Introduction to machine learning and `scikit-learn`

Part I: Supervised learning

QLSC 612 | 29 May 2025

By

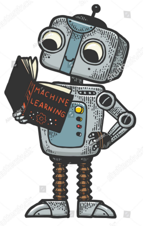
Michelle Wang & Mohammad Torabi

(reusing some of Nikhil Bhagwat's slides)

McGill UNIVERSITY

ORIGAMI Lab

neuro Montreal Neurological Institute-Hospital

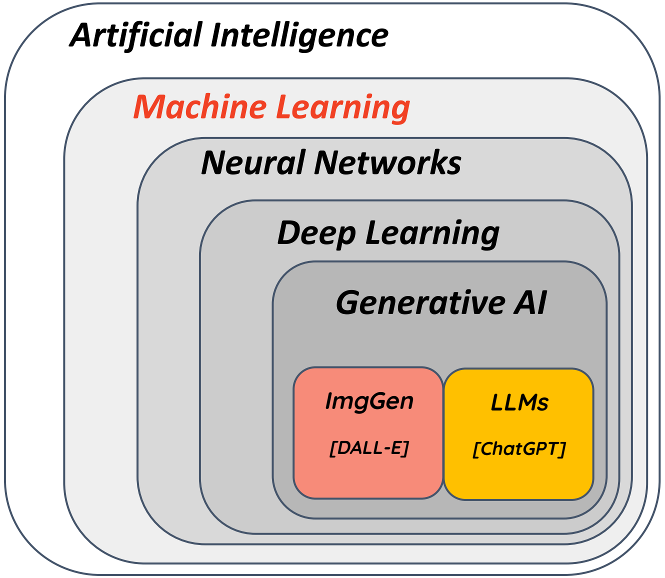


Outline

- Machine learning overview
- Supervised learning
  - Goal
  - Example models
  - Supervised learning with scikit-learn interface
  - Model evaluation
- Deep learning (brief)

Machine-learning - what, why, and when?

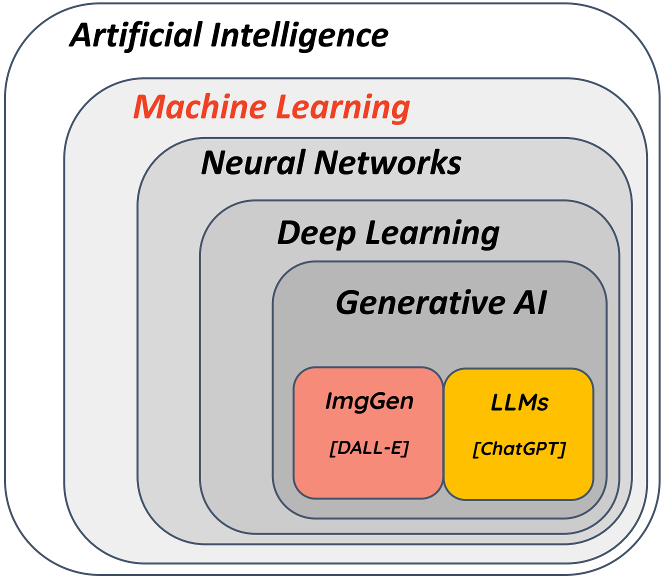
- What is Machine learning (ML)?
    - ML is the study of computer algorithms that improve automatically through “experience” and by the use of data.



Machine-learning - what, why, and when?

- What is Machine learning (ML)?
    - ML is the study of computer algorithms that improve automatically through “experience” and by the use of data.

- Why is it useful - especially in life sciences?
    - Biology, medicine, environmental sciences comprise phenomenons (e.g. a disease) with large number of variables.
    - We want to model complex relationships within these variables and **make accurate predictions**.



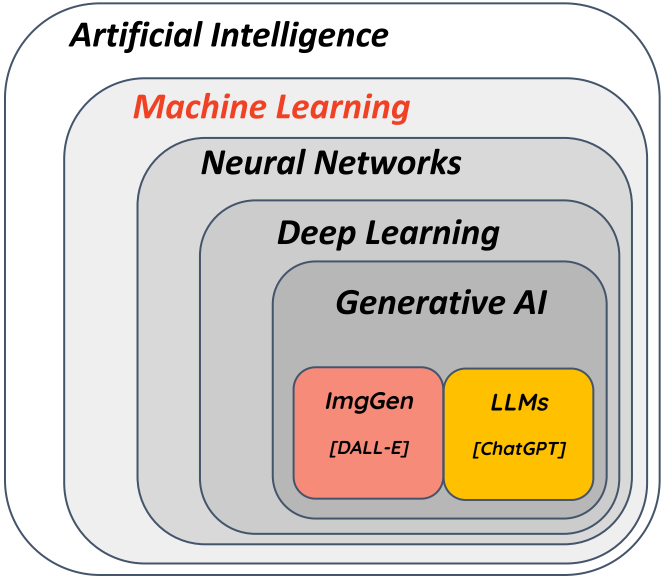
4

Machine-learning - what, why, and when?

- What is Machine learning (ML)?
  - ML is the study of computer algorithms that improve automatically through “experience” and by the use of data.

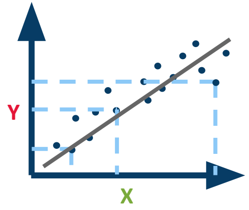
- Why is it useful - especially in life sciences?
  - Biology, medicine, environmental sciences comprise phenomenons (e.g. a disease) with large number of variables.
  - We want to model complex relationships within these variables and **make accurate predictions**.

- When do I use it?
  - You are interested in 1) prediction tasks or 2) low-dimensional representation.
  - **You have sufficient data.**

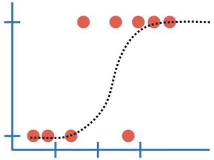


Types of ML Algorithms

- **Supervised** → labels are known
  - Regression → labels are continuous



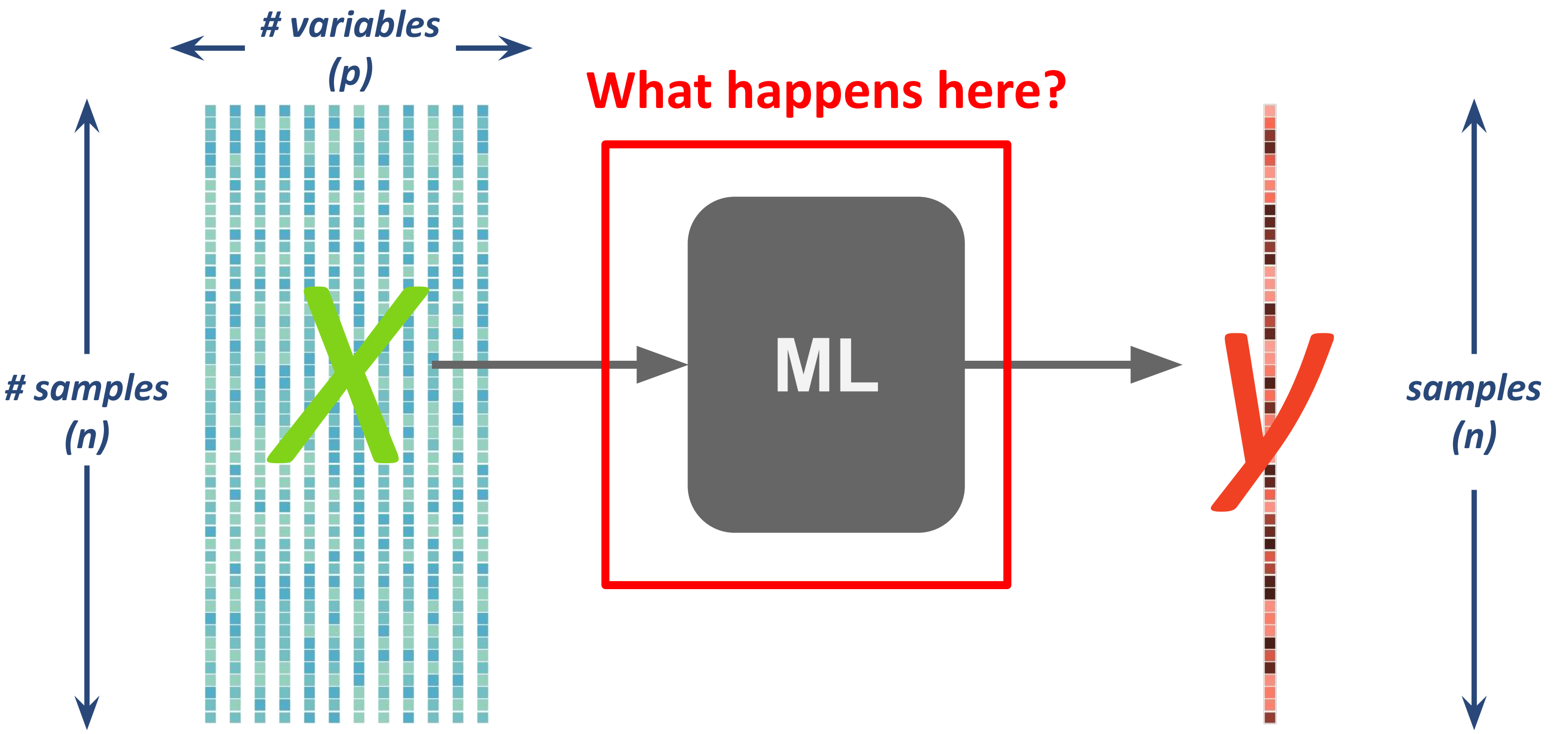
  - Classification → labels are discrete



- **Unsupervised** → labels are unknown
  - Associations, dimensionality reduction, clustering
  - Covered in Part 2

Machine learning terminology

Input (features, etc.) **Output (labels, targets, etc.)**



A typical supervised learning workflow

Decision points when developing a model

Features

X

Preprocessing
Missing values?
Normalization?
Feature selection?
Dimensionality reduction?

Prediction
Which algorithm?
Regularization?
Which hyperparameters?

Targets

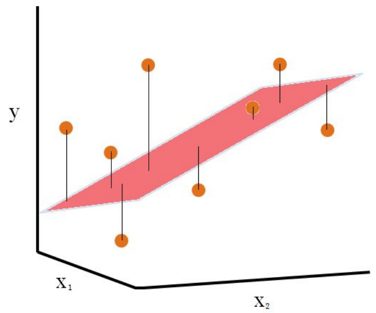
y

Supervised Learning: Models

- **Goal**: Learn parameters (or weights) of a model that maps X to y

Supervised Learning: Models

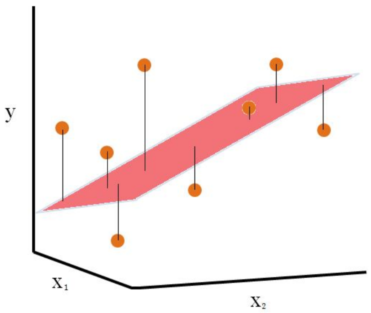
- **Goal**: Learn parameters (or weights) of a model that maps X to y
- Example models (see also scikit-learn documentation):
    - Linear / Logistic regression



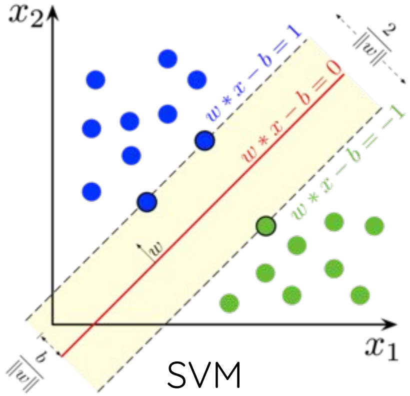
Linear Regression

Supervised Learning: Models

- **Goal**: Learn parameters (or weights) of a model that maps X to y
- Example models (see also scikit-learn documentation):
  - Linear / Logistic regression
  - Support vector machines



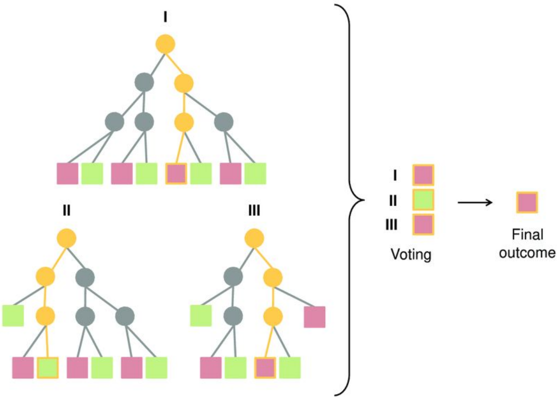
Linear Regression



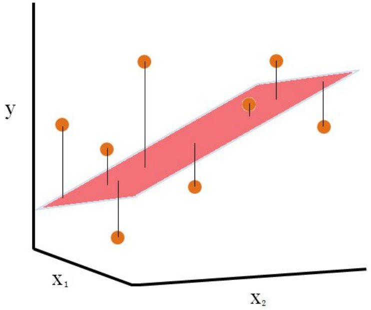
SVM

Supervised Learning: Models

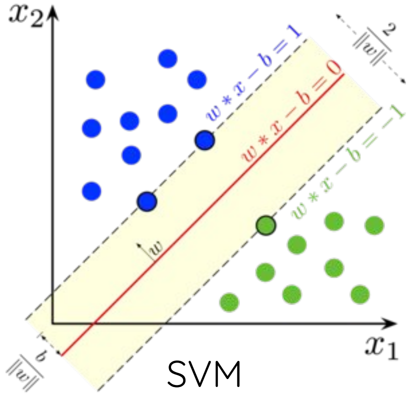
- **Goal**: Learn parameters (or weights) of a model that maps X to y
- Example models (see also [scikit-learn documentation]()):
  - Linear / Logistic regression
  - Support vector machines
  - Tree-ensembles: random forests, gradient boosting



Tree-ensembles



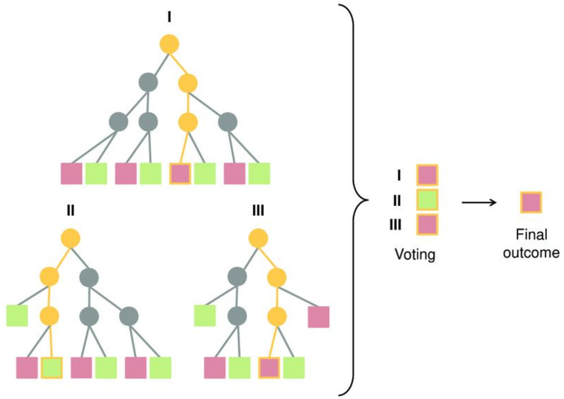
Linear Regression



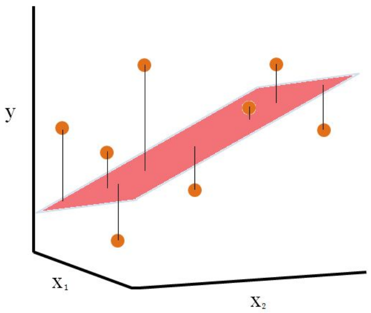
SVM

Supervised Learning: Models

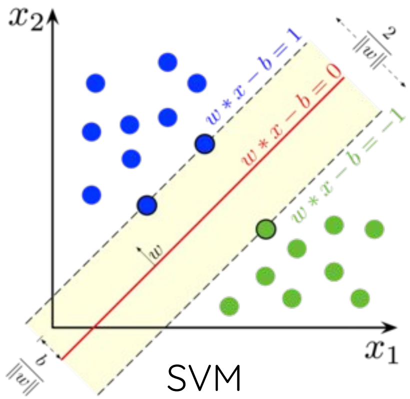
- **Goal**: Learn parameters (or weights) of a model that maps X to y
- Example models (see also scikit-learn documentation):
  - Linear / Logistic regression
  - Support vector machines
  - Tree-ensembles: random forests, gradient boosting
  - Artificial Neural networks



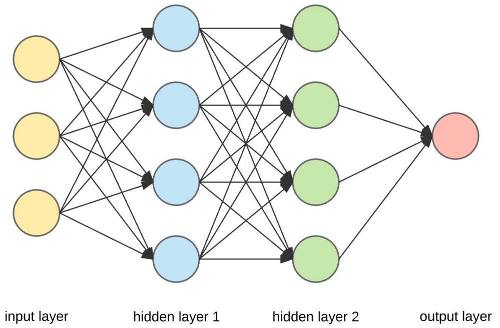
Tree-ensembles



Linear Regression



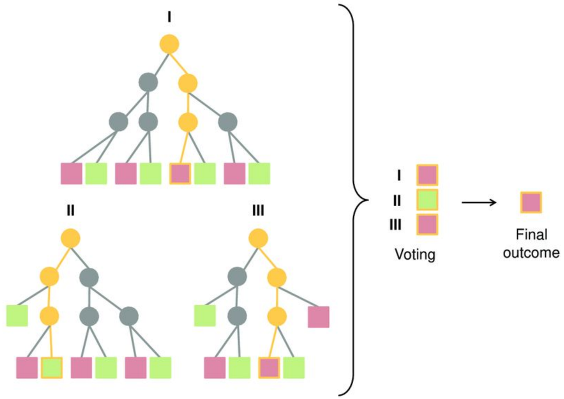
SVM



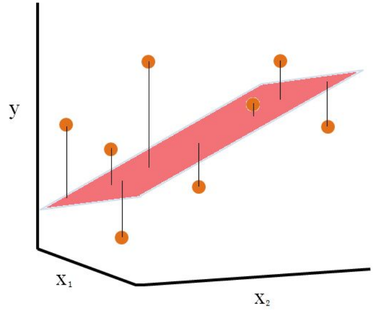
ANN

Supervised Learning: Models

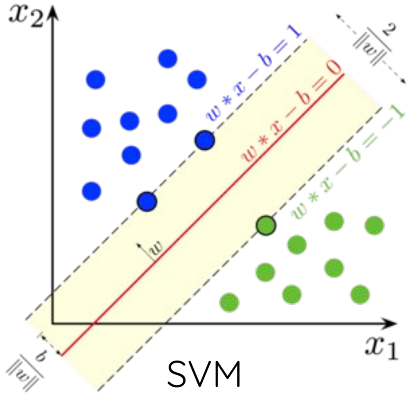
- **Goal**: Learn parameters (or weights) of a model that maps X to y
- Example models (see also scikit-learn documentation):
  - Linear / Logistic regression
  - Support vector machines
  - Tree-ensembles: random forests, gradient boosting
  - Artificial Neural networks
- **How are these models different from one another?**



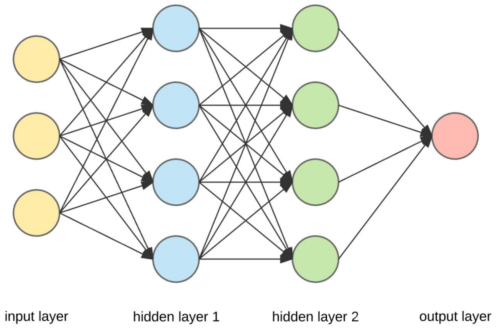
Tree-ensembles



Linear Regression



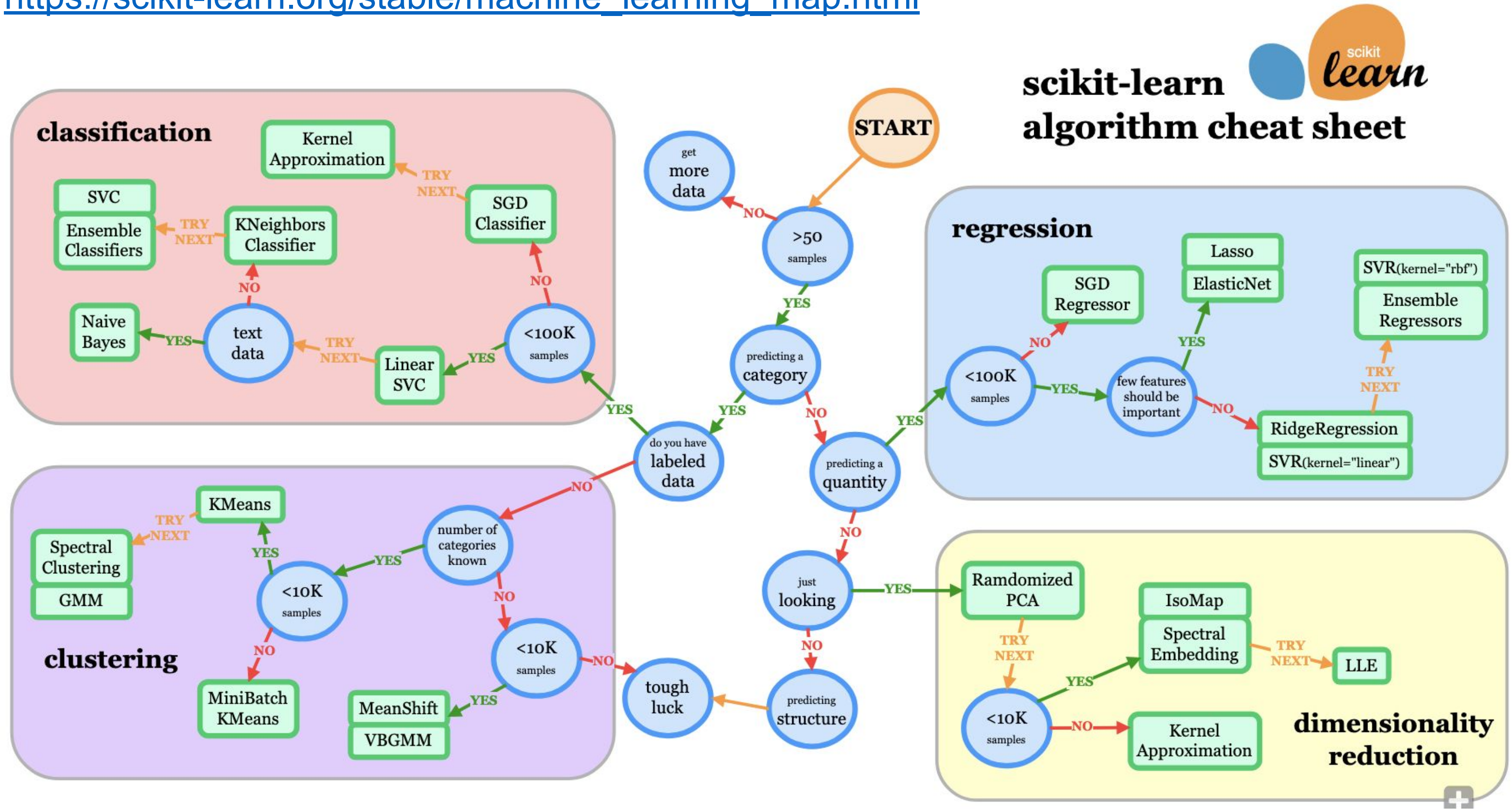
SVM



ANN

Some models make more sense in some situations

https://scikit-learn.org/stable/machine_learning_map.html



Questions?

Model fitting is easy with scikit-learn

Example with **linear regression**

```
# import
from sklearn.linear_model import Lasso


# data
X = [[0, 0], [1, 1]]
y = [0, 1]

# instantiate the model
model = Lasso()

# fit the model with data
model.fit(X, y)

# predict on new data
y_pred = model.predict([[1, 0]])
```

Change this to use different models/hyperparameters

I fitted my model, now what?

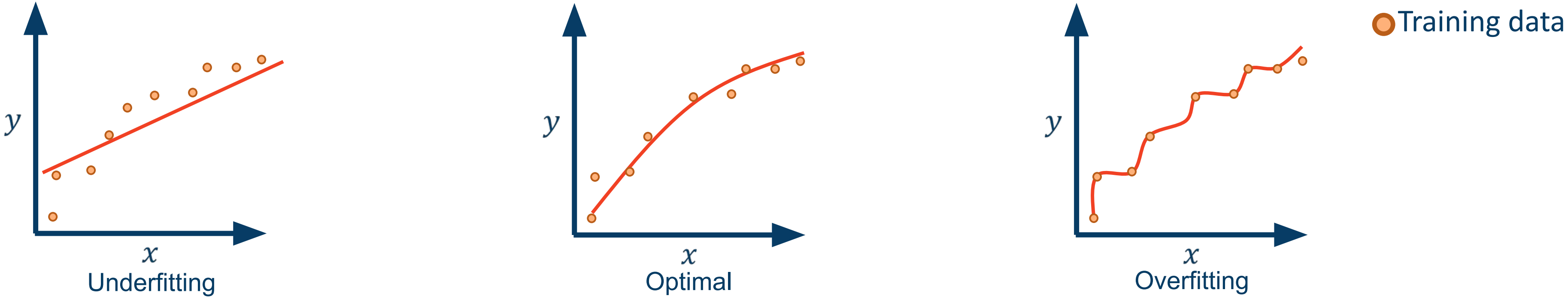
- Model evaluation metrics
  - **Regression**: R^2, mean squared error, mean absolute error, etc.
  - **Classification**: balanced accuracy, AUROC, confusion matrix, etc.
  - See https://scikit-learn.org/stable/modules/model_evaluation.html for more

- How does the model perform
  - On the data it was trained on?
  - On previously unseen data?

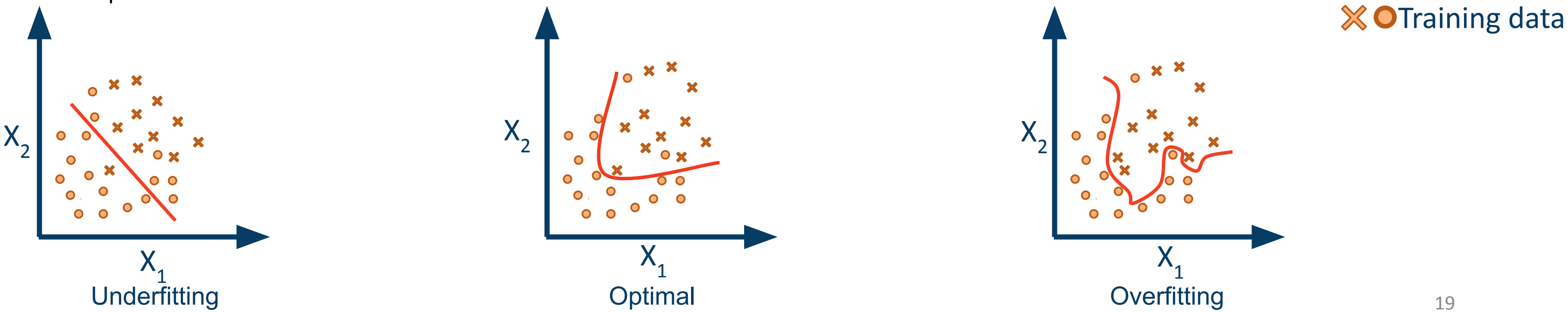
We want good generalizability on new data

Models can overfit (or underfit)

Example: **regression**

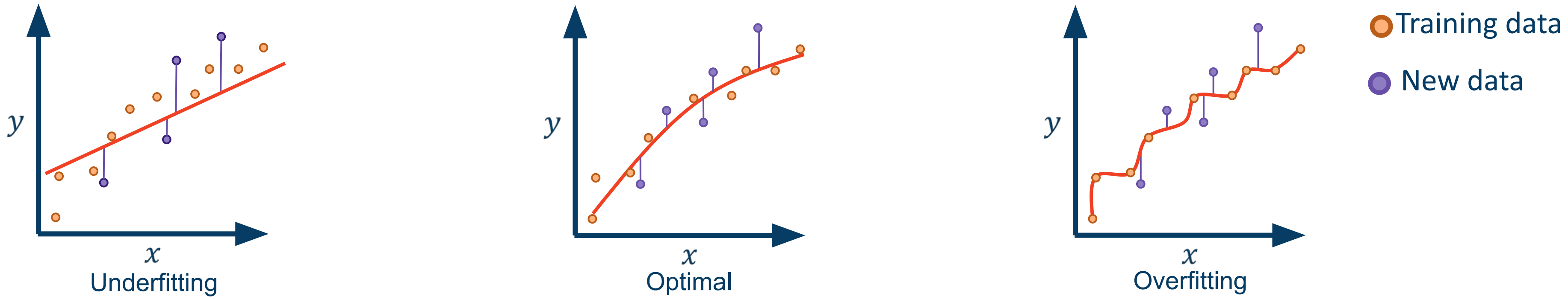


Example: **classification**

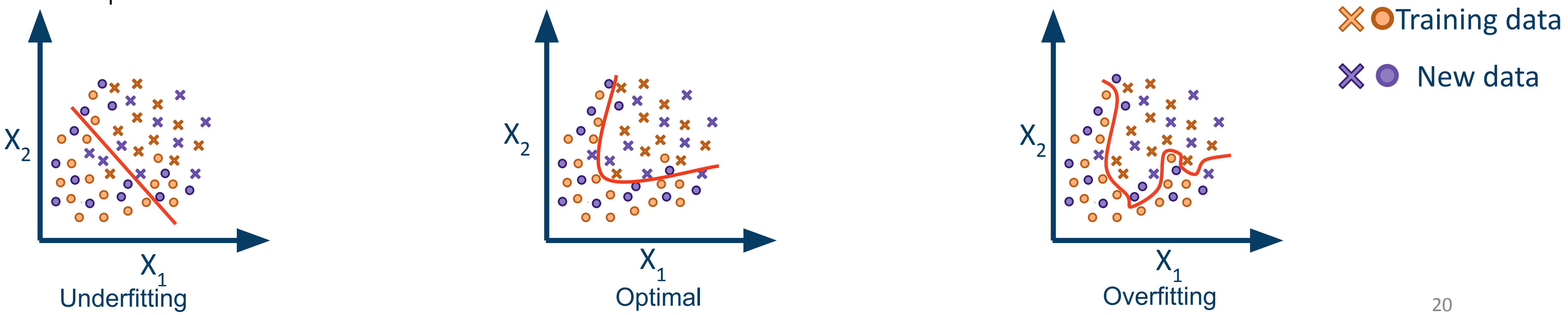


Models can overfit (or underfit)

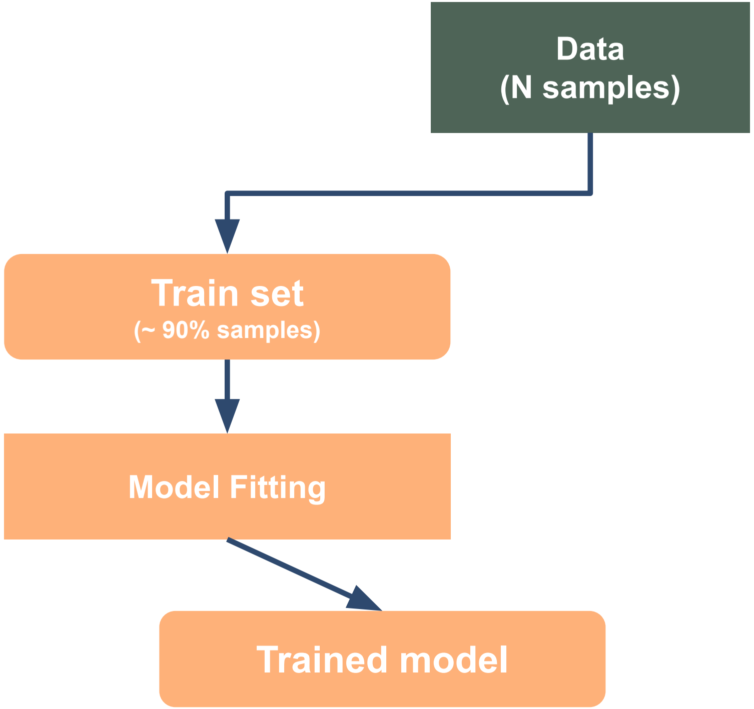
Example: **regression**



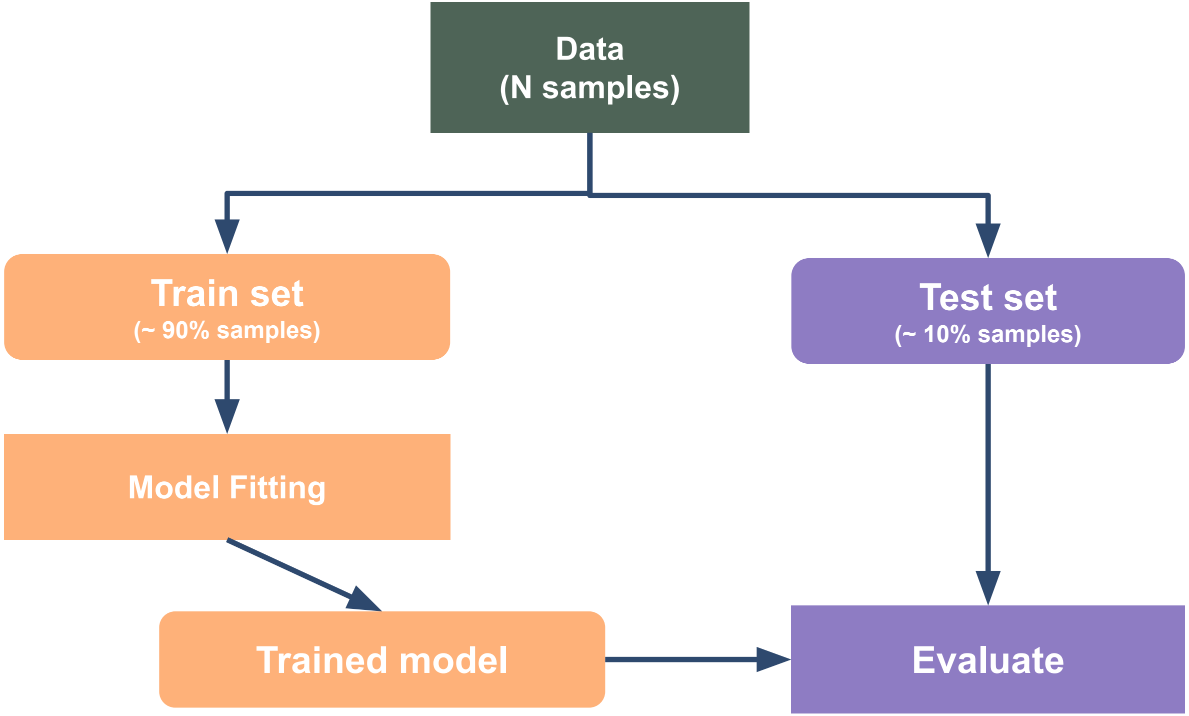
Example: **classification**



Split data into train and test sets

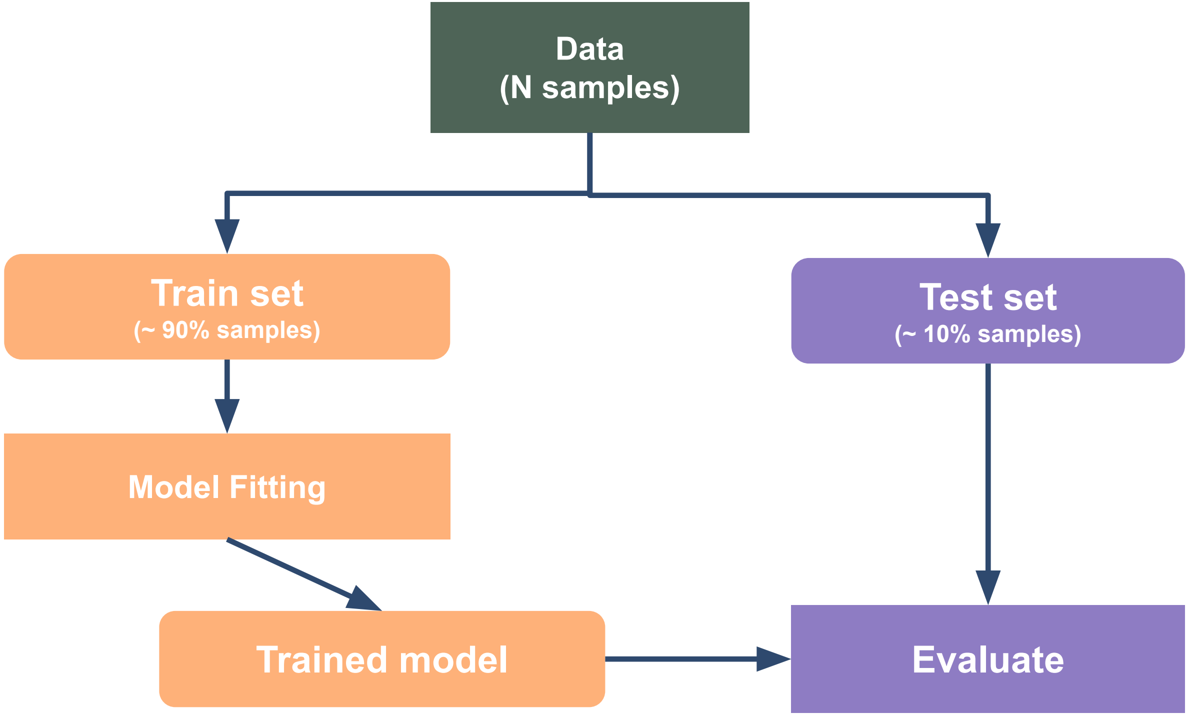


Split data into train and test sets



Exercise 1

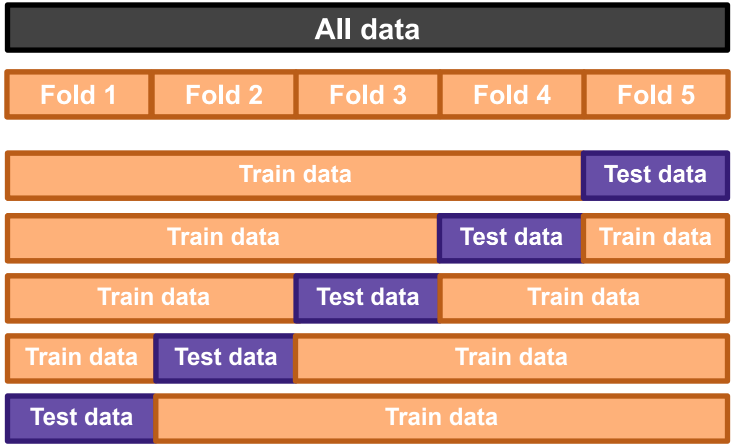
Split data into train and test sets



How to sample the train and test sets?

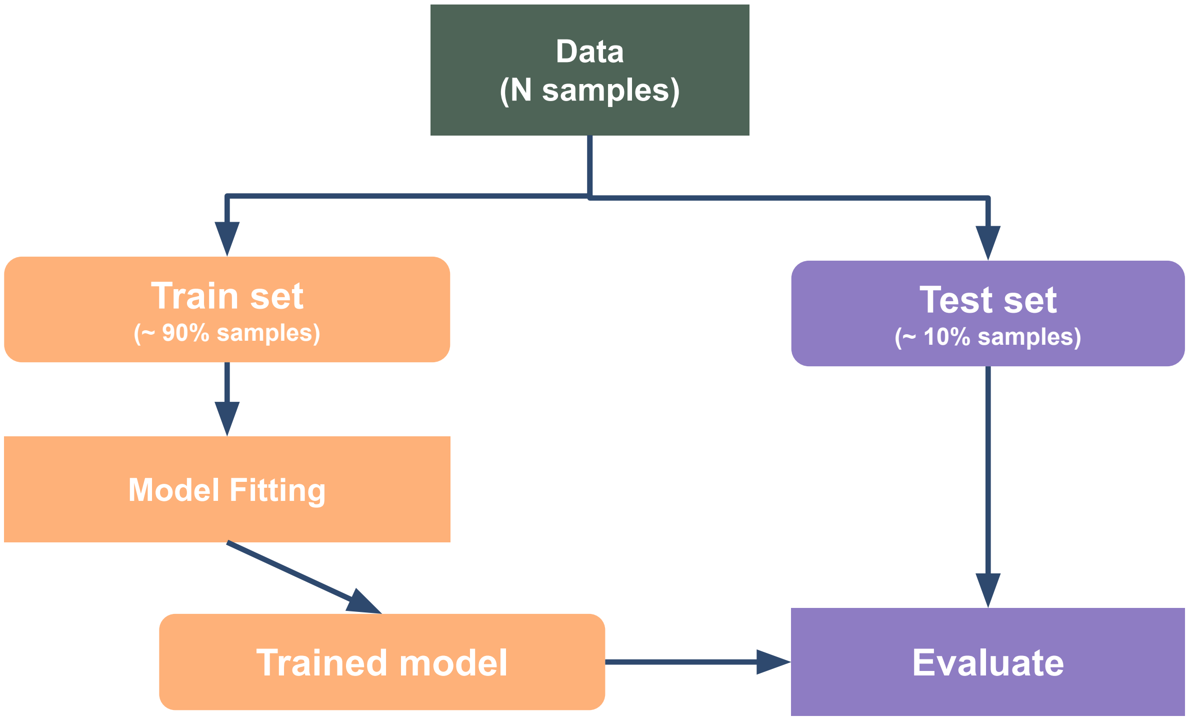
K-fold cross-validation

- How do we sample train and test sets?
  - Train set: learn model parameters
  - Test set (a.k.a held-out sample): Evaluate model performance
  - Repeat for different Train-Test splits
  - Report performance statistics over all test folds



Alternative method: shuffle-split (https://scikit-learn.org/stable/modules/cross_validation.html#shufflesplit)

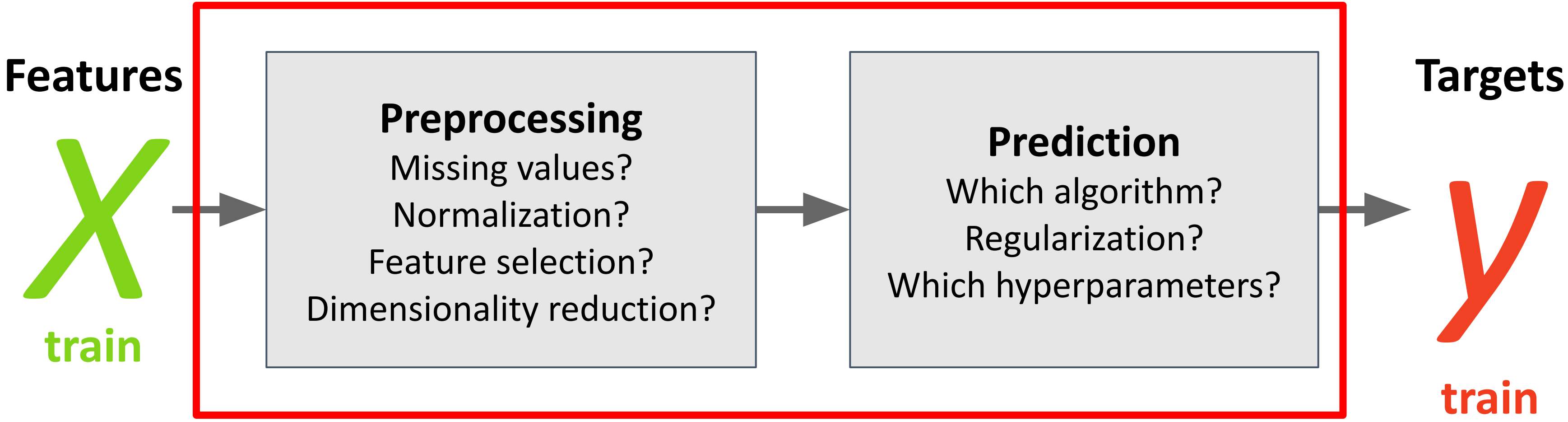
Split data into train and test sets



Be careful about data leakage/double-dipping!

A typical supervised learning workflow

Decision points when developing a model

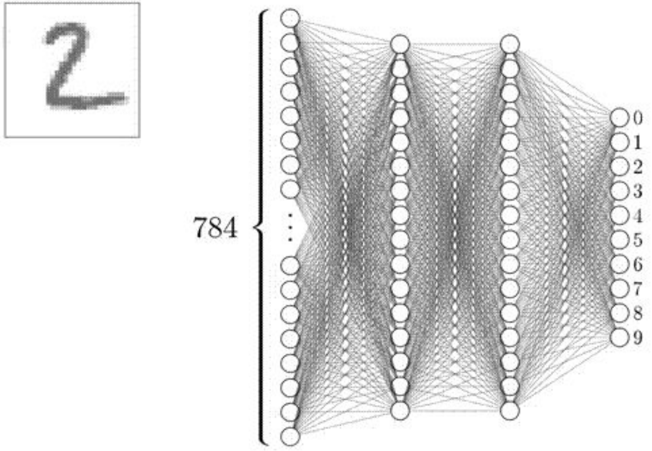


Do not use test data to make these decisions!
(z-score mean/std., hyperparameter tuning, etc.)

Exercise 2

Deep-learning

- Why the buzz?
  - Works amazing on spatio-temporal input
  - Highly flexible → universal function approximator

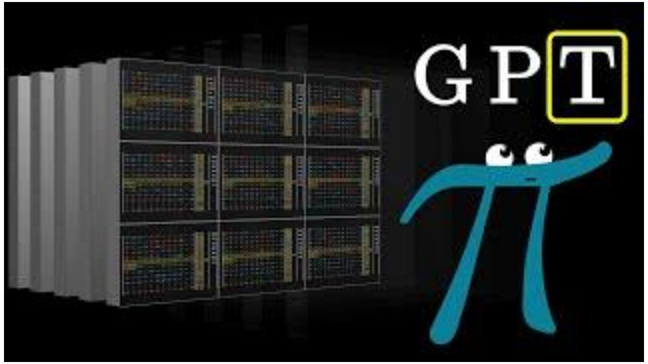


ANN for handwritten-digit images
(gif source: 3b1b)

Deep-learning

- Why the buzz?
  - Works amazing on spatio-temporal input
  - Highly flexible → universal function approximator

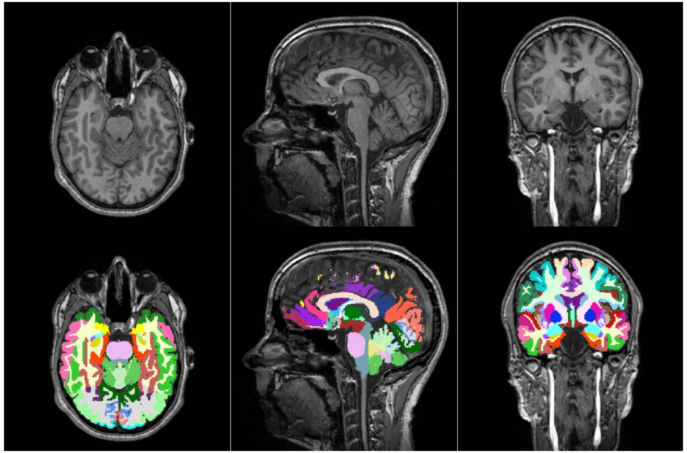
- What are the challenges?
  - Large number of parameters (175B!) → data hungry
  - Large number of hyper-parameters → difficult to train



LLM Transformers
(gif source: 3b1b)

Deep-learning

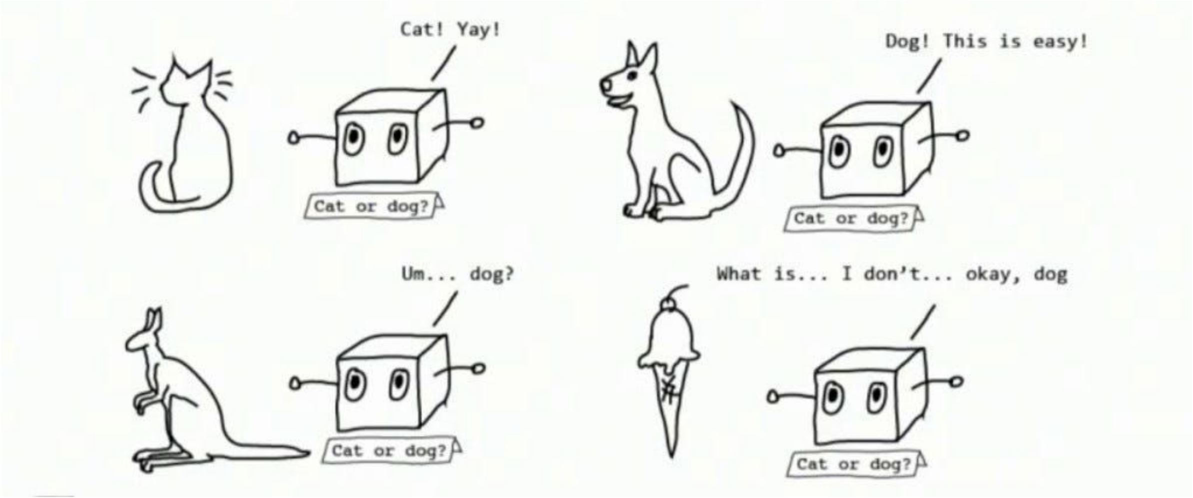
- Why the buzz?
  - Works amazing on spatio-temporal input
  - Highly flexible → universal function approximator
- What are the challenges?
  - Large number of parameters (175B!)→ data hungry
  - Large number of hyper-parameters → difficult to train
- When do I use it?
  - If you have highly-structured input, eg. medical images.
  - You have a lot of data and computational resources.



Source:
https://github.com/fepegar/torchio

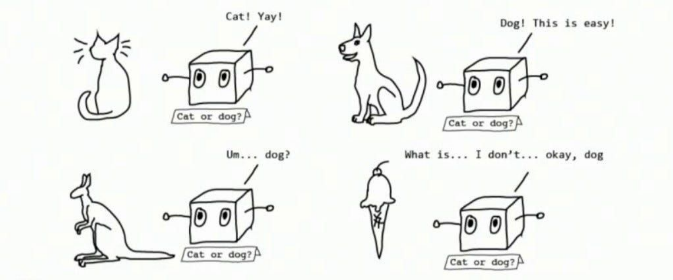
Pitfalls and Challenges

- Models do not generalize even after good CV performance
  - Implicit double-dipping
  - Dataset biases (eg. North-American demographics)
  - Noisy labels (eg. diagnosis definitions)
  - Data distribution shifts (eg. assay, scanner upgrades)



Pitfalls and Challenges

- Models do not generalize even after good CV performance
  - Implicit double-dipping
  - Dataset biases (eg. North-American demographics)
  - Noisy labels (eg. diagnosis definitions)
  - Data distribution shifts (eg. assay, scanner upgrades)



- Unnecessary complexity
  - Do I really need a giant deep-net or a simple linear model would do?

ML Novice Checklist

- Data
  - What is my n_features and n_samples?
  - Am I encoding categorical data correctly?
  - Am I using information (e.g. mean) from test set to preprocess (eg. z-score) the data?

ML Novice Checklist

- Data
  - What is my n_features and n_samples?
  - Am I encoding categorical data correctly?
  - Am I using information (e.g. mean) from test set to preprocess (eg. z-score) the data?

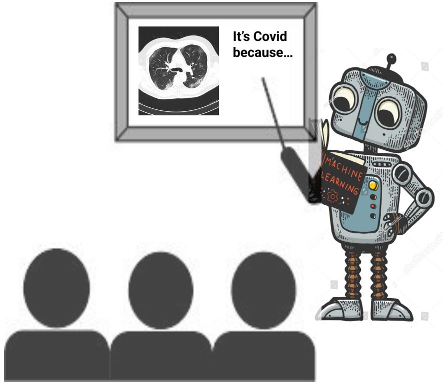
- Model
  - Do my performance metrics capture the practical use-case of interest?
  - What is the null / dummy model performance?
    - Classification: Predict majority class all the time
    - Regression: Predict the median value all the time
  - Am I interpreting model parameters (i.e. weights) correctly?

Takeaways

- Supervised ML is useful for **predictions** but **not really for explanations**
  - eg. image segmentation, prognosis, drug development

- Our job is to ensure **generalizability** of these models
  - Multitude of validations
  - Understanding model biases and limitations

- ***Engineering tools*** vs *Scientific discovery*
  - Interpretability and explainability



Explainable AI

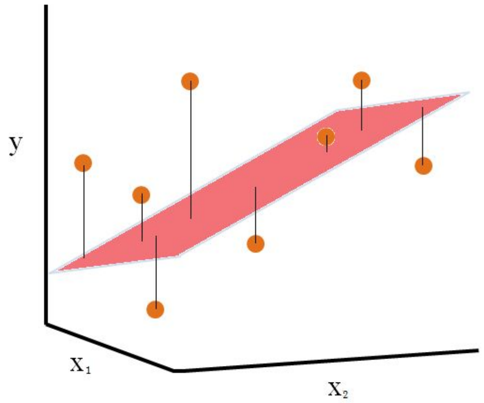
Useful resources

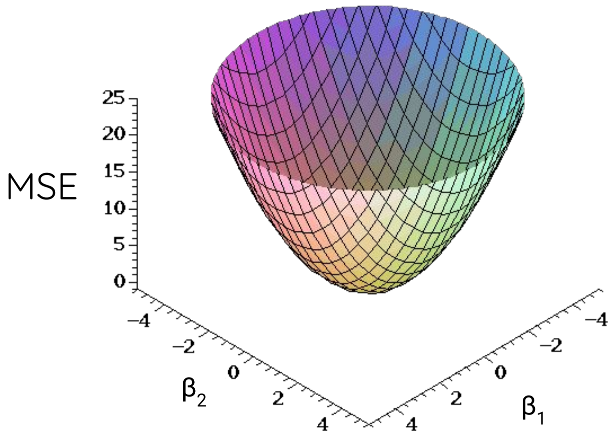
- https://scikit-learn.org/stable/user_guide.html
- `nilearn`, Python package for machine learning for brain images:
  https://nilearn.github.io/stable/index.html
- `skrub`, Python package machine learning for tabular data:
  https://skrub-data.org/stable/
- https://inria.github.io/scikit-learn-mooc/ml_concepts/slides.html
- https://www.3blue1brown.com/topics/linear-algebra
- 3Blue1Brown Gradient Descent:
  https://www.youtube.com/watch?v=IHZwWFHWa-w

If time permits…

Model Fitting

- How do we learn the model weights?
  - Example: Linear regression
  - Model: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$
  - Loss function: $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
  - Optimization: Gradient descent



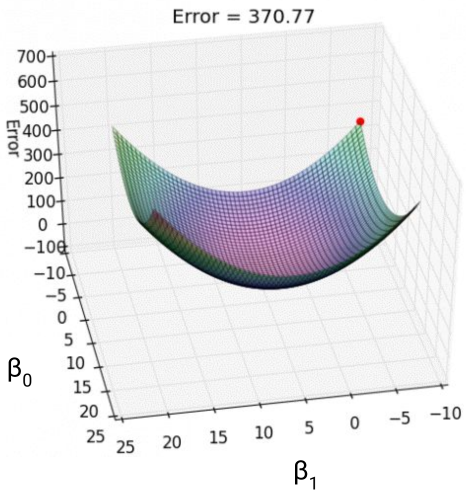


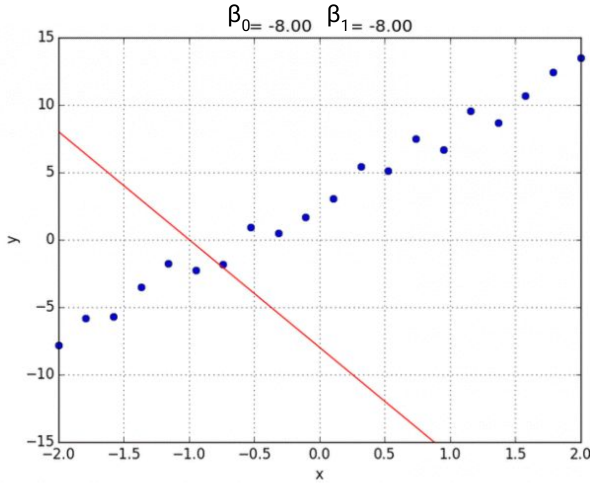
Model Fitting

- Gradient descent with a **single** input variable and **n** samples
  - Start with random weights (β_0 and β_1)
  - Compute loss (i.e. MSE)
  - Update weights based on the gradient

$$\hat{y}_i = \beta_0 + \beta_1 x_i$$

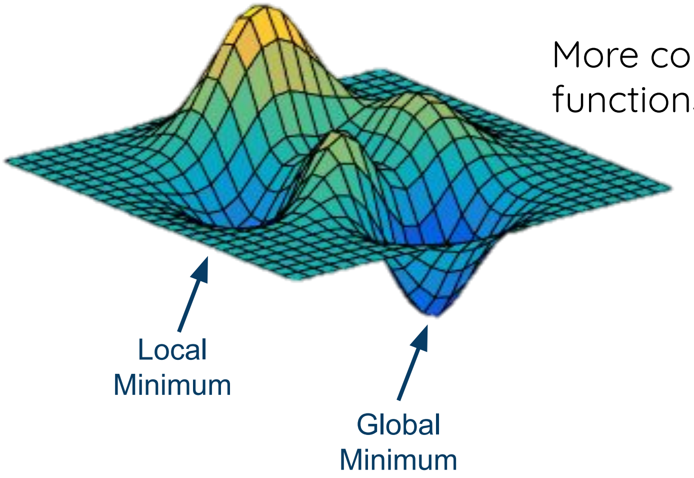
$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$





Model Fitting

- Gradient descent for complex models with non-convex loss functions
  - Start with random weights (β_0 and β_1)
  - Compute loss
  - Update weights based on the gradient



More complex models / loss functions (e.g. ANNs)

Model Fitting

- Can we control this fitting process to get a model with specific characteristics?

Model Fitting

- Can we control this fitting process to get a model with specific characteristics?
  - We have strong prior beliefs about what is a **plausible** model
    - e.g. I believe a disease symptom can be predicted with few genes.
  - Practical reasons
    - Prevent overfitting (n_features >> n_samples)

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_{\rho-1} x_{\rho-1} + \beta_\rho x_\rho$$

Model Fitting

- Can we control this fitting process to get a model with specific characteristics?
  - We have strong prior beliefs about what is a **plausible** model
    - e.g. I believe a disease symptom can be predicted with few genes.
  - Practical reasons
    - Prevent overfitting (n_features >> n_samples)

- **Yes! → Model regularization**

Model Fitting: Regularization

- How do we do it?
  - Modify the loss function
  - Constrain the learning process

- Examples:
  - L1 i.e. Lasso
  - L2 i.e. Ridge

Model Fitting: Regularization

- How do we do it?
  - Modify the loss function
  - Constrain the learning process

- Examples:
  - **L1 i.e. Lasso**
  - **L2 i.e. Ridge**

1) L1/Lasso: constrains parameters to be ***sparse***

$$MSE = \sum_{i=1}^{n} (y_i - \underbrace{[\beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j]}_{\hat{y}_i})^2 + \underbrace{\lambda \sum_{j=1}^{p} |\beta_j|}_{L_1}$$

2) L2/Ridge: constrains parameters to be ***small***

$$MSE = \sum_{i=1}^{n} (y_i - \underbrace{[\beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j]}_{\hat{y}_i})^2 + \underbrace{\lambda \sum_{j=1}^{p} \beta_j^2}_{L_2}$$

Nested-cross validation

Goal → Fit and evaluate Ridge model

1. Learn β that gives the best prediction on training data
2. Compute a new score on unseen data

Ridge model (L2 regularization)

$$MSE = \sum_{i=1}^{n} (y_i - [\, \beta_0 + \sum_{j=1}^{p} x_{ij} \beta_j \,])^2 + \alpha \sum_{j=1}^{p} \beta_j^2$$

model parameters

model hyper -parameters

Nested-cross validation

Goal → Fit and evaluate Ridge model

1. Learn β that gives the best prediction on training data
2. Compute a new score on unseen data
3. Repeat step 1-2 for a few values of α, fitting and testing several models (i.e. grid search)
4. Select the α value that obtains the best prediction

Ridge model (L2 regularization)

$$MSE = \sum_{i=1}^{n} (y_i - [\, \beta_0 + \sum_{j=1}^{p} x_{ij} \beta_j \,])^2 + \alpha \sum_{j=1}^{p} \beta_j^2$$

model parameters

model hyper -parameters

Nested-cross validation

Goal → Fit and evaluate Ridge model

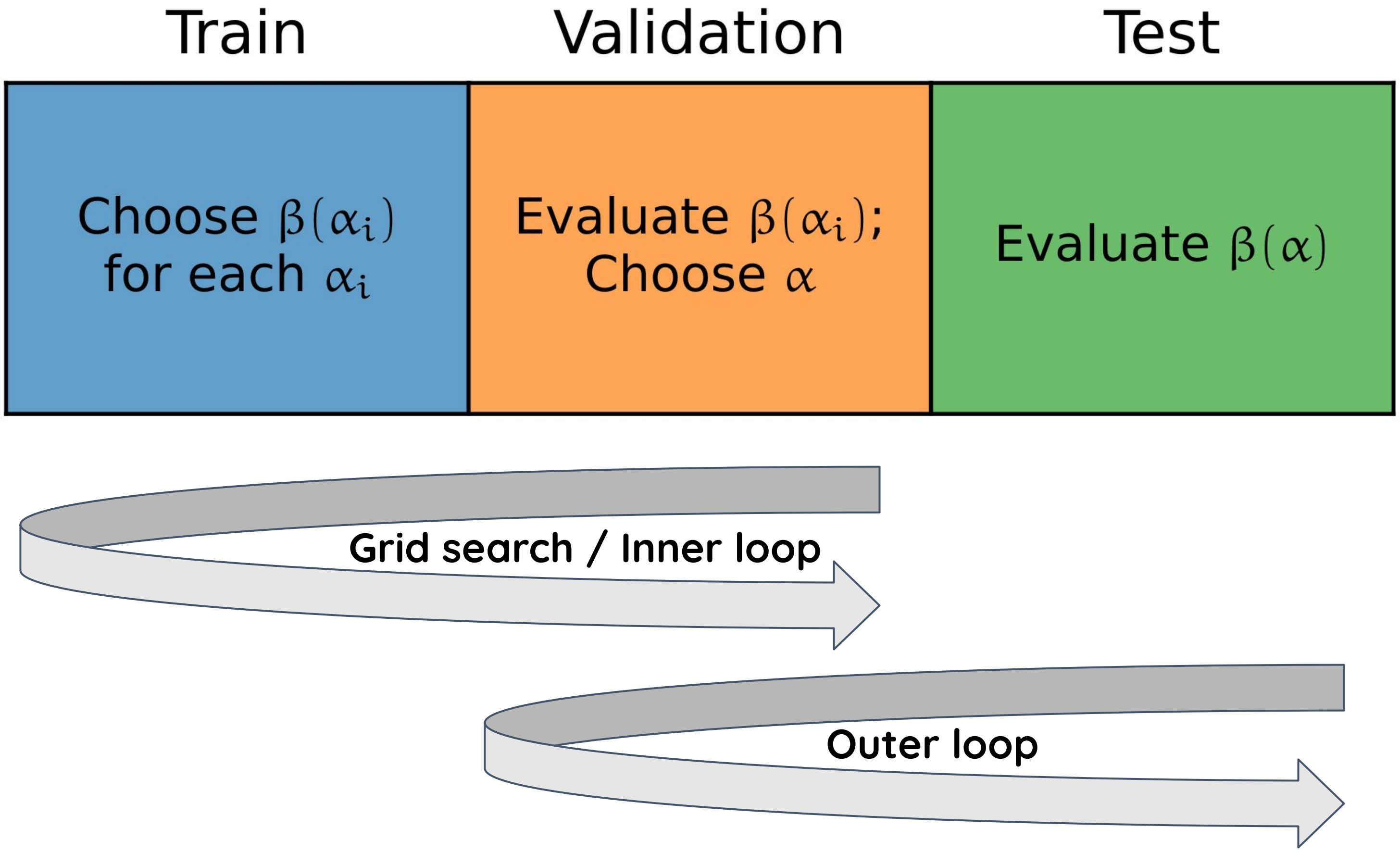
1. Learn β that gives the best prediction on training data
2. Compute a new score on unseen data
3. Repeat step 1-2 for a few values of α, fitting and testing several models (i.e. grid search)
4. Select the α value that obtains the best prediction
5. Evaluate the model with these β and α on another unseen data

Ridge model (L2 regularization)

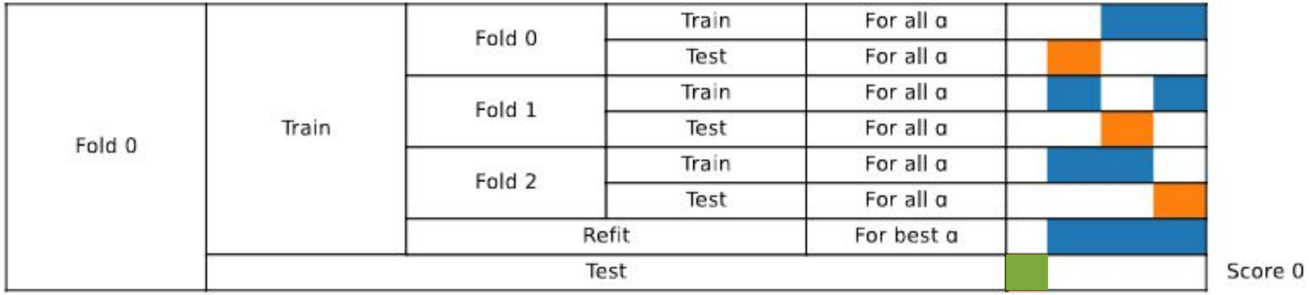
$$MSE = \sum_{i=1}^{n}\Big(y_i - \underbrace{\Big[\beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j\Big]}_{\text{model parameters}}\Big)^2 + \underbrace{\alpha}_{\text{model hyper-parameters}}\sum_{j=1}^{p}\beta_j^2$$

model parameters

model hyper -parameters


Train
Validation
Test
Choose $\beta(\alpha_i)$ for each α_i
Evaluate $\beta(\alpha_i)$; Choose α
Evaluate $\beta(\alpha)$
Grid search / Inner loop
Outer loop

Train	Validation	Test
Choose $\beta(\alpha_i)$ for each α_i	Evaluate $\beta(\alpha_i)$; Choose α	Evaluate $\beta(\alpha)$



K-fold cross-validation

- Split original data into “K” folds (outer loop)
  - n_folds == n_test_scores
- Split each fold into external “train” and “test” subsets
- Split train subset into M folds (inner loop)
- Split each internal fold into internal “train” and “test” (aka. validation) subsets