Supervised learning

Magnus Nielsen, SODAS, UCPH

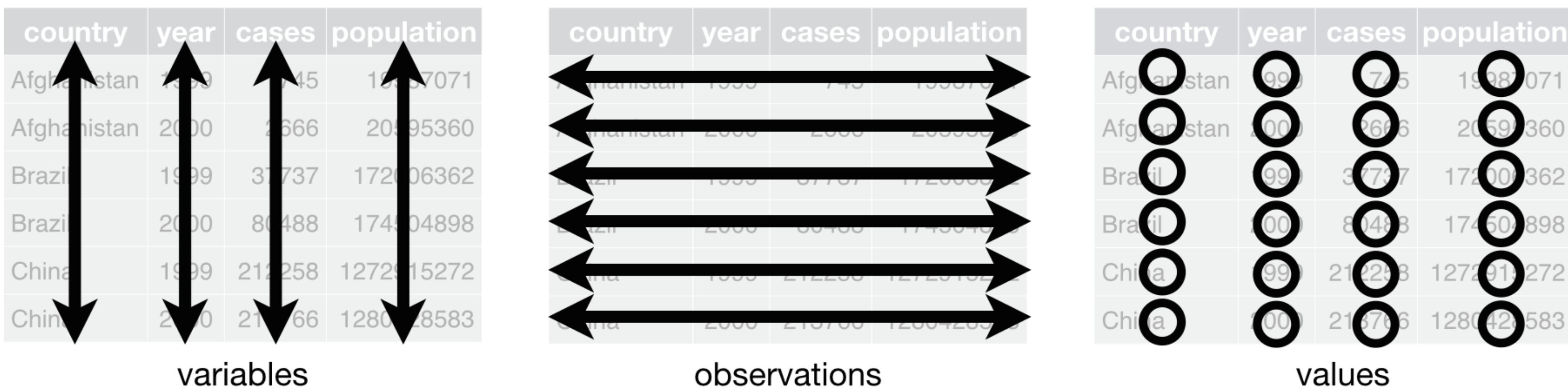
Agenda

- Supervised learning
- Regression & classification
- Tree methods
- Ensemble methods
  - Bagging
  - Boosting
- Neural networks

Supervised learning

Has a given target and uses structured datasets

- Every column is a variable
- Every row is an observation
- Every cell is a single value



A superficial tour of supervised learning

Focus on two things:

- Intuition about how these different models work
- Information about what hyperparameter(s) are key in each model
  - Most importantly, you will learn how to handle overfitting

I will give a tips and tricks summary for each model

I won’t go much into the math

Regression & classification

Question?

What’s the difference between regression & classification?

Regression

Target is a continuous value

Common in many situations:

- What is the income level of this person?
- How much wellbeing?
- Aggregates of classification tasks
    - What percentage will graduate?

How to measure performance

You probably know a lot of them already:

- Mean squared error
- Mean absolute error
- Mean absolute percentage error
- R^2

Often changes how models weigh extreme values

Classification

Categorical outcomes, which can be both binary and multiclass

Common in many situations:

- Who will commit a crime
- Who will graduate
- Who will become sick
  - With what sickness (multiclass)
- In which group does this instance belong

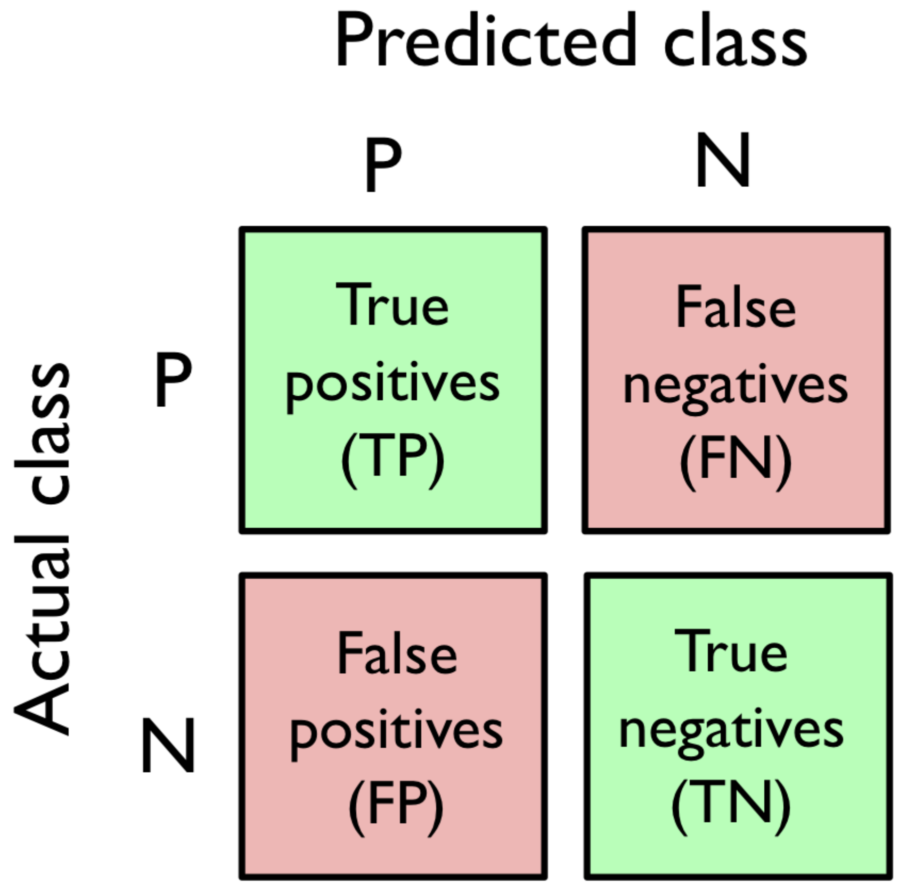
How to measure performance?

An intuitive starting point would be the accuracy of the classifier, i.e. what percentage of observations were correctly classified

$$Accuracy = \frac{True}{True + False}$$

We need to operationalize this mathematically

Introducing the confusion matrix



Binary confusion matrix

Source: Raschka & Mirjalili, 2022, ch. 6

A mathematical starting point

Rewriting accuracy using the confusion matrix

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

This is a fraction between 0 and 1

But it depends on the scenario

Could use precision if not classifying negative labels as positive is important

$$\frac{TP}{TP + FP}$$

Interpretation: ‘How many of the predicted positive labels are correct’

Could use recall if not missing positive is important

$$\frac{TP}{TP + FN}$$

Interpretation: ‘How many of the actual positive labels did we correctly classify’

`sklearn` has a whole list of performance metrics you can easily use, where other common ones are the F1 score and the ROC AUC (AUROC)

It matters

The results can be very dependent on the metric chosen, especially if classes are imbalanced

- Models can and will sometimes figure out that the best accuracy is achieved by always predicting the majority class

Tips:

- Use a baseline model
  - For classification, could be a model which always predict majority class
  - For regression, a model which predicts mean/median
- Look at the confusion matrix

Logistic Regression

Not to be forgotten

Linear combination with a logistic/sigmoid activation function to turn it into a probability

$$p(X) = \sigma(f(X, w)) = \frac{1}{1 + e^{-(Xw)}}$$

Regularized just like Lasso or Ridge models

Regularization implies we need to…

- Standardscale!

Tips and tricks

`sklearn.linear_model.LogisticRegression` for classification

- `penalty` specifies type of regularization
    - e.g. `l1` for absolute values, `l2` for squared values
- `C` specificies regularization strength
    - Lower values reduce overfitting

Tree methods

Decision tree

Split the data up into different groups one or more times, ending up with a flowchart!

Why?

- Good at capturing non-linear relationships
- Easily understandable (more about this in explainability session)

Why not?

- Bad at extrapolation
- I just like linear models

An example

The dataset

	Alcoholic mother	Born in Copenhagen	Becomes criminal
	1	1	1
	1	1	1
Yes	1	1	1
	1	1	1
	1	1	1
	0	1	0
	0	1	0
	0	0	0
	0	0	0
No	0	0	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0

Features/
Explanatory variables

Target/model outcome

Split the data into sections

The dataset

	Alcoholic mother	Born in Copenhagen	Becomes criminal
Yes	1	1	1
	1	1	1
	1	1	1
	1	1	1
	1	1	1
No	0	1	0
	0	1	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0

Features/
Explanatory variables

Target/model outcome

Model

How to split?

Split using intuition

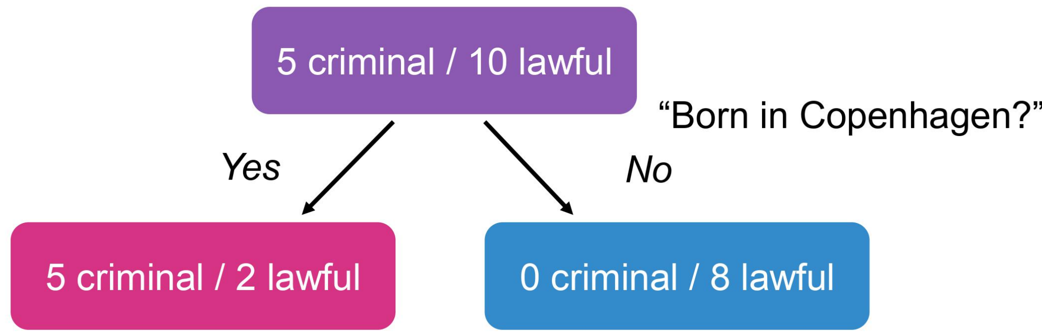
<u>The dataset</u>

	Alcoholic mother	Born in Copenhagen	Becomes criminal
	1	1	1
	1	1	1
Yes	1	1	1
	1	1	1
	1	1	1
	0	1	0
	0	1	0
	0	0	0
	0	0	0
No	0	0	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0

Features/
Explanatory variables

Target/model outcome

<u>Model</u>



Two new datasets

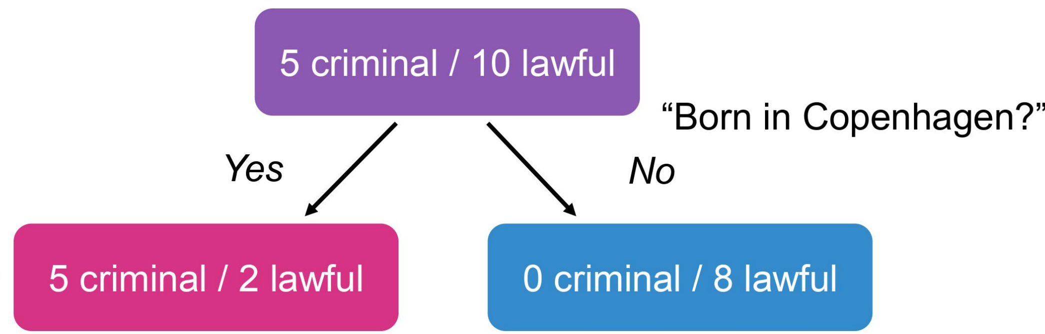
<u>The dataset</u>

	Alcoholic mother	Born in Copenhagen	Becomes criminal
Yes	1	1	1
	1	1	1
	1	1	1
	1	1	1
	1	1	1
	0	1	0
	0	1	0
No	0	0	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0

Features/
Explanatory variables

Target/model outcome

<u>Model</u>



Split nodes which are still impure

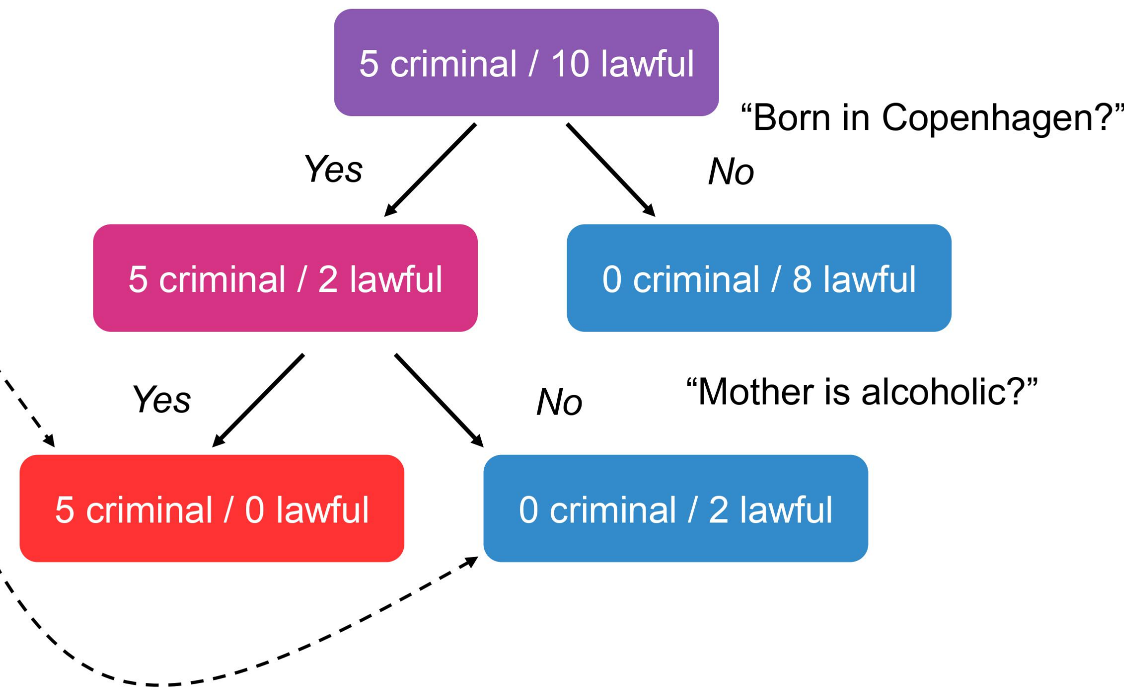
The dataset

	Alcoholic mother	Born in Copenhagen	Becomes criminal
Yes	1	1	1
	1	1	1
	1	1	1
	1	1	1
	1	1	1
No	0	1	0
	0	1	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0

Features/
Explanatory variables

Target/model outcome

Model



A better split

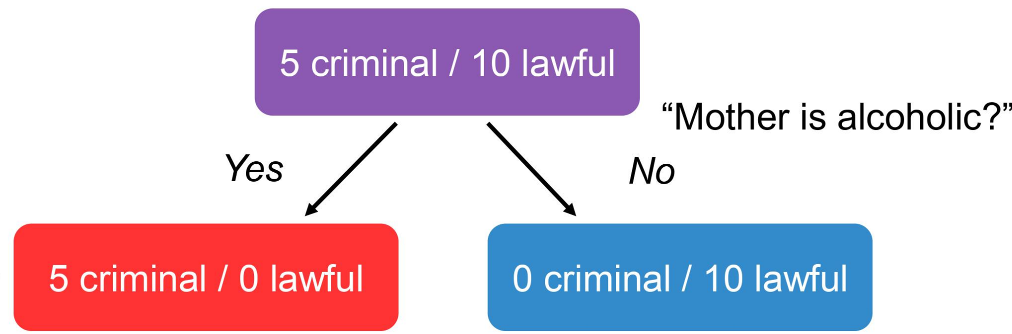
The dataset

	Alcoholic mother	Born in Copenhagen	Becomes criminal
Yes	1	1	1
	1	1	1
	1	1	1
	1	1	1
	1	1	1
No	0	1	0
	0	1	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0

Features/
Explanatory variables

Target/model outcome

Model



How to mathematically find best split

To make an algorithm, we need a measure of ‘goodness’ of fit

For classification we use separation of classes as a measure

- Commonly measured using Gini Impurity or Shannon entropy
- Calculate a weighted average, go with the split with lowest impurity/error

For regression, we use metrics such as mean squared error or mean absolute error

Shannon entropy

General formula

$P = [p, 1 - p]$
$\text{Entropy} = -(p \cdot \log(p) + (1 - p) \cdot \log(1 - p))$

0 criminal / 8 lawful

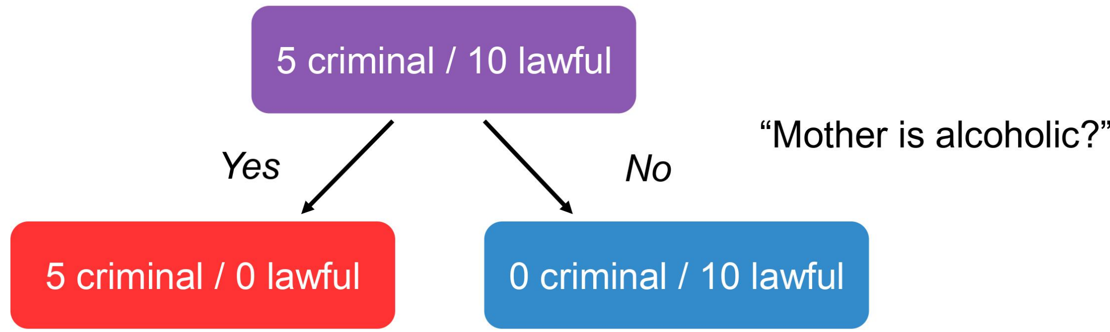
$P = [0, 1]$
$\text{Entropy} = -(0 \cdot \log(0) + 1 \cdot \log(1)) = \mathbf{0}$

5 criminal / 2 lawful

$P = [2/7, 5/7]$
$\text{Entropy} = -(2/7 \cdot \log(2/7) + 5/7 \cdot \log(5/7)) = \mathbf{0.59}$

Entropy of alcoholic mothers

<u>Split 1:</u>



$$\text{Split entropy} = 5/15 \cdot \mathbf{0} + 10/15 \cdot \mathbf{0} = \mathbf{0.00}$$

Check other splits

<u>Split 1:</u>

5 criminal / 10 lawful

"Mother is alcoholic?"

Yes | *No*

5 criminal / 0 lawful | 0 criminal / 10 lawful

$$\text{Split entropy} = \quad 5/15 \cdot \mathbf{0} \quad + \quad 10/15 \cdot \mathbf{0} \quad = \mathbf{0.00}$$

<u>Split 2:</u>

5 criminal / 10 lawful

"Born in Copenhagen?"

Yes | *No*

5 criminal / 2 lawful | 0 criminal / 8 lawful

$$\text{Split entropy} = \quad 7/15 \cdot \mathbf{0.59} \quad + \quad 8/15 \cdot \mathbf{0} \quad = \mathbf{0.28}$$

Tips and tricks

`sklearn.tree.DecisionTreeClassifier` for classification
`sklearn.tree.DecisionTreeRegressor` for regression

- `max_depth` specifies how deep the tree is
  - Smaller values reduce overfitting
- `min_samples_split` specifices how many observations need to be in a node for it to allow a split
  - Higher values reduce overfitting
- `min_samples_leaf` specifies how many observations need to be in an end node/leaf for it to be allowed
  - Higher values reduce overfitting

`sklearn` has a write-up on decision trees with more math, tips for practical use and more

Ensemble methods

How to aggregate models

In bagging, we do bootstrap aggregation

- Train many models, which all try to estimate the outcome using bootstrapped samples
- Can be different models, i.e. some which do well at linear relations and others for non-linear relations

In boosting, we ‘boost’ the models sequentially

- Train weak models in succession, where each models tries to increase performance based on previous models output

Bagging

The random forest is a bagging method using many decision trees

- All the knowledge you have about decision trees carry over!

Randomness

The new thing is two sources of randomness:

- for each tree, we randomly select the sample to train on (sampled with replacement, bootstrap)
- at each split, only consider a random subset of features

You need one of these two sources of randomness to avoid growing identical trees

This reduces overfitting, as

- some trees do not see the samples which other models overfit to
- some trees do not see the features in a split which other models used to overfit with

Tips and tricks

`sklearn.ensemble.RandomForestClassifier` for classification
`sklearn.ensemble.RandomForestRegressor` for regression

- All the parameters from decision trees carry over
- `n_estimators` controls how many trees are grown
  - Larger values are generally better, but more computationally expensive
- `max_features` controls how many features are considered at each split
  - Lower values reduce overfitting

Random forests are random, so remember a seed!

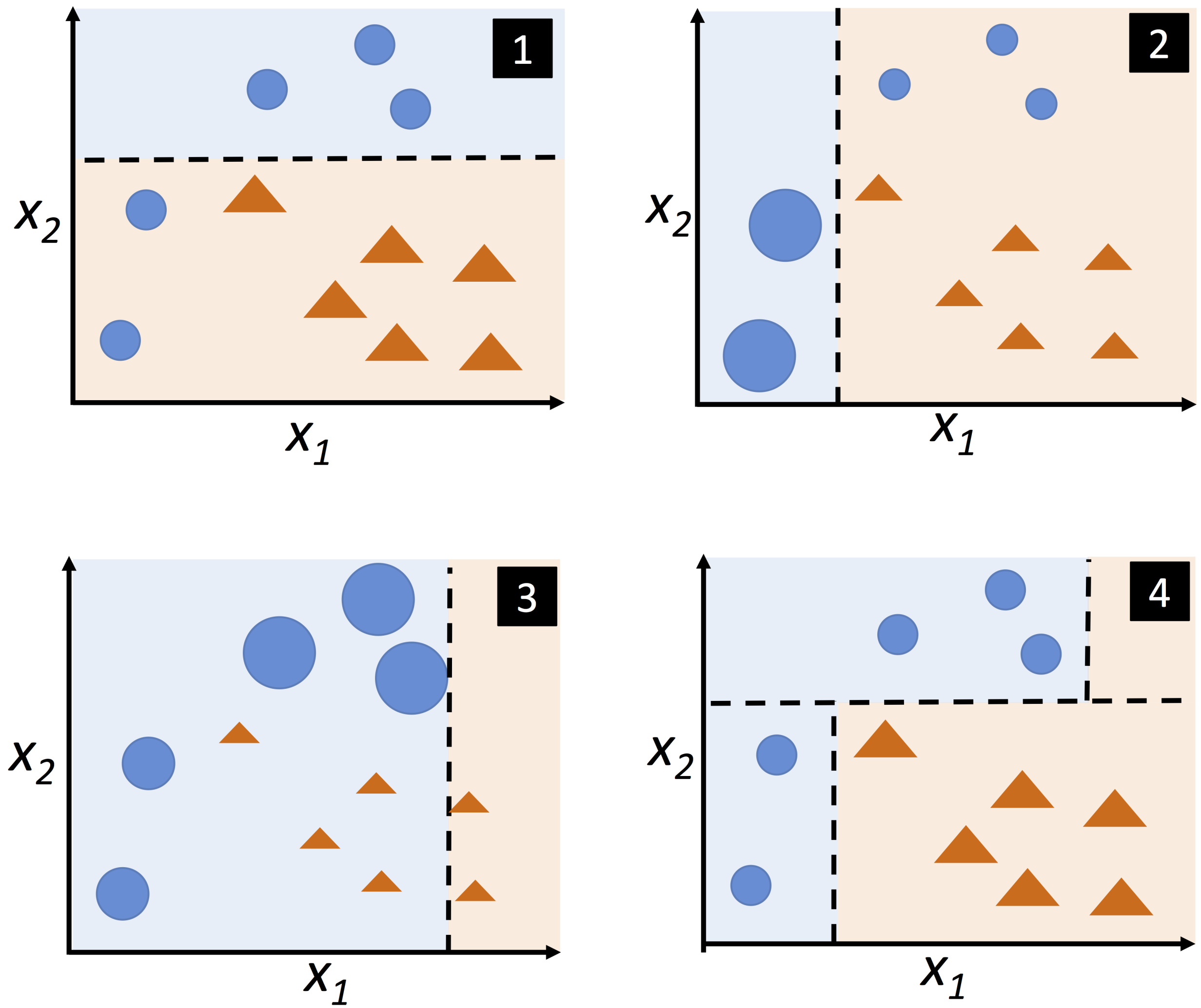
`sklearn` has a short write-up on random forests

Boosting

Use ‘weak learners’ (often decision trees of relatively shallow depth) which are built to correct each others mistakes

The AdaBoost (adaptive boosting) model does this by weighing the misclassified observations (high error in regression) higher in the next weak learning phase

Illustrated



Source: Raschka & Mirjalili, 2022, ch. 7

Tips and tricks

`sklearn.ensemble.AdaBoostClassifier` for classification,
`sklearn.ensemble.AdaBoostRegressor` for regression

- All the parameters from decision trees carry over
  - But designed with weak learners in mind
- `n_estimators` controls how many trees are grown
  - Lower values reduce overfitting
- `learning_rate` controls how much weight is given to misclassified observation
  - Lower values reduce overfitting

`sklearn` has a short write-up on AdaBoost

Neural networks

Neural networks

Neural networks can be extremely complicated

They can be both supervised and unsupervised

As an introduction, we focus on feed forward neural networks

Different models are created for and excel in different areas

- What you want or need to learn depends mostly on the input format
- For images, the literature is called computer vision (CV), for text it’s called natural language processing (NLP)

Feed forward neural networks

Following illustrations are from:

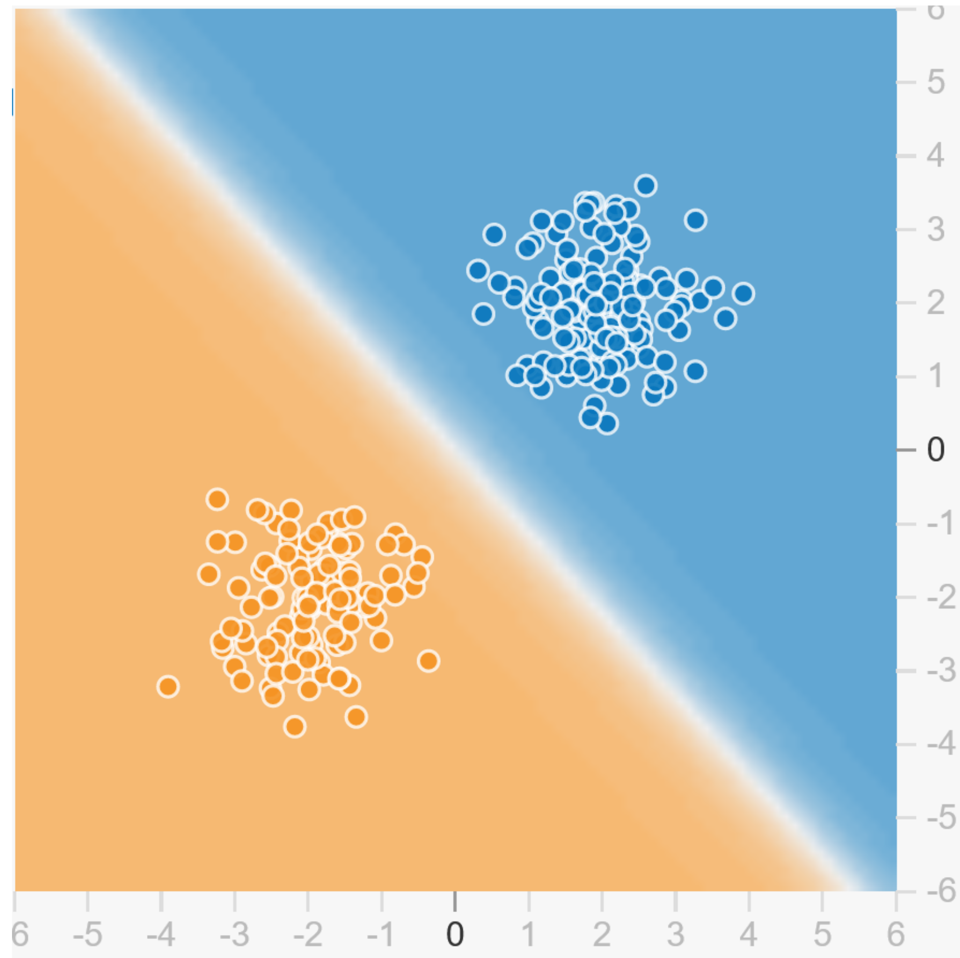
- Ulf Aslak, who used to work at SODAS and taught Social Data Science: Text Data and Deep Learning, and
- Tensorflows Neural Network Playground

In essence, feed forward neural networks are nested logistic regressions (roughly)

Back to logistic regression

Linear combination with an activation function (sigmoid/logistic function, σ)

$$\sigma(f(X, w)) = \frac{1}{1 + e^{-(Xw)}}$$



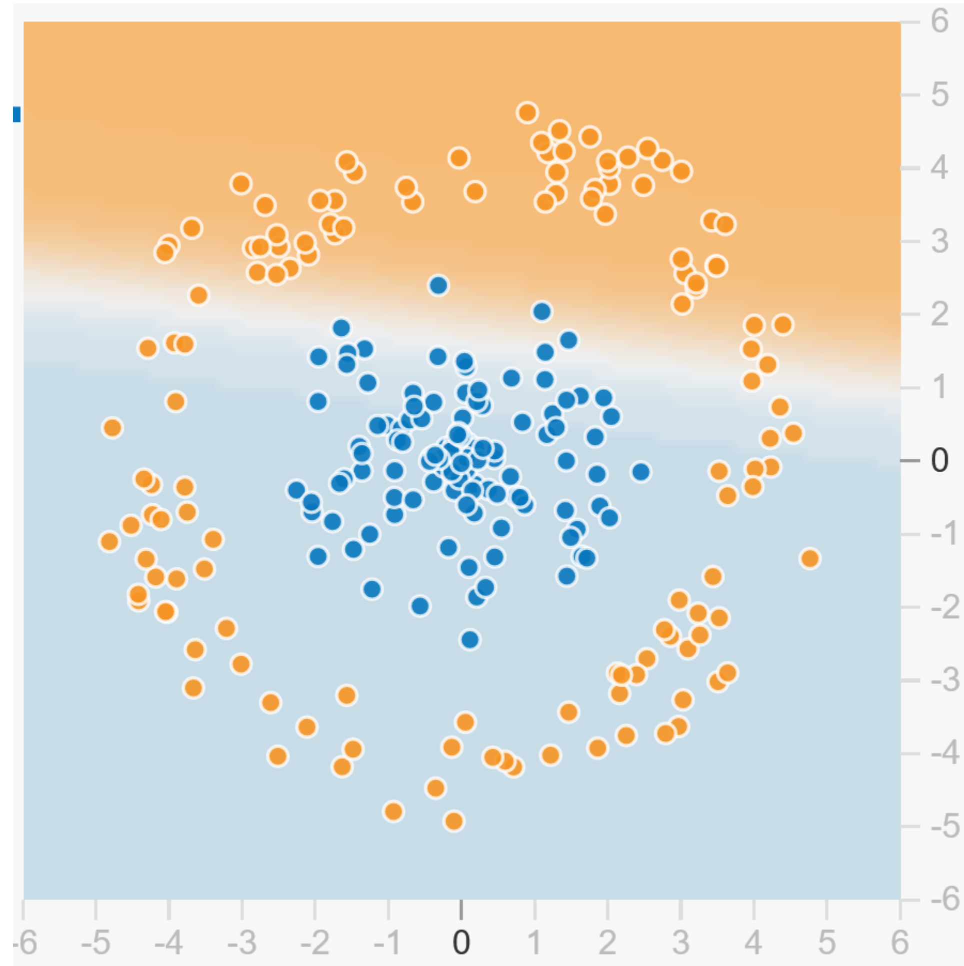
It's… linear

Not good at non-linear relationships

Would require input transformations

- Not always feasible

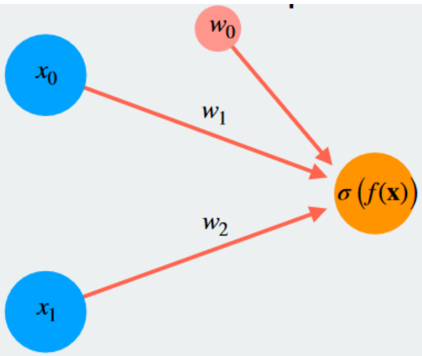
What if we limited ourselves to inputting just X_1 and X_2?



Question

How many logistic regressions do you think it would take to approximately separate the two groups?

First logistic regression



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

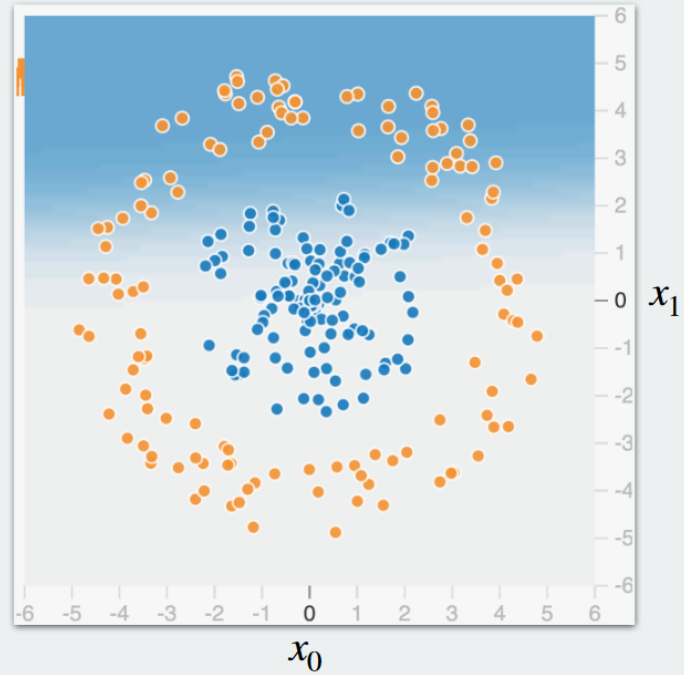
$\mathbf{X} =$

x_0	x_1
0.5	1.5
2.3	-1.7
...	...
4.2	-0.2
-1.9	2.3

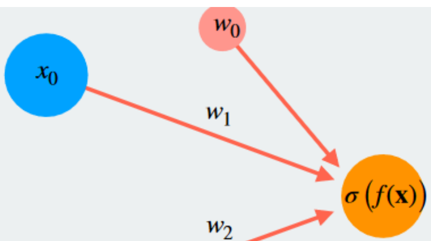
,

$z_0(\mathbf{X}) =$

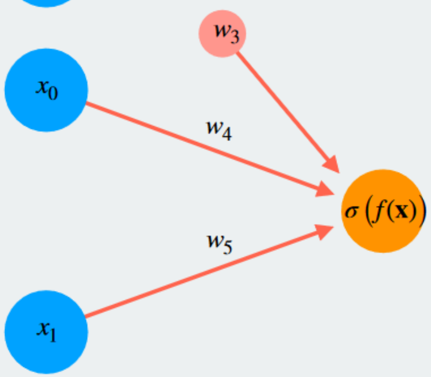
z_0
0.3
0.25
...
0.79
0.34



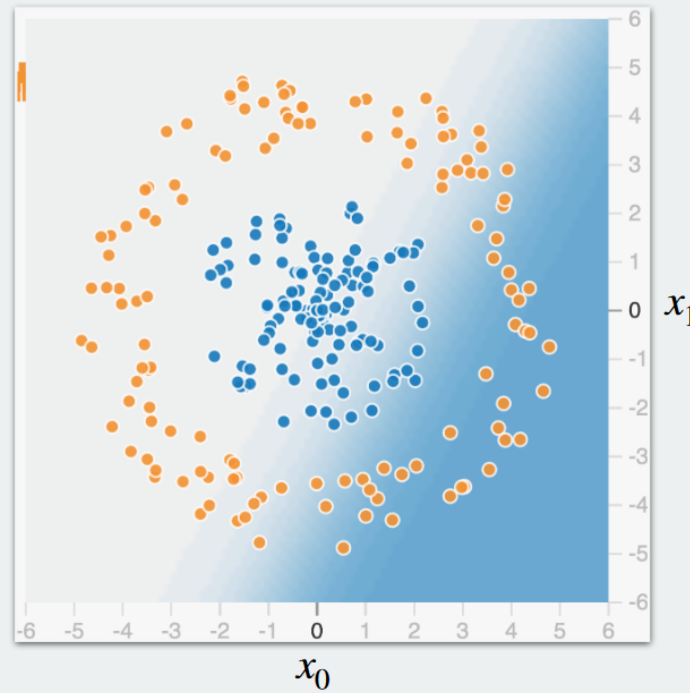
Second logistic regressions



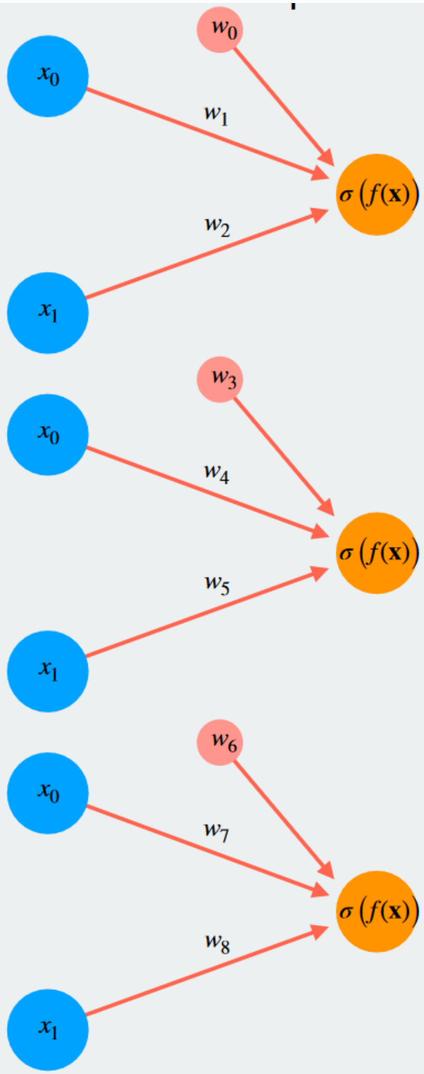
$$\sigma(w_0 + x_0w_1 + x_1w_2) = z_0(\mathbf{x})$$



$$\sigma(w_3 + x_0w_4 + x_1w_5) = z_1(\mathbf{x})$$



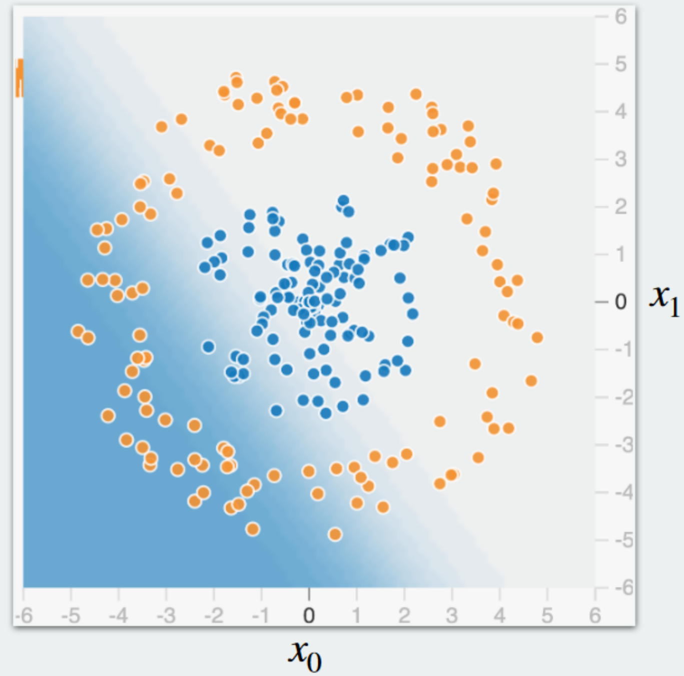
Third logistic regressions



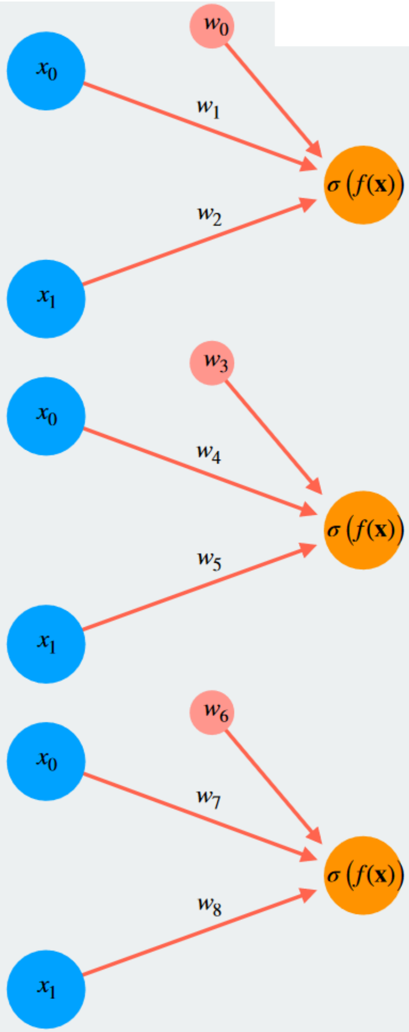
$$\sigma(w_0 + x_0 w_1 + x_1 w_2) = z_0(\mathbf{x})$$

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

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



What now?



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

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

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

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

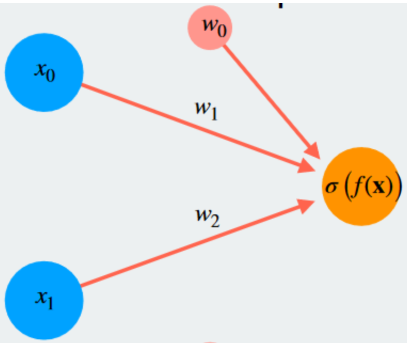
$\mathbf{Z} =$

z_0	z_1	z_2
0.3	0.75	0.78
0.25	0.1	0.95
...	...	
0.79	0.99	0.3
0.34	0.6	0.1

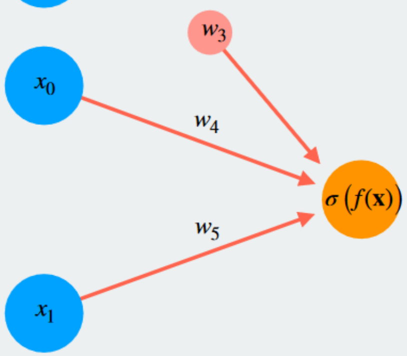
$\mathbf{y} =$

y
0
1
...
1
0

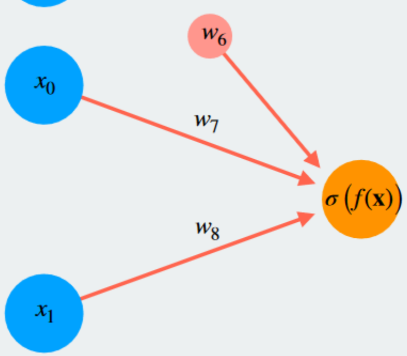
Fourth logistic regression



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



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



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

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

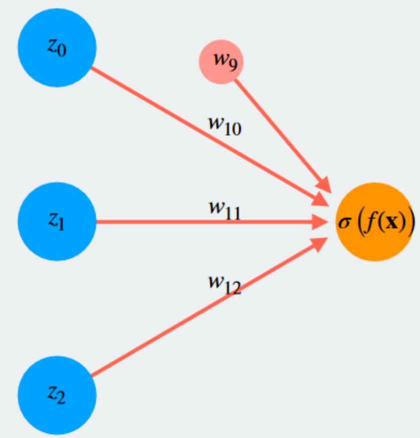
$\mathbf{Z} =$

z_0	z_1	z_2
0.3	0.75	0.78
0.25	0.1	0.95
...	...	
0.79	0.99	0.3
0.34	0.6	0.1

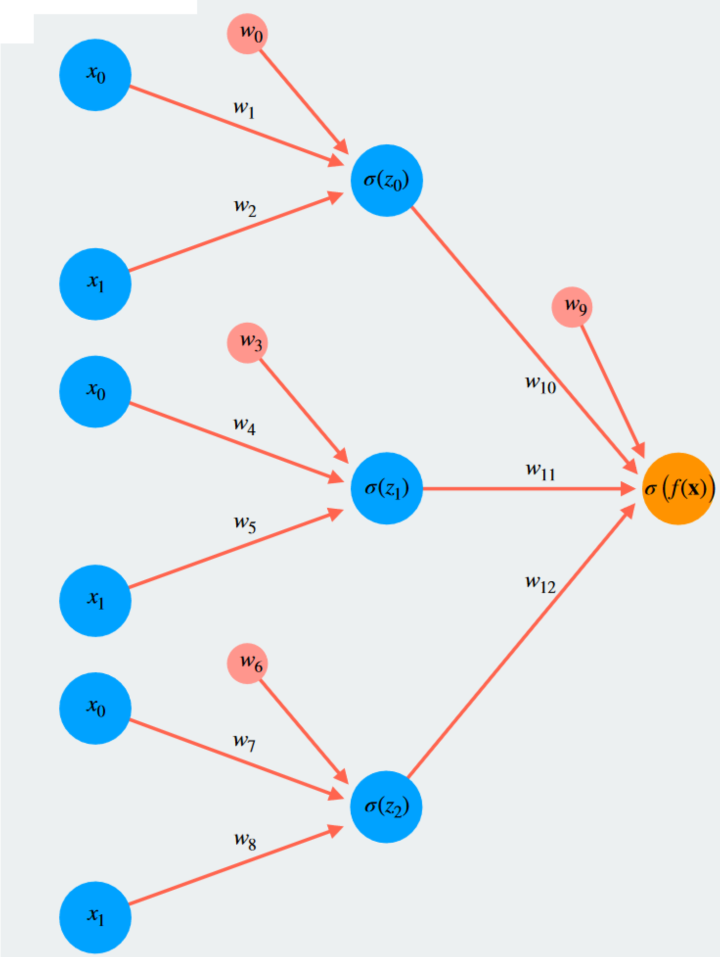
$\mathbf{y} =$

y
0
1
...
1
0

Solution: Why not use a logistic regression?



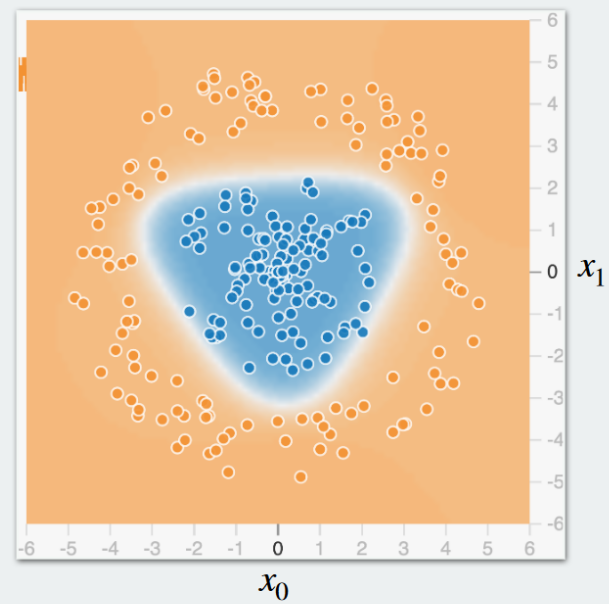
Succes!



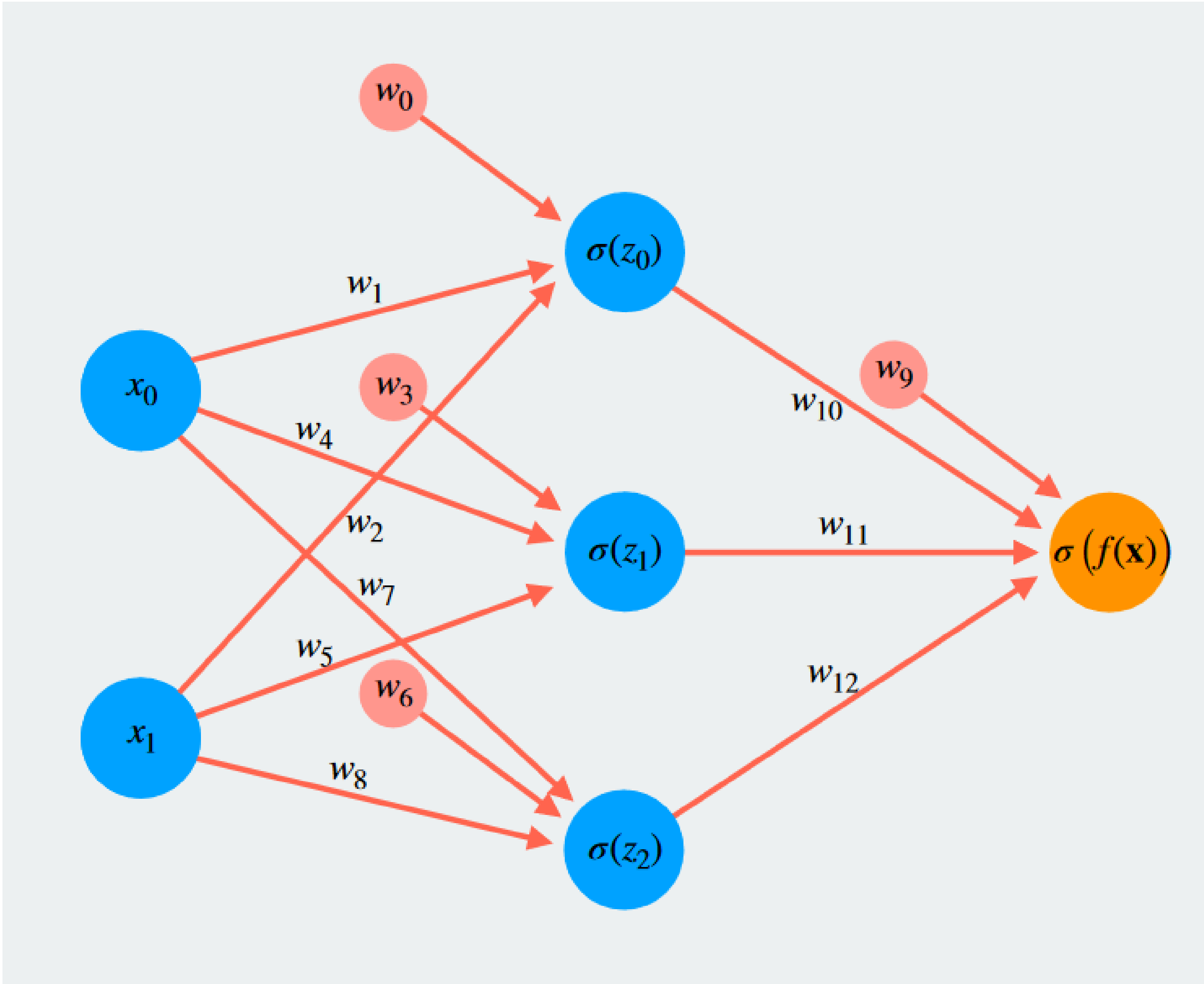
Mathematical form

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

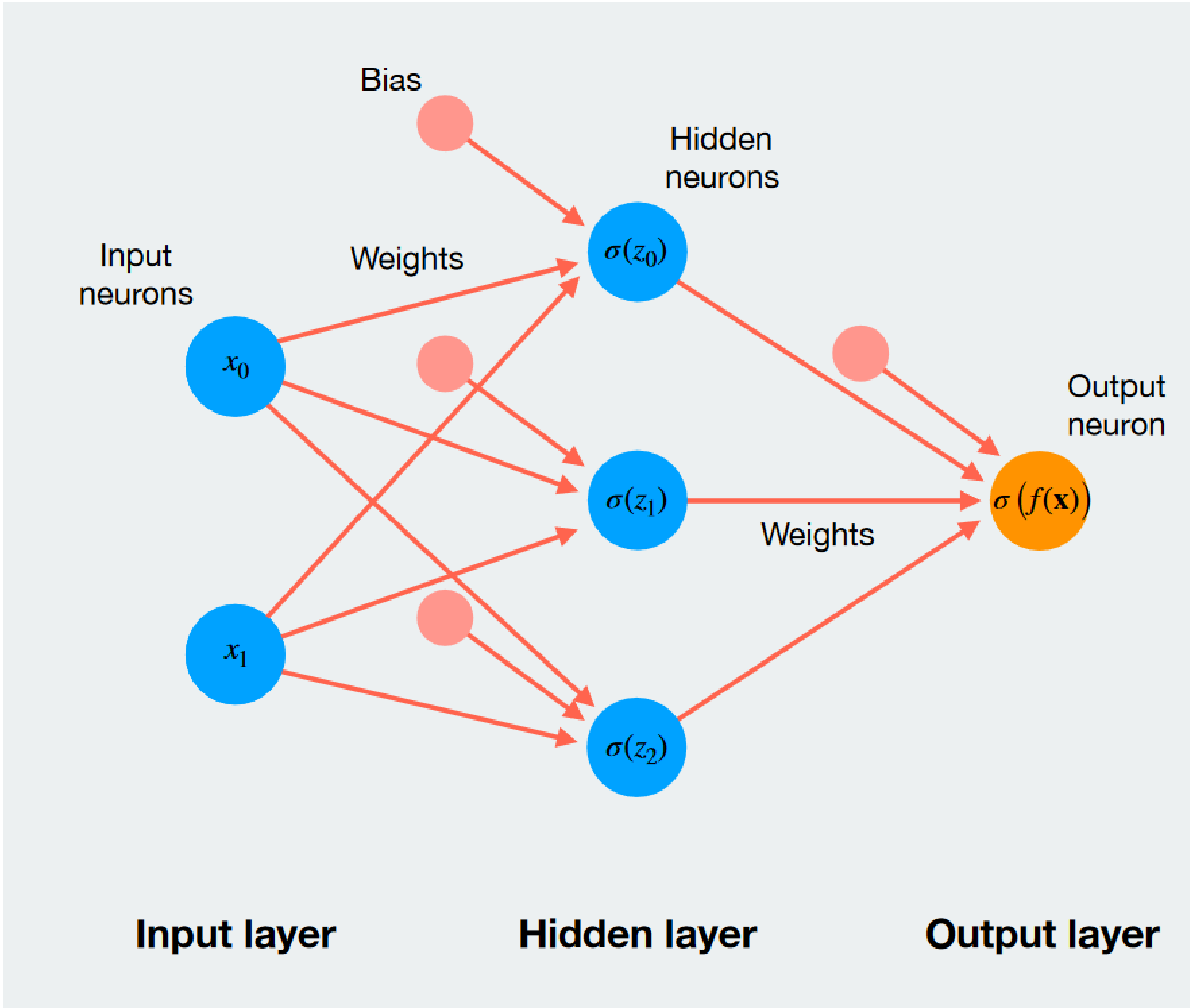
Solution



Compressing the graph



Labelling the graph



The Multilayer Perceptron

A fully connected feed forward neural network is called a Multilayer Perceptron (MLP)

By omitting the last logistic activation, the model is turned into a regression model

- Can also do multiple output neurons for multiple outcomes

The `sklearn` implementation is regularized (L2), so we should…

- Standardscale!

Tips and tricks

`sklearn.neural_network.MLPClassifier` for classification
`sklearn.neural_network.MLPRegressor` for regression

- `alpha` controls the amount of regularization
  - Higher values reduce overfitting
- `hidden_layer_sizes` is a tuple which controls amount of hidden layers and hidden neurons
  - e.g. (6, 7, 8) would give 3 hidden layers with 6, 7 and 8 hidden neurons
  - Fewer of both reduce overfitting
    - Default is one hidden layer with 100 hidden neurons

`sklearn` has a write-up on MLP with more math, tips for practical use and more

References

Raschka, S., Liu, Y. H., Mirjalili, V., & Dzhulgakov, D. (2022). Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python. Packt Publishing Ltd.

To the exercises!