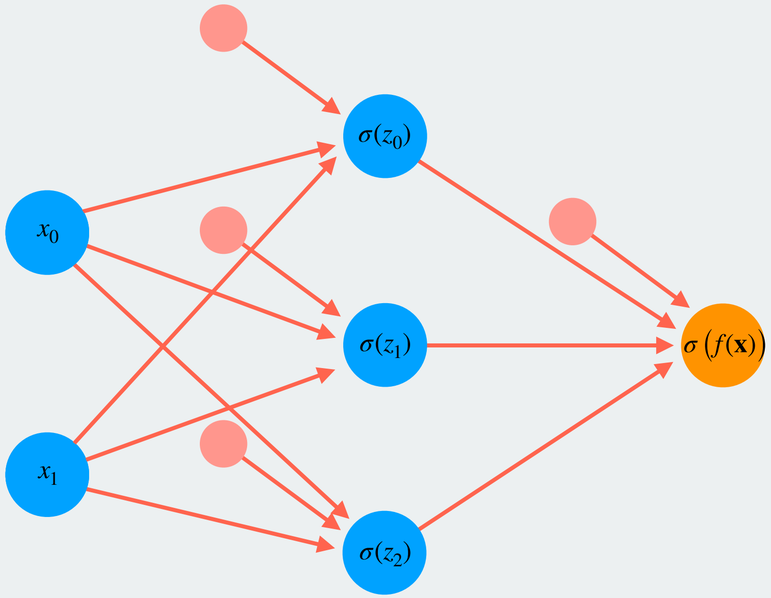


Mathematical form

$$\sigma\Big(w_{0,1} + \sigma(w_{0,0} + x_0 w_{1,0} + x_1 w_{2,0}) w_{1,1} \quad + \\ \sigma(w_{3,0} + x_0 w_{4,0} + x_1 w_{5,0}) w_{2,1} \quad + \\ \sigma(w_{6,0} + x_0 w_{7,0} + x_1 w_{8,0}) w_{3,1}\Big) = \sigma\Big(f(x)\Big)$$

$$\mathbf{b_0} = \begin{bmatrix} w_{0,0} \\ w_{3,0} \\ w_{6,0} \end{bmatrix} \quad \mathbf{W_0} = \begin{bmatrix} w_{1,0} & w_{2,0} \\ w_{4,0} & w_{5,0} \\ w_{7,0} & w_{8,0} \end{bmatrix} \quad \mathbf{b_1} = \begin{bmatrix} w_{0,1} \end{bmatrix} \quad \mathbf{W_1} = \begin{bmatrix} w_{1,1} & w_{2,1} & w_{3,1} \end{bmatrix}$$

Multilayer perceptron – the math



Mathematical form

$$\sigma\Big(w_{0,1} + \sigma(w_{0,0} + x_0 w_{1,0} + x_1 w_{2,0}) w_{1,1} \quad + \quad \sigma(w_{3,0} + x_0 w_{4,0} + x_1 w_{5,0}) w_{2,1} \quad + \quad \sigma(w_{6,0} + x_0 w_{7,0} + x_1 w_{8,0}) w_{3,1}\Big) = \sigma\left(f(x)\right)$$

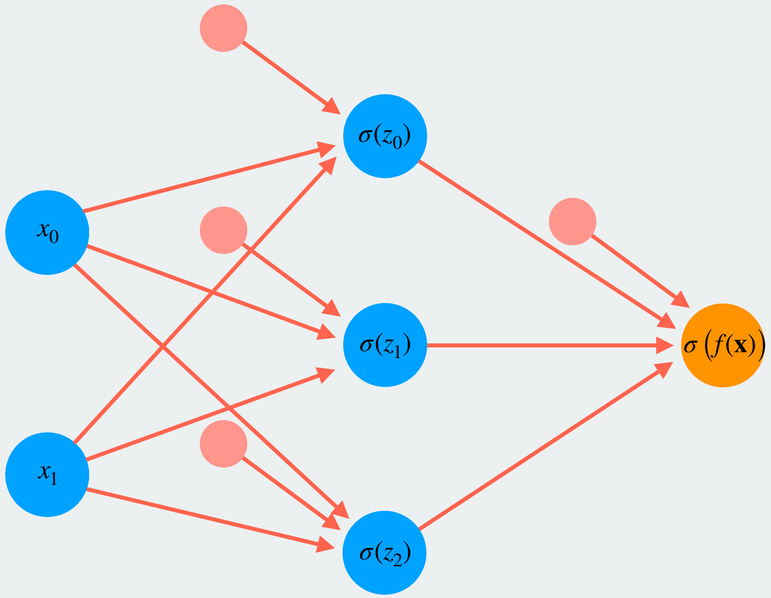
$\Leftrightarrow$

$$\sigma\Big(\mathbf{b_1} + \mathbf{W_1}\sigma\left(\mathbf{b_0} + \mathbf{W_0}\mathbf{x}\right)\Big) = \sigma\left(f(x)\right)$$

$$\mathbf{b_0} = \begin{bmatrix} w_{0,0} \\ w_{3,0} \\ w_{6,0} \end{bmatrix} \quad \mathbf{W_0} = \begin{bmatrix} w_{1,0} & w_{2,0} \\ w_{4,0} & w_{5,0} \\ w_{7,0} & w_{8,0} \end{bmatrix} \quad \mathbf{b_1} = \begin{bmatrix} w_{0,1} \end{bmatrix} \quad \mathbf{W_1} = \begin{bmatrix} w_{1,1} & w_{2,1} & w_{3,1} \end{bmatrix}$$

and $\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}$

Multilayer perceptron – the math



$$\mathbf{b_0} = \begin{bmatrix} w_{0,0} \\ w_{3,0} \\ w_{6,0} \end{bmatrix} \quad \mathbf{W_0} = \begin{bmatrix} w_{1,0} & w_{2,0} \\ w_{4,0} & w_{5,0} \\ w_{7,0} & w_{8,0} \end{bmatrix} \qquad \mathbf{b_1} = \begin{bmatrix} w_{0,1} \end{bmatrix} \quad \mathbf{W_1} = \begin{bmatrix} w_{1,1} & w_{2,1} & w_{3,1} \end{bmatrix}$$

Mathematical form

$$\begin{aligned} \sigma\Big(w_{0,1} + \sigma(w_{0,0} + x_0 w_{1,0} + x_1 w_{2,0}) w_{1,1} \quad &+ \\ \sigma(w_{3,0} + x_0 w_{4,0} + x_1 w_{5,0}) w_{2,1} \quad &+ \\ \sigma(w_{6,0} + x_0 w_{7,0} + x_1 w_{8,0}) w_{3,1}\Big) &= \sigma\left(f(x)\right) \end{aligned}$$

$$\Leftrightarrow$$

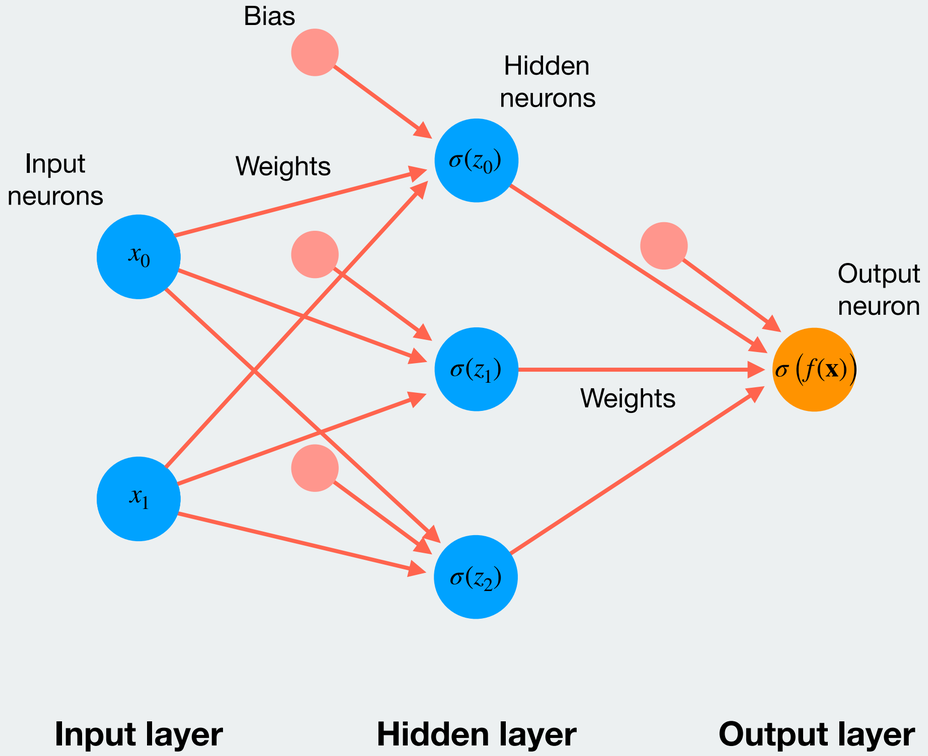
$$\sigma\left(\mathbf{b_1} + \mathbf{W_1}\sigma\left(\mathbf{b_0} + \mathbf{W_0}\mathbf{x}\right)\right) = \sigma\left(f(x)\right)$$

$$\Leftrightarrow$$

$$\sigma\left(\mathbf{b_n} + \mathbf{W_n}\mathbf{a_{n-1}}\right) = \mathbf{a_n}$$

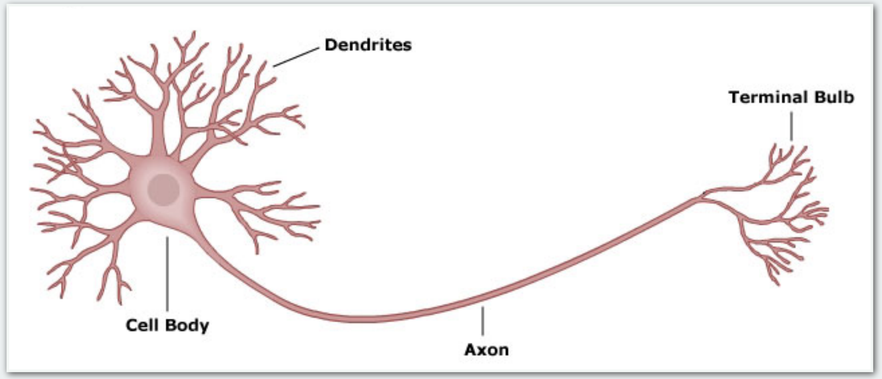
and $\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}$

Multilayer perceptron – the structure



Multilayer perceptron – why “Neural”?

Brain neuron

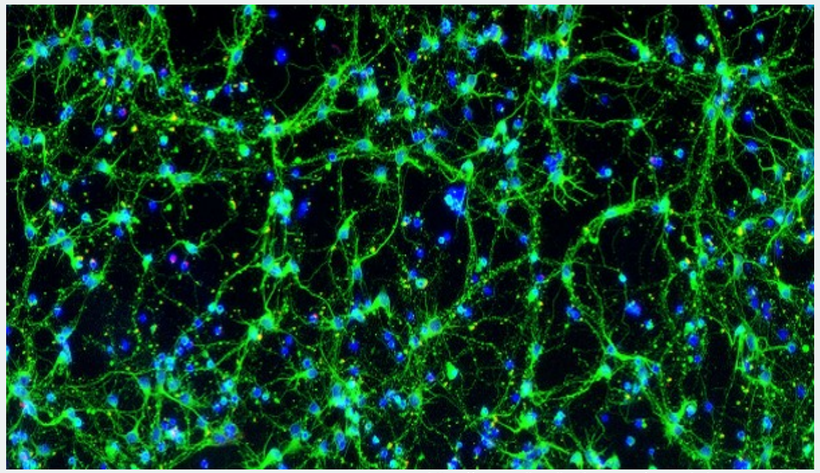


https://neurofantastic.com/brain/2017/4/13/brain-computation-is-a-lot-more-analog-than-we-thought

3D artist impression

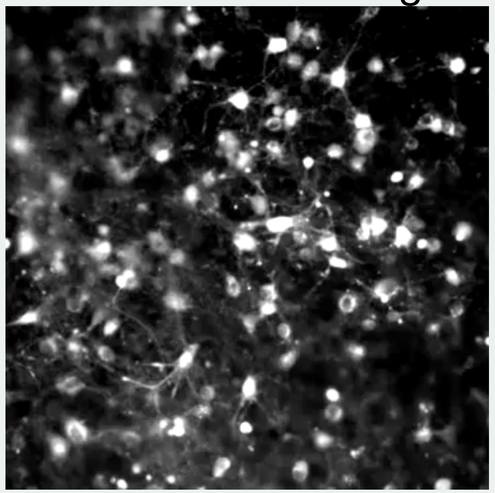
https://alsnewstoday.com/2017/07/25/als-study-shows-how-excessive-dna-repetitions-trigger-neuron-deaths/

Rat neuron image



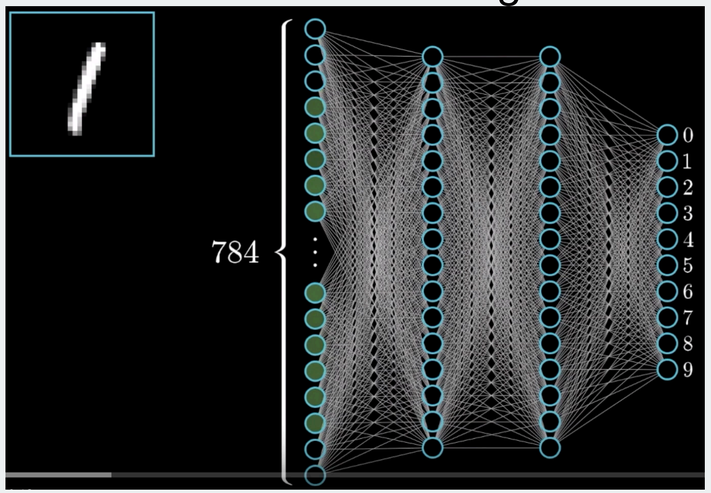
https://www.cellapplications.com/expanded-neuron-offering

Live rat neurons firing



https://www.youtube.com/watch?v=yy994HpFudc

Live artificial neurons firing



https://www.youtube.com/watch?v=aircAruvnKk&t=1s

Multilayer perceptron – why use non-linearities?

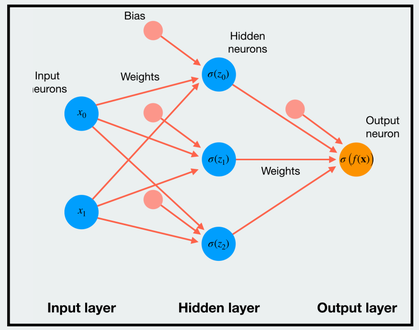
Mathematical form

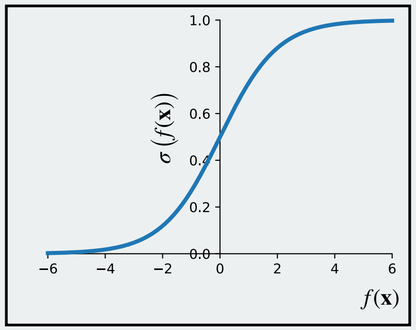
$$\sigma\Big(w_{0,1} + \sigma(w_{0,0} + x_0 w_{1,0} + x_1 w_{2,0})w_{1,1} \quad + \\ \sigma(w_{3,0} + x_0 w_{4,0} + x_1 w_{5,0})w_{2,1} \quad + \\ \sigma(w_{6,0} + x_0 w_{7,0} + x_1 w_{8,0})w_{3,1}\Big) = \sigma\Big(f(x)\Big)$$

Mathematical form without non-linearities

$$w_{0,1} + (w_{0,0} + x_0 w_{1,0} + x_1 w_{2,0})w_{1,1} + \\ (w_{3,0} + x_0 w_{4,0} + x_1 w_{5,0})w_{2,1} + \\ (w_{6,0} + x_0 w_{7,0} + x_1 w_{8,0})w_{3,1} = f(x)$$

Reminders





Multilayer perceptron – why use non-linearities?

Mathematical form

$$\sigma\Big(w_{0,1} + \sigma(w_{0,0} + x_0 w_{1,0} + x_1 w_{2,0})w_{1,1} \quad + \\ \sigma(w_{3,0} + x_0 w_{4,0} + x_1 w_{5,0})w_{2,1} \quad + \\ \sigma(w_{6,0} + x_0 w_{7,0} + x_1 w_{8,0})w_{3,1}\Big) = \sigma\Big(f(x)\Big)$$

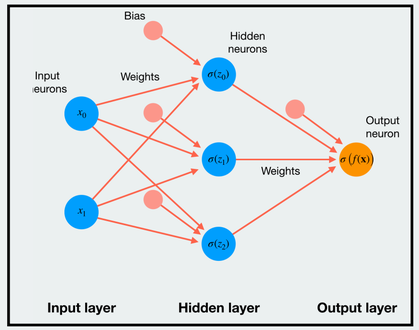
Mathematical form without non-linearities

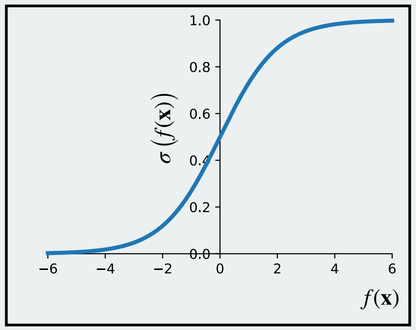
$$w_{0,1} + (w_{0,0} + x_0 w_{1,0} + x_1 w_{2,0})w_{1,1} + \\ (w_{3,0} + x_0 w_{4,0} + x_1 w_{5,0})w_{2,1} + \\ (w_{6,0} + x_0 w_{7,0} + x_1 w_{8,0})w_{3,1} = f(x)$$

$$\Leftrightarrow$$

$$x_0(w_{1,0}w_{1,1} + w_{4,0}w_{2,1} + w_{7,0}w_{3,1}) + \\ x_1(w_{2,0}w_{1,1} + w_{5,0}w_{2,1} + w_{8,0}w_{3,1}) + \\ w_{0,1} + w_{0,0}w_{1,1} + w_{3,0}w_{2,1} + w_{6,0}w_{3,1} = f(x)$$

Reminders





Multilayer perceptron – why use non-linearities?

Mathematical form

$$\sigma\Big(w_{0,1} + \sigma(w_{0,0} + x_0 w_{1,0} + x_1 w_{2,0})w_{1,1} \quad + \\ \sigma(w_{3,0} + x_0 w_{4,0} + x_1 w_{5,0})w_{2,1} \quad + \\ \sigma(w_{6,0} + x_0 w_{7,0} + x_1 w_{8,0})w_{3,1}\Big) = \sigma\Big(f(x)\Big)$$

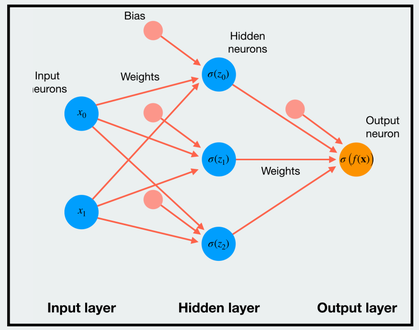
Mathematical form without non-linearities

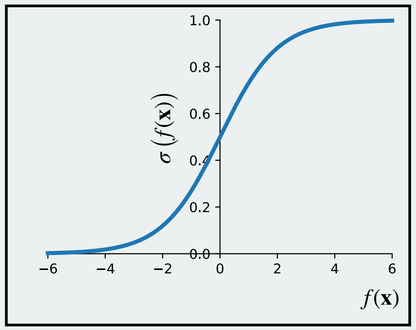
$$w_{0,1} + (w_{0,0} + x_0 w_{1,0} + x_1 w_{2,0})w_{1,1} + \\ (w_{3,0} + x_0 w_{4,0} + x_1 w_{5,0})w_{2,1} + \\ (w_{6,0} + x_0 w_{7,0} + x_1 w_{8,0})w_{3,1} = f(x)$$

$$\Leftrightarrow$$

$$x_0(w_{1,0}w_{1,1} + w_{4,0}w_{2,1} + w_{7,0}w_{3,1}) + \\ x_1(w_{2,0}w_{1,1} + w_{5,0}w_{2,1} + w_{8,0}w_{3,1}) + \\ w_{0,1} + w_{0,0}w_{1,1} + w_{3,0}w_{2,1} + w_{6,0}w_{3,1} = f(x)$$

Reminders





Multilayer perceptron – why use non-linearities?

Mathematical form

$$\sigma\Big(w_{0,1} + \sigma(w_{0,0} + x_0 w_{1,0} + x_1 w_{2,0})w_{1,1} \quad + \\ \sigma(w_{3,0} + x_0 w_{4,0} + x_1 w_{5,0})w_{2,1} \quad + \\ \sigma(w_{6,0} + x_0 w_{7,0} + x_1 w_{8,0})w_{3,1}\Big) = \sigma\Big(f(x)\Big)$$

Mathematical form without non-linearities

$$w_{0,1} + (w_{0,0} + x_0 w_{1,0} + x_1 w_{2,0})w_{1,1} + \\ (w_{3,0} + x_0 w_{4,0} + x_1 w_{5,0})w_{2,1} + \\ (w_{6,0} + x_0 w_{7,0} + x_1 w_{8,0})w_{3,1} = f(x)$$

$$\Leftrightarrow$$

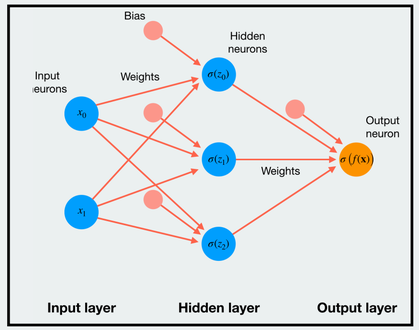
$$x_0(w_{1,0}w_{1,1} + w_{4,0}w_{2,1} + w_{7,0}w_{3,1}) + \\ x_1(w_{2,0}w_{1,1} + w_{5,0}w_{2,1} + w_{8,0}w_{3,1}) + \\ w_{0,1} + w_{0,0}w_{1,1} + w_{3,0}w_{2,1} + w_{6,0}w_{3,1} = f(x)$$

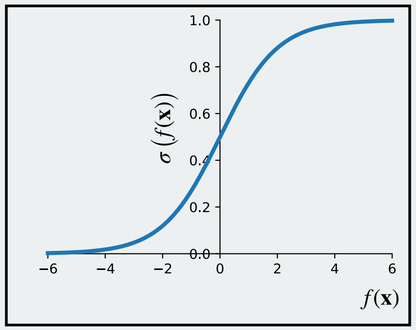
$$\Leftrightarrow$$

$$x_0 w_a + x_1 w_b + w_c = f(x)$$

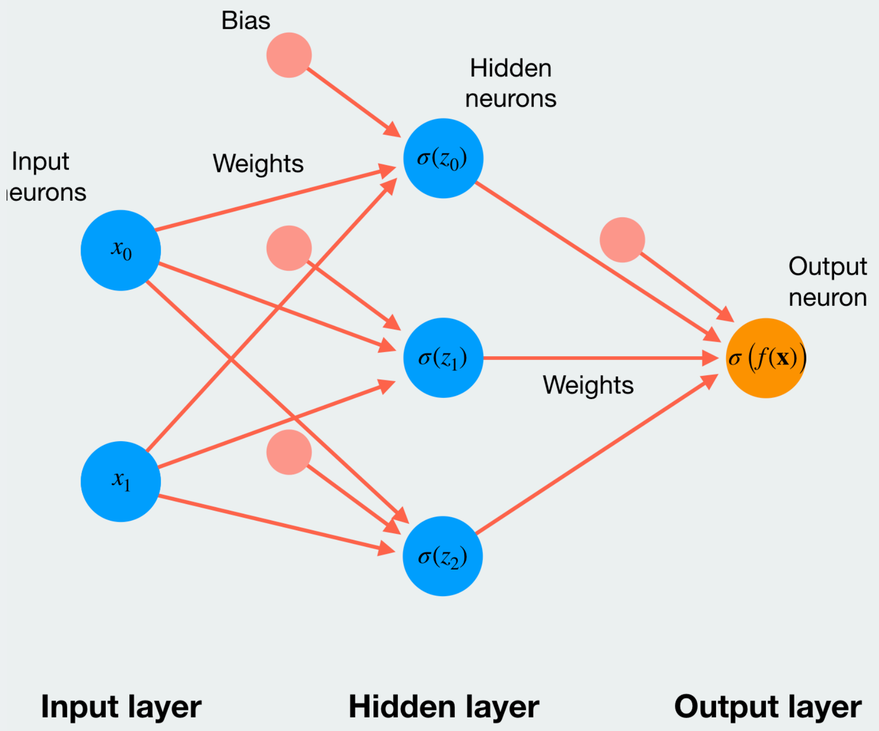
omg! **nested** linear regression is actually **just** linear regression

Reminders





Multilayer perceptron – overview

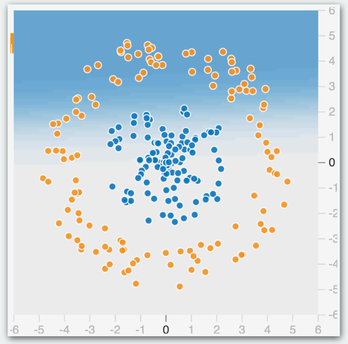


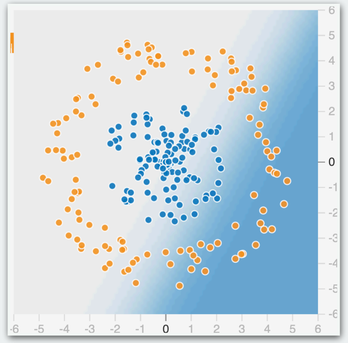
- Forward-coupled logistic regressions
- Activation function **necessary**
- Number of *layers* refers to number of layers of weights (left, 2)
- Can be *deep,* i.e. have many *hidden* layers
- Dense/fully-connected layers
- Can have multiple output neurons
- More general name: **Feed forward neural network**

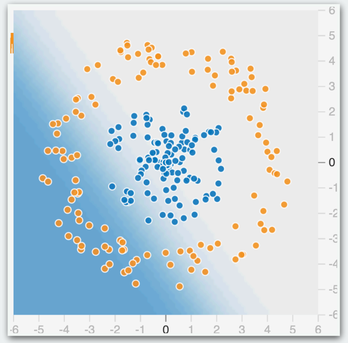
Model fitting

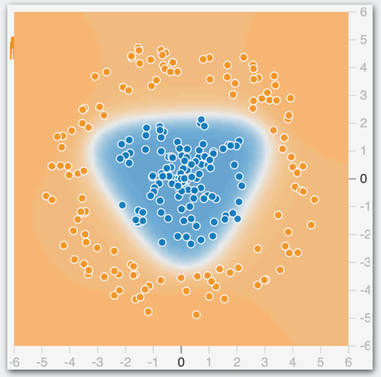
Finding the weights $\mathbf{W}$, that gives you the best prediction

How do we find the weights that give the best classification?

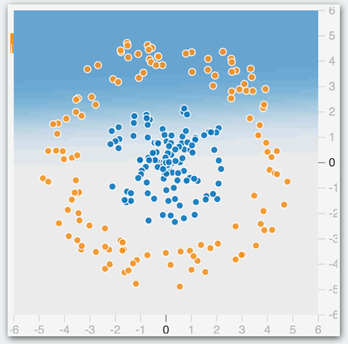


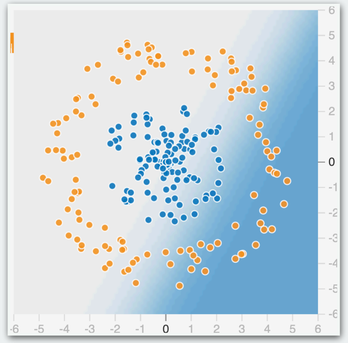


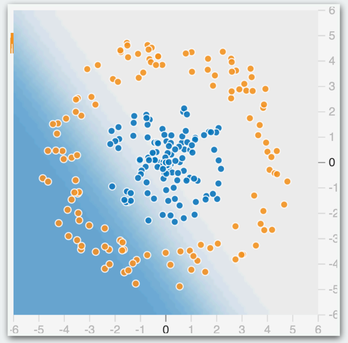


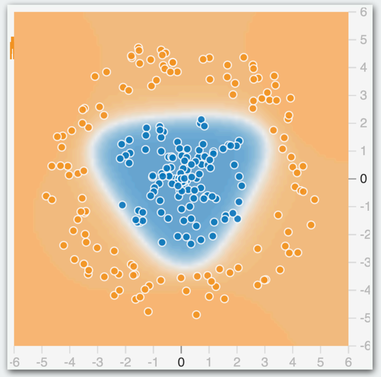


How do we find the weights that give the best classification?









predicted from **X** given **W**

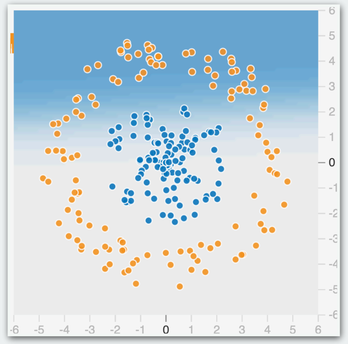
$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.96 \\ 0.10 \\ 0.04 \\ \dots \\ 0.70 \\ 0.02 \\ 0.99 \end{bmatrix}$$

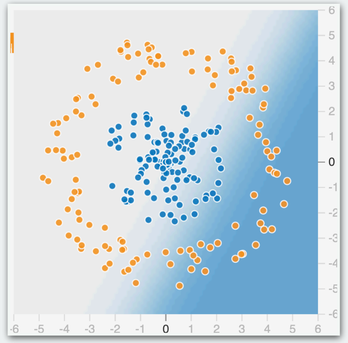
true

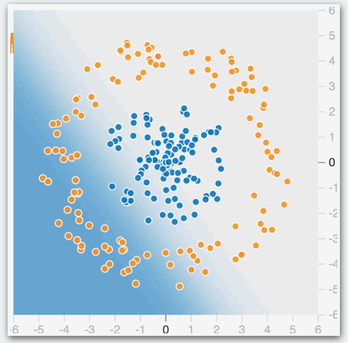
$$\mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \dots \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

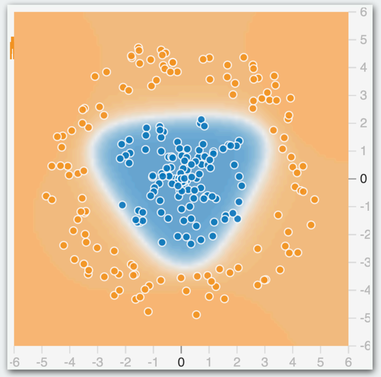
$$\begin{aligned} C(\mathbf{W}) &= \frac{1}{N}\sum_i (\tilde{y}_i - y_i)^2 \\ &= (0.96 - 1)^2 \\ &+ (0.10 - 0)^2 \\ &+ (0.04 - 0)^2 \\ &+ \dots \\ &+ (0.70 - 1)^2 \\ &+ (0.02 - 0)^2 \\ &+ (0.99 - 1)^2 \end{aligned}$$

How do we find the weights that give the best classification?









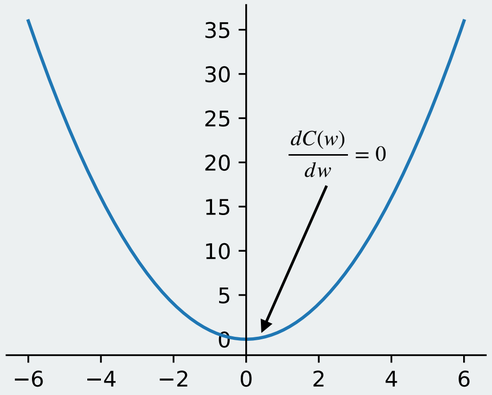
predicted from **X** given **W**

$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.96 \\ 0.10 \\ 0.04 \\ \cdots \\ 0.70 \\ 0.02 \\ 0.99 \end{bmatrix}$$

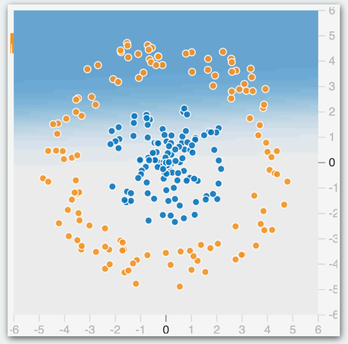
true

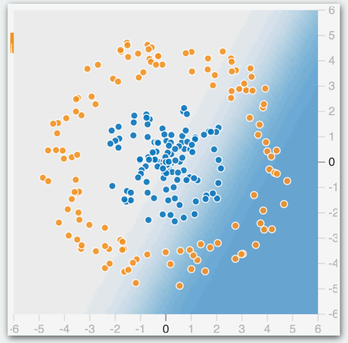
$$\mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \cdots \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

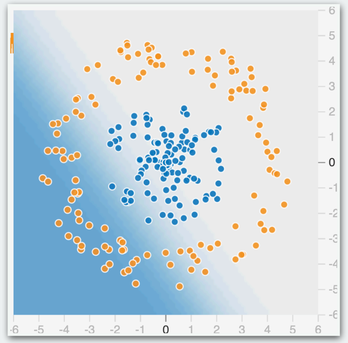
$$\begin{aligned} C(\mathbf{W}) &= \frac{1}{N}\sum_i (\tilde{y}_i - y_i)^2 \\ &= (0.96 - 1)^2 \\ &+ (0.10 - 0)^2 \\ &+ (0.04 - 0)^2 \\ &+ \ldots \\ &+ (0.70 - 1)^2 \\ &+ (0.02 - 0)^2 \\ &+ (0.99 - 1)^2 \end{aligned}$$

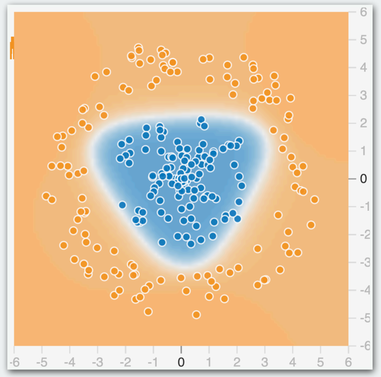


How do we find the weights that give the best classification?









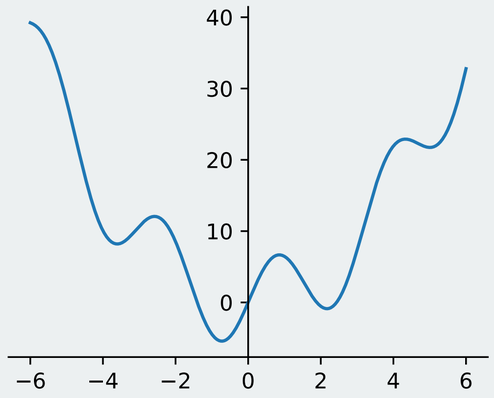
predicted from **X** given **W**

$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.96 \\ 0.10 \\ 0.04 \\ \dots \\ 0.70 \\ 0.02 \\ 0.99 \end{bmatrix}$$

true

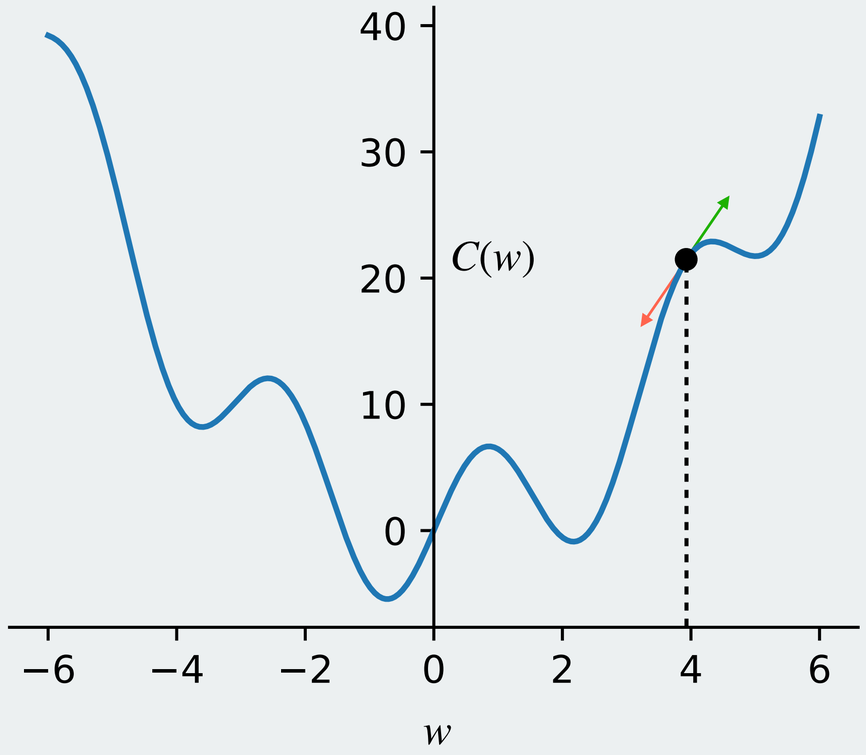
$$\mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \dots \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{aligned} C(\mathbf{W}) &= \frac{1}{N}\sum_i (\tilde{y}_i - y_i)^2 \\ &= (0.96 - 1)^2 \\ &+ (0.10 - 0)^2 \\ &+ (0.04 - 0)^2 \\ &+ \dots \\ &+ (0.70 - 1)^2 \\ &+ (0.02 - 0)^2 \\ &+ (0.99 - 1)^2 \end{aligned}$$



How do we find the weights that give the best classification?

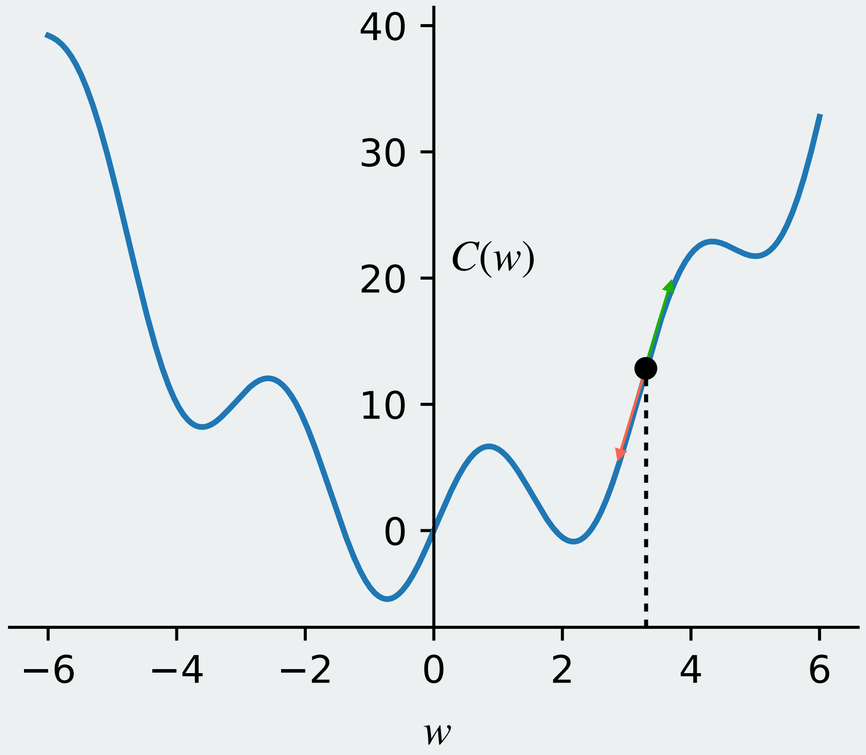
> Gradient Descent



$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.82 \\ 0.47 \\ 0.21 \\ \cdots \\ 0.41 \\ 0.74 \\ 0.23 \end{bmatrix}$$

How do we find the weights that give the best classification?

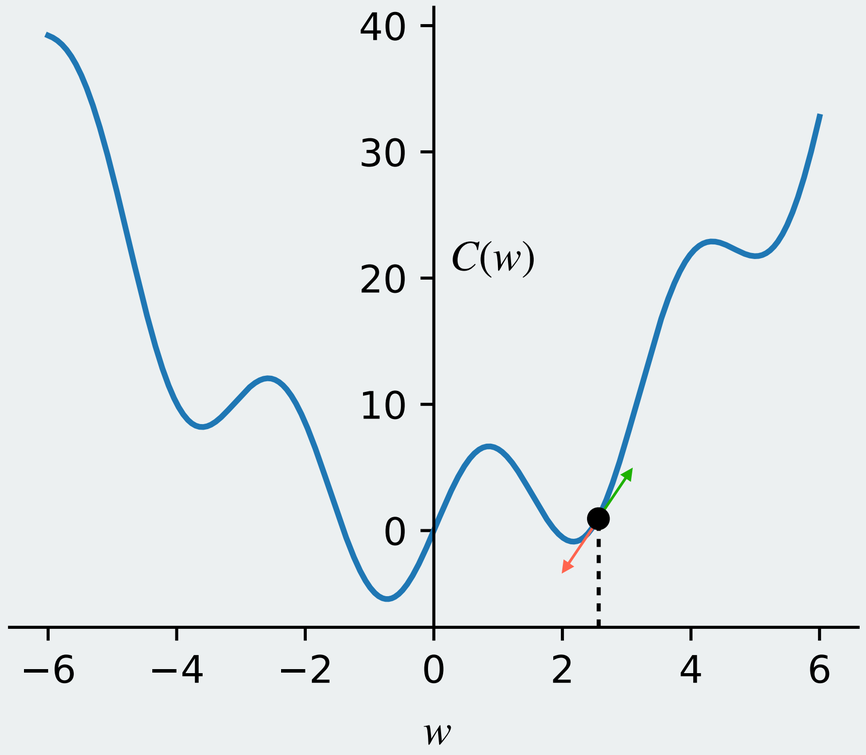
> Gradient Descent



$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.82 \\ 0.32 \\ 0.12 \\ \cdots \\ 0.25 \\ 0.85 \\ 0.11 \end{bmatrix}$$

How do we find the weights that give the best classification?

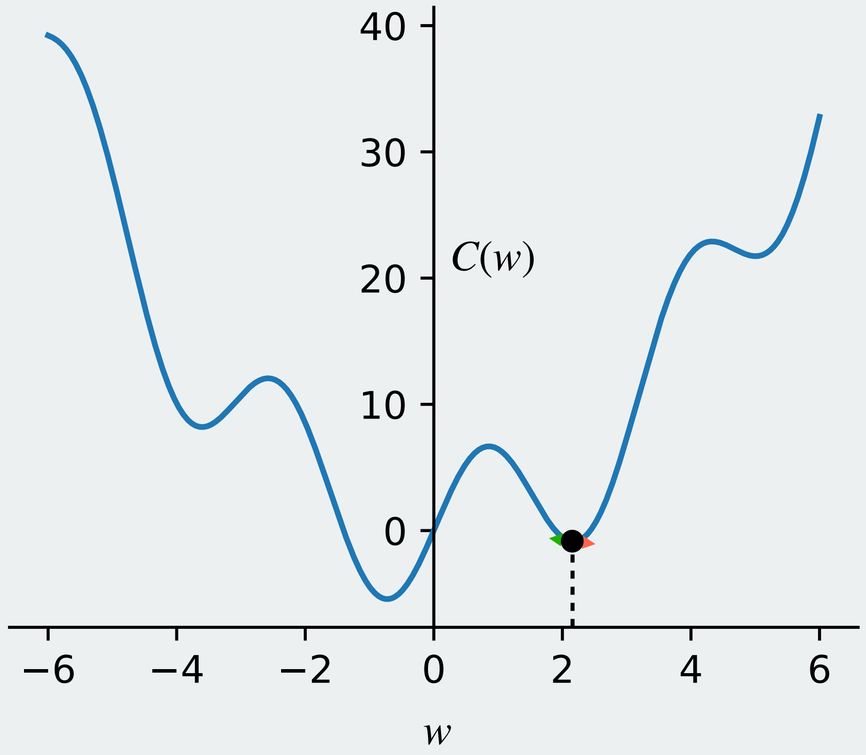
> Gradient Descent



$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.96 \\ 0.08 \\ 0.02 \\ \cdots \\ 0.12 \\ 0.96 \\ 0.01 \end{bmatrix}$$

How do we find the weights that give the best classification?

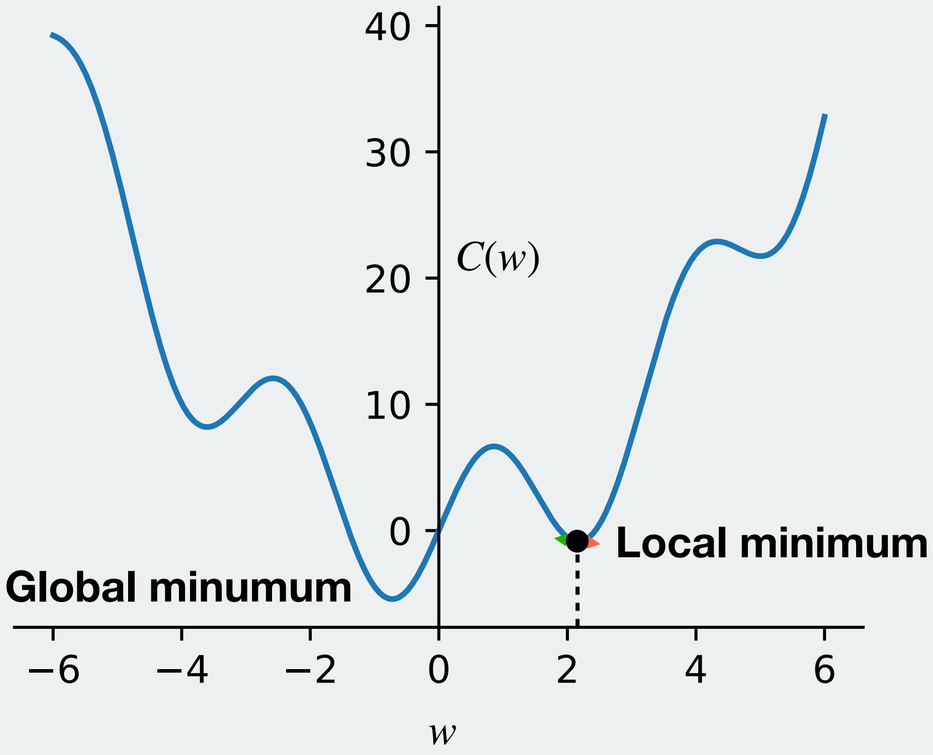
> Gradient Descent



$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.99 \\ 0.02 \\ 0.00 \\ \cdots \\ 0.01 \\ 1.00 \\ 0.00 \end{bmatrix}$$

How do we find the weights that give the best classification?

> Gradient Descent

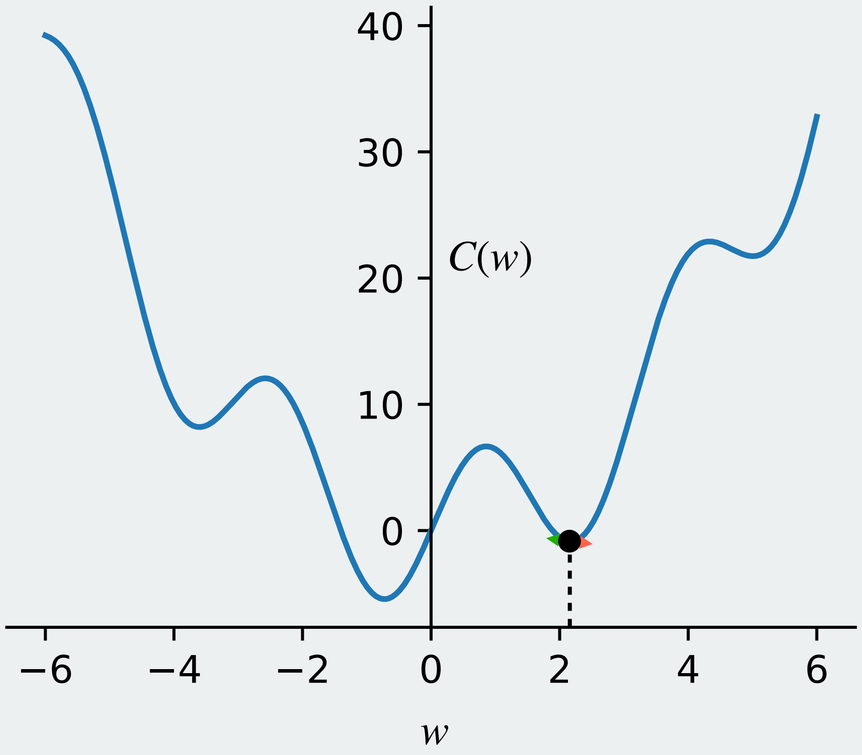


$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.99 \\ 0.02 \\ 0.00 \\ \cdots \\ 0.01 \\ 1.00 \\ 0.00 \end{bmatrix}$$

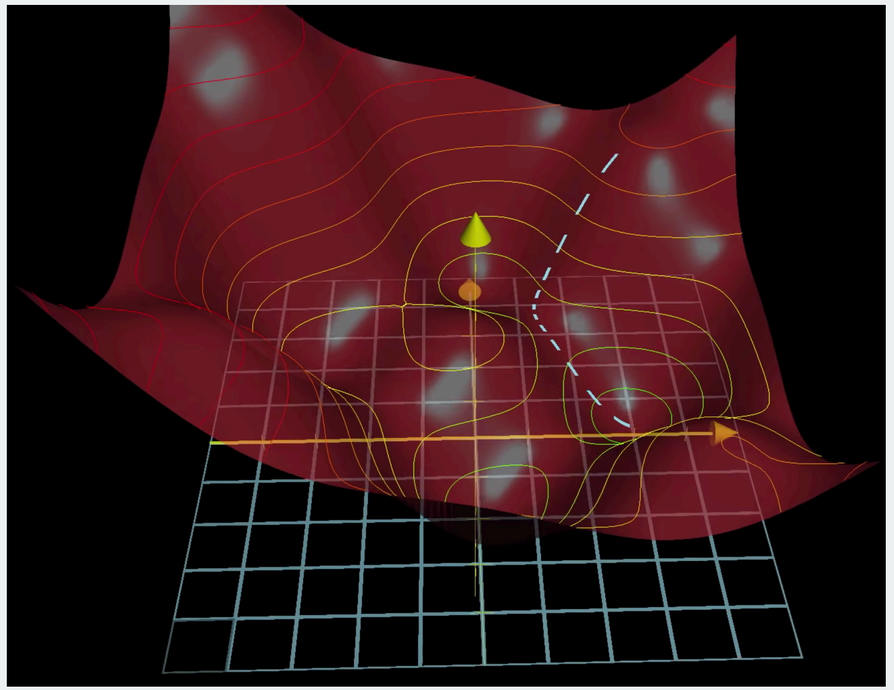
How do we find the weights that give the best classification?

> Gradient Descent

1D



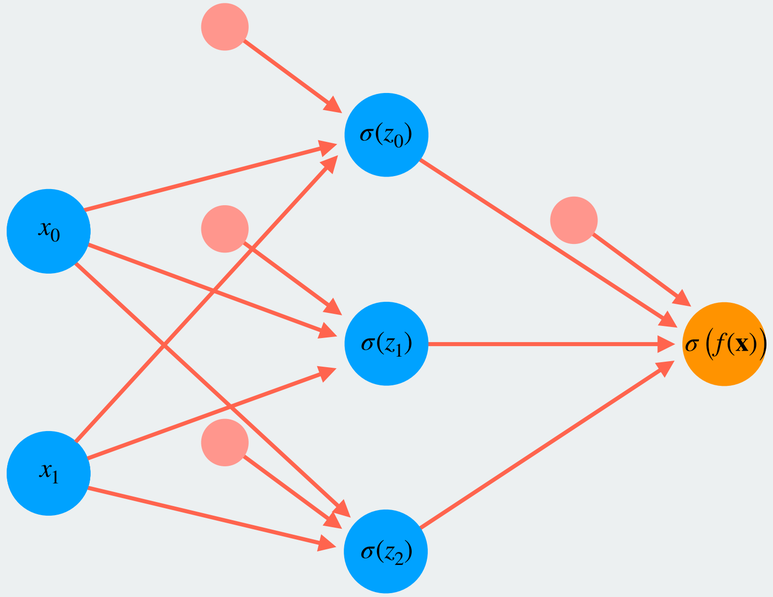
2D



How do we find the weights that give the best classification?

> Gradient Descent

13D? Or higher?

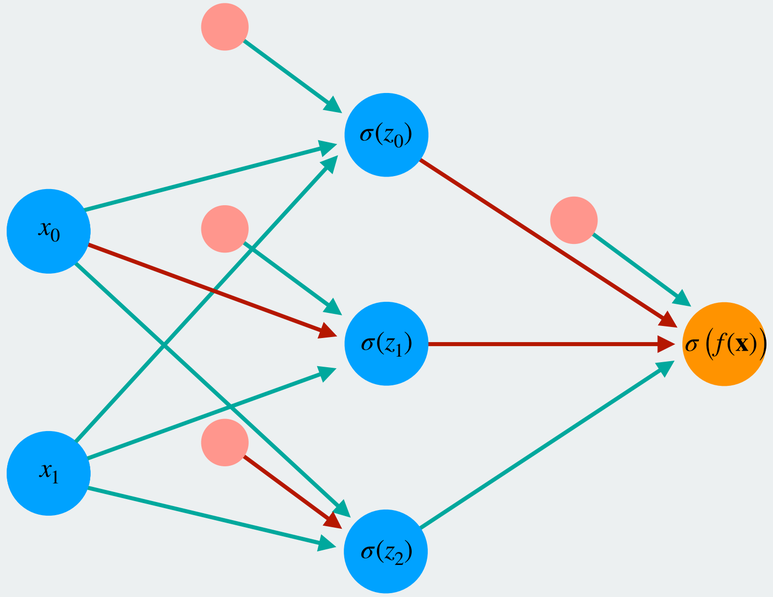


$$\mathbf{W} = \begin{bmatrix} w_{0,0} \\ w_{1,0} \\ w_{2,0} \\ w_{3,0} \\ w_{4,0} \\ w_{5,0} \\ w_{6,0} \\ w_{7,0} \\ w_{8,0} \\ w_{0,1} \\ w_{1,1} \\ w_{2,1} \\ w_{3,1} \end{bmatrix}$$

How do we find the weights that give the best classification?

> Gradient Descent

13D? Or higher?

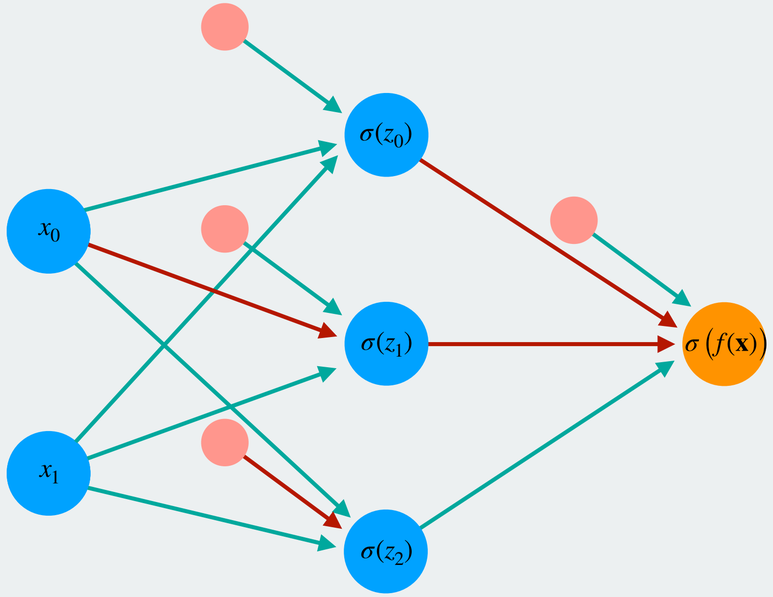


$$\mathbf{W} = \begin{bmatrix} w_{0,0} \\ w_{1,0} \\ w_{2,0} \\ w_{3,0} \\ w_{4,0} \\ w_{5,0} \\ w_{6,0} \\ w_{7,0} \\ w_{8,0} \\ w_{0,1} \\ w_{1,1} \\ w_{2,1} \\ w_{3,1} \end{bmatrix} \qquad -\nabla C(\mathbf{W}) = \begin{bmatrix} -0.23 \\ 1.32 \\ 0.20 \\ 0.38 \\ -1.23 \\ 0.01 \\ 1.20 \\ -2.12 \\ 0.73 \\ 2.17 \\ -0.23 \\ -0.93 \\ 0.45 \end{bmatrix}$$

How do we find the weights that give the best classification?

> Gradient Descent

13D? Or higher?



$$\mathbf{W} = \begin{bmatrix} w_{0,0} \\ w_{1,0} \\ w_{2,0} \\ w_{3,0} \\ w_{4,0} \\ w_{5,0} \\ w_{6,0} \\ w_{7,0} \\ w_{8,0} \\ w_{0,1} \\ w_{1,1} \\ w_{2,1} \\ w_{3,1} \end{bmatrix} \qquad -\nabla C(\mathbf{W}) = \begin{bmatrix} -0.23 \\ 1.32 \\ 0.20 \\ 0.38 \\ -1.23 \\ 0.01 \\ 1.20 \\ -2.12 \\ 0.73 \\ 2.17 \\ -0.23 \\ -0.93 \\ 0.45 \end{bmatrix}$$

$$\mathbf{W} = \mathbf{W}^{\mathbf{old}} + r\left(-\nabla C(\mathbf{W}^{\mathbf{old}})\right)$$

How it all hangs together

(1) The model

Input → Neural network function → **Output**

Datapoints, **X**

Parameters: Weights, **W**

Prediction, $\tilde{\mathbf{y}}$

(2) Its performance

Input → Cost function → **Output**

Weights, **W**

Many training examples

One number, **the cost**

(3) The cost function gradient in **W**

$$-\nabla C(\mathbf{W}) = \begin{bmatrix} -0.23 \\ 1.32 \\ 0.20 \\ 0.38 \\ -1.23 \\ 0.01 \\ 1.20 \\ -2.12 \\ 0.73 \\ 2.17 \\ 0.54 \\ -0.23 \\ -0.93 \\ 0.45 \end{bmatrix}$$

(4) Updating **W**

$$\mathbf{W} = \mathbf{W}^{\mathbf{old}} + r\left(-\nabla C(\mathbf{W}^{\mathbf{old}})\right)$$

(5) Repeat 3 and 4

r is usually called the *learning rate*

How it all hangs together

Find the gradients with **Backpropagation** … next week

(1) The model

Input → Neural network function → **Output**

Datapoints, **X**

Parameters: Weights, **W**

Prediction, $\tilde{\mathbf{y}}$

(2) Its performance

Input → Cost function → **Output**

Weights, **W**

Many training examples

One number, **the cost**

(3) The cost function gradient in **W**

$$-\nabla C(\mathbf{W}) = \begin{bmatrix} -0.23 \\ 1.32 \\ 0.20 \\ 0.38 \\ -1.23 \\ 0.01 \\ 1.20 \\ -2.12 \\ 0.73 \\ 2.17 \\ 0.54 \\ -0.23 \\ -0.93 \\ 0.45 \end{bmatrix}$$

(4) Updating **W**

$$\mathbf{W} = \mathbf{W}^{\mathbf{old}} + r\left(-\nabla C(\mathbf{W}^{\mathbf{old}})\right)$$

(5) Repeat 3 and 4

r is usually called the *learning rate*