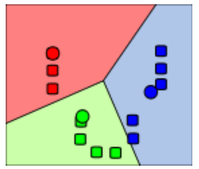
From (chemical) data to information

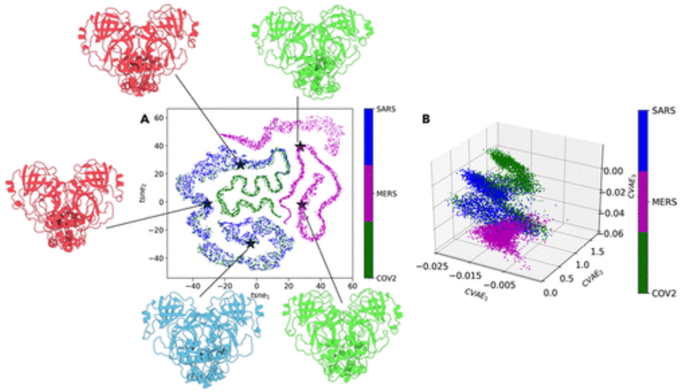
Introduction to Machine learning for Chemistry Lecture 2

Dr Antonia Mey

antonia.mey@ed.ac.uk

Room 214 JBB





Topics overview

Lecture 1

- What is machine learning?
- Examples of machine learning (in Chemistry)
- Introduction to **unsupervised learning**:
  - Clustering (k-means and others)
  - How does actual input data look like?
- Molecular fingerprints and nomenclature
- Introduction to **supervised learning:**
  - What is a classification problem?

Topics overview

Lecture 1

- What is machine learning?
- Examples of machine learning (in Chemistry)
- Introduction to **unsupervised learning**:
  - Clustering (k-means and others)
  - How does actual input data look like?
- Molecular fingerprints and nomenclature
- Introduction to **supervised learning:**
  - What is a classification problem?

Lecture 2

- Regressions
- Unsupervised learning continued:
  - Dimensionality reduction (PCA)
  - t-SNE
- Classifications in practice:
  - Random Forests
  - Multilayer perceptrons

Topics overview

Lecture 1

- What is machine learning?
- Examples of machine learning (in Chemistry)
- Introduction to **unsupervised learning**:
  - Clustering (k-means and others)
  - How does actual input data look like?
- Molecular fingerprints and nomenclature
- Introduction to **supervised learning:**
  - What is a classification problem?

Lecture 2

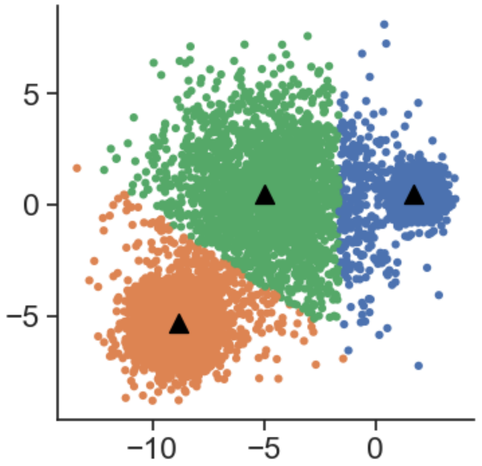
- Regressions
- Unsupervised learning continued:
  - Dimensionality reduction (PCA)
  - t-SNE
- Classifications in practice:
  - Random Forests
  - Multilayer perceptrons

Learning outcomes:

- Understand the main pillars of machine learning
- Know about different clustering techniques as part of unsupervised learning
- Be able to use common nomenclature used in machine learning
- Use PCA to reduce the dimensions of your data set
- Understand how a regression problem can be cast as a machine learning problem
- Be aware of how random forests and multilayer perceptrons can be used in a classification problem

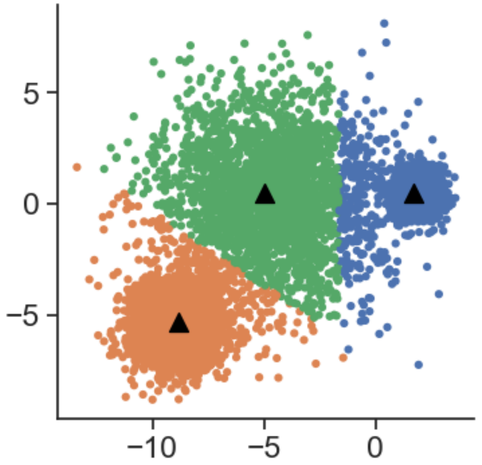
Recall from Lecture 1

Different clustering algorithm identify groups in datasets

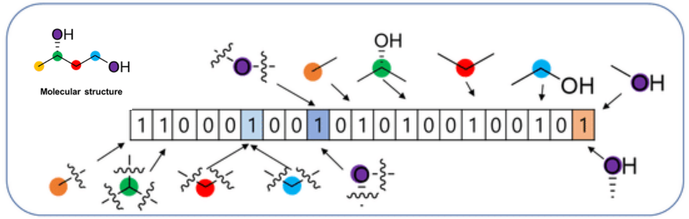


Recall from Lecture 1

Different clustering algorithm identify groups in datasets



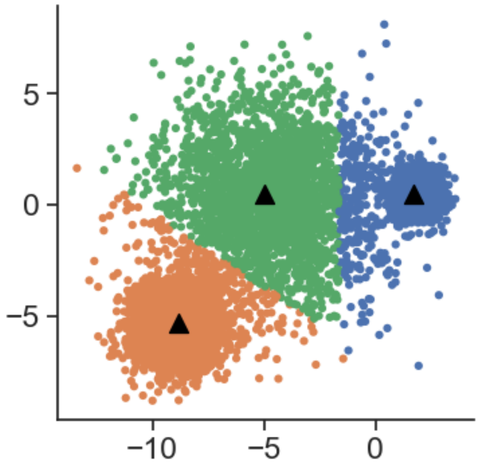
Features are used in input as feature vectors and can take many different forms



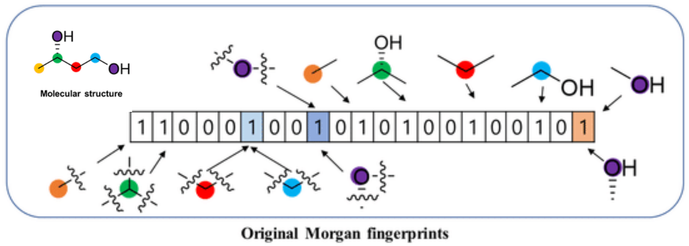
Original Morgan fingerprints

Recall from Lecture 1

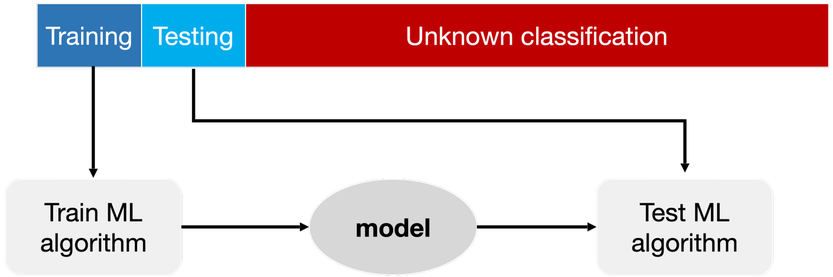
Different clustering algorithm identify groups in datasets



Features are used in input as feature vectors and can take many different forms

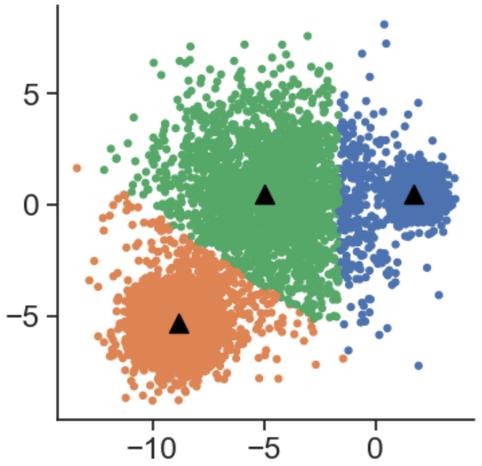


A classification model uses labelled data as input to train a model that can be used as a classifier

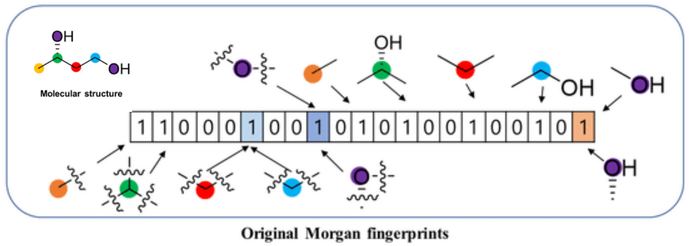


Recall from Lecture 1

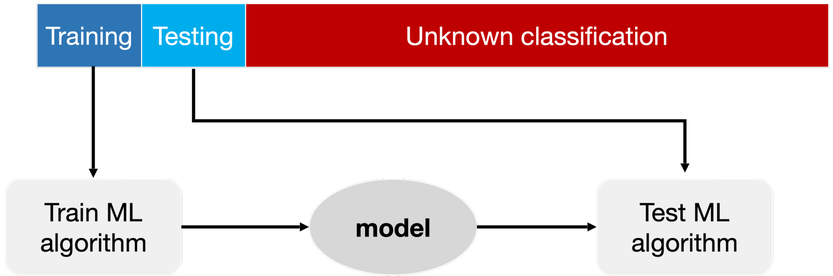
Different clustering algorithm identify groups in datasets



Features are used in input as feature vectors and can take many different forms



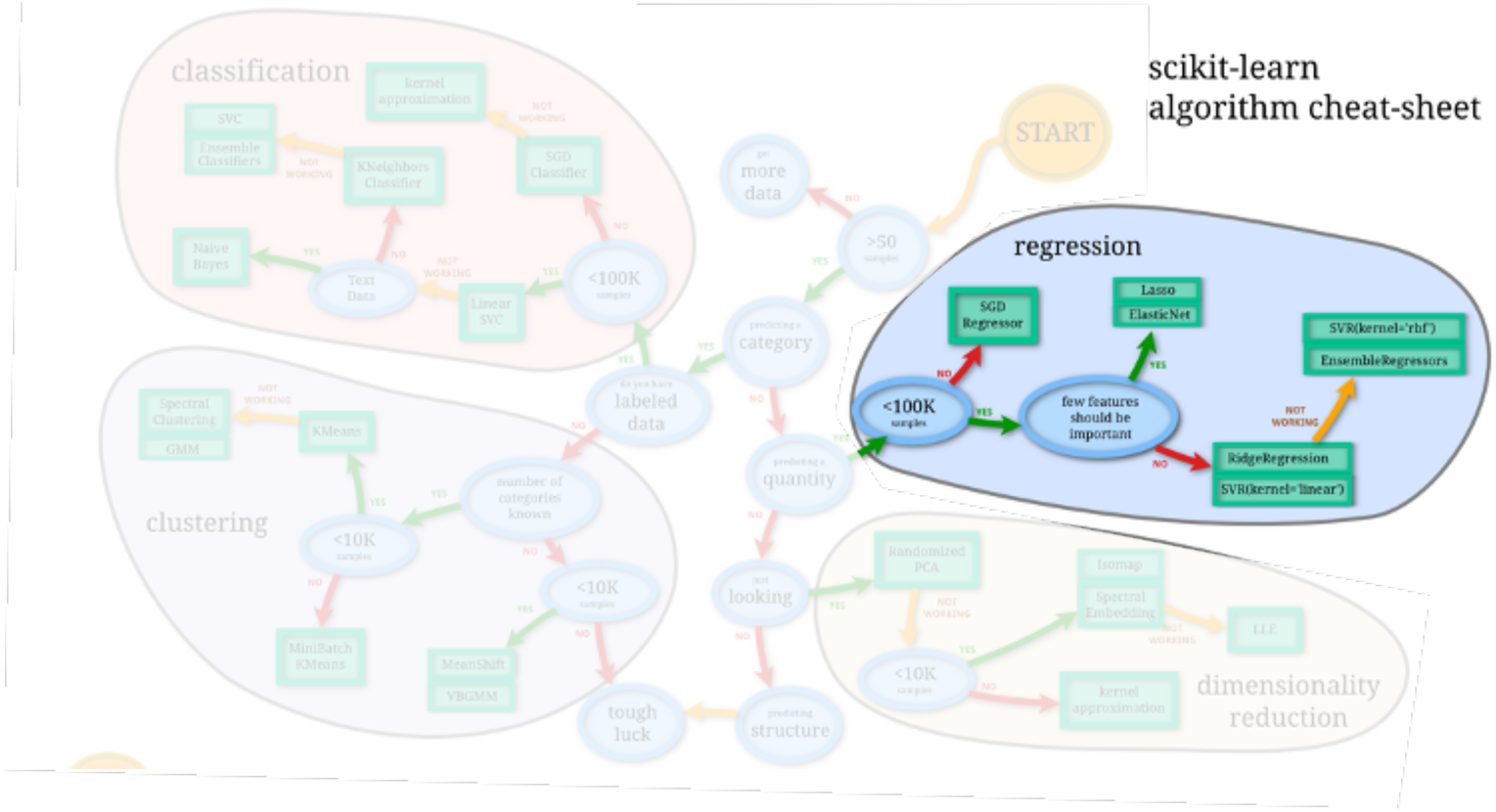
A classification model uses labelled data as input to train a model that can be used as a classifier



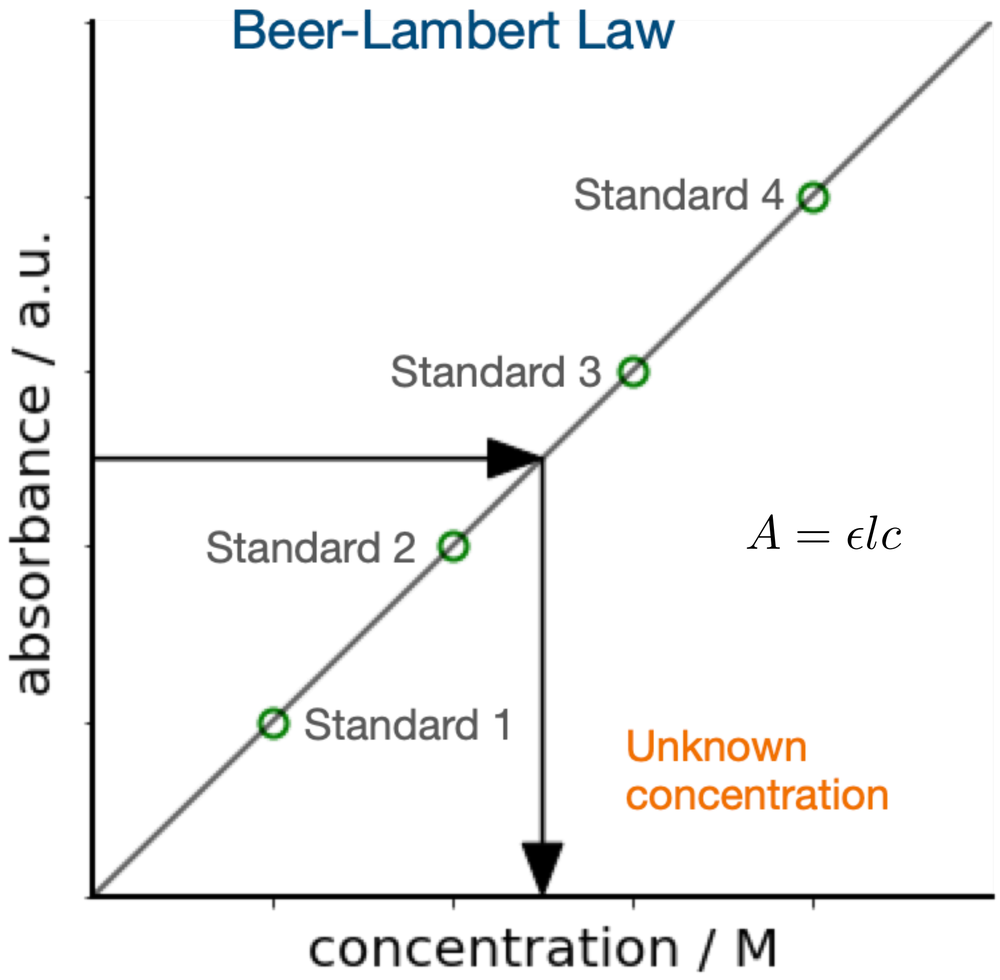
result \ real	Dog	Cat
Dog	90	10
Cat	12	88

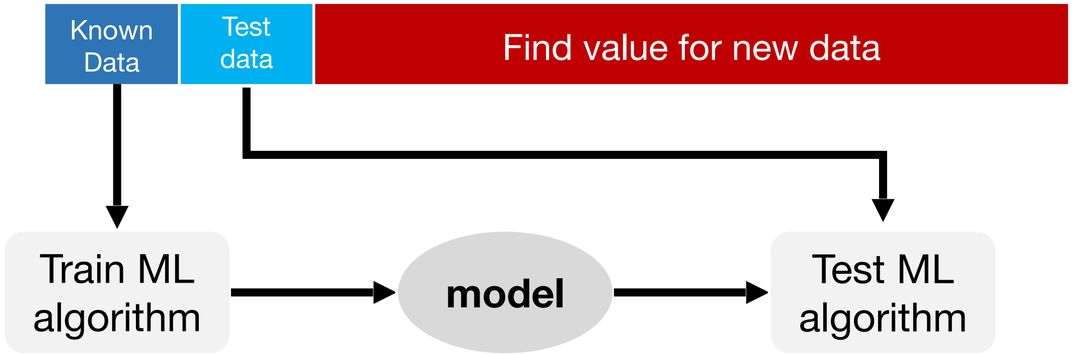
A confusion matrix allows the assessment of how good a model is.

The Data Mining World



Linear regression: an example





Now the model is the line of best fit that will allow us to identify an unknown concentration

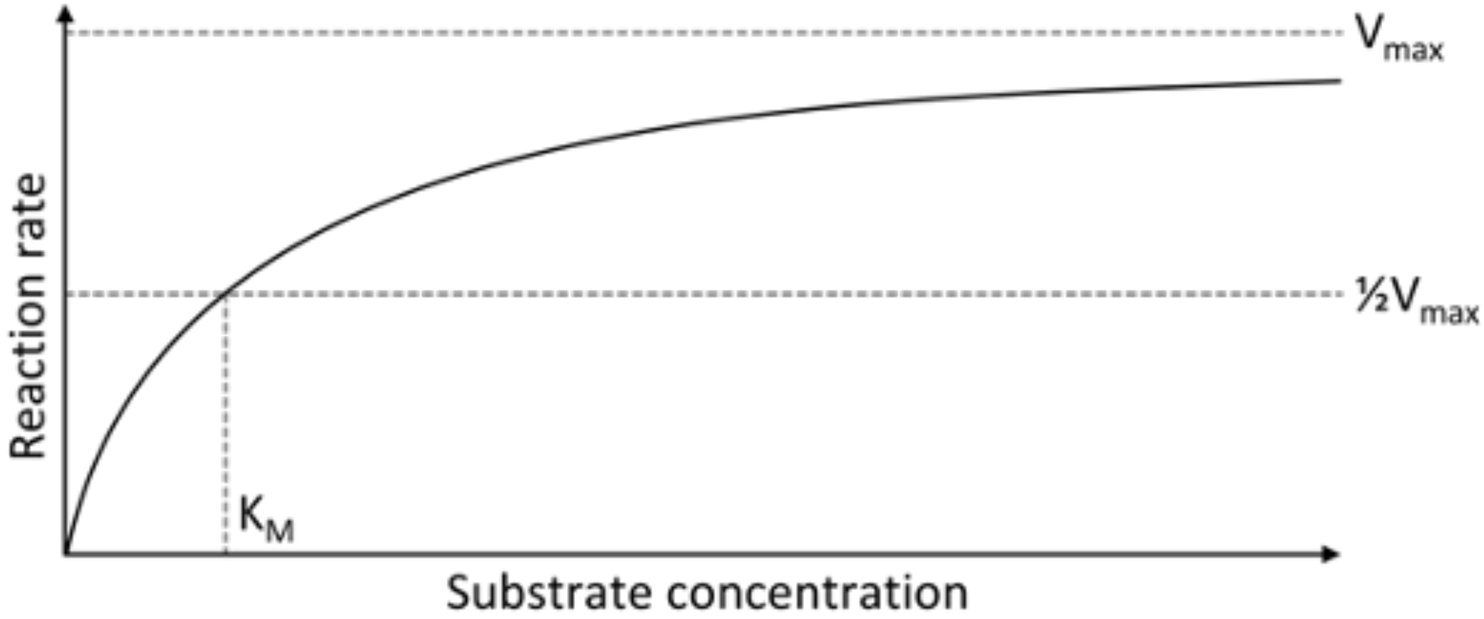
How do we find the line of best fit?

Non-linear Least squares

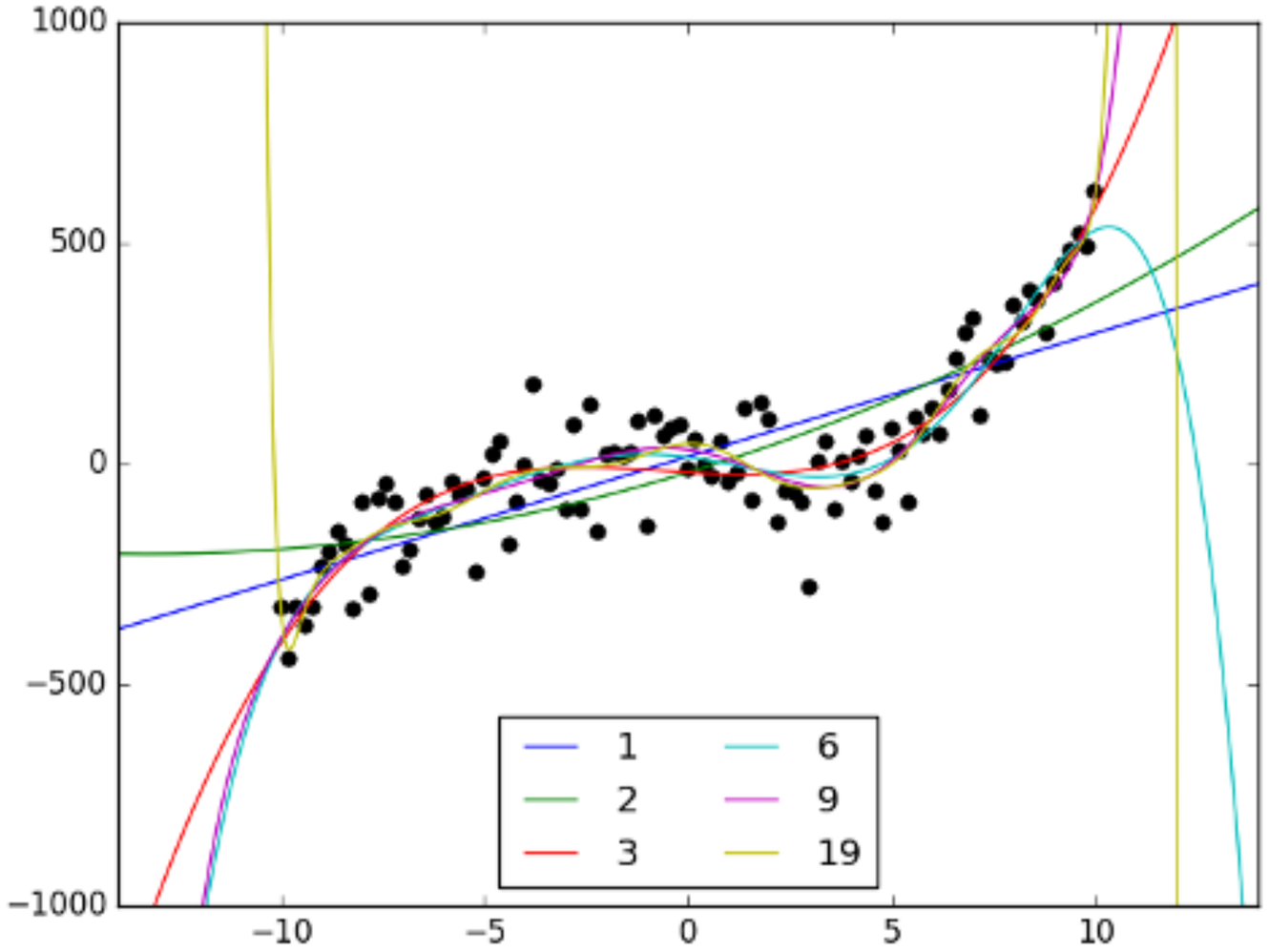
Non-linear combinations of model parameters, e.g. Michaelis Menten

$$\frac{d[P]}{dt} = \frac{V_{\max}[S]}{K_{\mathrm{M}} + [S]}$$

$$f(x, a, b) = \frac{ax}{b + x}$$



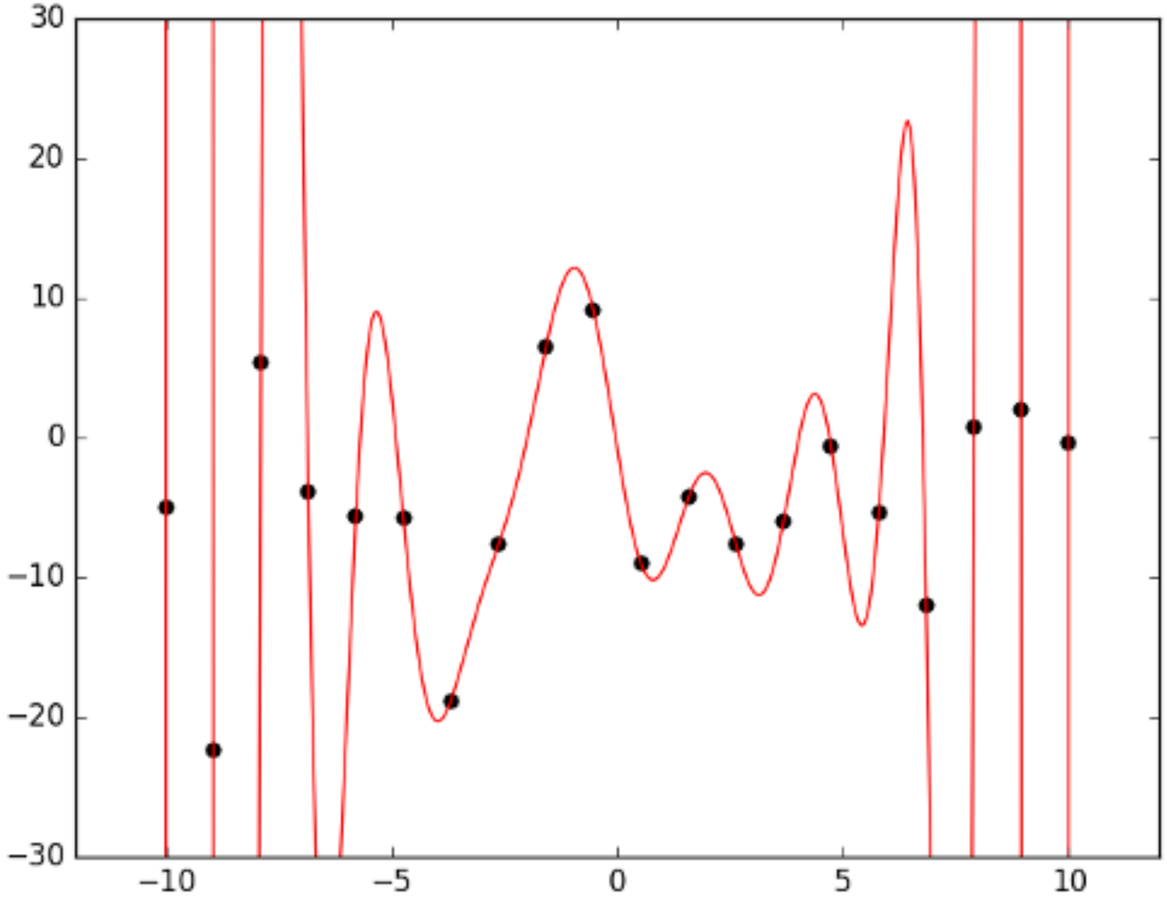
The more parameters the better the fit?



”With four parameters I can fit an elephant, and with five I can make him wiggle his trunk”

John von Neumann

The more parameters the better the fit?

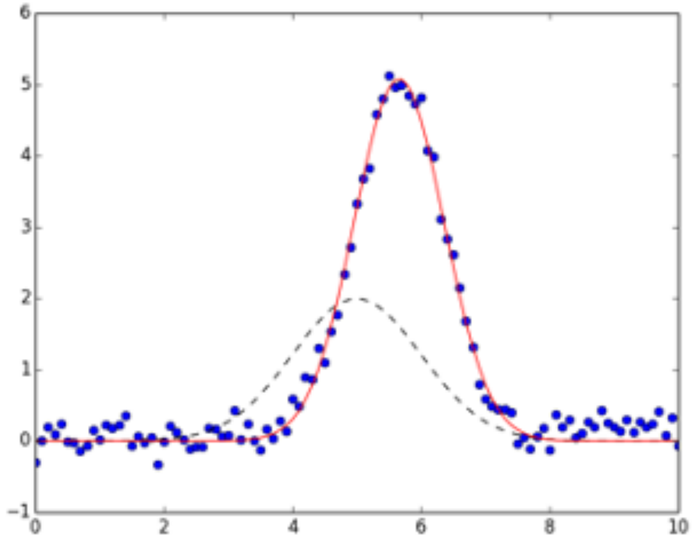


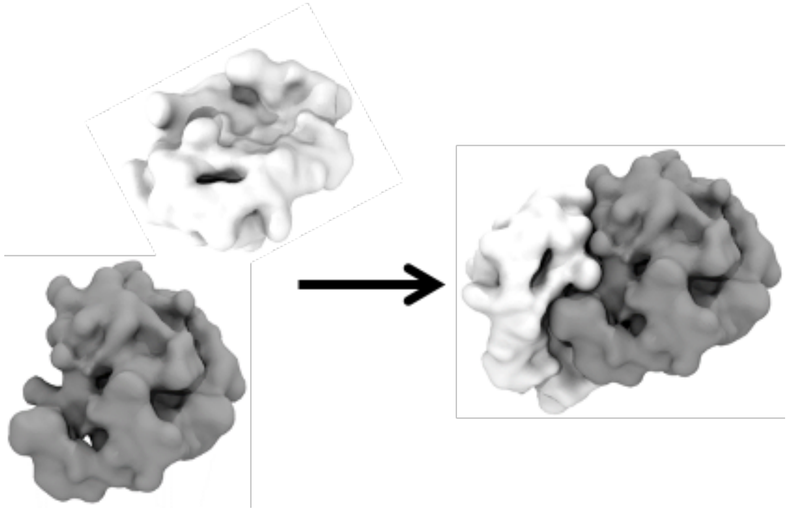
"With four parameters I can fit an elephant, and with five I can make him wiggle his trunk"

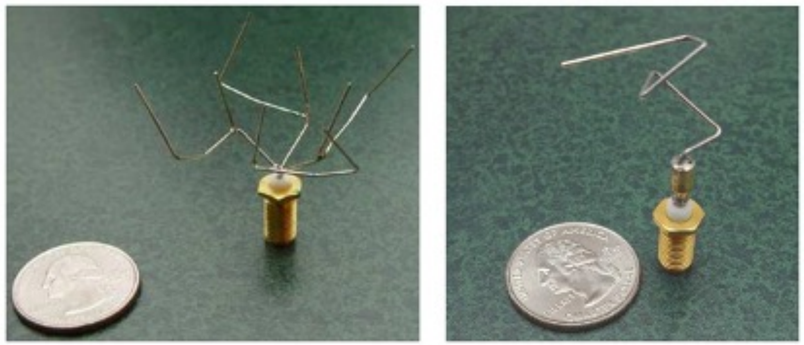
John von Neumann

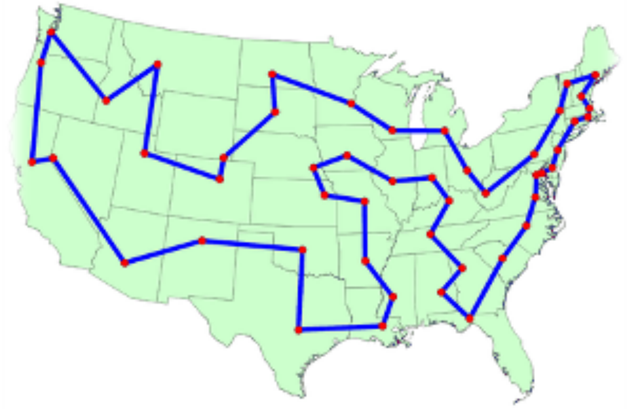
N points can be perfectly fitted with an N-1 order polynomial

Data Fitting is an Optimisation Problem









Optimisation problem mathematically

Optimisation problem mathematically

1) Define the parameters **x** of your problem

Optimisation problem mathematically

1) Define the parameters **x** of your problem

2) Let y=f(**x**) a function associating a score to every parameter combination in a subset of Euclidean space $\mathbf{R}^n$

Optimisation problem mathematically

1) Define the parameters **x** of your problem

2) Let y=f(**x**) a function associating a score to every parameter combination in a subset of Euclidean space $\mathbf{R}^n$

3) find **x*** such that f(**x***) < f(**x**) for all **x** (a.k.a. *minimization*)

Definitions:

- *parameters **x***: (sometimes) degrees of freedom
- *subset of Euclidean space*: search space
- *f(x)*: fitness / scoring / objective / cost function

Optimisation problem mathematically

1) Define the parameters **x** of your problem

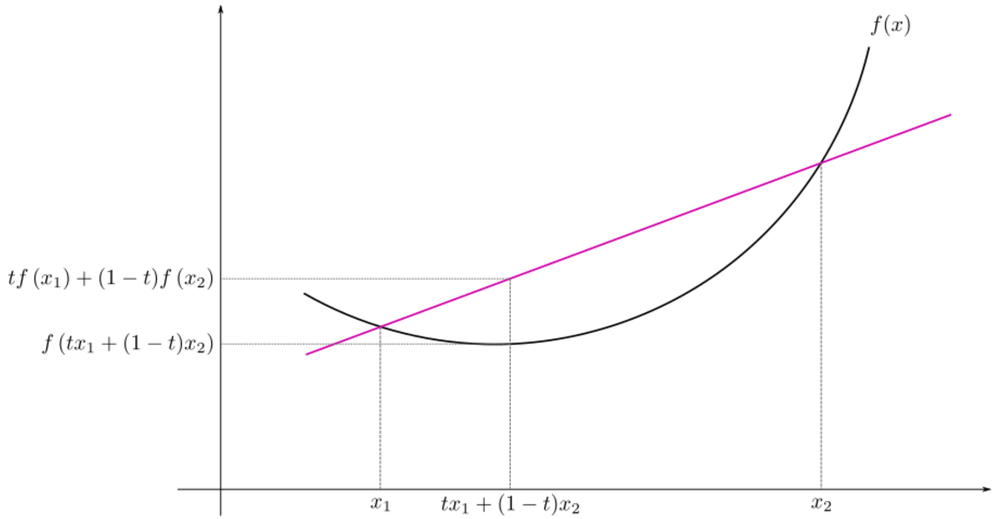
2) Let y=f(**x**) a function associating a score to every parameter combination in a subset of Euclidean space $\mathbf{R}^n$

3) find **x*** such that f(**x***) < f(**x**) for all **x** (a.k.a. *minimization*)

Definitions:

- *parameters **x***: (sometimes) degrees of freedom
- *subset of Euclidean space*: search space
- *f(x)*: fitness / scoring / objective / cost function

Convex function: iff the segments connecting any two points on the function, lie above it



the function has *only one* minimum

demonstrating that a problem is convex is a *big* deal!

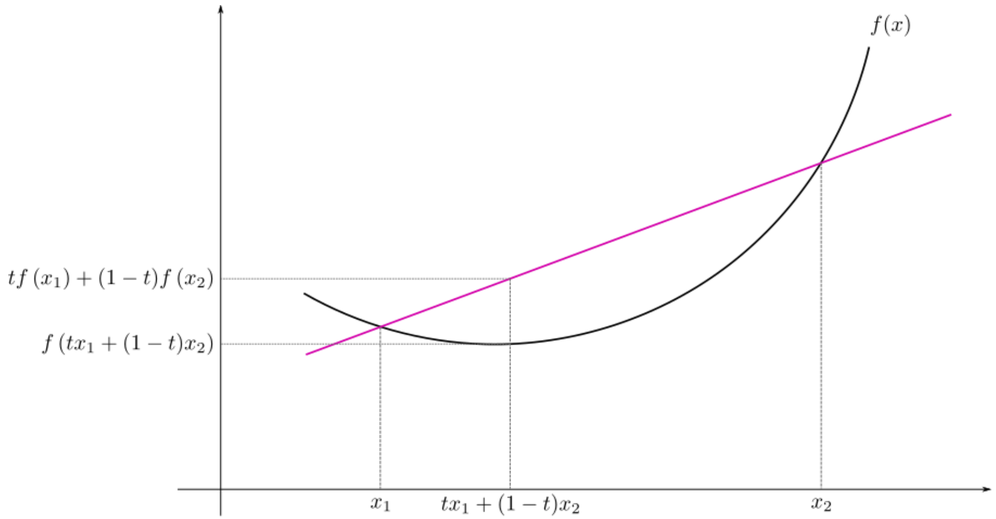
Optimisation problem mathematically

1) Define the parameters **x** of your problem

2) Let y=f(**x**) a function associating a score to every parameter combination in a subset of Euclidean space $\mathbf{R}^n$

3) find **x*** such that f(**x***) < f(**x**) for all **x** (a.k.a. *minimization*)

Convex function: iff the segments connecting any two points on the function, lie above it



Definitions:

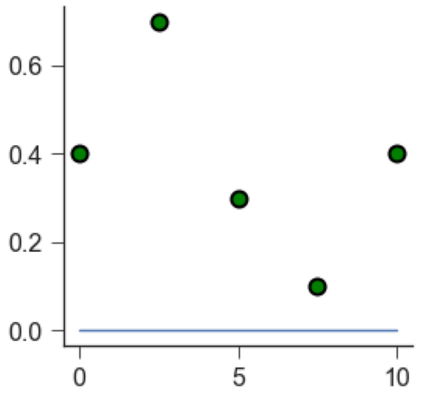
- *parameters **x***: (sometimes) degrees of freedom
- *subset of Euclidean space*: search space
- *f(x)*: fitness / scoring / objective / cost function

the function has *only one* minimum

demonstrating that a problem is convex is a *big* deal!

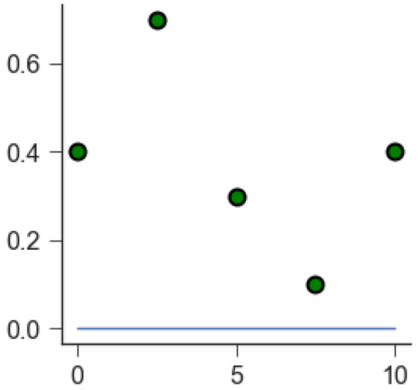
What is a good objective function?

Linear regression: in ML language

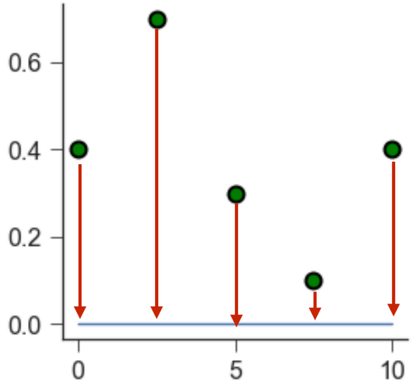


The blue line is meant to **fit** the green data points.

Linear regression: in ML language

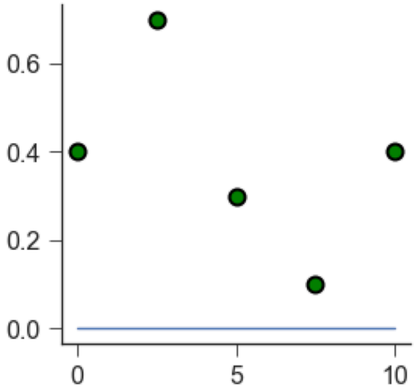


The blue line is meant to **fit** the green data points.

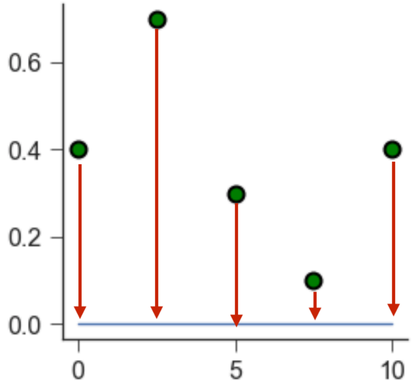


Red arrows are a called **residuals** and small arrows mean a good fit.

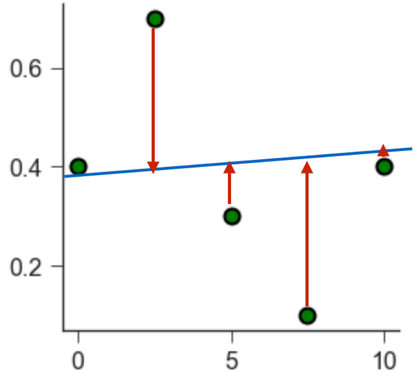
Linear regression: in ML language



The blue line is meant to **fit** the green data points.

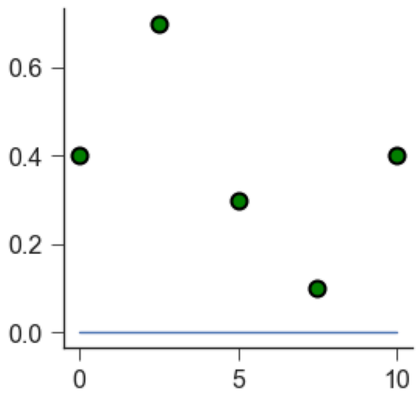


Red arrows are a called **residuals** and small arrows mean a good fit.

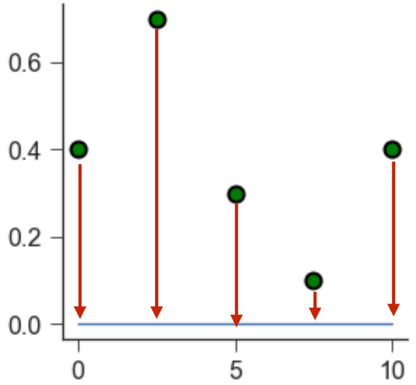


Mean square error is a good **loss function,** because it is **convex.**

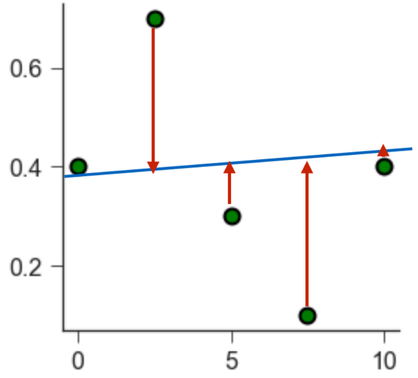
Linear regression: in ML language



The blue line is meant to **fit** the green data points.



Red arrows are a called **residuals** and small arrows mean a good fit.



Mean square error is a good **loss function,** because it is **convex.**

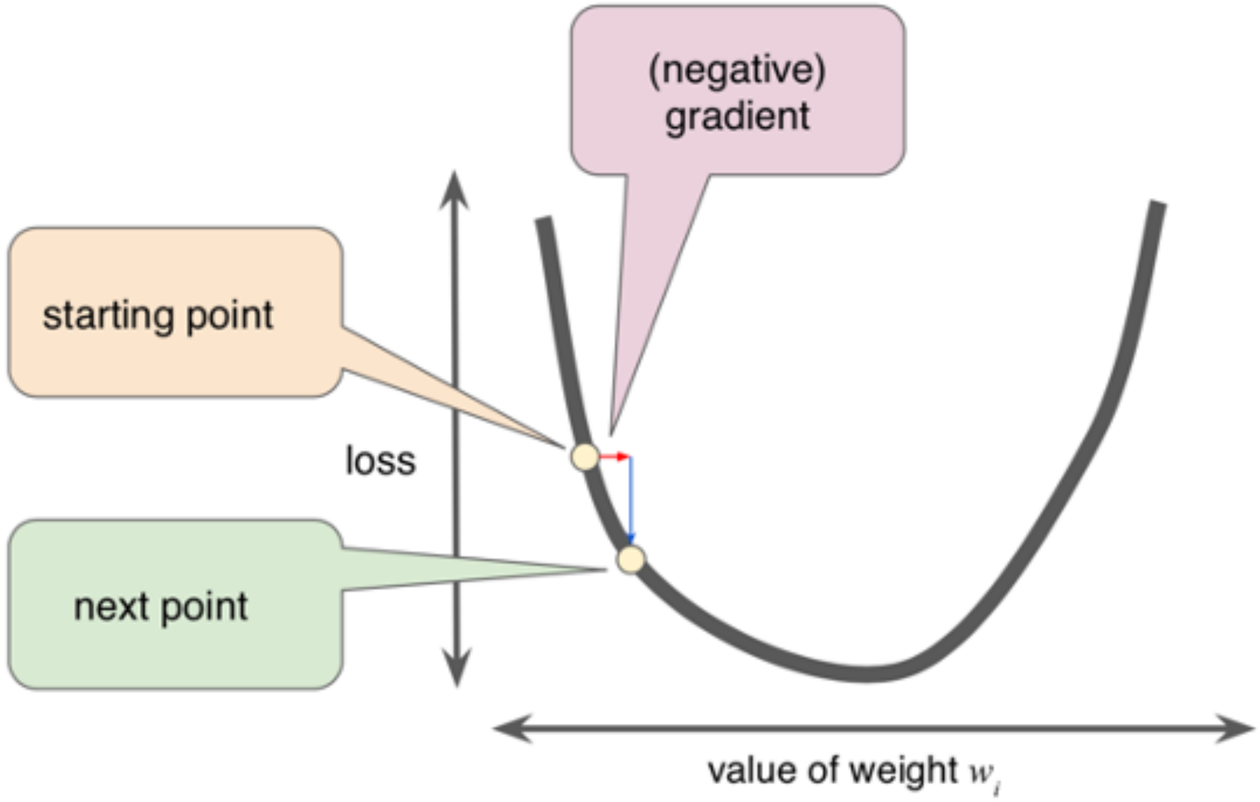
$$\mathrm{MSE} = \frac{1}{n}\sum(Y_i - \hat{Y}_i)^2$$

n is the number of data points,
Y_i is the observed value (i.e. the measured data point),
$\hat{Y}_i$ is the predicted value (i.e. the value that lies on the line of best fit).

Gradient Descent

$$\text{MSE} = \frac{1}{n}\sum(Y_i - \hat{Y}_i)^2$$

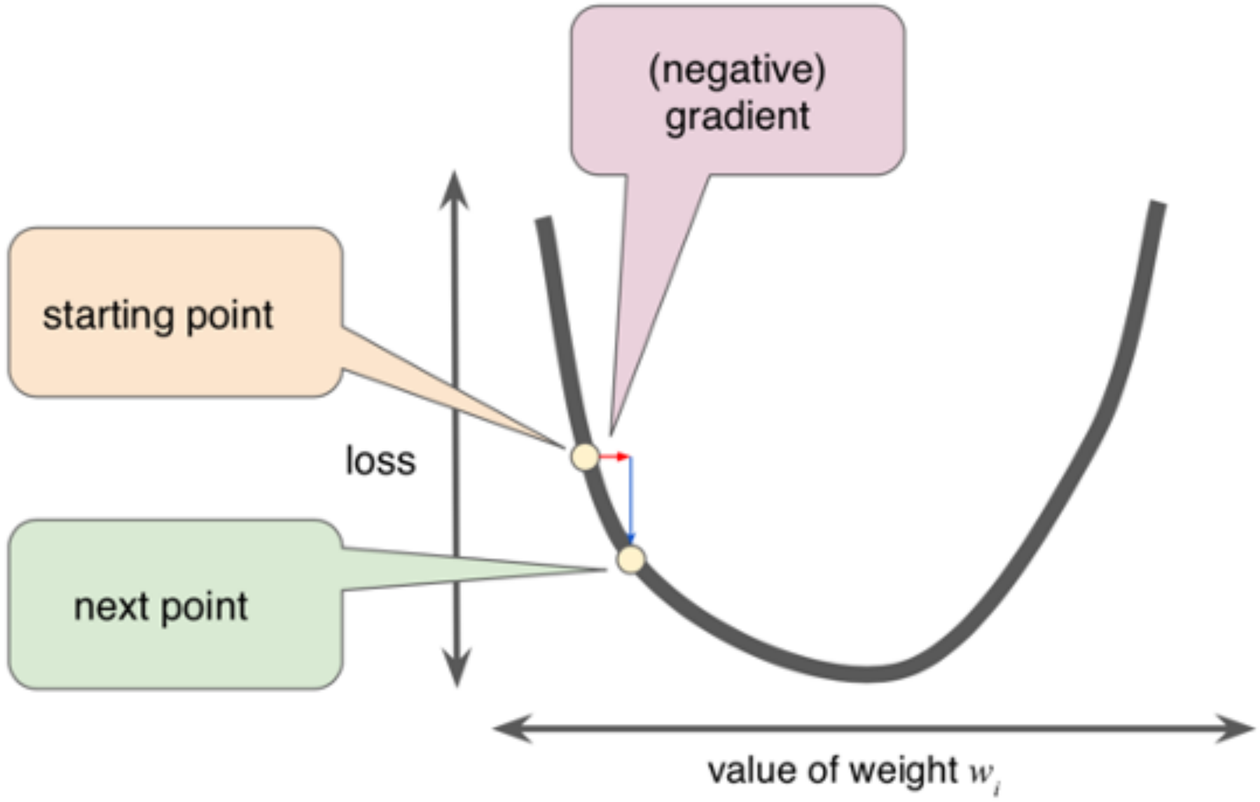
Objective: Find the minimum of the loss function (see optimisation problem)



Gradient Descent

$$\text{MSE} = \frac{1}{n}\sum(Y_i - \hat{Y}_i)^2$$

Objective: Find the minimum of the loss function (see optimisation problem)

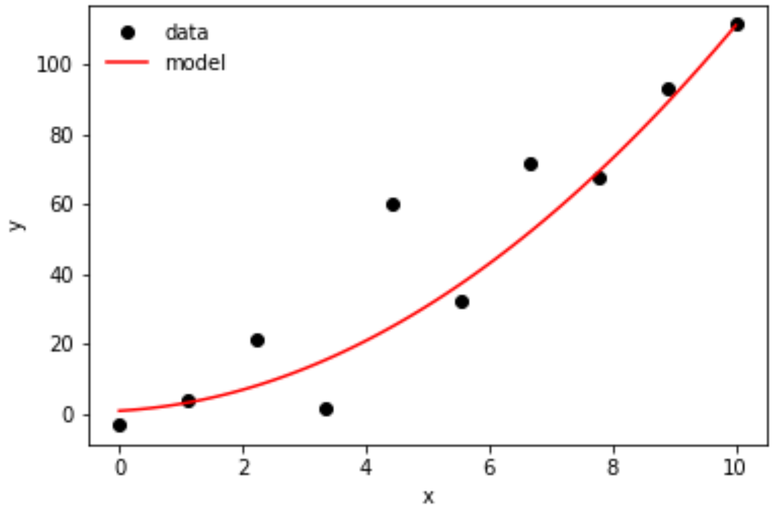


In machine learning, the step size of the gradient descent is called the **learning rate**

There is an *optimal learning* rate for every regression problem

Choosing the **best** optimiser is an optimisation problem in itself!

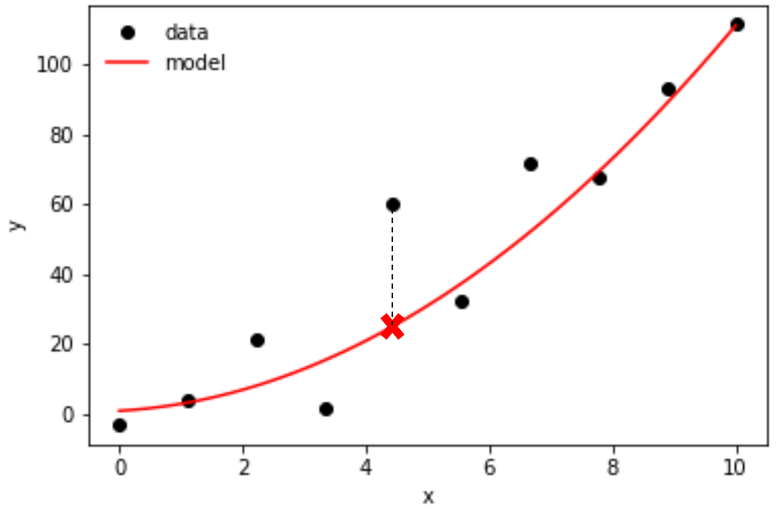
Linear Least Squares: A Summary



Model predicts values for each datapoint

$$\hat{y}_i = f(x_i; a, b, \ldots)$$

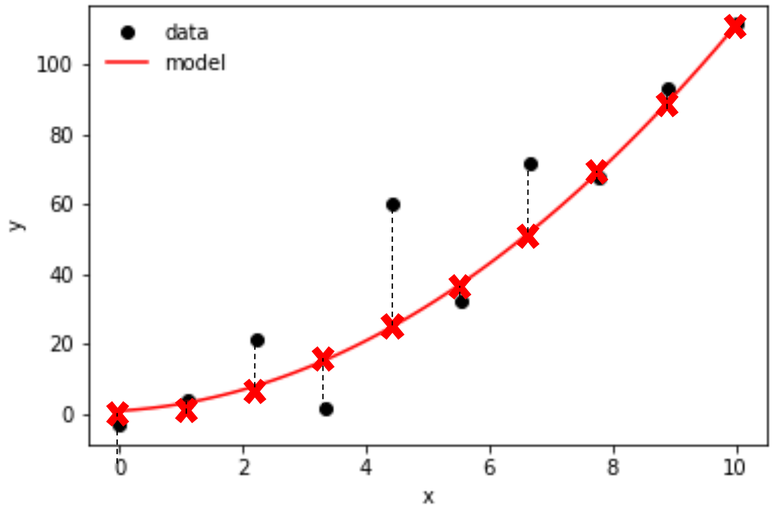
Linear Least Squares: A Summary



Model predicts values for each datapoint

$$\hat{y}_i = f(x_i; a, b, \ldots)$$

Linear Least Squares: A Summary



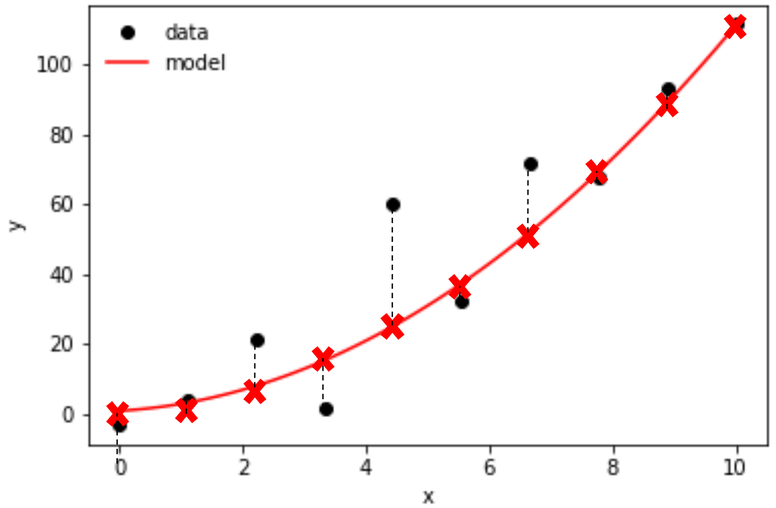
Model predicts values for each datapoint

$$\hat{y}_i = f(x_i; a, b, \ldots)$$

Residuals quantify prediction error

$$r_i = y_i - f(x_i; a, b, \ldots)$$

Linear Least Squares: A Summary



Model predicts values for each datapoint

$$\hat{y}_i = f(x_i; a, b, \ldots)$$

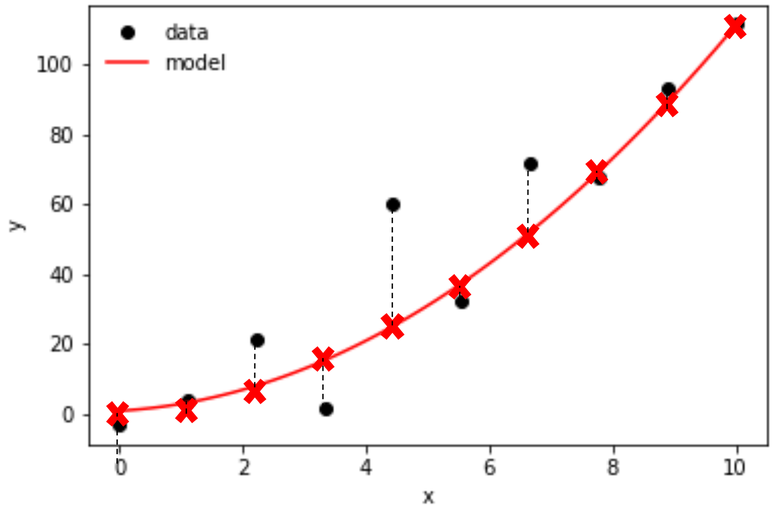
Residuals quantify prediction error

$$r_i = y_i - f(x_i; a, b, \ldots)$$

The best model minimizes the sum of squared residuals (“loss function”)

$$E(a, b, \ldots) = \sum_{i=1}^{N} r_i^2$$

Linear Least Squares: A Summary



Model predicts values for each datapoint

$$\hat{y}_i = f(x_i; a, b, \ldots)$$

Residuals quantify prediction error

$$r_i = y_i - f(x_i; a, b, \ldots)$$

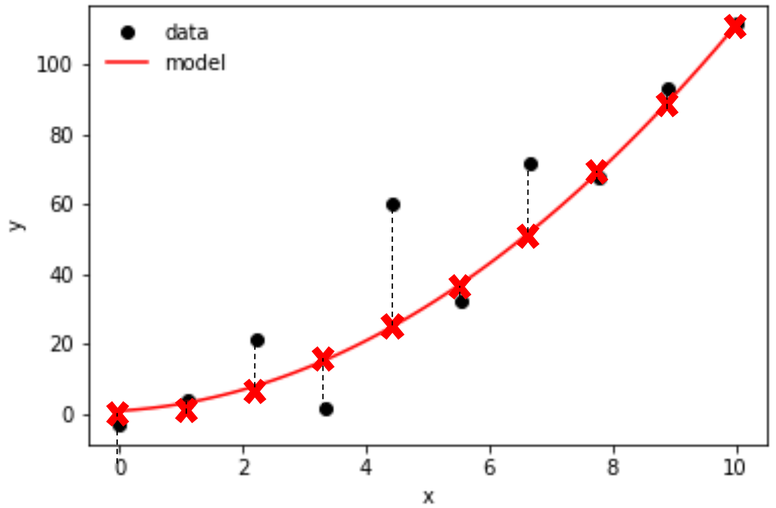
The best model minimizes the sum of squared residuals ("loss function")

$$E(a, b, \ldots) = \sum_{i=1}^{N} r_i^2$$

Solve: $\nabla E = 0$

-> find the minimum of the loss function
Note: This can be in many dimensions!

Linear Least Squares: A Summary



Model predicts values for each datapoint

$$\hat{y}_i = f(x_i; a, b, \ldots)$$

Residuals quantify prediction error

$$r_i = y_i - f(x_i; a, b, \ldots)$$

The best model minimizes the sum of squared residuals ("loss function")

$$E(a, b, \ldots) = \sum_{i=1}^{N} r_i^2$$

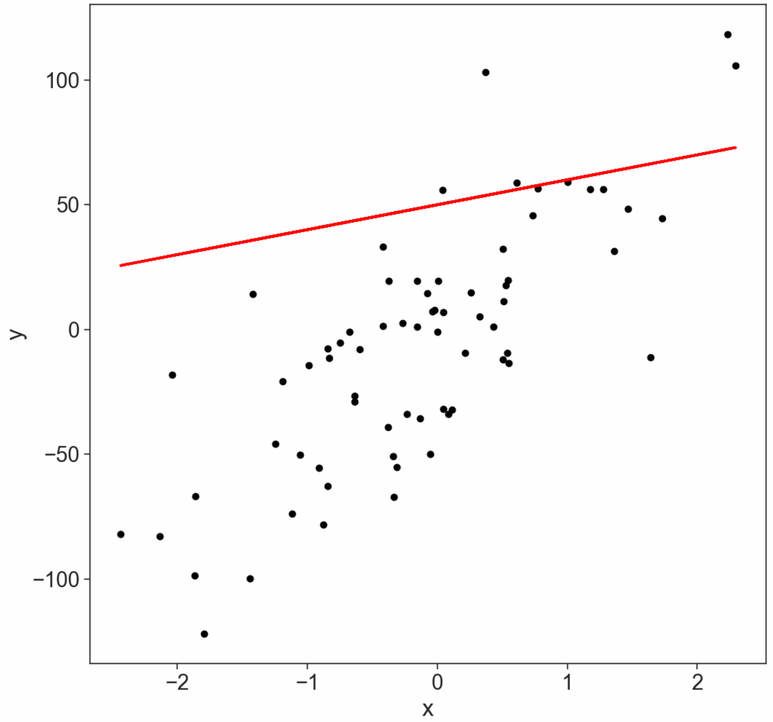
Solve: $\nabla E = 0$

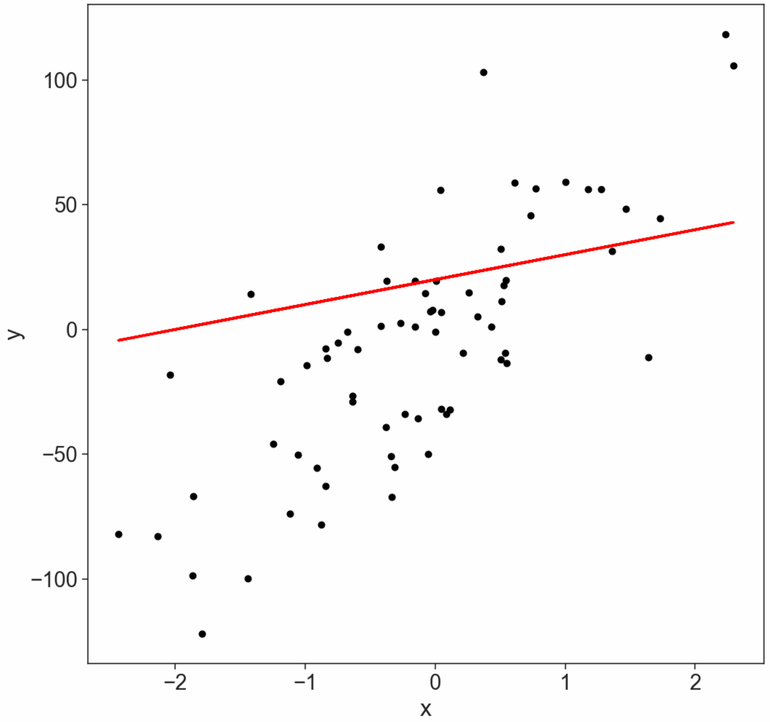
-> find the minimum of the loss function
Note: This can be in many dimensions!

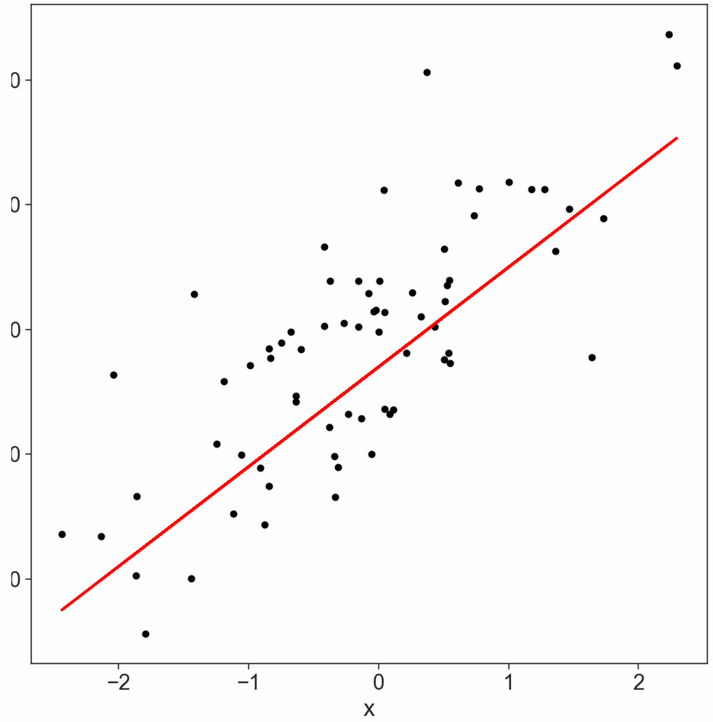
Use least squares (i.e. analytical solution) if:

- there are more data points than parameters (overdetermined)
- Model's parameters combine linearly (linear least squares)

Gradient Descent: learning rate and initial conditions

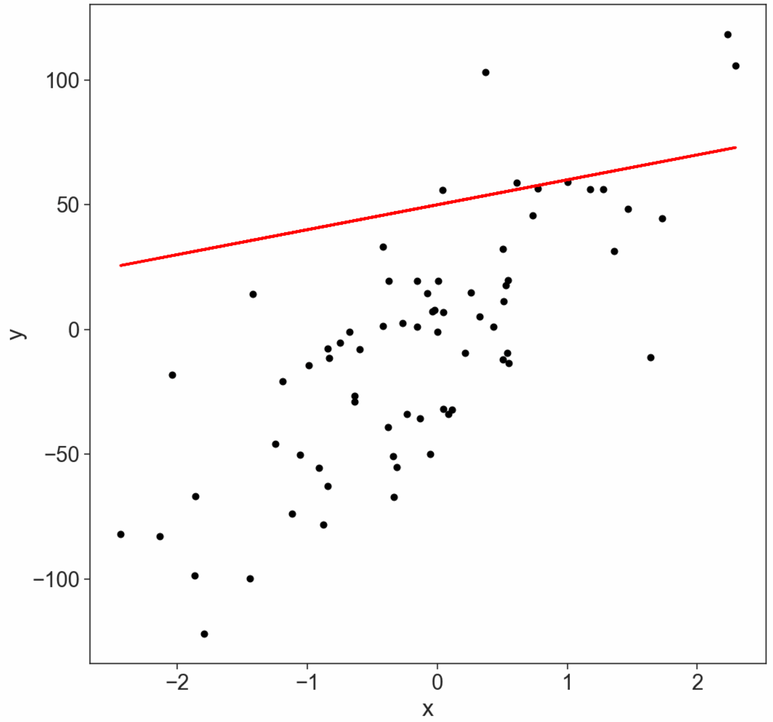


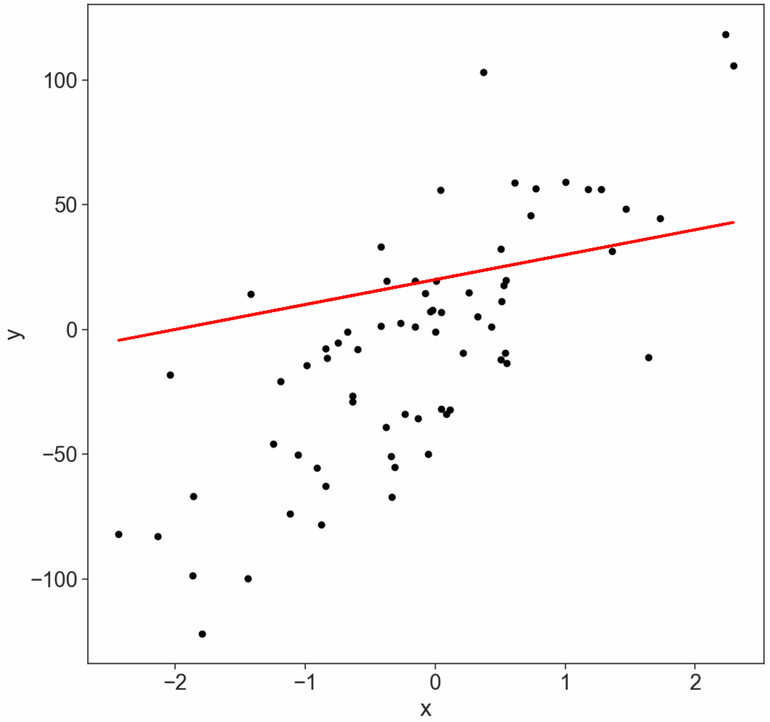


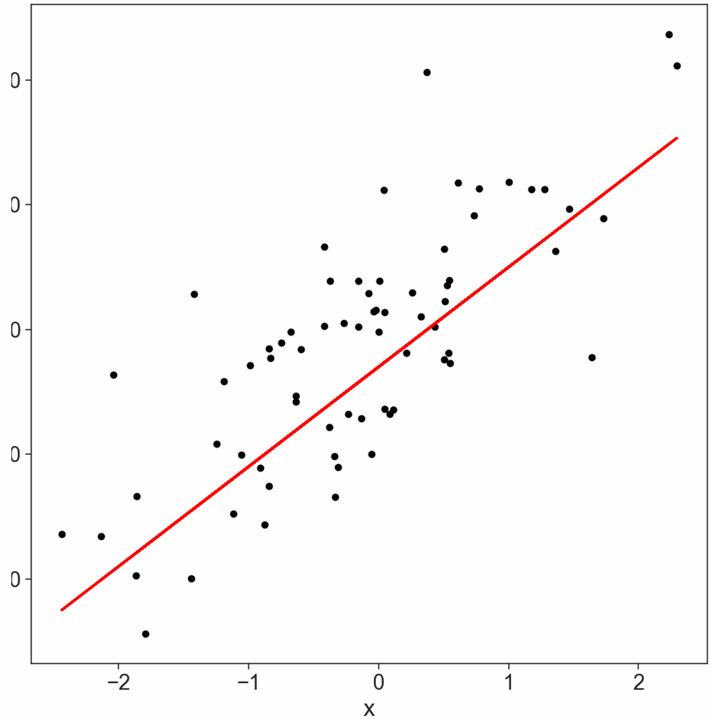


Low learning rate
bad starting point
424 steps

Gradient Descent: learning rate and initial conditions

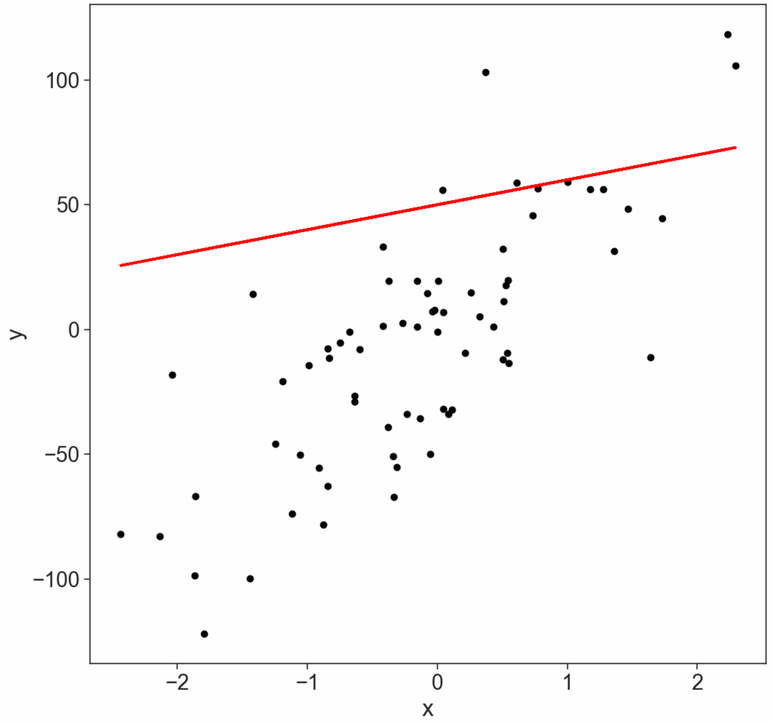


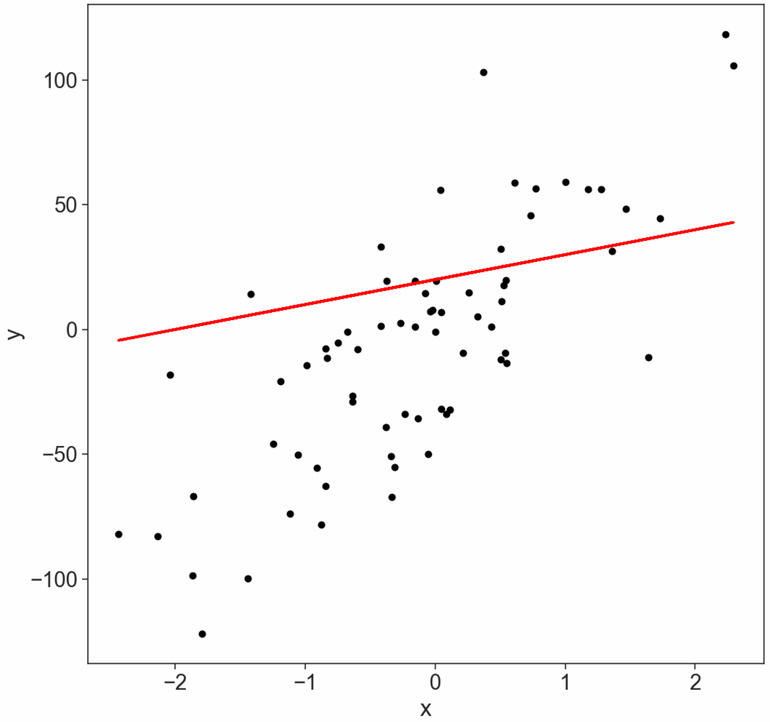


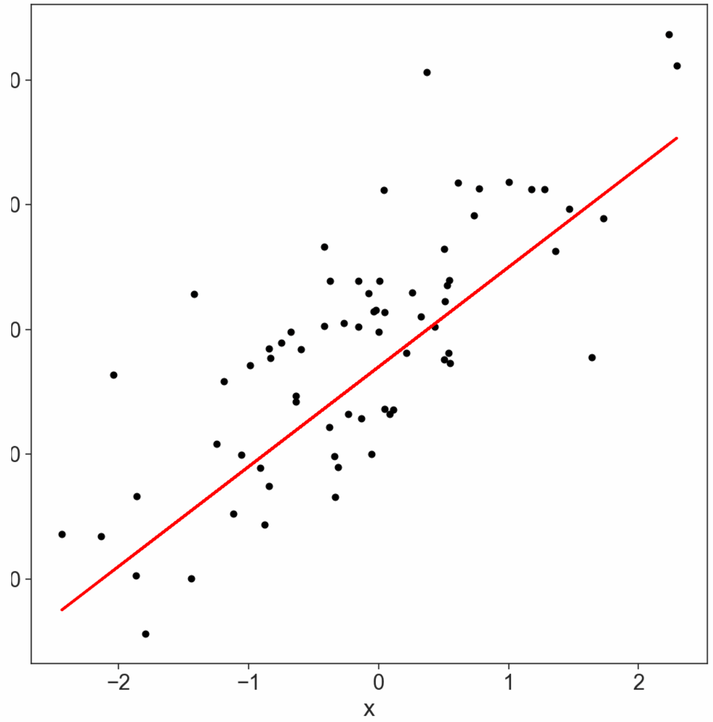


Low learning rate
bad starting point
424 steps

Gradient Descent: learning rate and initial conditions

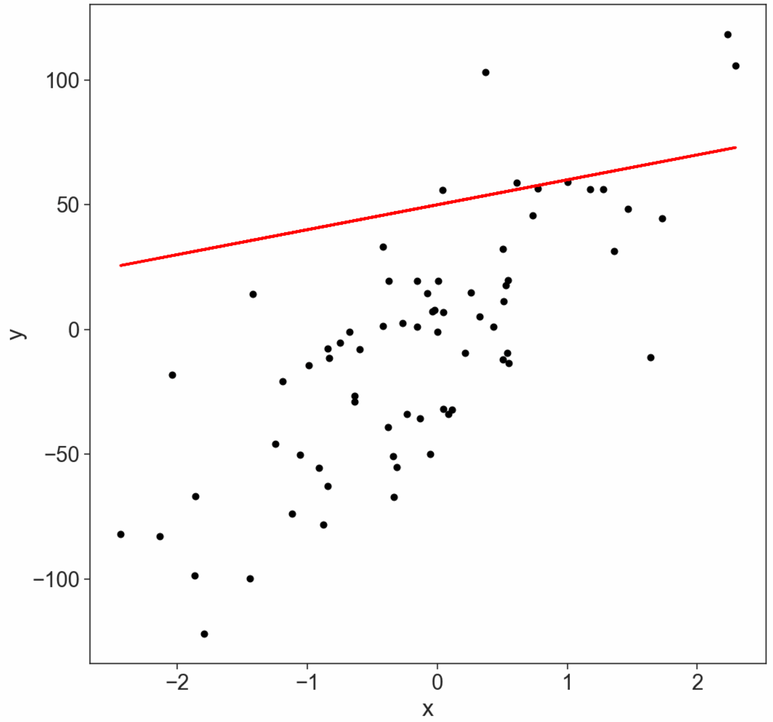




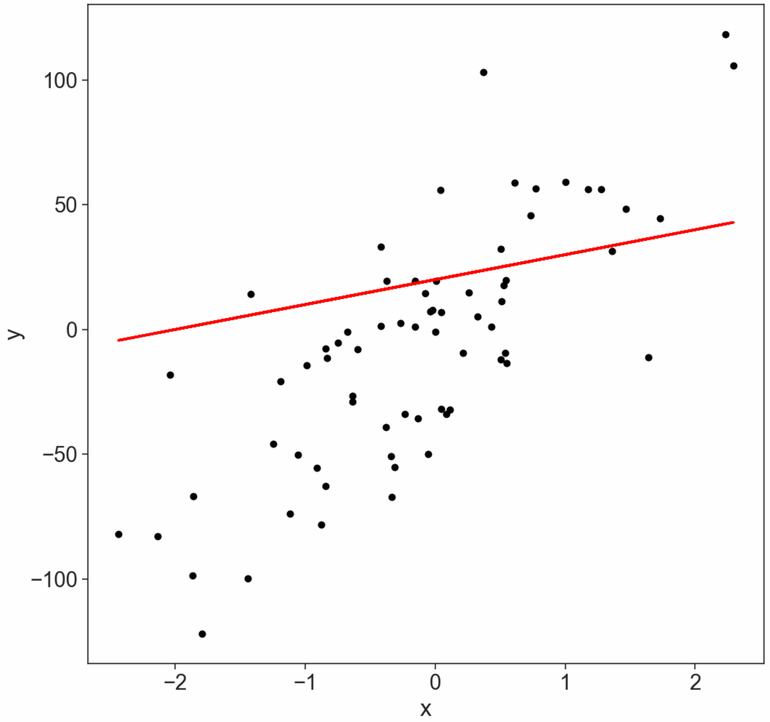


Low learning rate
bad starting point
424 steps

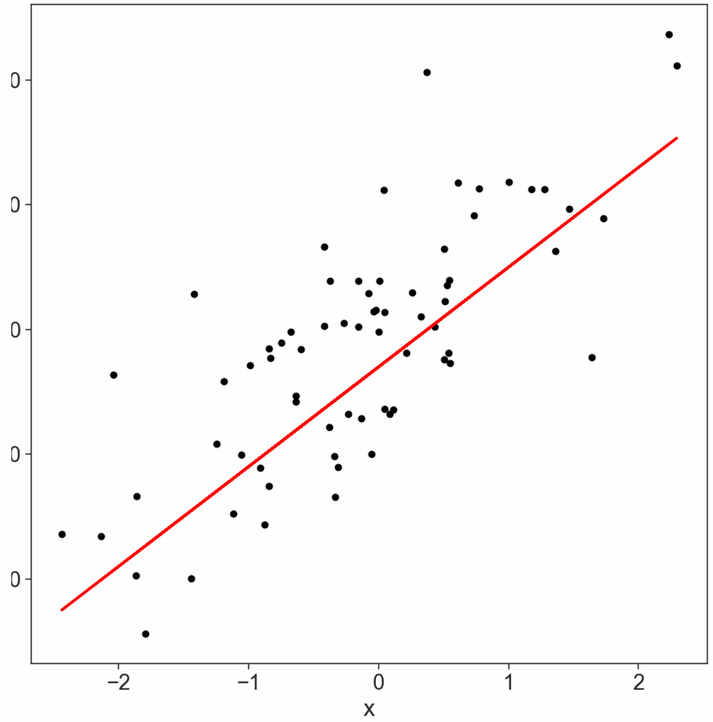
Gradient Descent: learning rate and initial conditions



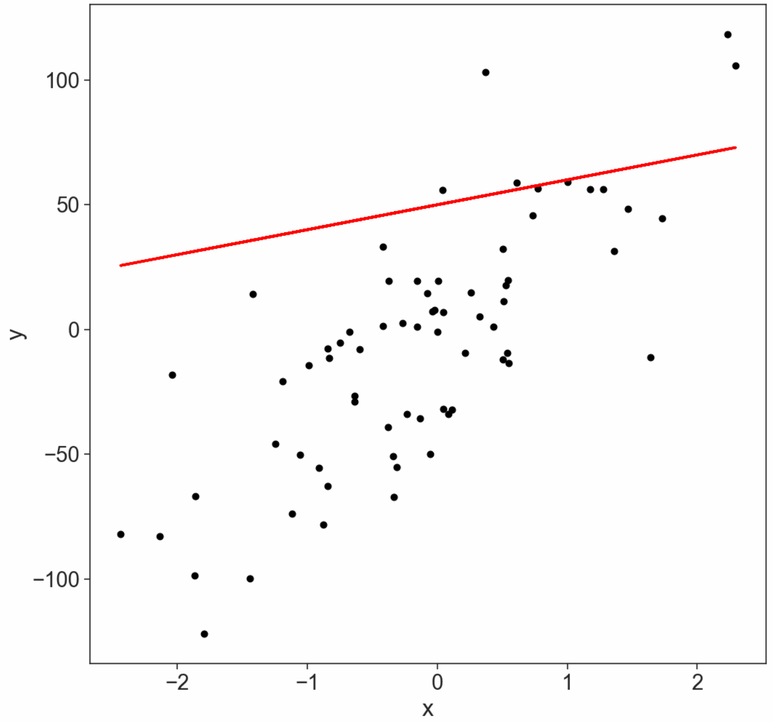
Low learning rate
bad starting point
424 steps



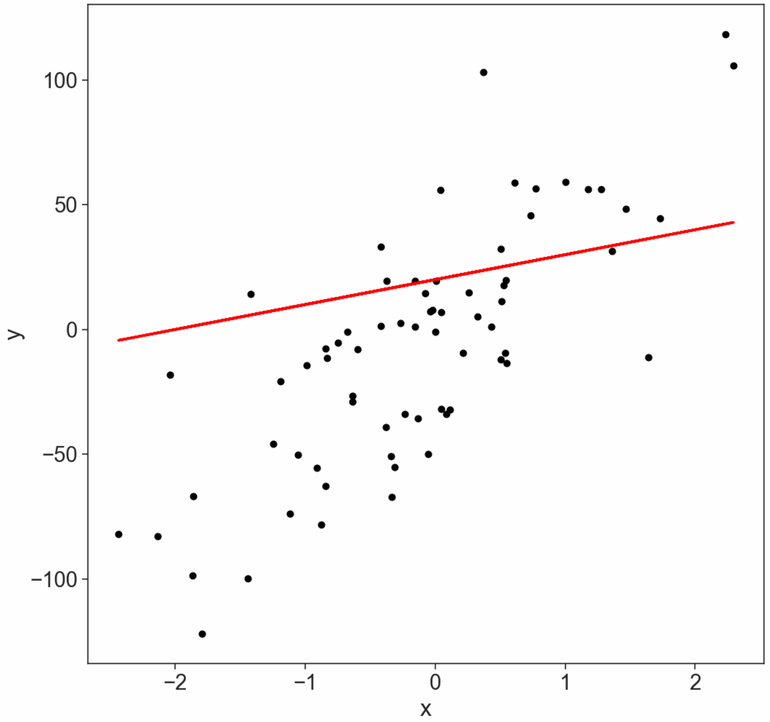
High learning rate
bad starting point
52 steps



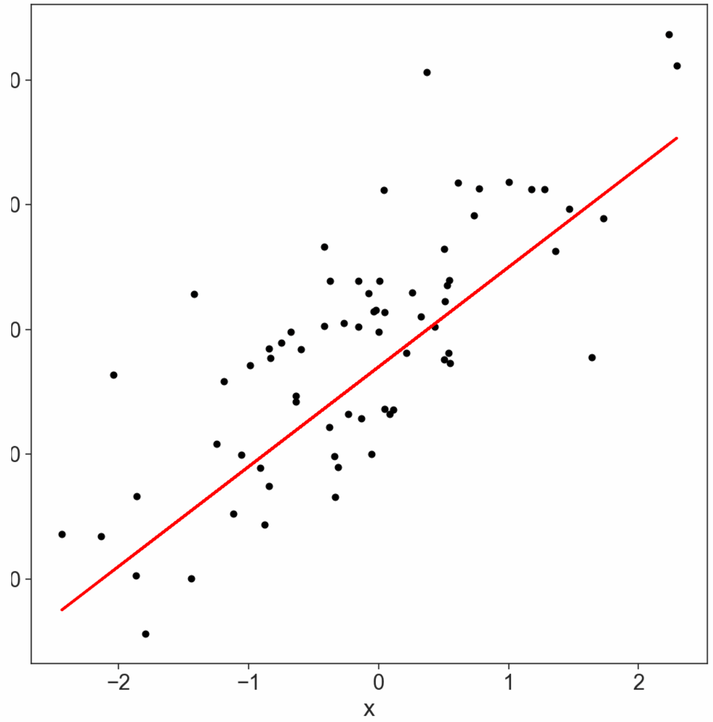
Gradient Descent: learning rate and initial conditions



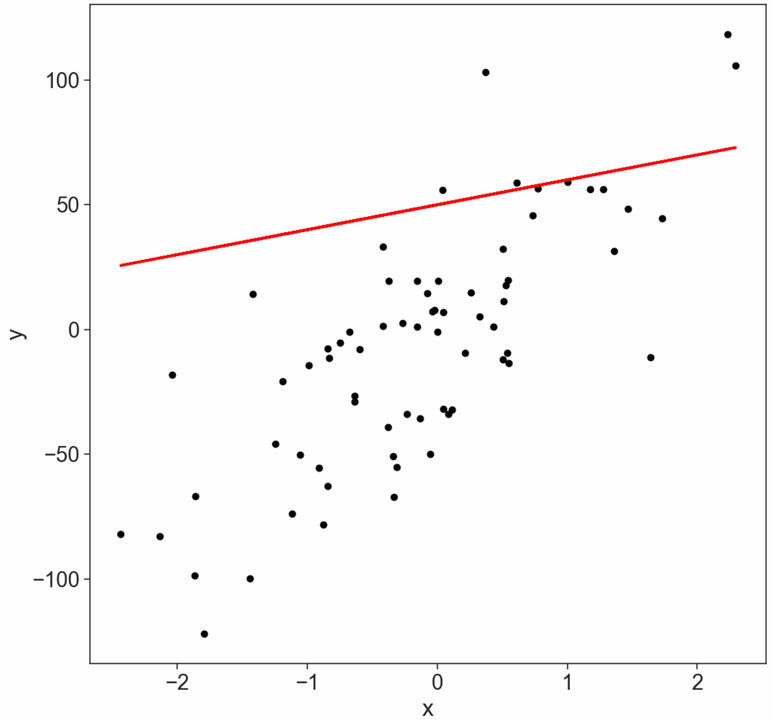
Low learning rate
bad starting point
424 steps



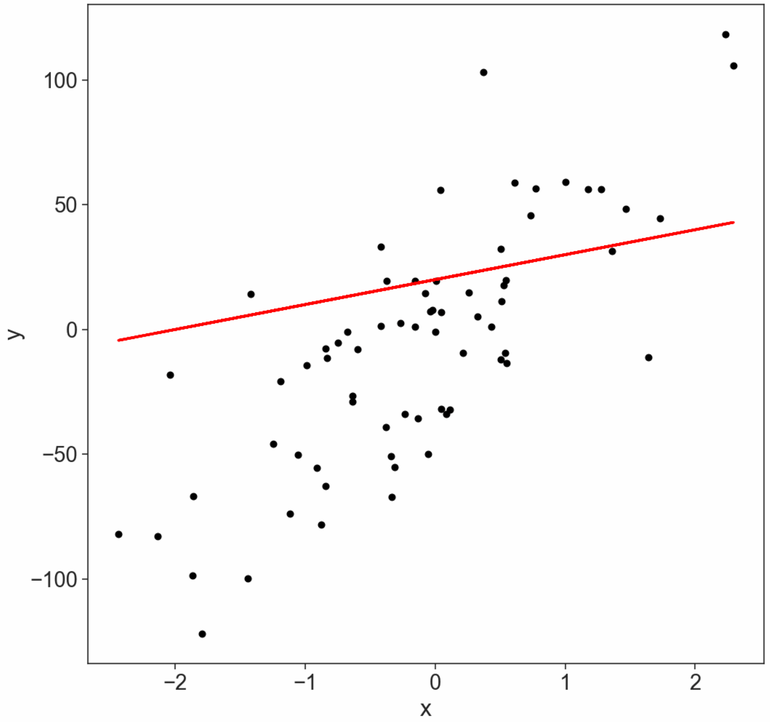
High learning rate
bad starting point
52 steps



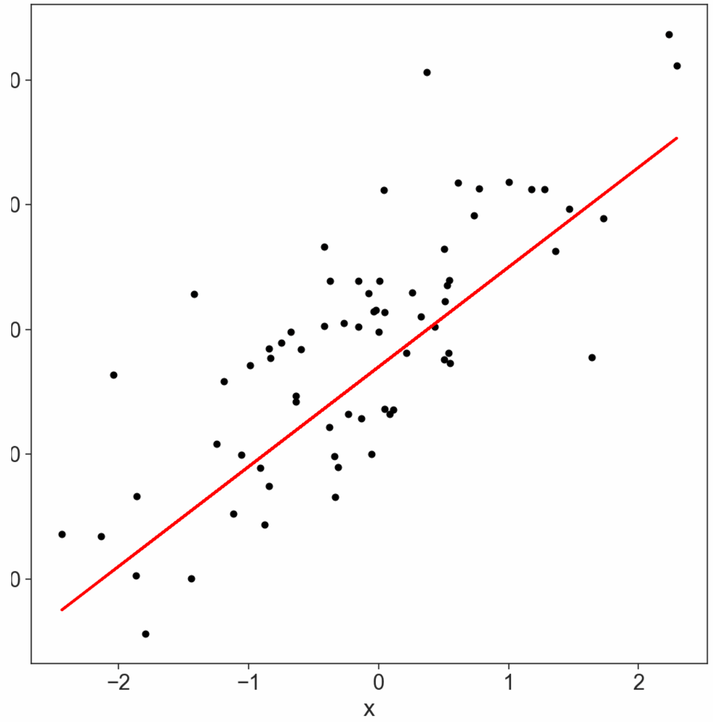
Gradient Descent: learning rate and initial conditions



Low learning rate
bad starting point
424 steps



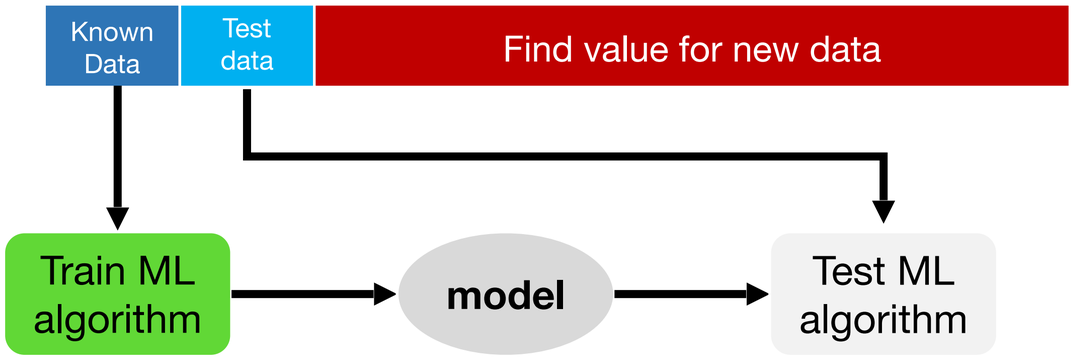
High learning rate
bad starting point
52 steps



Low learning rate
Good starting point
379 steps

Learning Algorithms for regression problems

- Decision Tree Regression (DT)
- Random Forests (RF)
- Neural Network Regressions (ANN)
- Support Vector Machine (SVM)
- **Linear Regression (LR)**
- *Non-Linear Regression*
- K Nearest Neighbor (KNN)
- Gaussian Regression
- ...

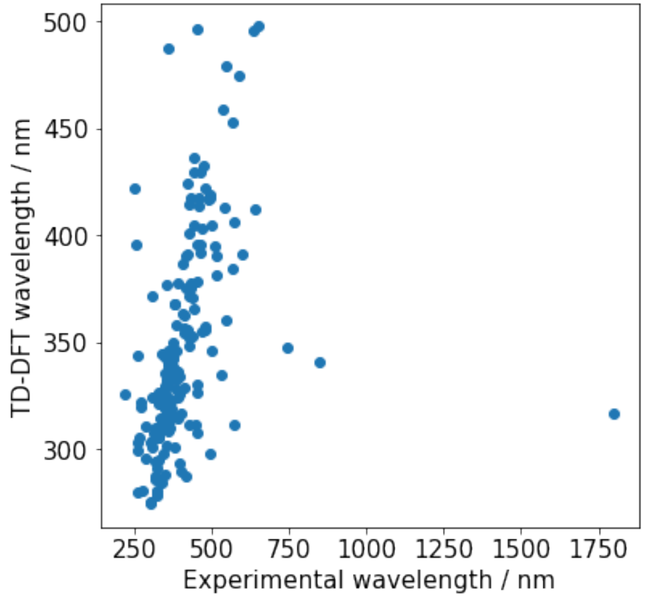


Linear and non-linear examples in chemistry

Can you think of linear/non-linear regression models you have used in your degree?

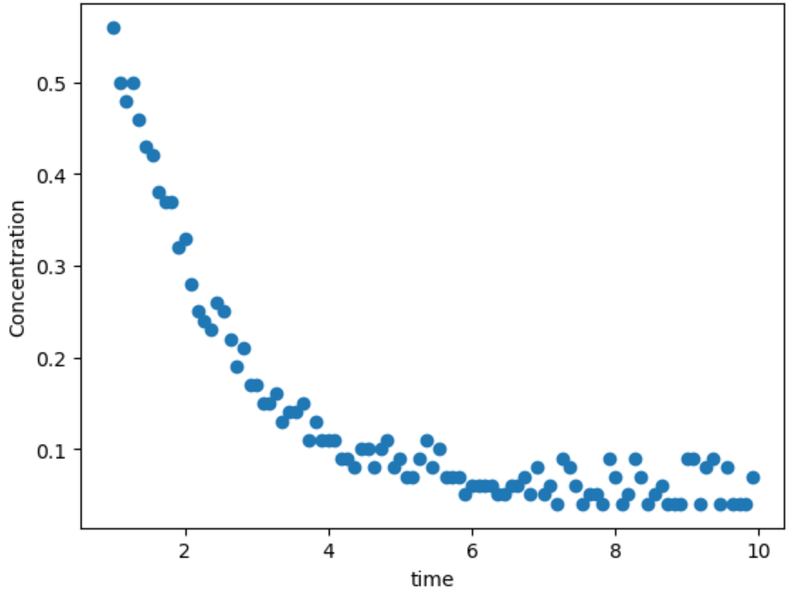
Can you think of examples in multiple dimensions?

Comparing experimental UV-Vis data with DFT calculations



$$f(x) = mx + b$$

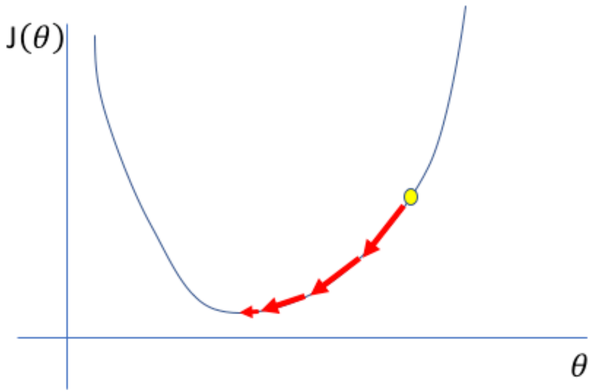
Variation of the concentration of a drug molecule in blood over time



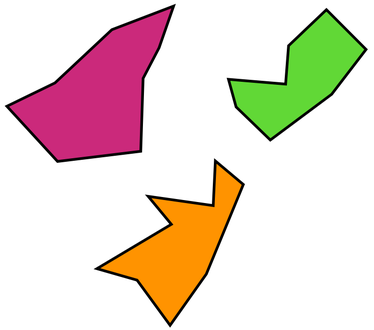
$$f(x) = a\exp(kx) + b$$

Hyper parameters open up an array of modelling choices

Learning rate

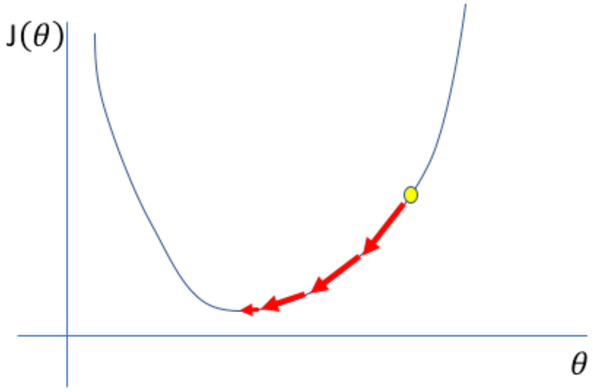


Hyper parameters open up an array of modelling choices

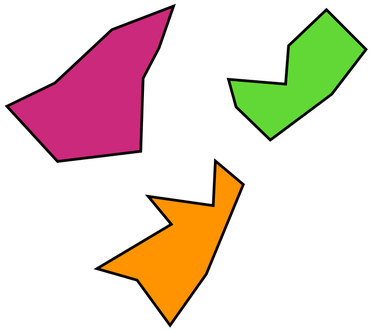


N-clusters

Learning rate



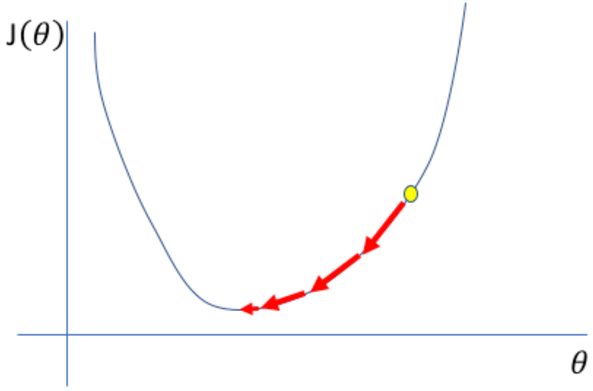
Hyper parameters open up an array of modelling choices



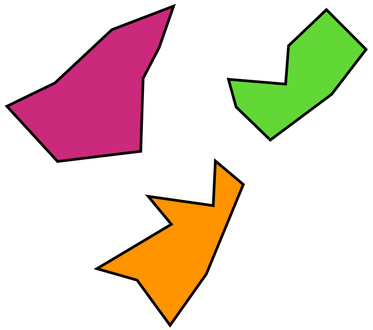
N-clusters

How many dimensions for reduction

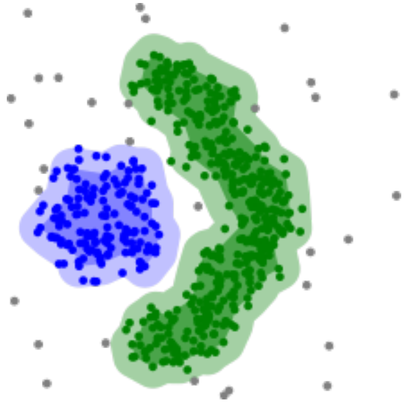
Learning rate



Hyper parameters open up an array of modelling choices



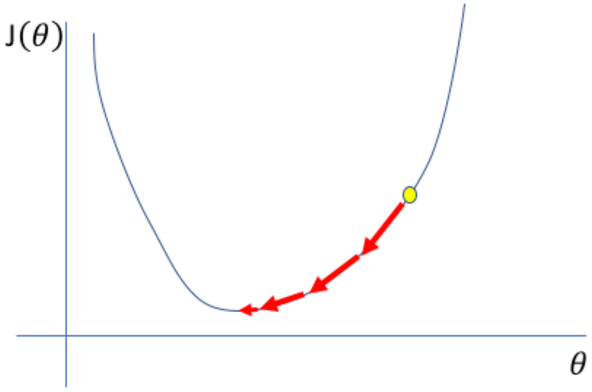
N-clusters



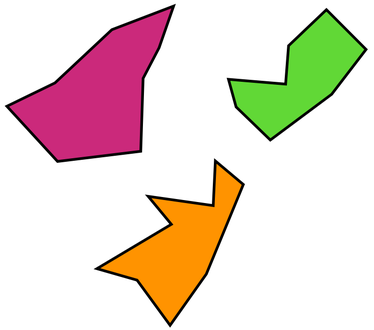
epsilon

How many dimensions for reduction

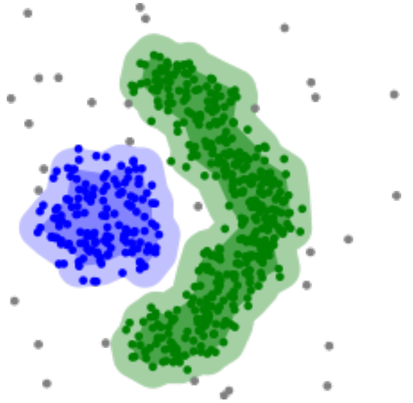
Learning rate



Hyper parameters open up an array of modelling choices



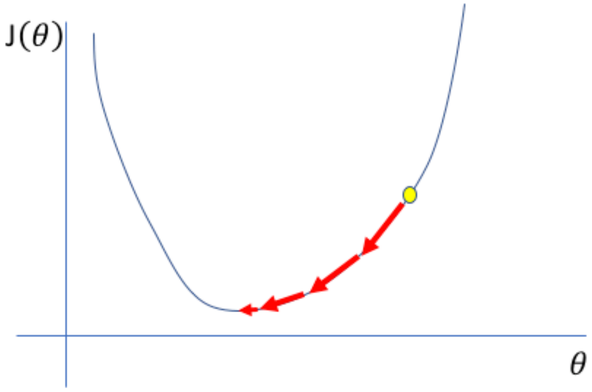
N-clusters



epsilon

How many dimensions for reduction

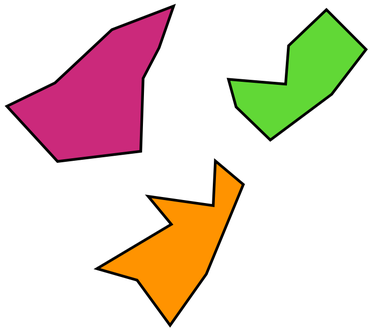
Learning rate



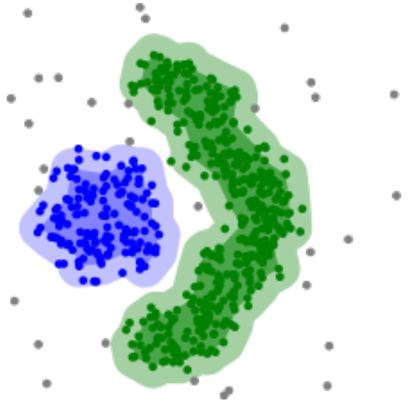
Optimiser	Year	Learning Rate	Gradient
Momentum	1964		✓
AdaGrad	2011	✓	
RMSprop	2012	✓	
Adadelta	2012	✓	
Nesterov	2013		✓
Adam	2014	✓	✓
AdaMax	2015	✓	✓
Nadam	2015	✓	✓
AMSGrad	2018	✓	✓

Choice of optimiser to minimise the loss

Hyper parameters open up an array of modelling choices



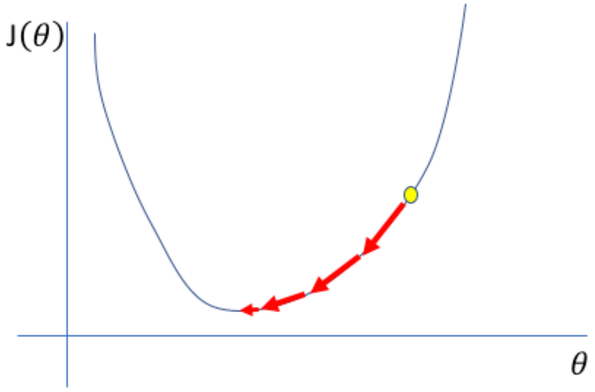
N-clusters



Choice of loss function

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Learning rate



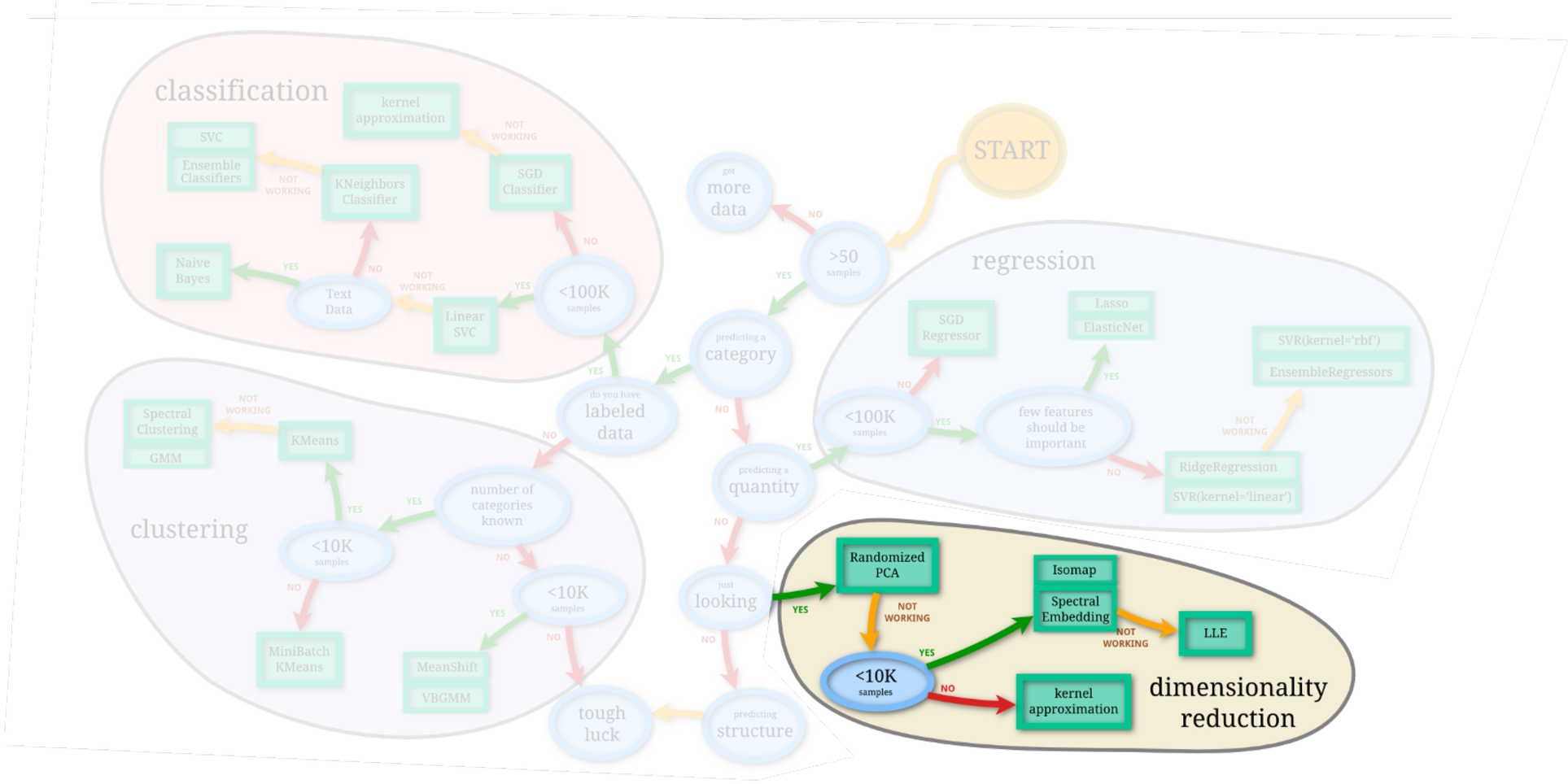
How many dimensions for reduction

epsilon

Optimiser	Year	Learning Rate	Gradient
Momentum	1964		✓
AdaGrad	2011	✓	
RMSprop	2012	✓	
Adadelta	2012	✓	
Nesterov	2013		✓
Adam	2014	✓	✓
AdaMax	2015	✓	✓
Nadam	2015	✓	✓
AMSGrad	2018	✓	✓

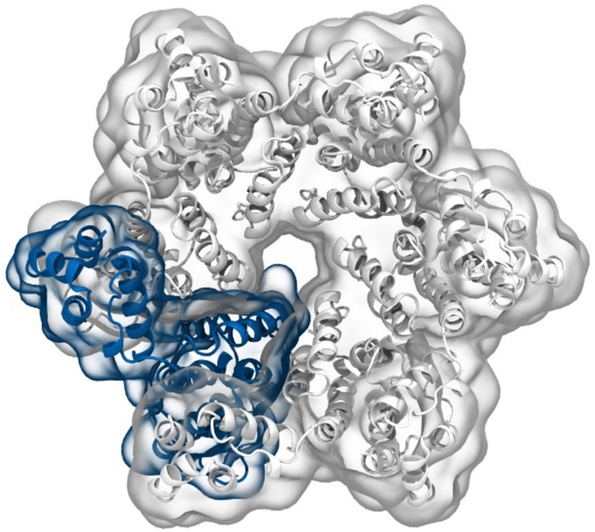
Choice of optimiser to minimise the loss

The Data Mining World — Dimensionality Reduction

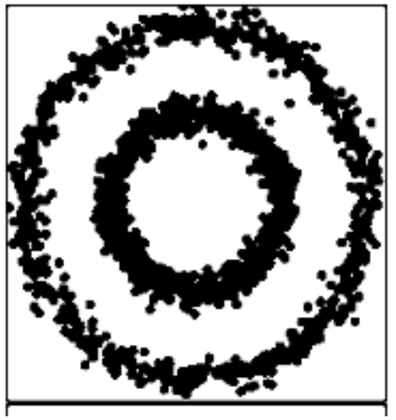


From **scikit-learn.org**

What are the ‘dominant’ dimensions?



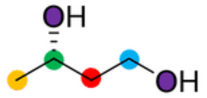
≠



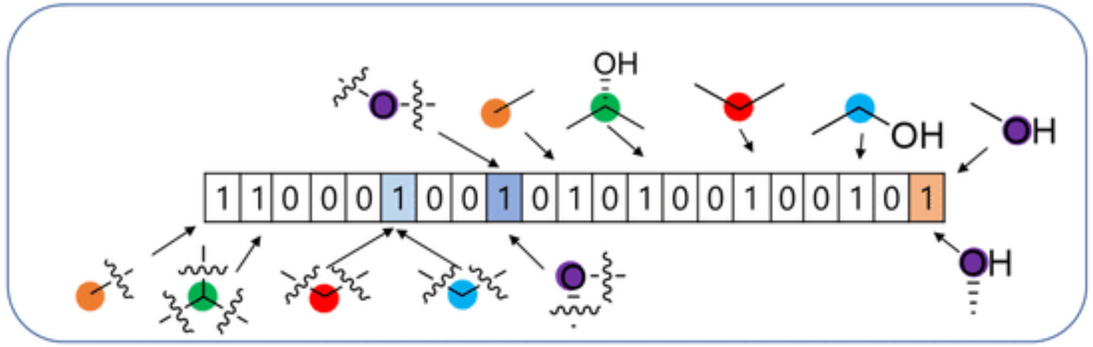
Dimensionality reduction

Feature vectors

```
['ATOM:ACE 1 CH3 1 x',
 'ATOM:ACE 1 CH3 1 y',
 'ATOM:ACE 1 CH3 1 z',
 'ATOM:ACE 1 C 4 x',
 'ATOM:ACE 1 C 4 y',
 'ATOM:ACE 1 C 4 z',
 'ATOM:ACE 1 O 5 x',
 'ATOM:ACE 1 O 5 y',
 'ATOM:ACE 1 O 5 z',
 'ATOM:ALA 2 N 6 x',
 'ATOM:ALA 2 N 6 y',
 'ATOM:ALA 2 N 6 z',
 'ATOM:ALA 2 CA 8 x',
 'ATOM:ALA 2 CA 8 y',
 'ATOM:ALA 2 CA 8 z',
 'ATOM:ALA 2 CB 10 x',
 'ATOM:ALA 2 CB 10 y',
 'ATOM:ALA 2 CB 10 z',
 'ATOM:ALA 2 C 14 x',
 'ATOM:ALA 2 C 14 y',
 'ATOM:ALA 2 C 14 z',
 'ATOM:ALA 2 O 15 x',
 'ATOM:ALA 2 O 15 y',
 'ATOM:ALA 2 O 15 z',
 'ATOM:NME 3 N 16 x',
 'ATOM:NME 3 N 16 y',
 'ATOM:NME 3 N 16 z',
 'ATOM:NME 3 C 18 x',
 'ATOM:NME 3 C 18 y',
 'ATOM:NME 3 C 18 z']
```



Molecular structure

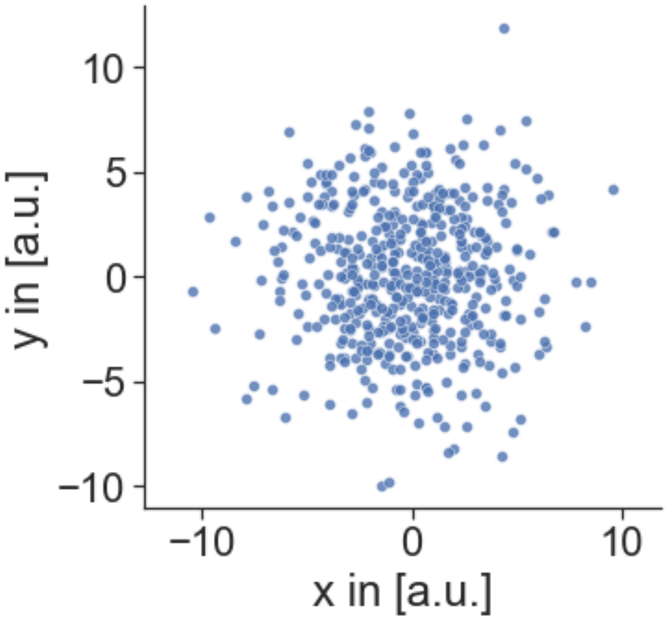


Original Morgan fingerprints

- Clustering on high dimensional data is difficult because often **not enough data is available.**
- How can you identify the **most important features before e.g. clustering?**

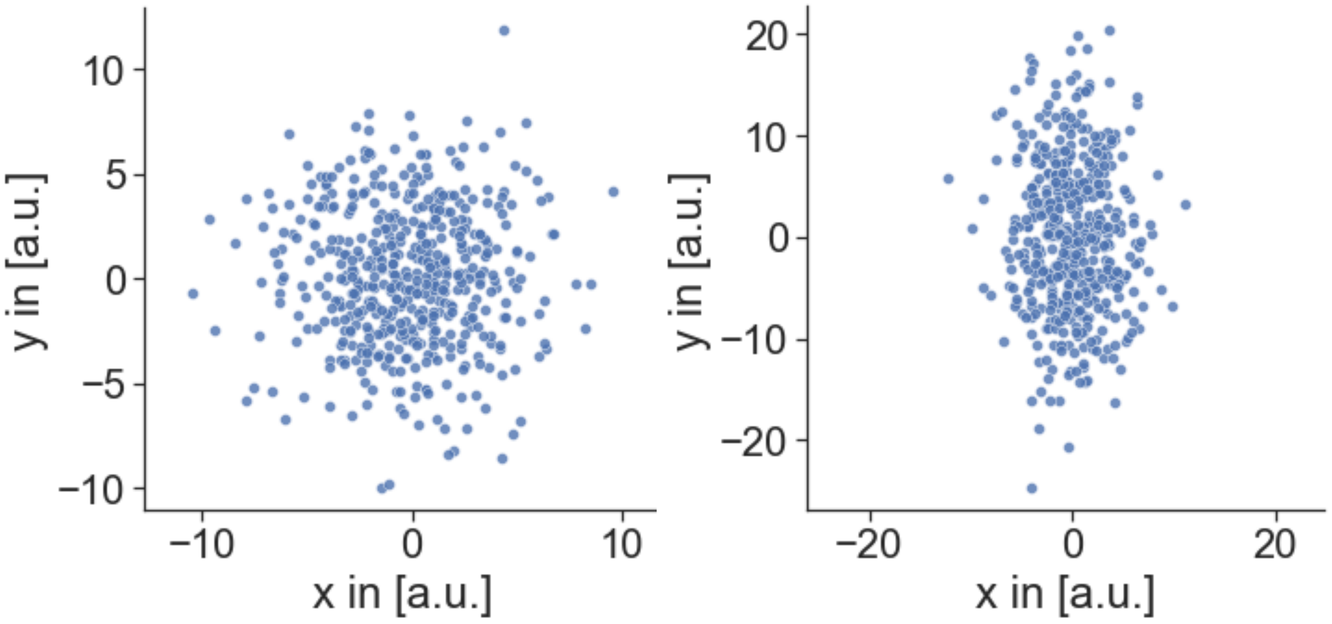
Principal component analysis (PCA)

- Principle component analysis finds the **dominant component** in the feature space
- A regression finds the **correlation** of the data in feature space



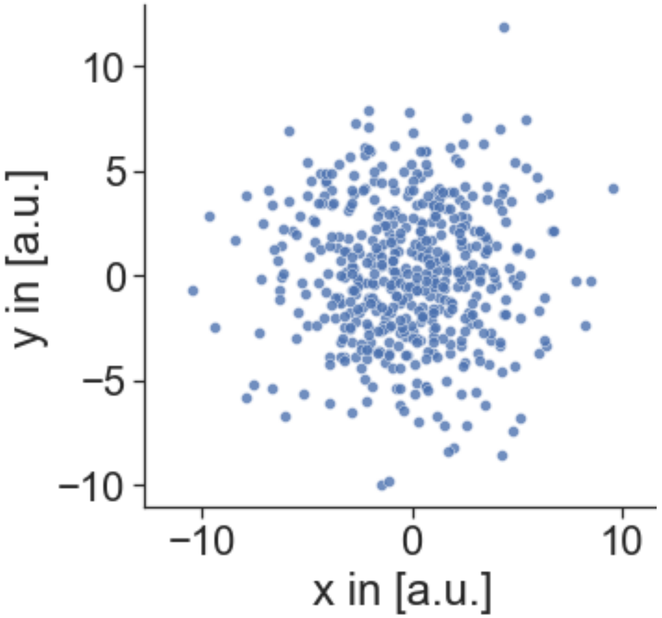
Principal component analysis (PCA)

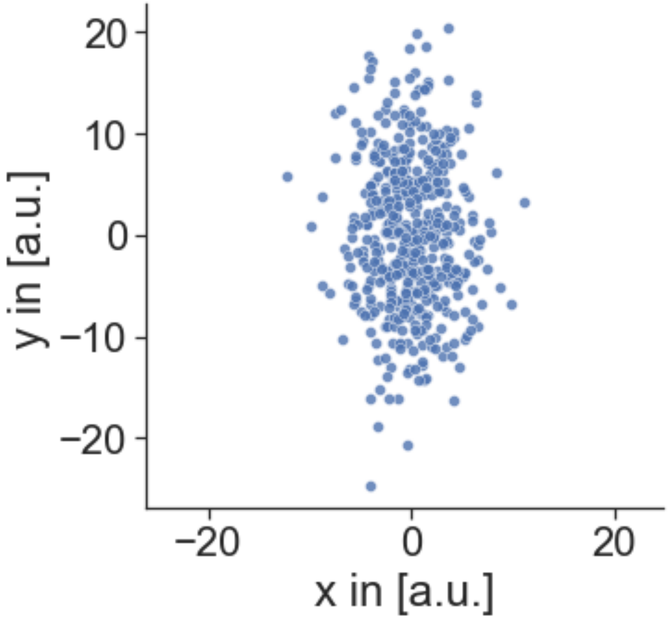
- Principle component analysis finds the **dominant component** in the feature space
- A regression finds the **correlation** of the data in feature space

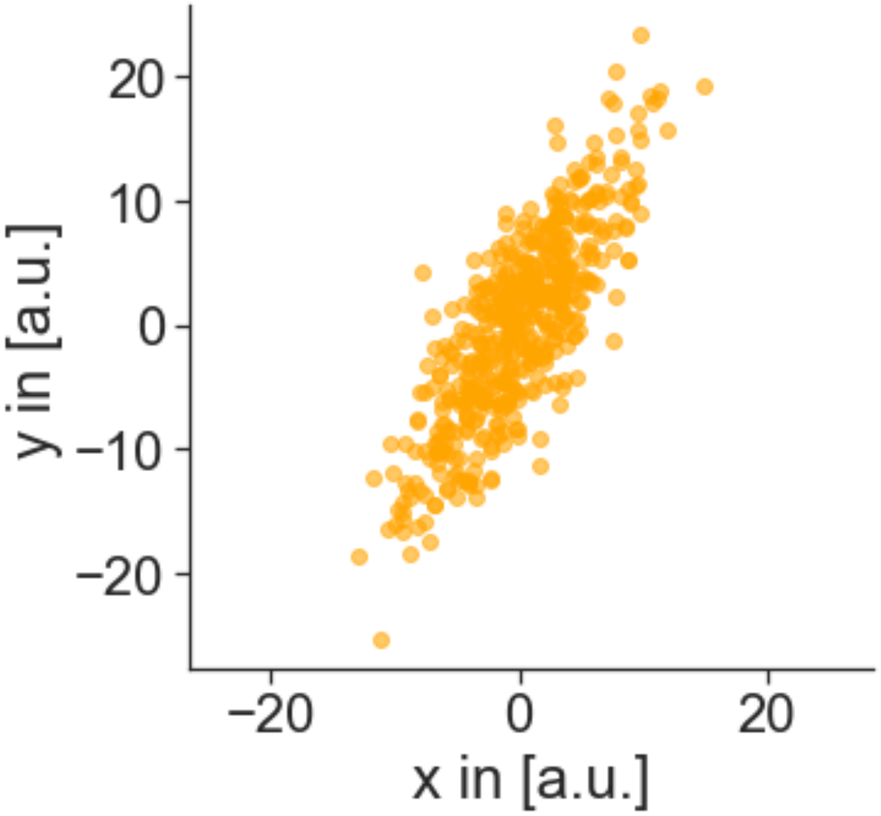


Principal component analysis (PCA)

- Principle component analysis finds the **dominant component** in the feature space
- A regression finds the **correlation** of the data in feature space

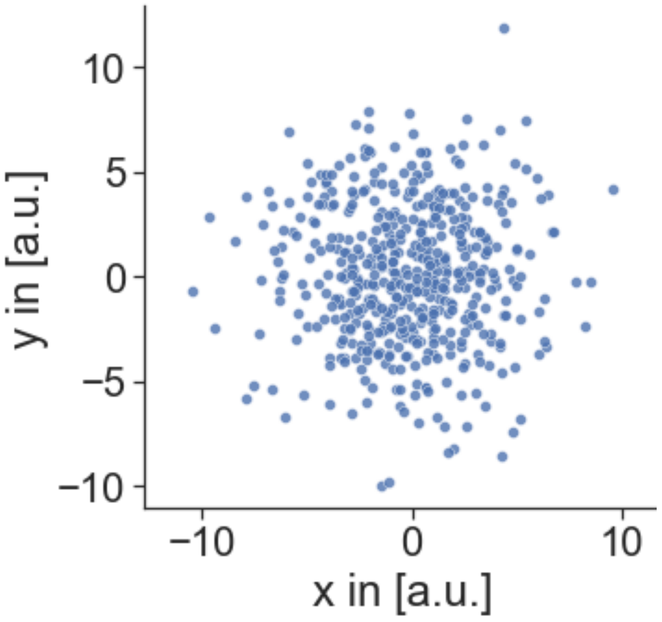


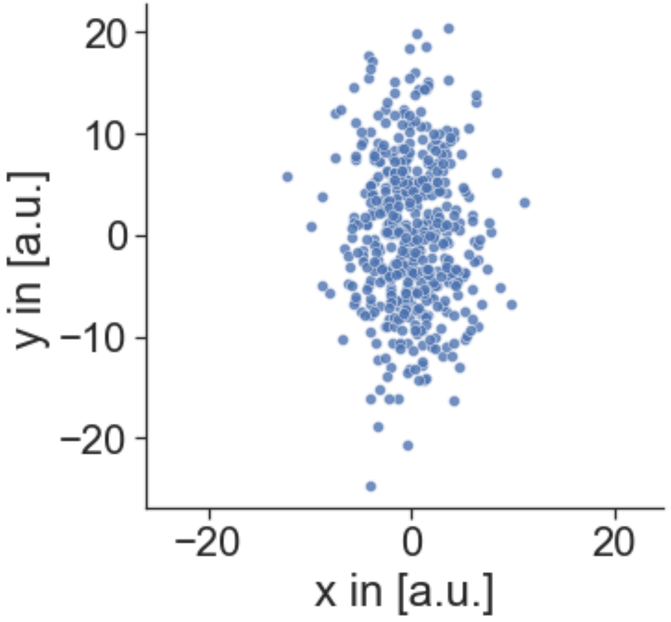


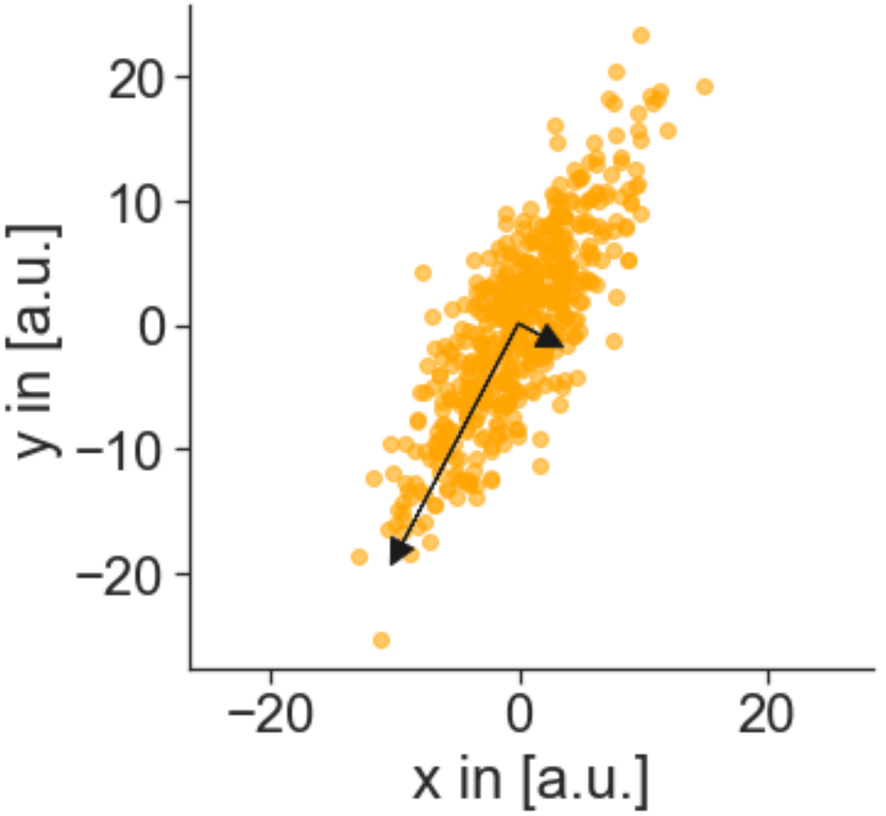


Principal component analysis (PCA)

- Principle component analysis finds the **dominant component** in the feature space
- A regression finds the **correlation** of the data in feature space

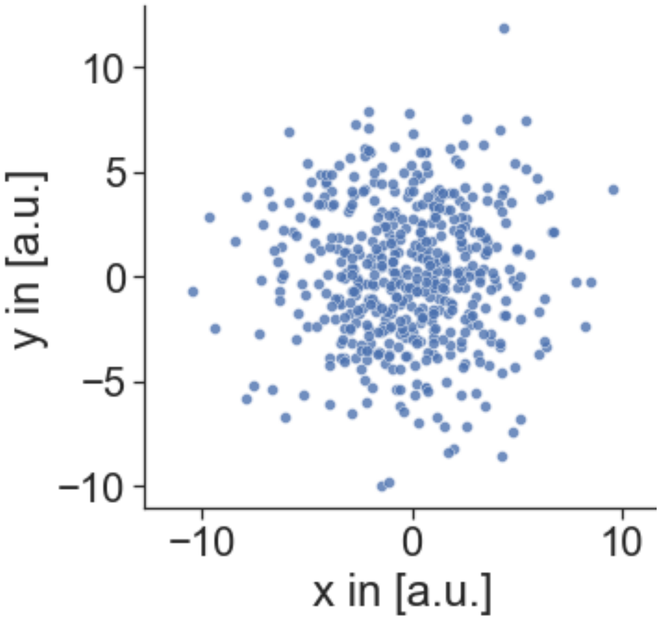


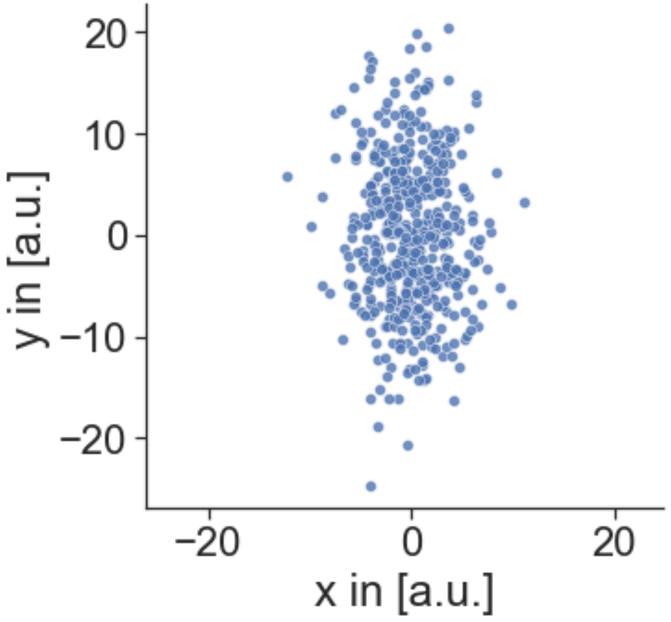


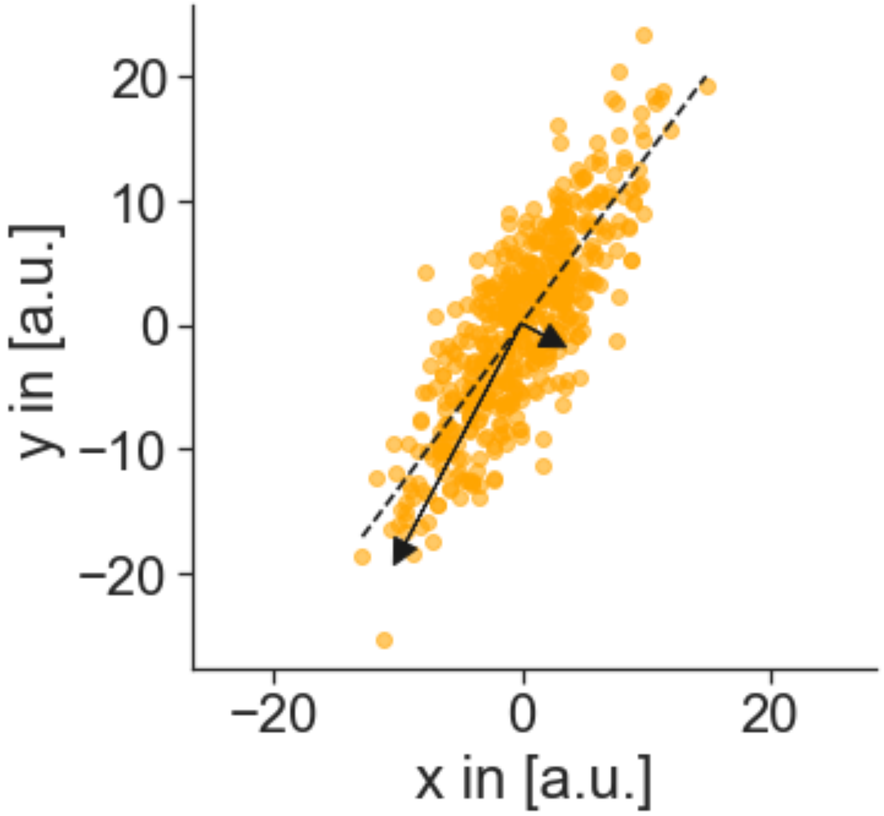


Principal component analysis (PCA)

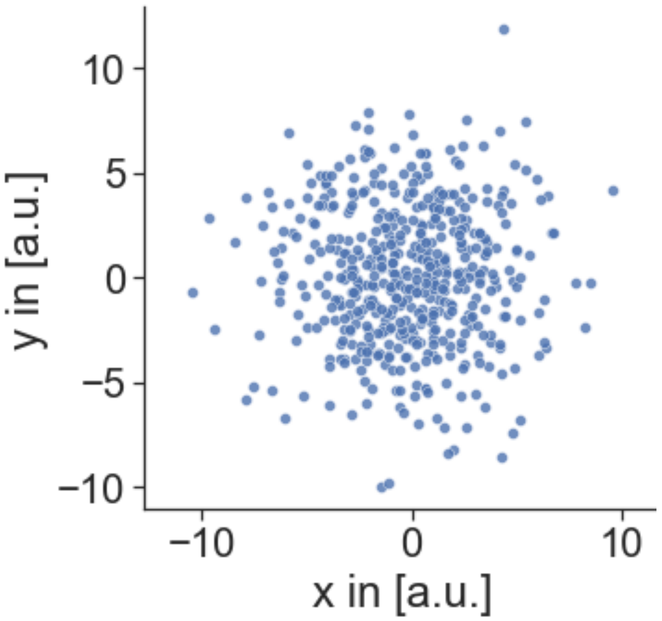
- Principle component analysis finds the **dominant component** in the feature space
- A regression finds the **correlation** of the data in feature space

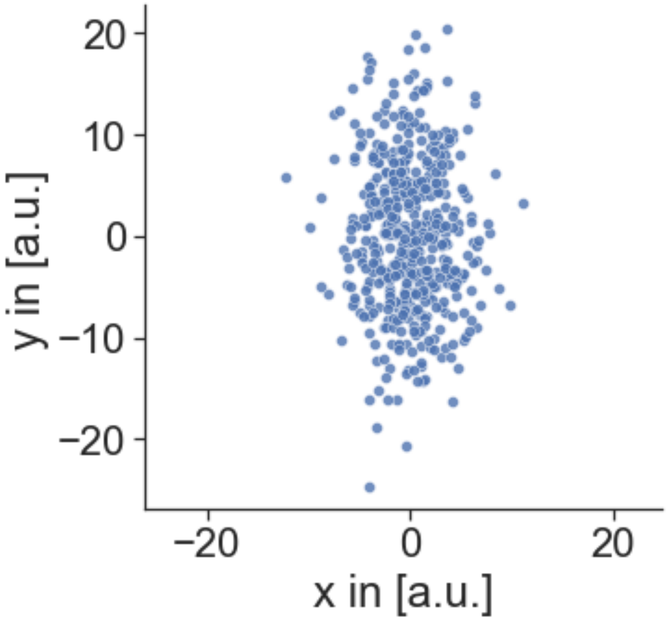


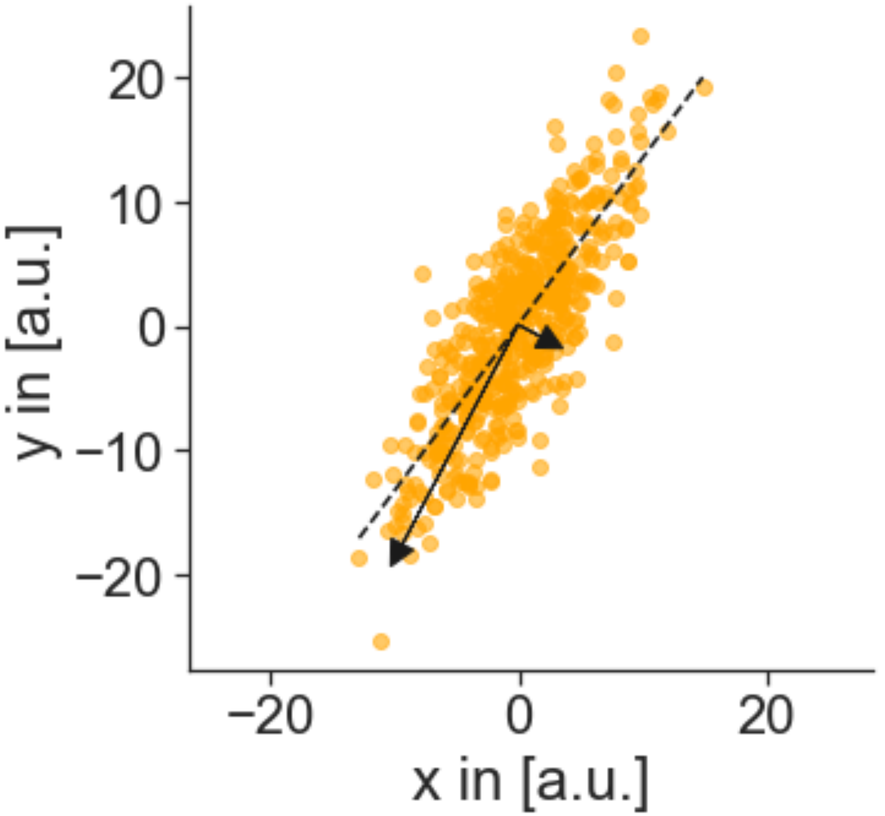




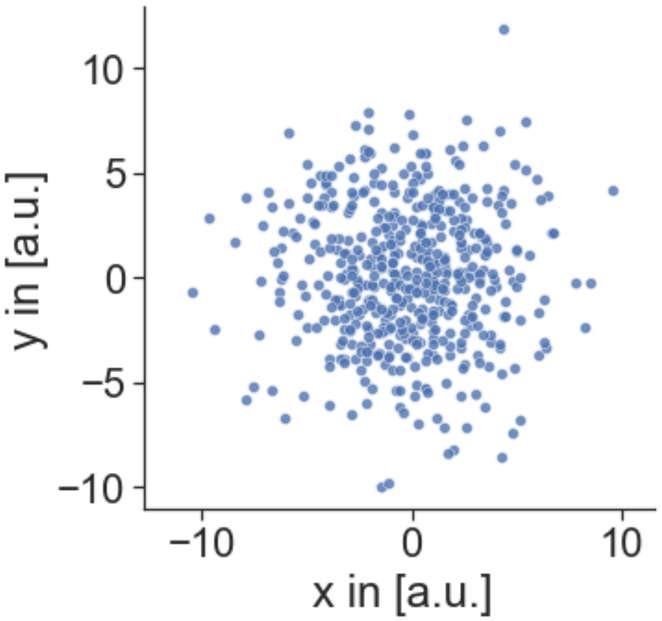
Principal component analysis (PCA)

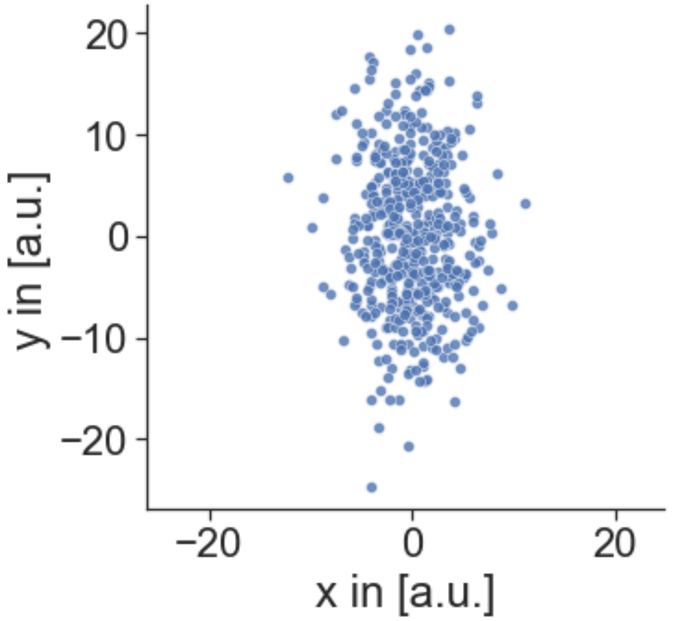


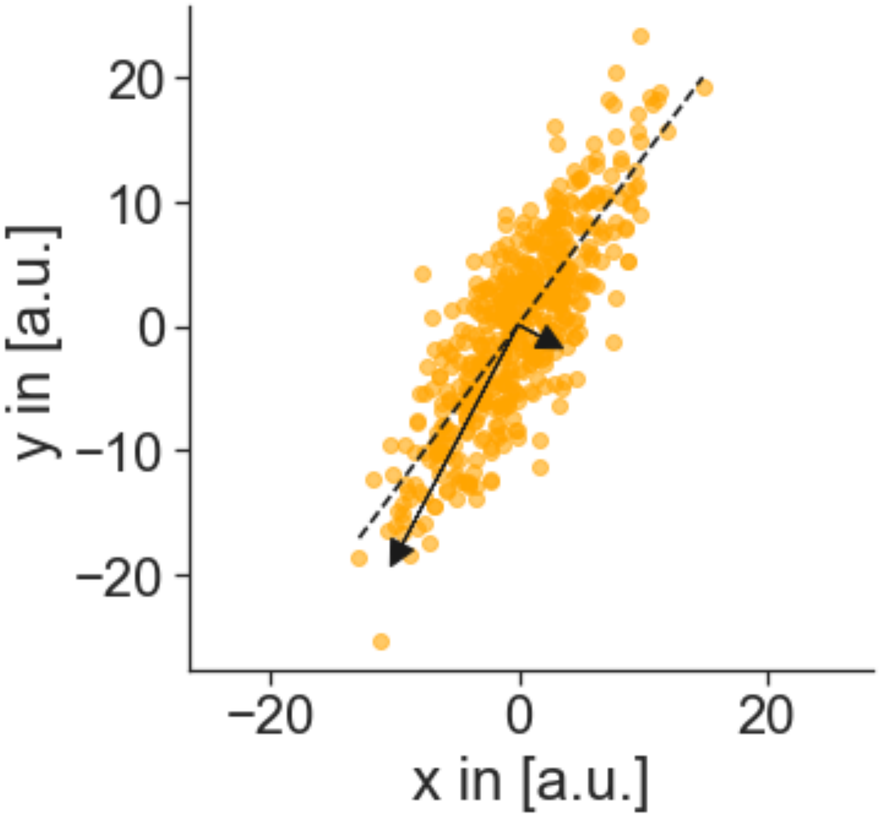




Principal component analysis (PCA)

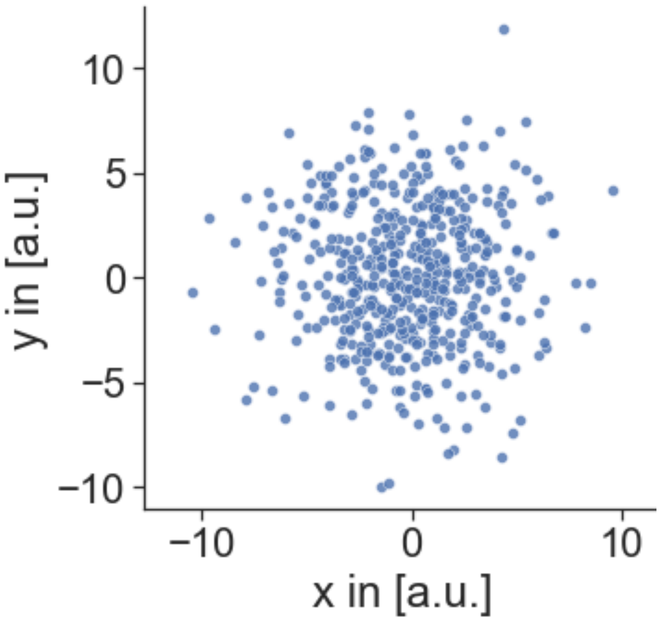


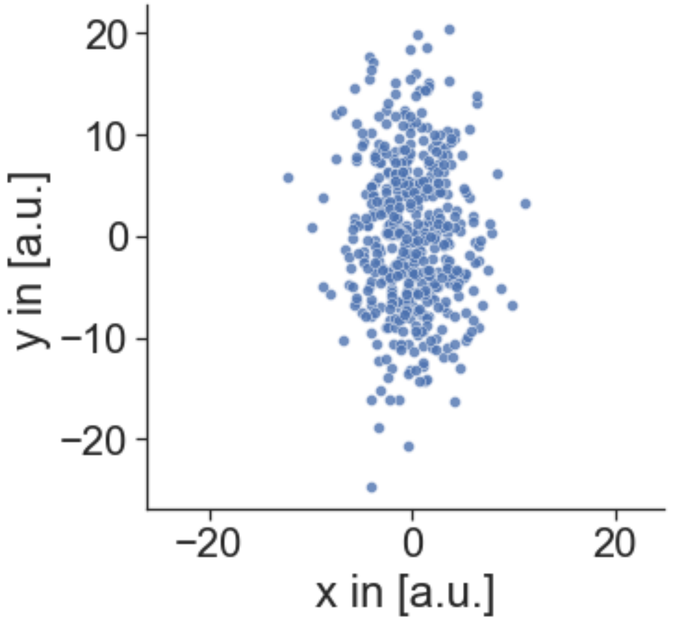


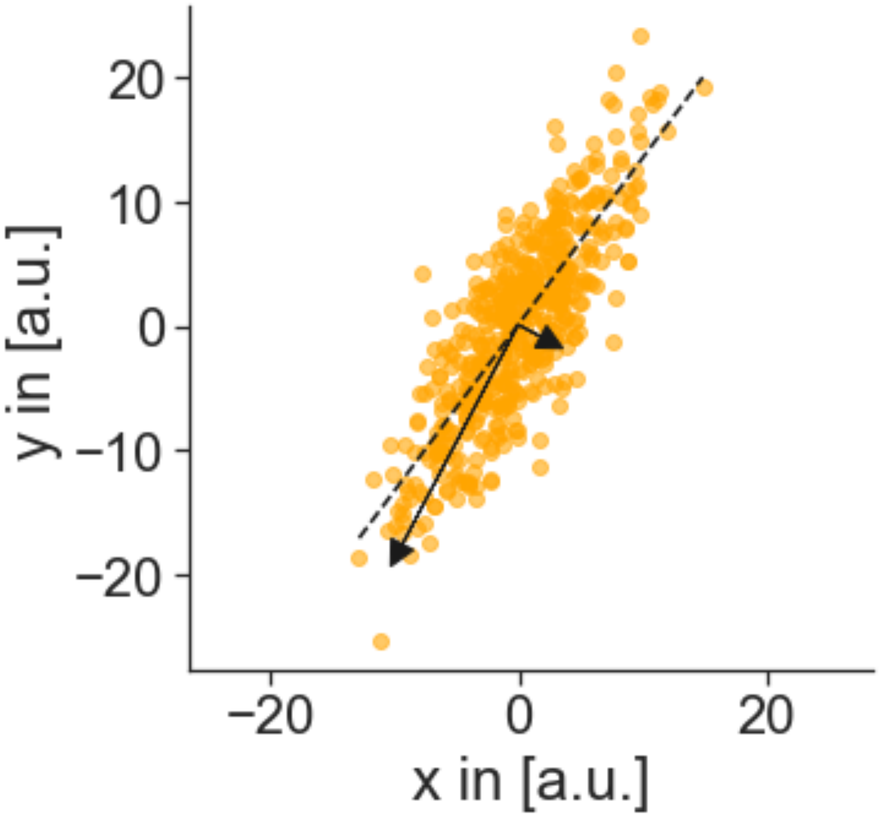


- PCA is an **orthogonal linear transformation** that **maximises the variance** across the first component
- A linear regression fit **minimises the error** with regard to all data points.
- PCA can be used as a tool for **dimensionality reduction**

Principal component analysis (PCA)

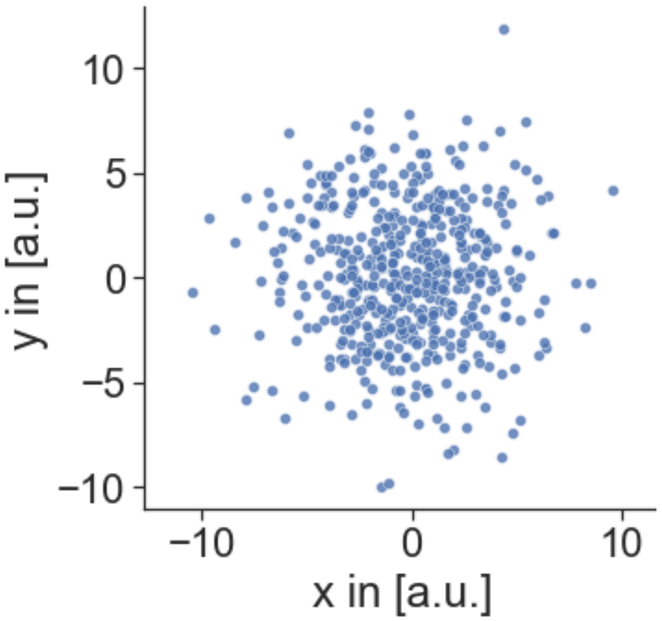


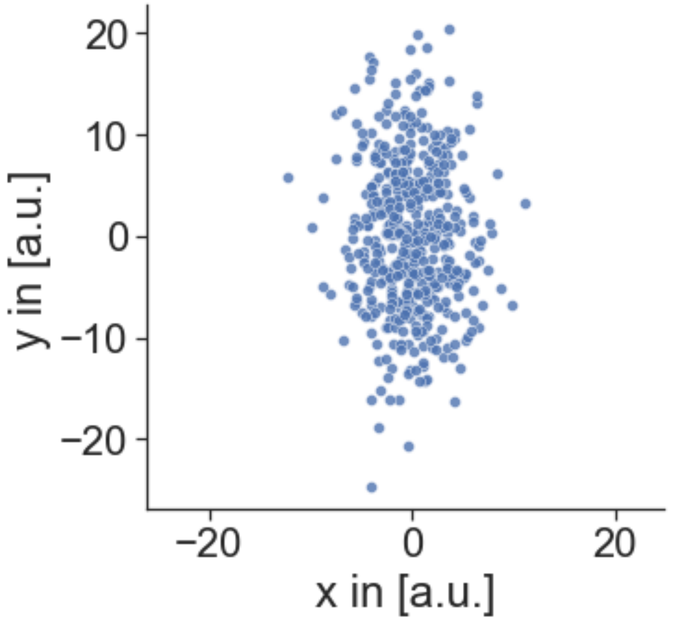


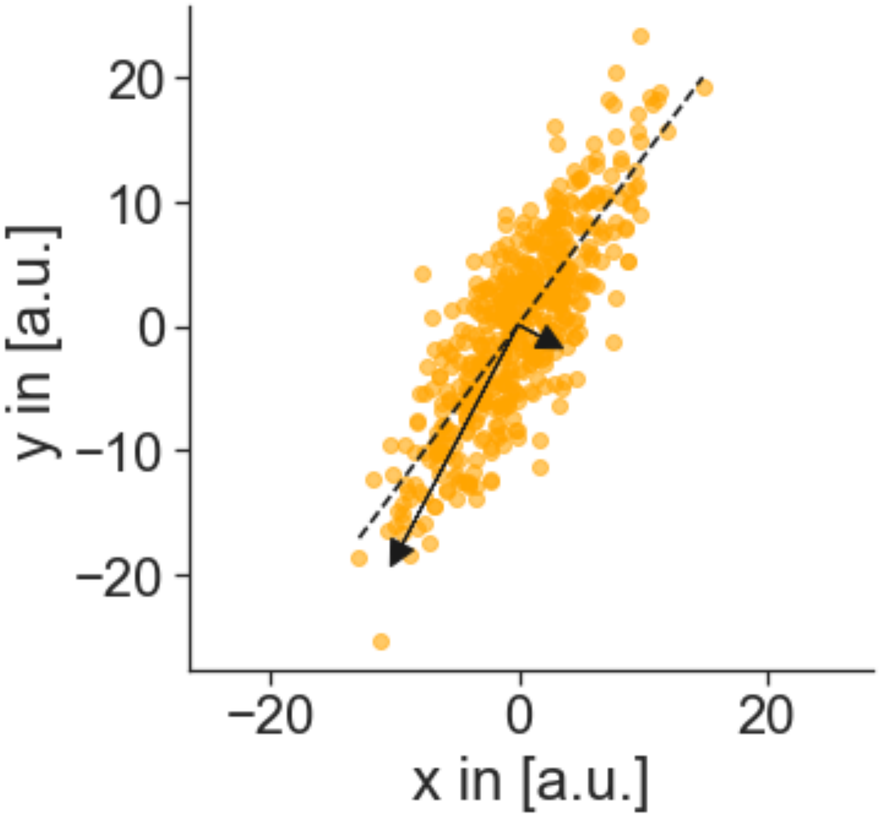


- PCA is an **orthogonal linear transformation** that **maximises the variance** across the first component
- A linear regression fit **minimises the error** with regard to all data points.
- PCA can be used as a tool for **dimensionality reduction**

Principal component analysis (PCA)







- PCA is an **orthogonal linear transformation** that **maximises the variance** across the first component
- A linear regression fit **minimises the error** with regard to all data points.
- PCA can be used as a tool for **dimensionality reduction**

Principal component analysis (PCA)

- PCA is an **orthogonal linear transformation** that **maximises the variance** across the first component
- A linear regression fit **minimises the error** with regard to all data points.
- PCA can be used as a tool for **dimensionality reduction**

Principal component analysis (PCA)

- PCA is an **orthogonal linear transformation** that **maximises the variance** across the first component
- A linear regression fit **minimises the error** with regard to all data points.
- PCA can be used as a tool for **dimensionality reduction**

$\mathbf{C}(0)$ Is the covariance matrix of the data **X**

Solving the generalised eigenvalue problem

$$\mathbf{C}(0)\mathbf{W} = \mathbf{W}\boldsymbol{\Sigma}$$

Gives an eigenvector matrix **W** that will allow the transform of the original data **X** onto a new basis **T** that maximises the variance.

$$\mathbf{T} = \mathbf{X}\mathbf{W}$$

It is possible to choose m eigenvectors to project onto by only using the first m-columns of **W**.

Principal component analysis (PCA)

- PCA is an **orthogonal linear transformation** that **maximises the variance** across the first component
- A linear regression fit **minimises the error** with regard to all data points.
- PCA can be used as a tool for **dimensionality reduction**

$\mathbf{C}(0)$ Is the covariance matrix of the data **X**

Solving the generalised eigenvalue problem

$$\mathbf{C}(0)\mathbf{W} = \mathbf{W}\boldsymbol{\Sigma}$$

Gives an eigenvector matrix **W** that will allow the transform of the original data **X** onto a new basis **T** that maximises the variance.

$$\mathbf{T} = \mathbf{X}\mathbf{W}$$

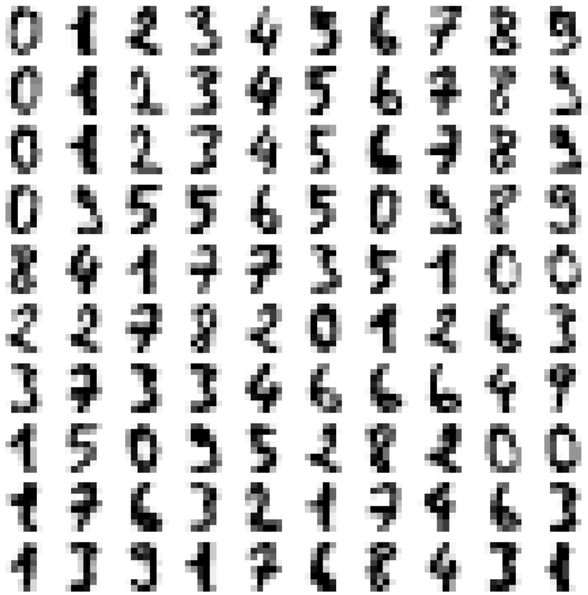
It is possible to choose m eigenvectors to project onto by only using the first m-columns of **W**.

The would be the 'dominant' features!

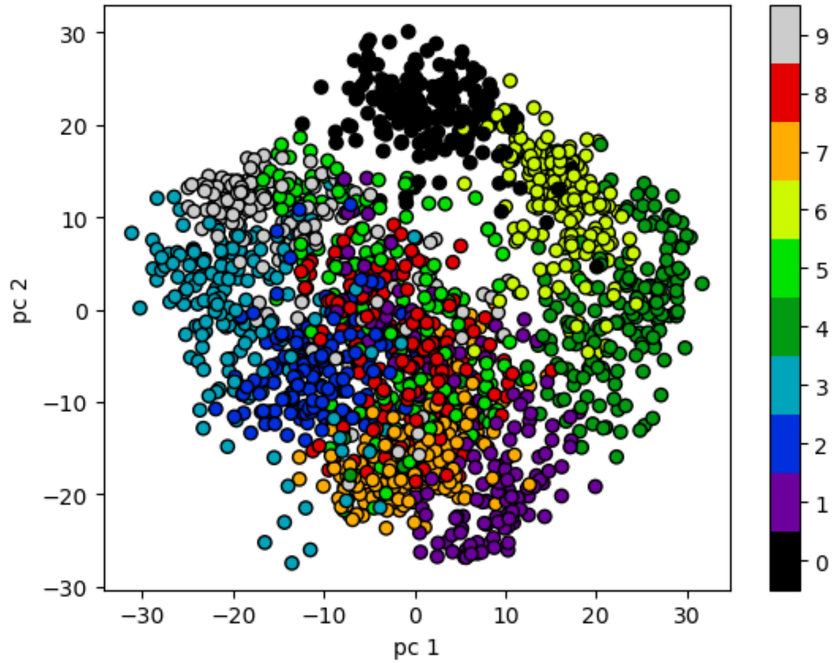
PCA example

- PCA is an **orthogonal linear transformation** that **maximises the variance** across the first component
- A linear regression fit **minimises the error** with regard to all data points.
- PCA can be used as a tool for **dimensionality reduction**

MNIST database of written digits



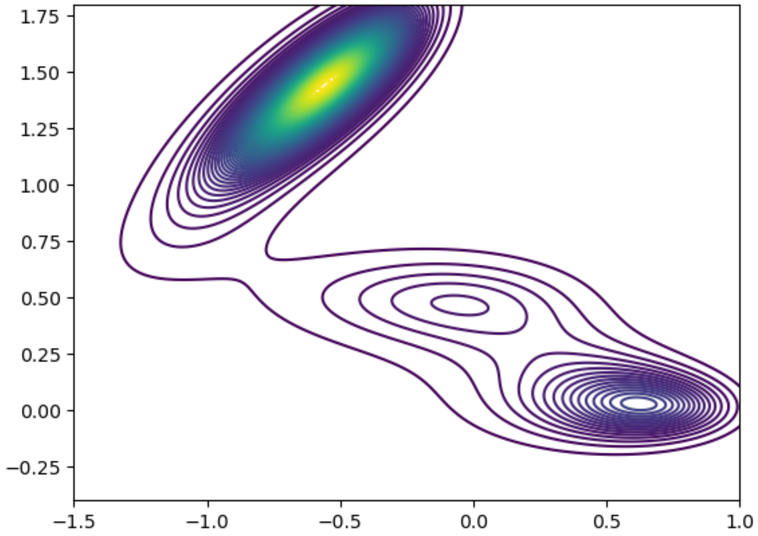
Input is a vector of 64 dimensions 8x8 pixel digits

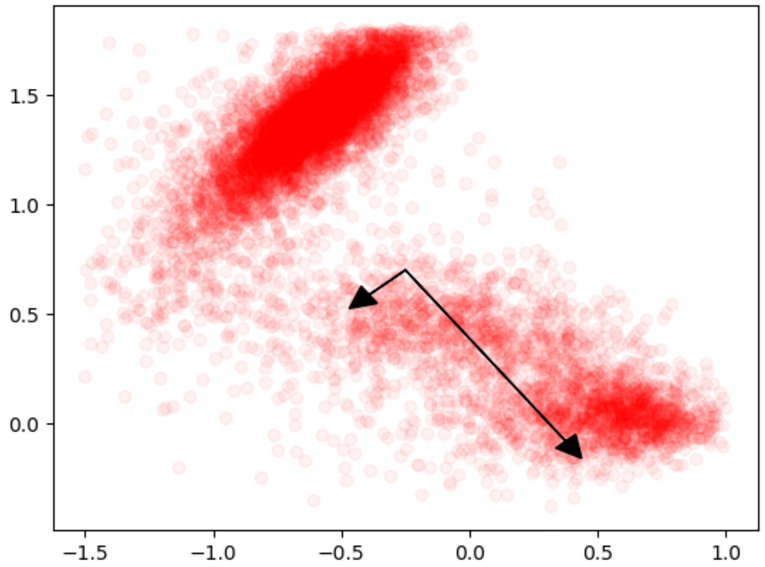


2 principle components manage to project a few of the digits in a similar area of space!

PCA example

- PCA is an **orthogonal linear transformation** that **maximises the variance** across the first component
- A linear regression fit **minimises the error** with regard to all data points.
- PCA can be used as a tool for **dimensionality reduction**



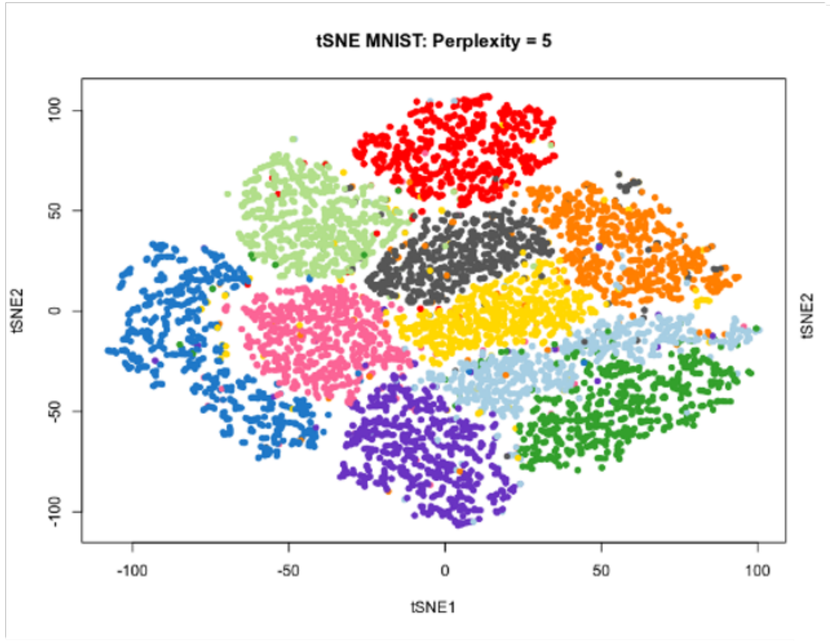


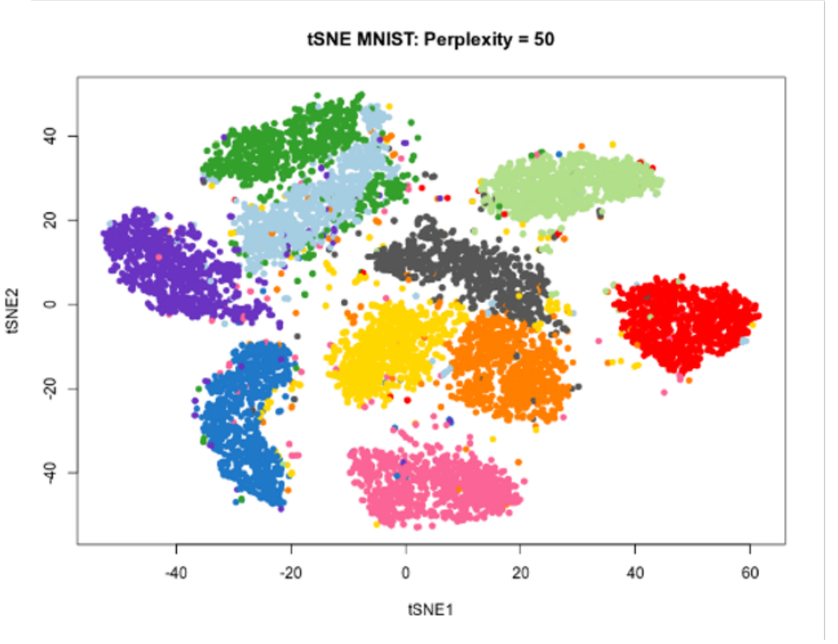
Finding the dominant reaction coordinate of a potential energy surface of a reaction!

T-distributed Stochastic Neighbour Embedding (t-SNE)

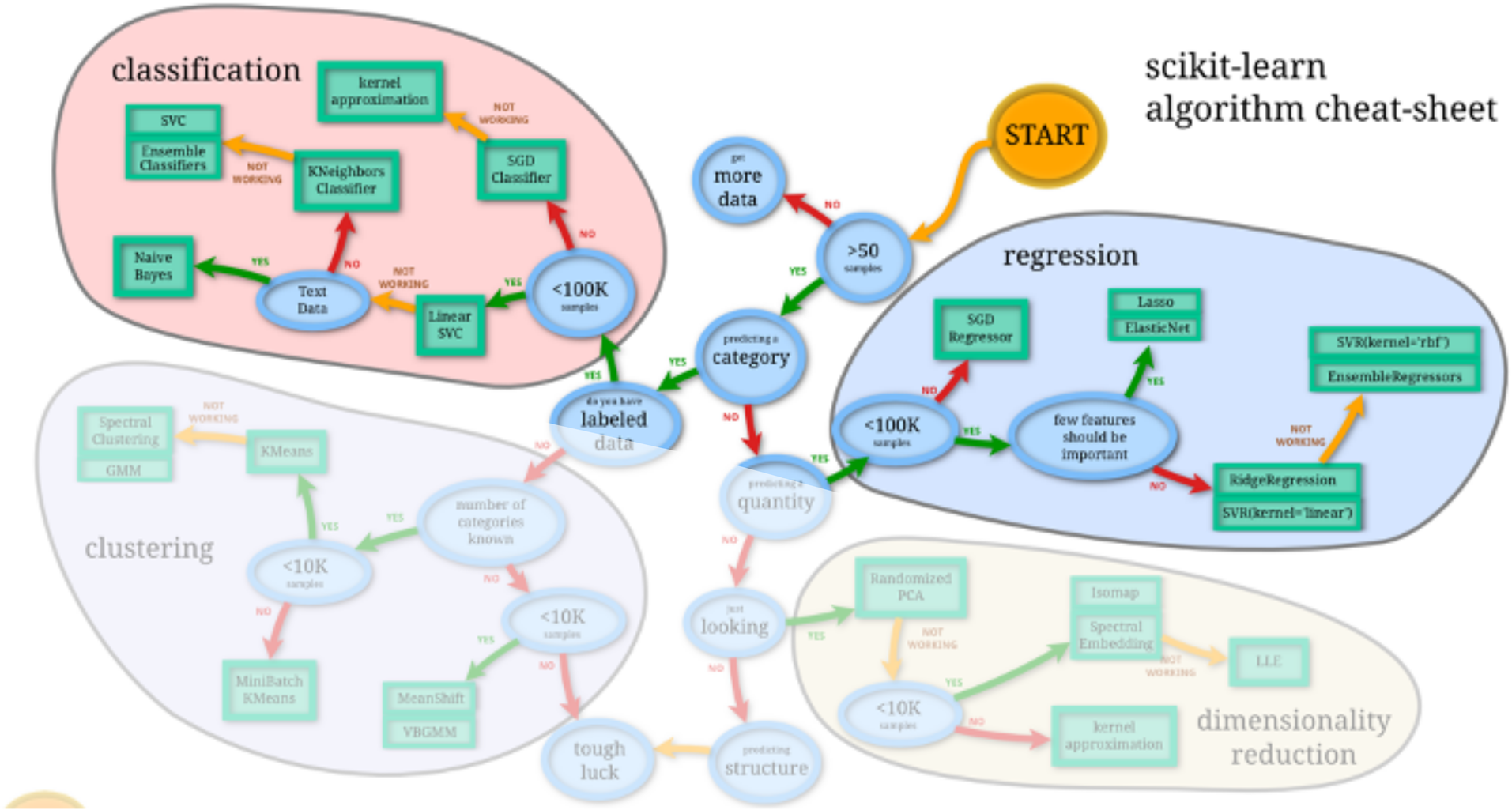
- Useful for visualisation, project high-dimensional data in 2 or 3 dimensions.
- Controlled by one main parameters: “perplexity”
- Relative distance between points not quantitatively meaningful

MNIST: database of written digits





ML Algorithms for classification and regression



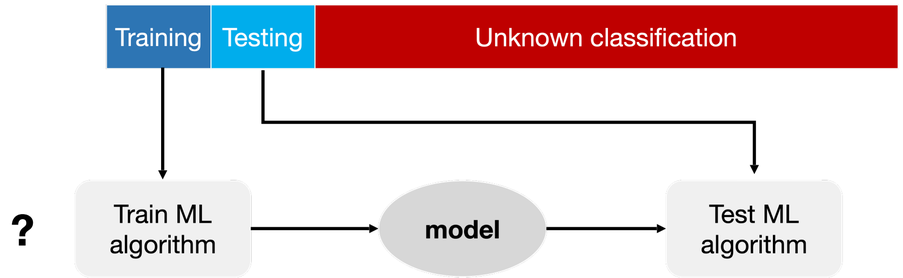
Learning Algorithms

Classification task

- **Decision Tree (DT)**
- **Random Forests (RF)**
- **Artificial Neural Network (ANN)**
- Support Vector Machine (SVM)
- Logistic Regression (LOGRES)
- Naïve Bayes (NB)
- K Nearest Neighbor (KNN)
- ...

Regression task

- **Decision Tree Regression (DT)**
- **Random Forests (RF)**
- **Neural Network Regressions (ANN)**
- Support Vector Machine (SVM)
- Linear Regression (LR)
- K Nearest Neighbor (KNN)
- Gaussian Regression
- ...



How do I pick the best learning algorithm?

Learning algorithms quality criteria:

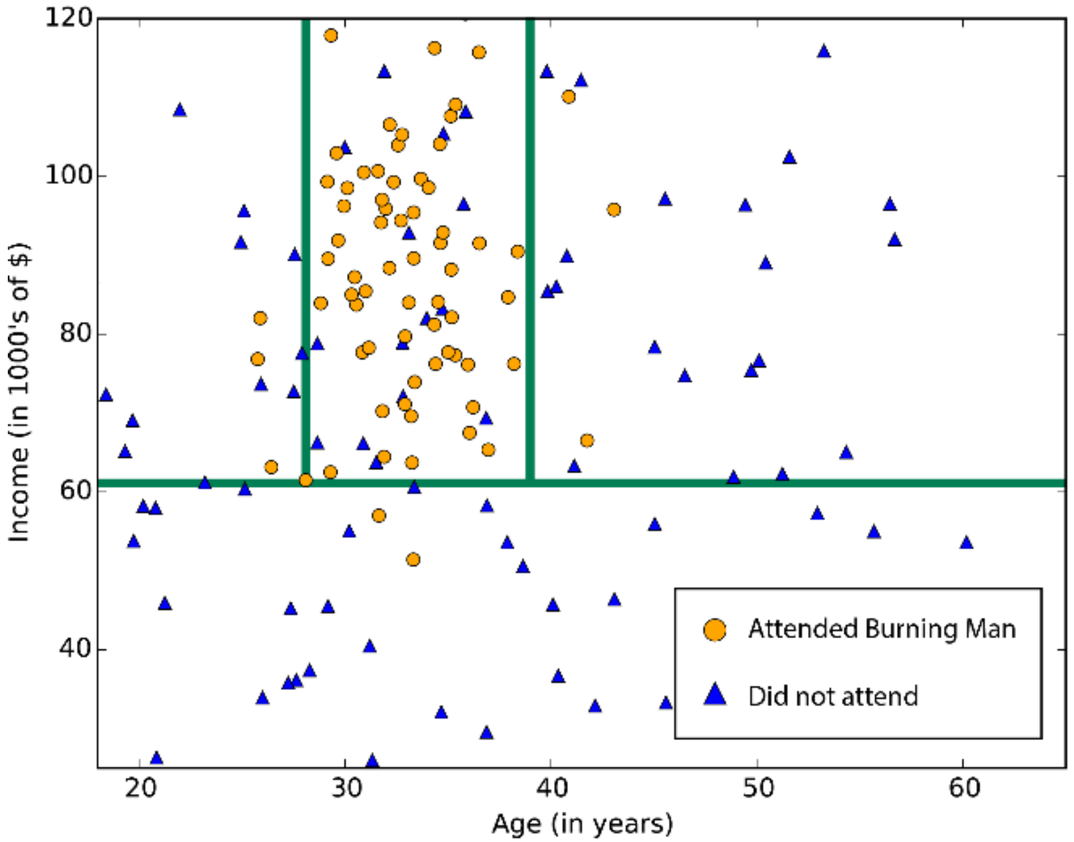
- **accuracy**: percentage of correct classification

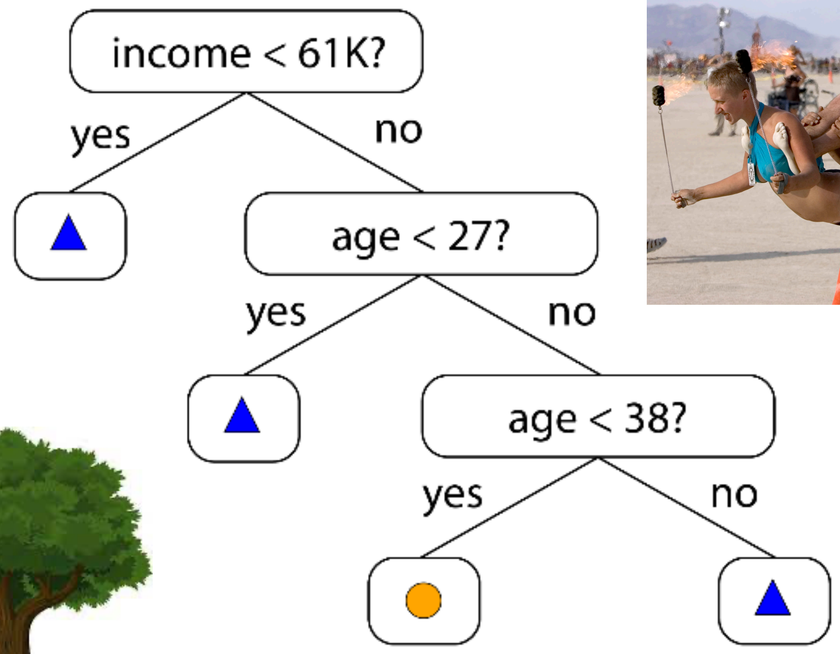
How do I pick the best learning algorithm?

Learning algorithms quality criteria:

- **accuracy**: percentage of correct classification
- **robustness**: handling noise and missing values
- **efficiency**: time to construct and use the model
- **scalability**: efficiency in memory requirements
- **interpretability**: how much the model is understandable

Decision Trees (DT)





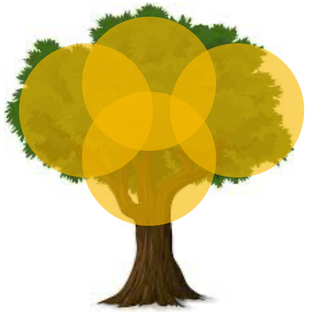
- Subdivides features space in sectors
- Can overfit if space subdivision becomes too fine

azure.microsoft.com/en-us/documentation/articles/machine-learning-algorithm-choice

Bootstrap Aggregating (Bagging)

training set *D* with *m* examples

D= | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |



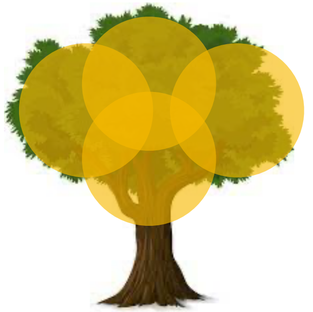
L. Breiman, *Bagging Predictors,* in *Machine* Learning, Springer, 1996

Bootstrap Aggregating (Bagging)

A weighted sum of weak classifiers creates a single strong classifier

training set *D* with *m* examples

D= | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |



L. Breiman, *Bagging Predictors,* in *Machine* Learning, Springer, 1996

Bootstrap Aggregating (Bagging)

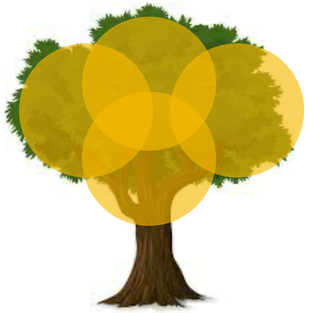
A weighted sum of weak classifiers creates a single strong classifier

Useful when a small change to training set causes large change in the output classifier (“learner is unstable”)

training set *D* with *m* examples

D=

1	2	3	4	5	6	7	8	9



L. Breiman, *Bagging Predictors,* in *Machine* Learning, Springer, 1996

Bootstrap Aggregating (Bagging)

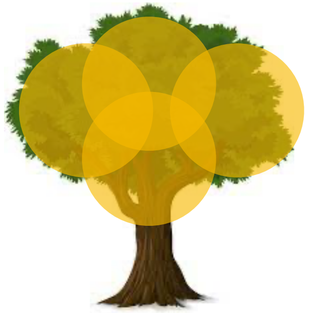
A weighted sum of weak classifiers creates a single strong classifier

Useful when a small change to training set causes large change in the output classifier ("learner is unstable")

training set *D* with *m* examples

D= | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Create *N* bootstrap samples S drawing *m* random examples from *D* *with replacement*



L. Breiman, *Bagging Predictors,* in *Machine* Learning, Springer, 1996

Bootstrap Aggregating (Bagging)

A weighted sum of weak classifiers creates a single strong classifier

Useful when a small change to training set causes large change in the output classifier (“learner is unstable”)

training set *D* with *m* examples

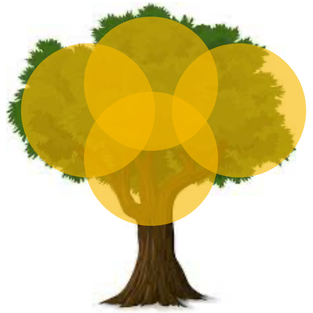
Create *N* bootstrap samples S drawing *m* random examples from *D* *with replacement*

D=

1	2	3	4	5	6	7	8	9

S[0]=

5	1	7	2	7	9	2	6	5



L. Breiman, *Bagging Predictors,* in *Machine* Learning, Springer, 1996

Bootstrap Aggregating (Bagging)

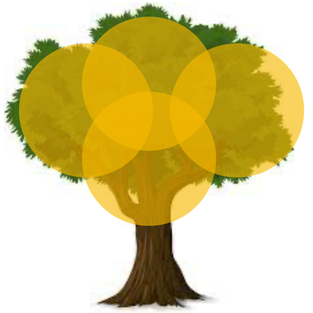
A weighted sum of weak classifiers creates a single strong classifier

Useful when a small change to training set causes large change in the output classifier ("learner is unstable")

training set *D* with *m* examples

Create *N* bootstrap samples S drawing *m* random examples from *D* *with replacement*

D=	1	2	3	4	5	6	7	8	9
S[0]=	5	1	7	2	7	9	2	6	5
S[1]=	9	4	7	1	2	8	9	7	6
S[2]=	0	8	2	0	9	7	7	0	1
	...								
S[*N*]=	1	2	3	4	5	6	7	8	9



L. Breiman, *Bagging Predictors,* in *Machine* Learning, Springer, 1996

Bootstrap Aggregating (Bagging)

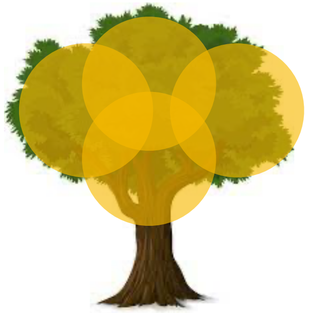
A weighted sum of weak classifiers creates a single strong classifier

Useful when a small change to training set causes large change in the output classifier ("learner is unstable")

training set *D* with *m* examples

Create *N* bootstrap samples S drawing *m* random examples from *D* *with replacement*

D=	1	2	3	4	5	6	7	8	9
S[0]=	5	1	7	2	7	9	2	6	5
S[1]=	9	4	7	1	2	8	9	7	6
S[2]=	0	8	2	0	9	7	7	0	1
	...								
S[*N*]=	1	2	3	4	5	6	7	8	9



Training: for every S, build a distinct classifier C using the same learning algorithm

L. Breiman, *Bagging Predictors,* in *Machine* Learning, Springer, 1996

Bootstrap Aggregating (Bagging)

A weighted sum of weak classifiers creates a single strong classifier

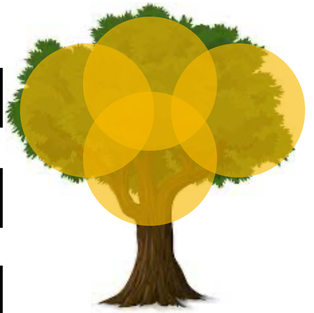
Useful when a small change to training set causes large change in the output classifier ("learner is unstable")

training set *D* with *m* examples

Create *N* bootstrap samples S drawing *m* random examples from *D* *with replacement*

D=	1	2	3	4	5	6	7	8	9
S[0]=	5	1	7	2	7	9	2	6	5
S[1]=	9	4	7	1	2	8	9	7	6
S[2]=	0	8	2	0	9	7	7	0	1
	...								
S[*N*]=	1	2	3	4	5	6	7	8	9

→C[0] →C[1] →C[2] →C[*N*]



Training: for every S, build a distinct classifier C using the same learning algorithm

L. Breiman, *Bagging Predictors,* in *Machine* Learning, Springer, 1996

Random Forests (RF)

- **Data bagging**: creates *N* decision trees trained on bagged data

L. Breiman, *Random Forests*, Machine Learning, 2001

Random Forests (RF)

- **Data bagging**: creates *N* decision trees trained on bagged data
- **Feature bagging**: Given *M* features, every tree learns on $m \ll M$ randomly selected features

L. Breiman, *Random Forests*, Machine Learning, 2001

Random Forests (RF)

- **Data bagging**: creates *N* decision trees trained on bagged data
- **Feature bagging**: Given *M* features, every tree learns on $m \ll M$ randomly selected features
- Classification based on **voting** of resulting *forest*

L. Breiman, *Random Forests*, Machine Learning, 2001

Random Forests (RF)

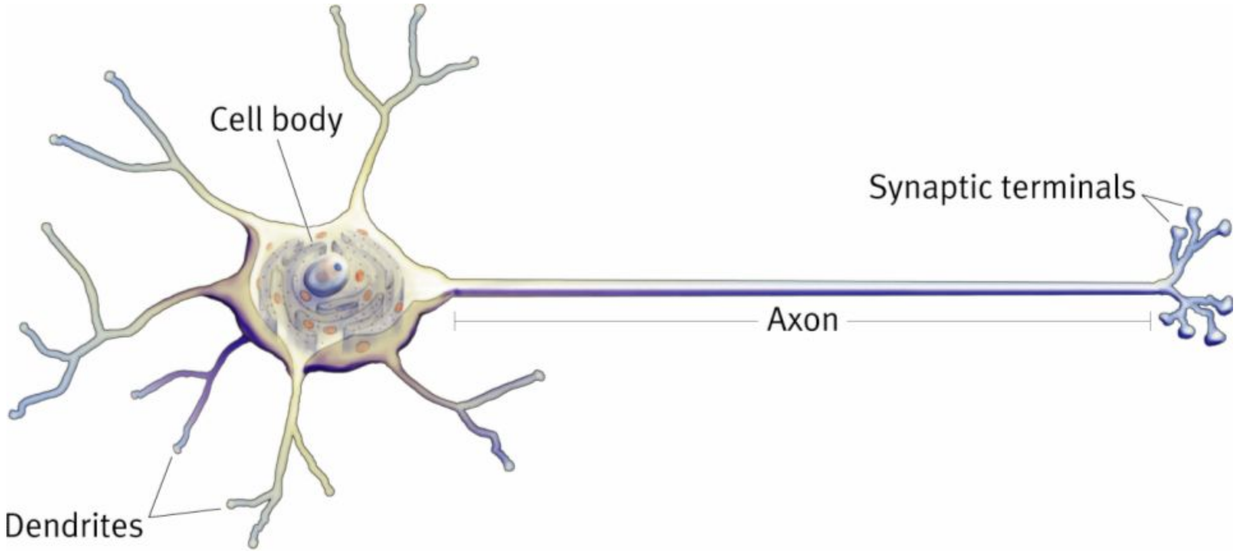
- **Data bagging**: creates *N* decision trees trained on bagged data
- **Feature bagging**: Given *M* features, every tree learns on $m \ll M$ randomly selected features
- Classification based on **voting** of resulting *forest*

Advantages:

- does *not* overfit
- Can handle thousands of features
- estimates what variables are important for classification

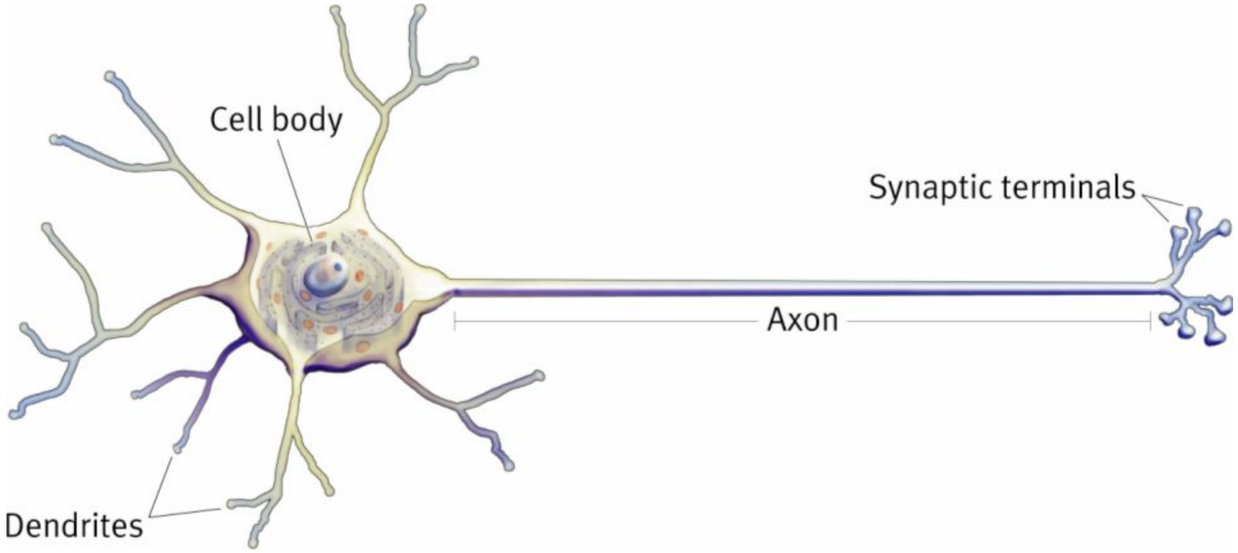
L. Breiman, *Random Forests*, Machine Learning, 2001

Artificial Neural Network



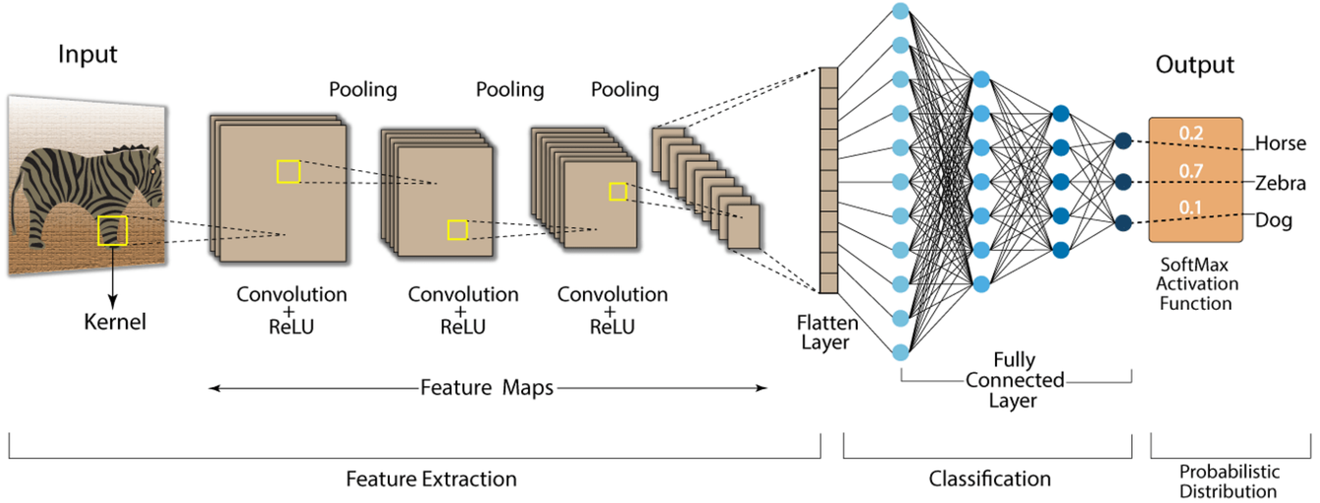
A **neuron** fires if input signal is above a threshold

Artificial Neural Network



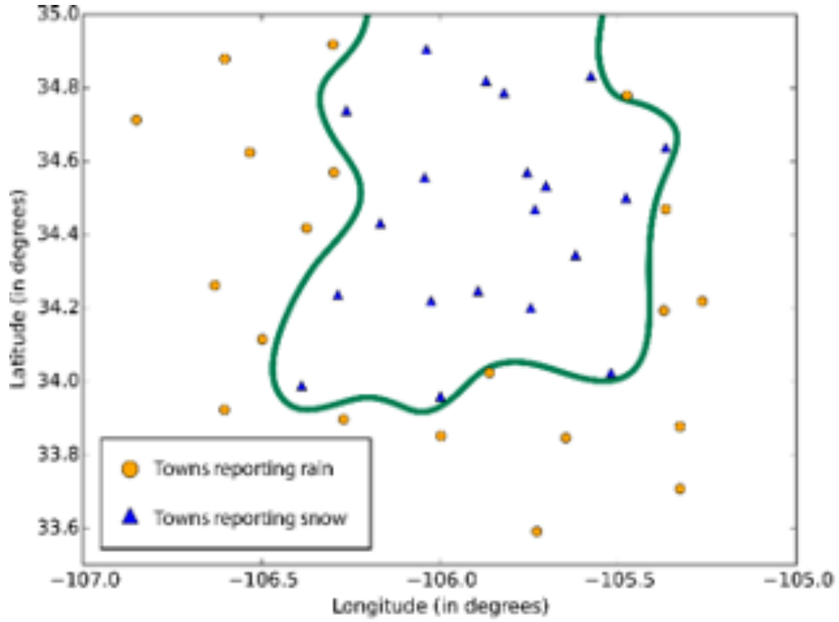
A **neuron** fires if input signal is above a threshold

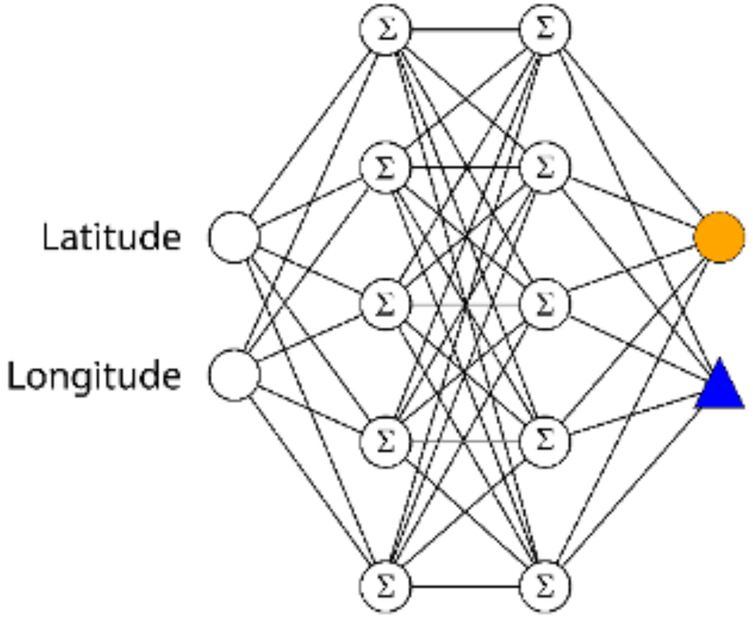
Many different complicated architectures are possible



Artificial Neural Network

Neurons can be arranged in **networks**
Hidden layers enable producing any complex boundary (in classification) or data fitting model (in regression)

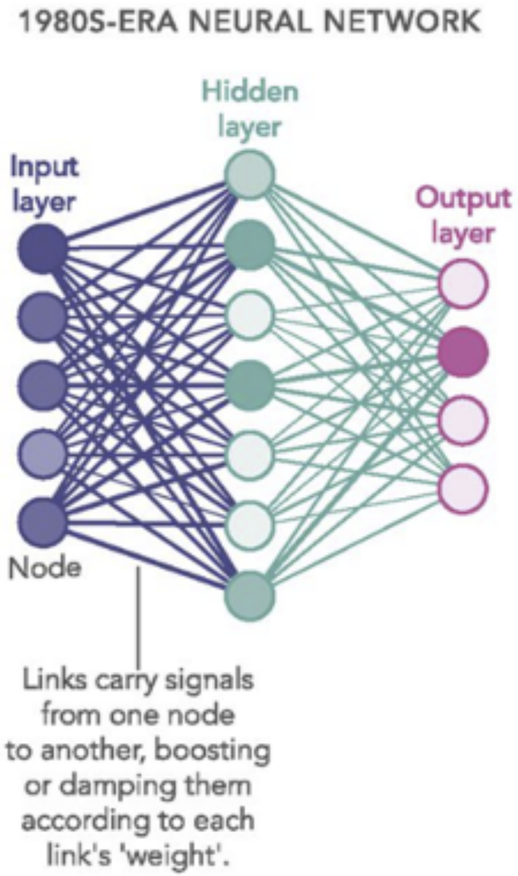


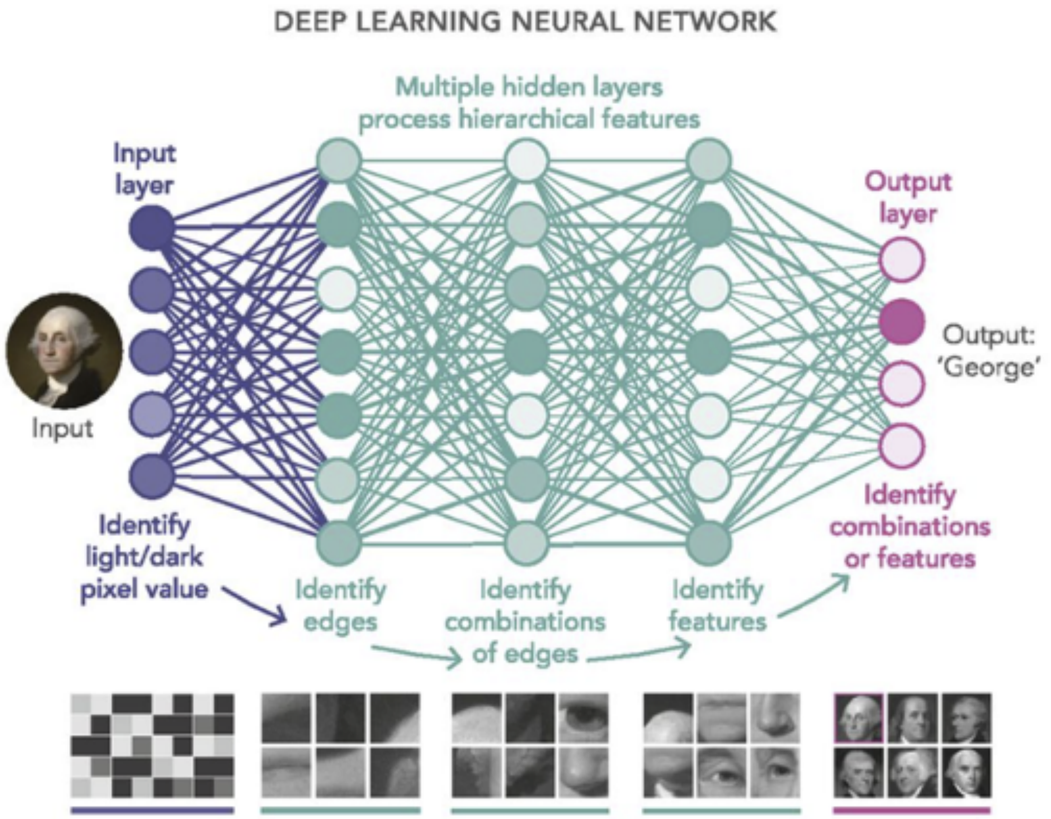


Training the ANN = finding synapses weights w_i minimizing error

ANN can fit any data, but are not easily interpretable

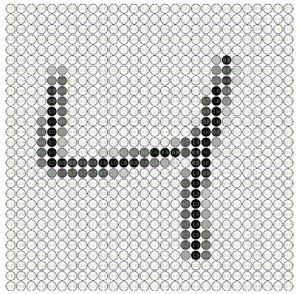
Multilayer perceptron





M.M. Waldrop, *News Feature: What are the limits of deep learning?*, ***PNAS***, 2019

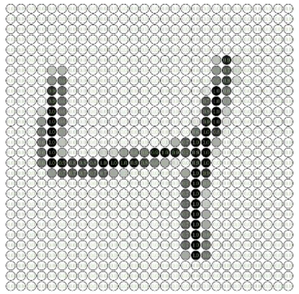
Multiple Output classification using a Multilayer Perceptron

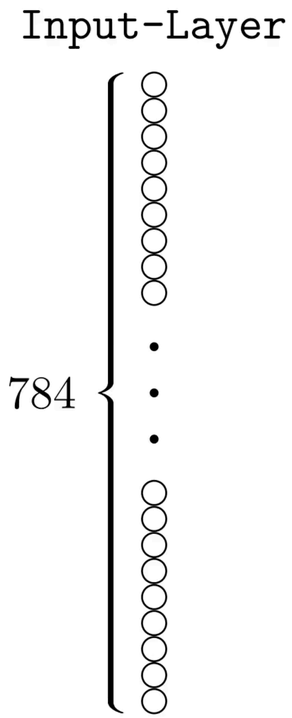


Artificial Neural Network (ANN)

4

Multiple Output classification using a Multilayer Perceptron

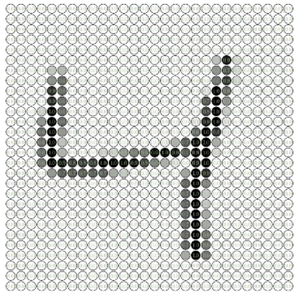


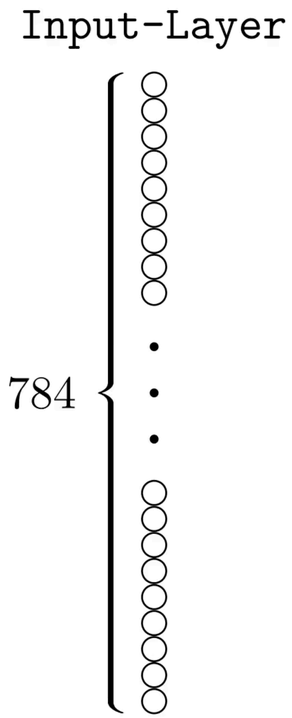


4

https://www.i-am.ai

Multiple Output classification using a Multilayer Perceptron

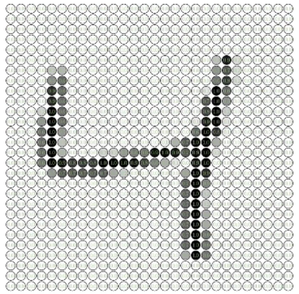


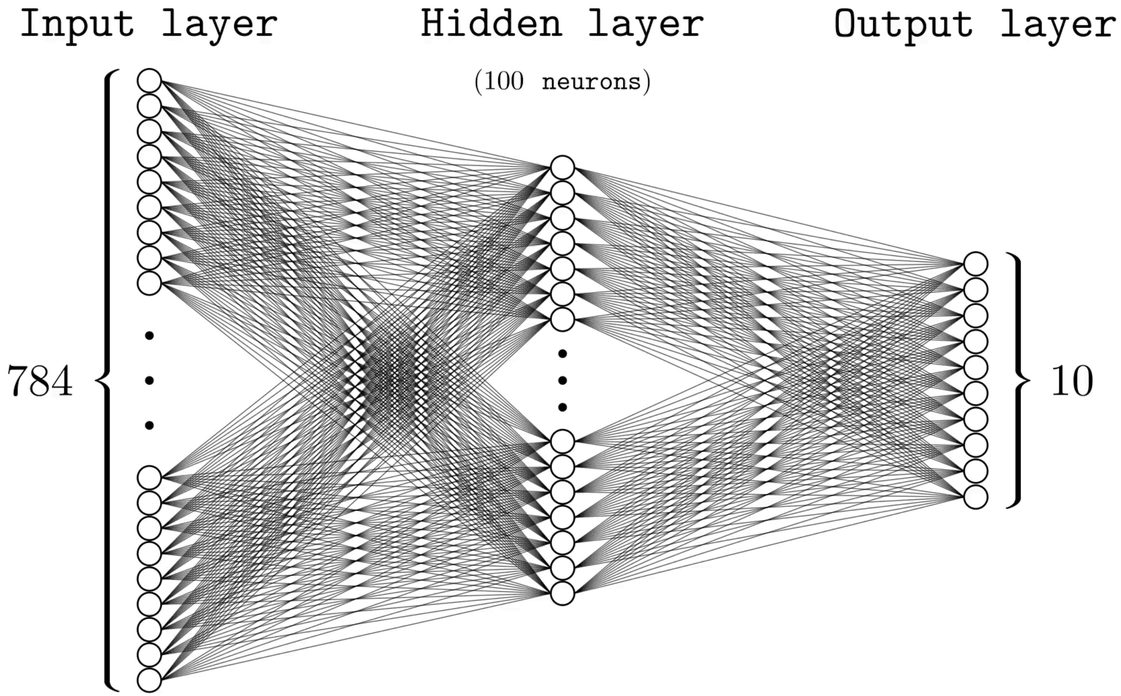


4

https://www.i-am.ai

Multiple Output classification using a Multilayer Perceptron

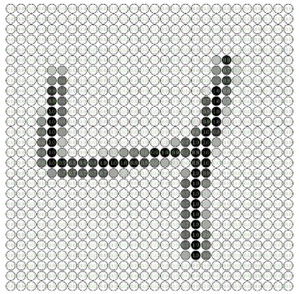


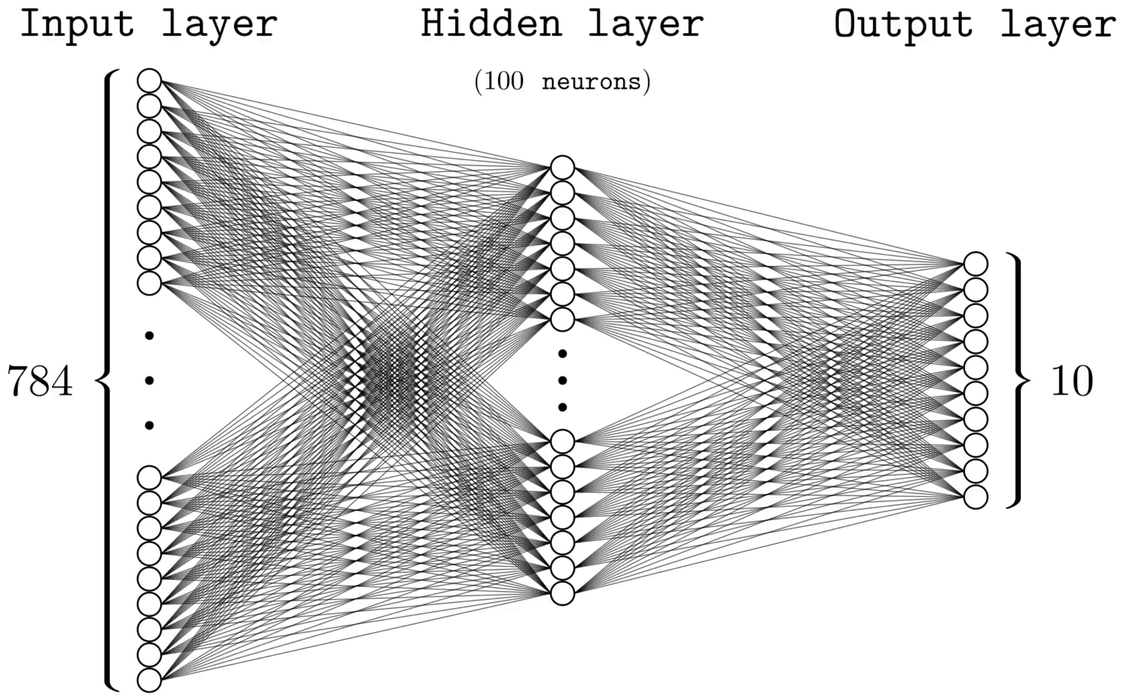


4

https://www.i-am.ai

Multiple Output classification using a Multilayer Perceptron





4

https://www.i-am.ai

Multilayer perceptron: the maths

Weighted input/output Weighted matrix input Bias

$$z^{(1)} = W^{(0)} a^{(0)} + b^{(1)}$$

$$a^{(l)} = \varphi^{(l)}(z^{(l)}) = \varphi^{(l)}\left(W^{(l-1)} a^{(l-1)} + b^{(l)}\right).$$

Activation function

Activation functions:

Multilayer perceptron: the maths

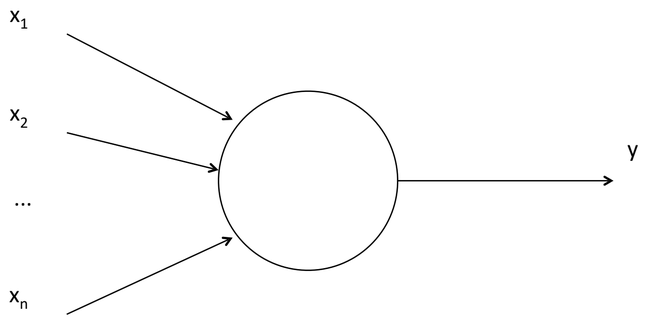
Weighted input/output Weighted matrix input Bias

$$z^{(1)} = W^{(0)} a^{(0)} + b^{(1)}$$

$$a^{(l)} = \varphi^{(l)}(z^{(l)}) = \varphi^{(l)}\left(W^{(l-1)} a^{(l-1)} + b^{(l)}\right).$$

Activation function

Activation functions:



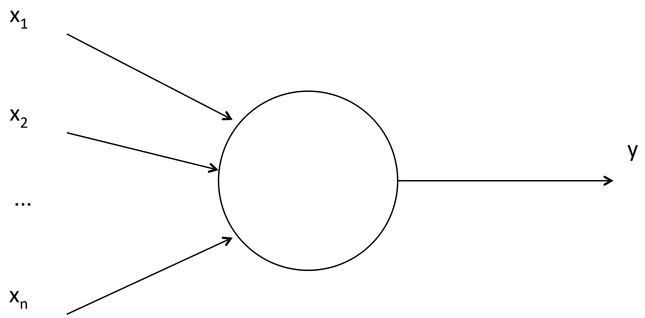
Multilayer perceptron: the maths

Weighted input/output Weighted matrix input Bias

$$z^{(1)} = W^{(0)} a^{(0)} + b^{(1)}$$

$$a^{(l)} = \varphi^{(l)}(z^{(l)}) = \varphi^{(l)}\left(W^{(l-1)} a^{(l-1)} + b^{(l)}\right).$$

Activation function



$f(a_1w_1 + a_2w_2 + ... + a_nw_n+b) = z$

Activation functions:

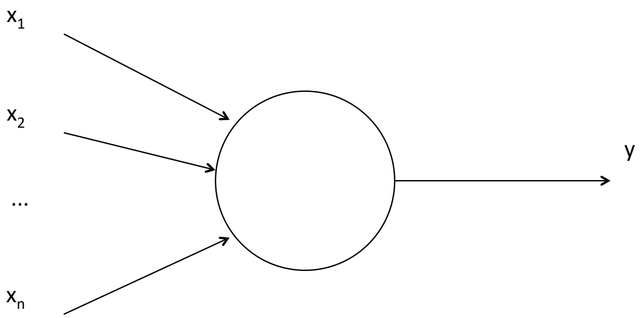
Multilayer perceptron: the maths

Weighted input/output Weighted matrix input Bias

$$z^{(1)} = W^{(0)} a^{(0)} + b^{(1)}$$

$$a^{(l)} = \varphi^{(l)}(z^{(l)}) = \varphi^{(l)}\left(W^{(l-1)} a^{(l-1)} + b^{(l)}\right).$$

Activation function

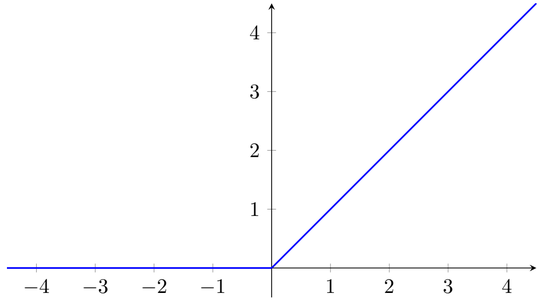


$$f(a_1w_1 + a_2w_2 + \ldots + a_nw_n + b) = z$$

Activation functions:

ReLu:

$$\mathrm{ReLU}(z) = \begin{cases} 0, & z \leq 0 \\ z, & z > 0 \end{cases}$$



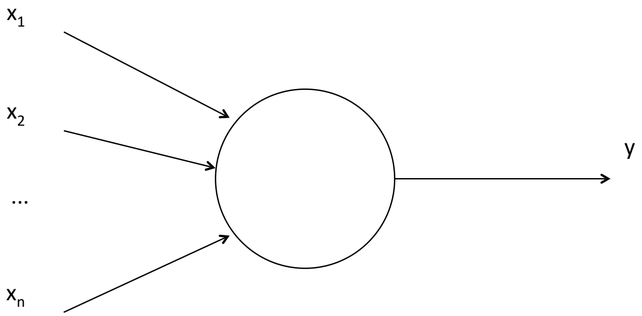
Multilayer perceptron: the maths

Weighted input/output Weighted matrix input Bias

$$z^{(1)} = W^{(0)} a^{(0)} + b^{(1)}$$

$$a^{(l)} = \varphi^{(l)}(z^{(l)}) = \varphi^{(l)}\left(W^{(l-1)} a^{(l-1)} + b^{(l)}\right).$$

Activation function

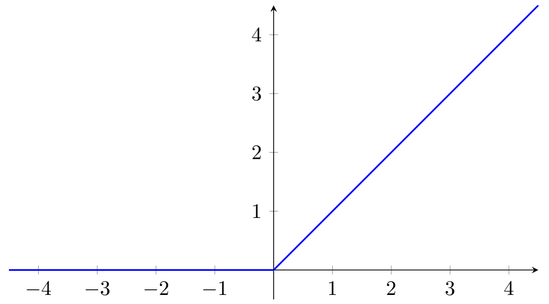


$$f(a_1w_1 + a_2w_2 + \ldots + a_nw_n + b) = z$$

Activation functions:

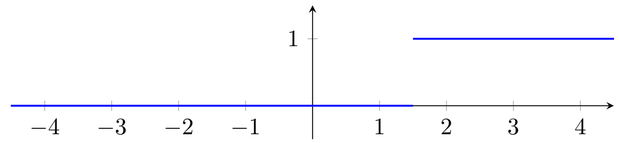
ReLu:

$$\mathrm{ReLU}(z) = \begin{cases} 0, & z \leq 0 \\ z, & z > 0 \end{cases}$$



Binary Threshold:

$$s_t(z) = \begin{cases} 0, & z \leq t \\ 1, & z > t \end{cases}$$



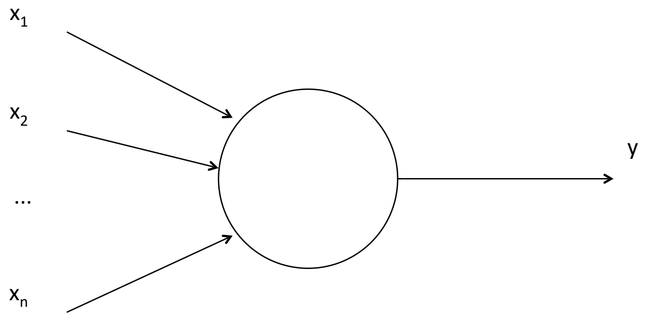
Multilayer perceptron: the maths

Weighted input/output Weighted matrix input Bias

$$z^{(1)} = W^{(0)} a^{(0)} + b^{(1)}$$

$$a^{(l)} = \varphi^{(l)}(z^{(l)}) = \varphi^{(l)}\left(W^{(l-1)} a^{(l-1)} + b^{(l)}\right).$$

Activation function

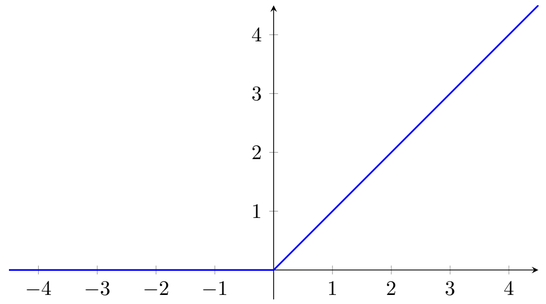


$$f(a_1w_1 + a_2w_2 + \ldots + a_nw_n + b) = z$$

Activation functions:

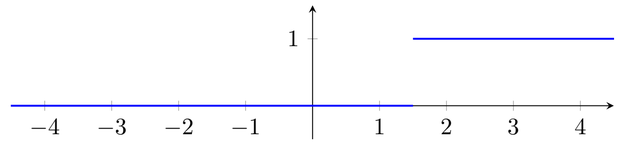
ReLu:

$$\mathrm{ReLU}(z) = \begin{cases} 0, & z \leq 0 \\ z, & z > 0 \end{cases}$$



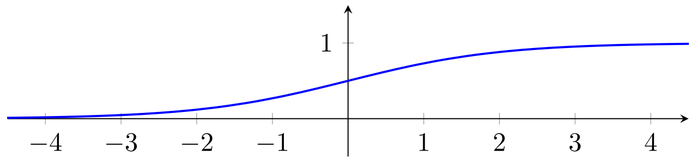
Binary Threshold:

$$s_t(z) = \begin{cases} 0, & z \leq t \\ 1, & z > t \end{cases}$$



Sigmoid:

$$\mathrm{sig}(z) = \frac{1}{1 + e^{-z}}$$



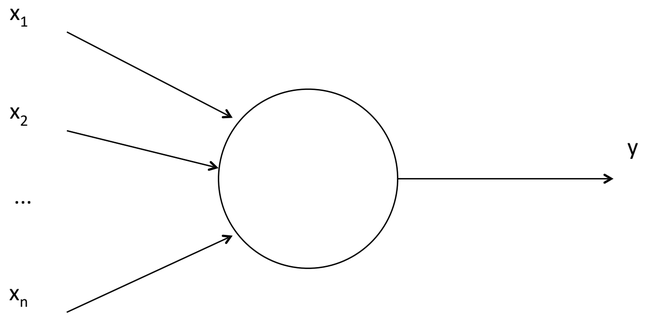
Multilayer perceptron: the maths

Weighted input/output Weighted matrix input Bias

$$z^{(1)} = W^{(0)} a^{(0)} + b^{(1)}$$

$$a^{(l)} = \varphi^{(l)}(z^{(l)}) = \varphi^{(l)}\left(W^{(l-1)} a^{(l-1)} + b^{(l)}\right).$$

Activation function

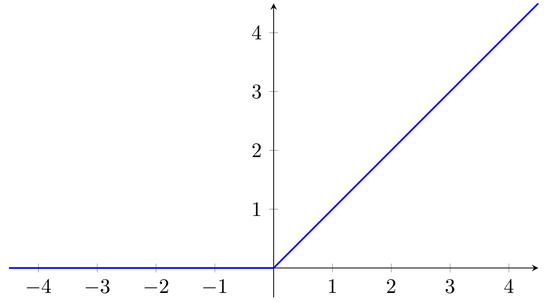


$$f(a_1w_1 + a_2w_2 + \ldots + a_nw_n + b) = z$$

Activation functions:

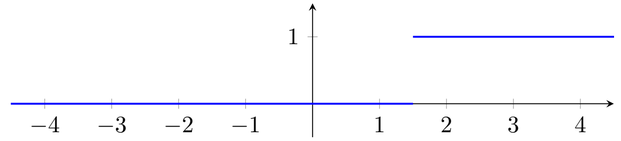
ReLu:

$$\mathrm{ReLU}(z) = \begin{cases} 0, & z \leq 0 \\ z, & z > 0 \end{cases}$$



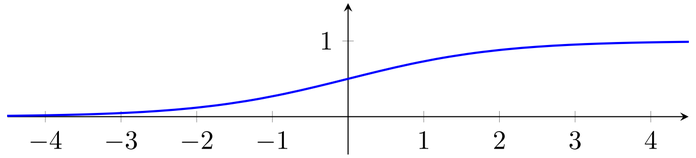
Binary Threshold:

$$s_t(z) = \begin{cases} 0, & z \leq t \\ 1, & z > t \end{cases}$$



Sigmoid:

$$\mathrm{sig}(z) = \frac{1}{1 + e^{-z}}$$



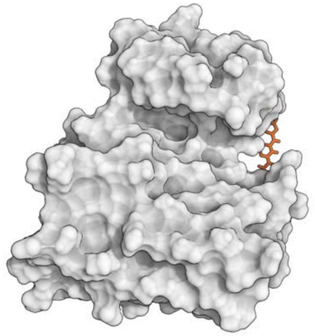
Softmax:

$$\sigma(z) = \frac{1}{\sum_{j=0}^{k-1} e^{z_j}} \begin{pmatrix} e^{z_0} \\ e^{z_1} \\ \vdots \\ e^{z_{k-1}} \end{pmatrix}$$

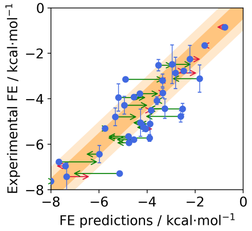
Predicting binding affinities from simulation and experimental data

Jenke Scheen

$$\Delta G_{\mathrm{bind},L} = -k_B T \ln \frac{k_{\mathrm{on}}}{k_{\mathrm{off}}}$$



Learning binding affinity corrections

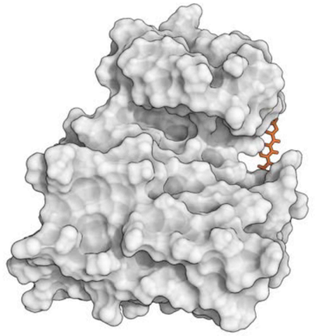


Scheen *et al. J. Chem. Inf. Model. 60, 11, 5331–5339 (2020)*

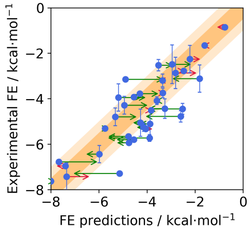
Predicting binding affinities from simulation and experimental data

Jenke Scheen

$$\Delta G_{\mathrm{bind},L} = -k_B T \ln \frac{k_{\mathrm{on}}}{k_{\mathrm{off}}}$$



Learning binding affinity corrections

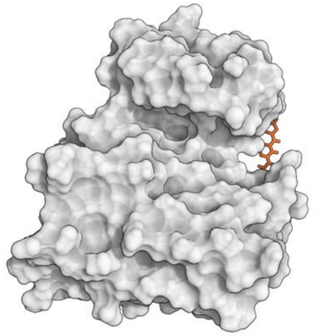


Scheen *et al. J. Chem. Inf. Model. 60, 11, 5331–5339 (2020)*

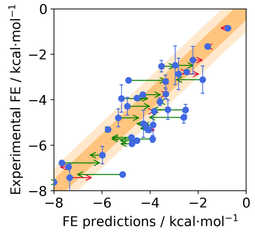
Predicting binding affinities from simulation and experimental data

Jenke Scheen

$$\Delta G_{\mathrm{bind},L} = -k_B T \ln \frac{k_{\mathrm{on}}}{k_{\mathrm{off}}}$$

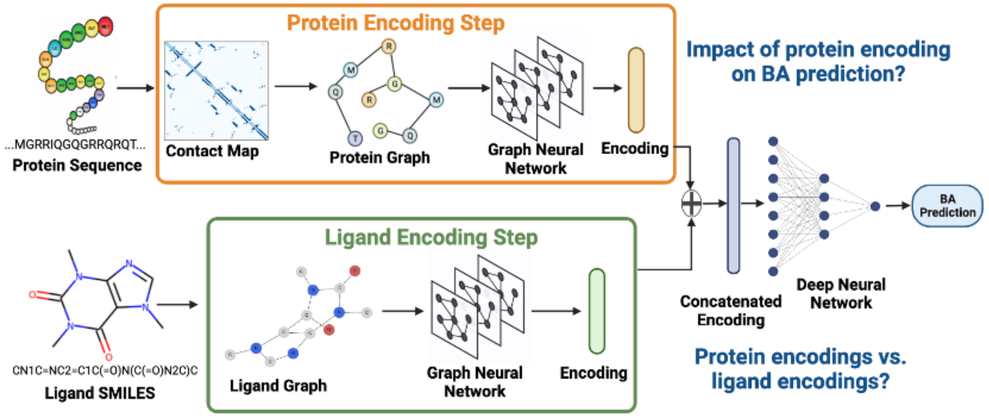


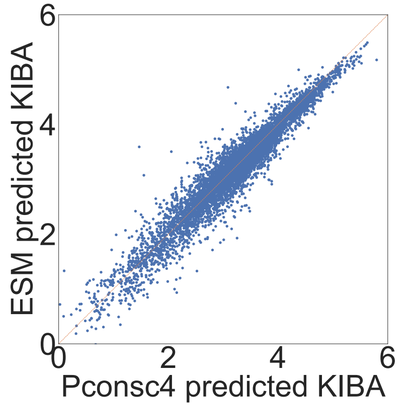
Learning binding affinity corrections



Scheen *et al. J. Chem. Inf. Model. 60, 11, 5331–5339 (2020)*

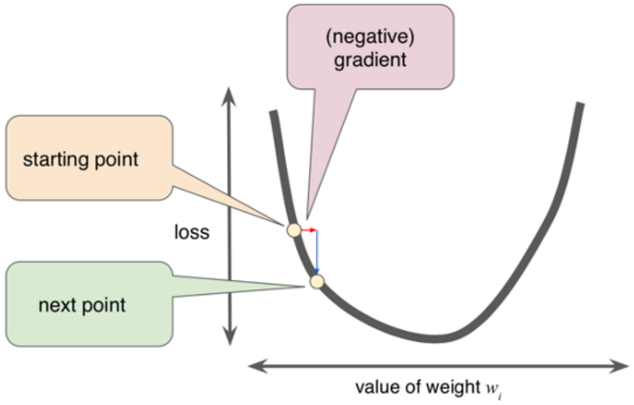
Rohan Gorantla





From mathematical concepts to applications in peptides

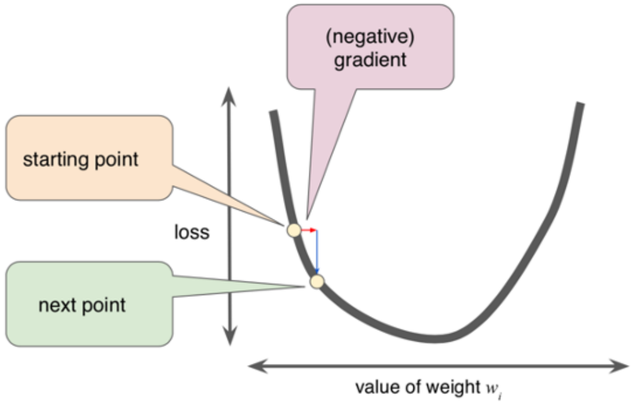
Katarina Karoni



Katerina works on Continual learning problems.
She developed a new optimiser for Multirate learning and proved its convergence.

From mathematical concepts to applications in peptides

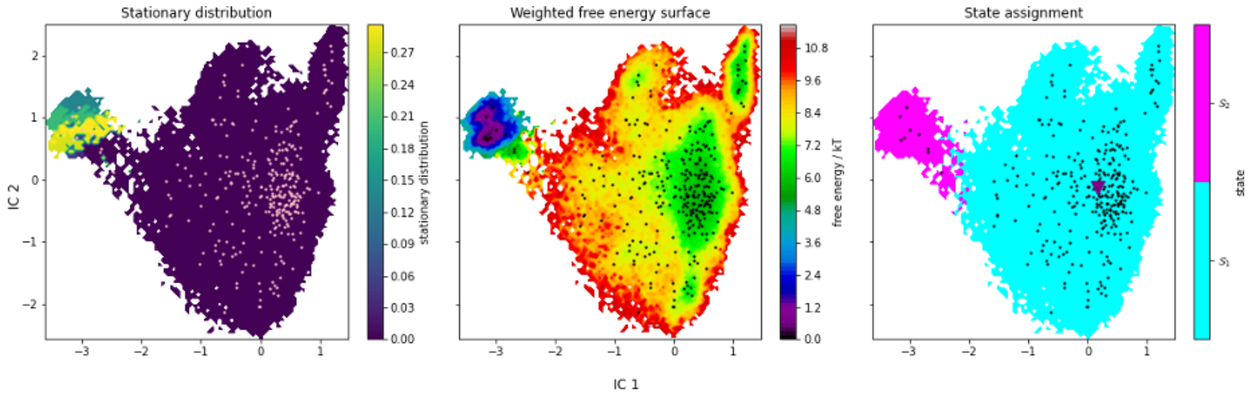
Katarina Karoni

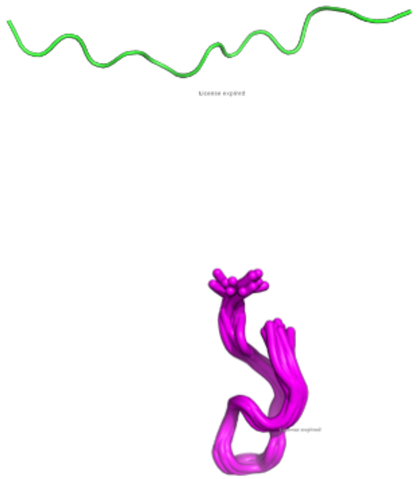


Katerina works on Continual learning problems.
She developed a new optimiser for Multirate learning and proved its convergence.

How can we find good transition paths between conformations?

Ryan Zhu

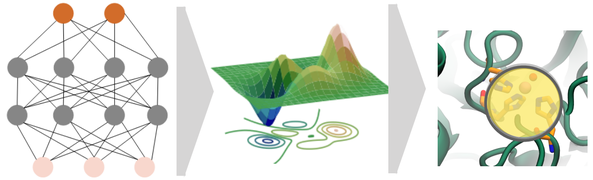


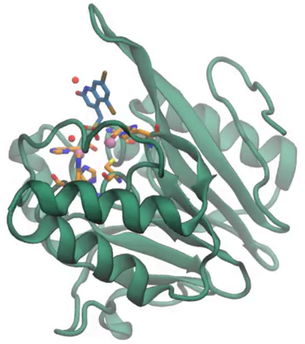


Understanding layers of complexity with ML models

Jasmin Güven

Neural Network potentials for metalloproteins

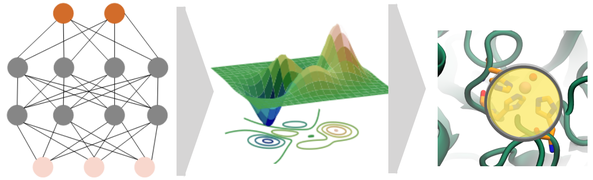


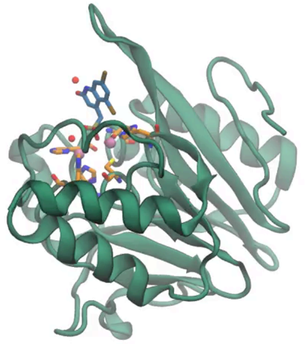


Understanding layers of complexity with ML models

Jasmin Güven

Neural Network potentials for metalloproteins

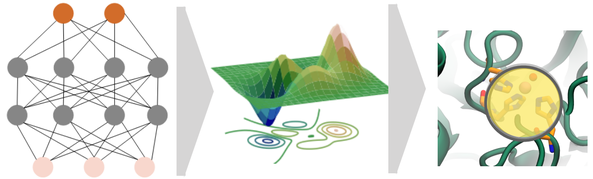


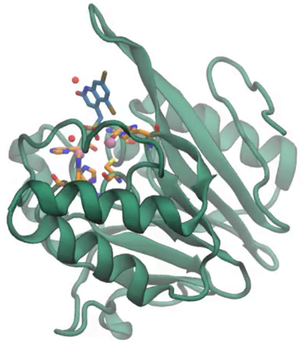


Understanding layers of complexity with ML models

Jasmin Güven

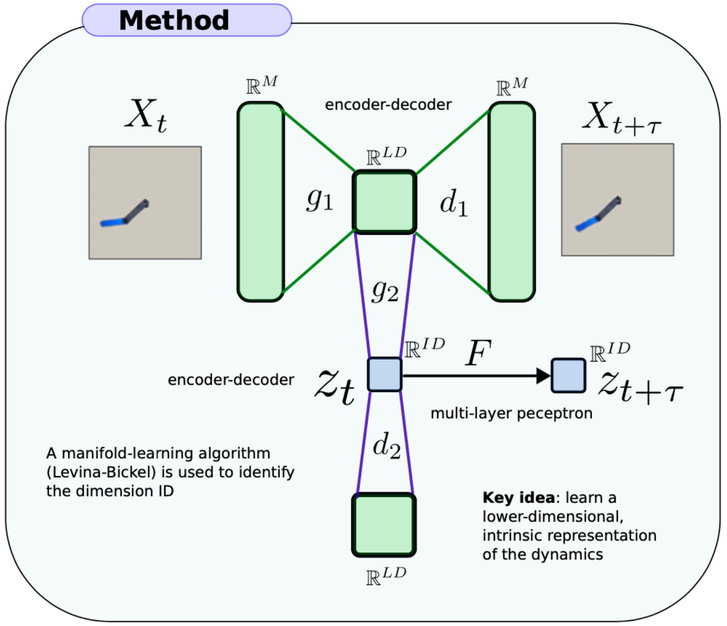
Neural Network potentials for metalloproteins



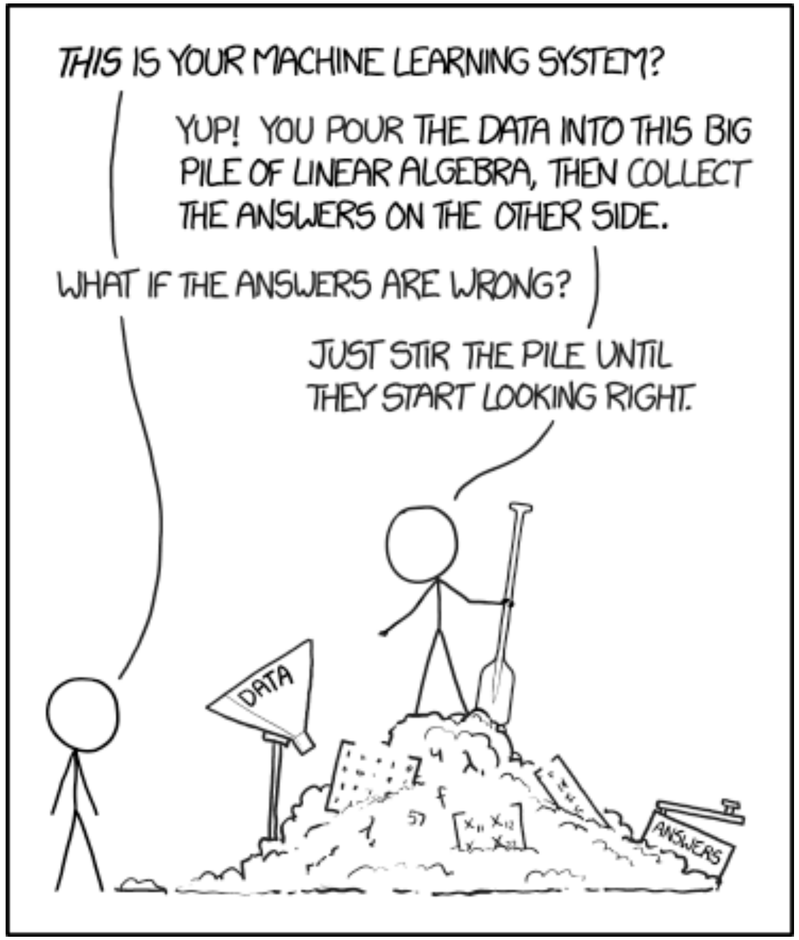


Dominic Philips

Automated Discovery of Fundamental Variables in Experiments and or Simulation



Valuable lessons on machine learning



https://xkcd.com/1838/

Something you liked 😎👍

Something you think could be improved 😬👎