

# Machine learning Part 2

## Dimensionality reduction & cross-validation

Jérôme Dockès & Nikhil Bhagwat

QLS course 2021-07-30



**McGill**  
UNIVERSITY



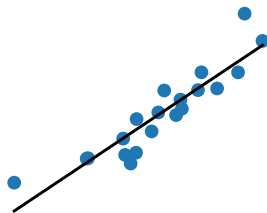
ORIGAMI  
Lab

# Recap of part 1

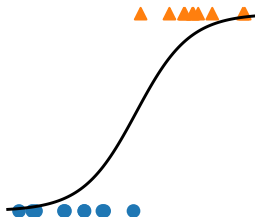
## Supervised learning

- Regression: least-squares linear regression
- Classification: logistic regression

Linear regression



Logistic regression



# Recap of part 1

## Supervised learning

- Regression: least-squares linear regression
- Classification: logistic regression

## Regularization

- $\ell_2$  a.k.a. ridge regularization

# Recap of part 1

## Supervised learning

- Regression: least-squares linear regression
- Classification: logistic regression

## Regularization

- $\ell_2$  a.k.a. ridge regularization

## Model evaluation and selection

- Out-of-sample generalization; independent test set
- Performance metrics:
  - regression: mean squared error
  - classification: accuracy, ROC curve
- Cross-validation

# Recap of part 1

## Supervised learning

- Regression: least-squares linear regression
- Classification: logistic regression

## Regularization

- $\ell_2$  a.k.a. ridge regularization

## Model evaluation and selection

- Out-of-sample generalization; independent test set
- Performance metrics:
  - regression: mean squared error
  - classification: accuracy, ROC curve
- Cross-validation

Don't remember? watch Part 1 again!

# Notation & vocabulary

## Supervised learning framework

$$Y = f(X) + E \quad (1)$$

- $Y \in \mathbb{R}$ : output (a.k.a. target, dependent variable) to predict

# Notation & vocabulary

## Supervised learning framework

$$Y = f(X) + E \quad (1)$$

- $Y \in \mathbb{R}$ : output (a.k.a. target, dependent variable) to predict
- $X \in \mathbb{R}^p$ : features (a.k.a. inputs, regressors, descriptors, independent variables)

# Notation & vocabulary

## Supervised learning framework

$$Y = f(X) + E \quad (1)$$

- $Y \in \mathbb{R}$ : output (a.k.a. target, dependent variable) to predict
- $X \in \mathbb{R}^p$ : features (a.k.a. inputs, regressors, descriptors, independent variables)
- $E \in \mathbb{R}$ : unmodelled noise



# Notation & vocabulary

## Supervised learning framework

$$Y = f(X) + E \quad (1)$$

- $Y \in \mathbb{R}$ : output (a.k.a. target, dependent variable) to predict
- $X \in \mathbb{R}^p$ : features (a.k.a. inputs, regressors, descriptors, independent variables)
- $E \in \mathbb{R}$ : unmodelled noise
- $f$ : the function we try to approximate

## Example: linear regression

$$Y = \beta_0 + \langle X, \beta \rangle + E \quad (2)$$

$$= \beta_0 + \sum_{j=1}^p X_j \beta_j + E \quad (3)$$

"learning" = estimating  $\beta_0 \in \mathbb{R}$  and  $\beta \in \mathbb{R}^p$

# How to estimate parameters: Empirical Risk Minimization

Given  $n$  training examples  $\mathbf{X} \in \mathbb{R}^{n \times p}$ ,  $\mathbf{y} \in \mathbb{R}^n$ ,  
find  $\hat{\beta}_0 \in \mathbb{R}$ ,  $\hat{\beta} \in \mathbb{R}^p$   
that minimize the empirical risk:

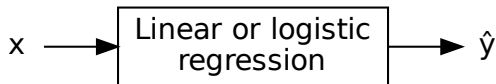
$$\|\mathbf{y} - \hat{\mathbf{y}}\|_2^2 = \|\mathbf{y} - \hat{\beta}_0 - \mathbf{X} \hat{\beta}\|_2^2 \quad (4)$$

$$= \sum_{i=1}^n (\mathbf{y}_i - \hat{\beta}_0 - \sum_{j=1}^p \mathbf{x}_{ij} \hat{\beta}_j)^2 \quad (5)$$

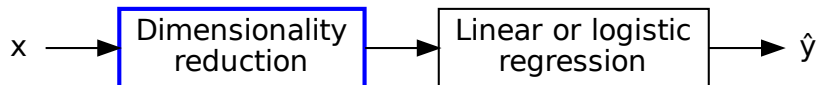
"Fitting" the parameters to  $\mathbf{X}, \mathbf{y}$ .

# Dimensionality reduction

Until now



Add a step in the pipeline: simplifying the inputs



# Dimensionality reduction

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 \mathbf{X}_{:,1} + \hat{\beta}_2 \mathbf{X}_{:,2} + \cdots + \hat{\beta}_p \mathbf{X}_{:,p} \quad (6)$$

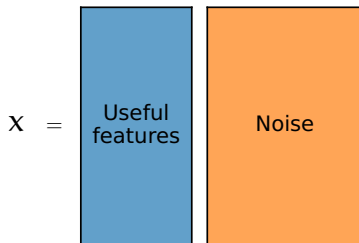
Problems when the number of features  $p$  becomes large

- Bigger errors on test data (larger variance of predictions)
- Numerical stability issues
- Computational cost and memory usage

# Simulated data for linear regression

- Generate  $\mathbf{X} \in \mathbb{R}^{n \times 3}$ ,  $\boldsymbol{\beta} \in \mathbb{R}^3$ , and  $\mathbf{y} = \mathbf{X} \boldsymbol{\beta} \in \mathbb{R}^n$
- Append columns containing random noise to  $\mathbf{X}$
- Now  $\mathbf{X} \in \mathbb{R}^{n \times p}$ , with  $p \geq 3$ , but only the first 3 columns are linked with  $\mathbf{y}$
- Split into training and testing tests and evaluate a linear regression model: what happens when  $p$  becomes large?

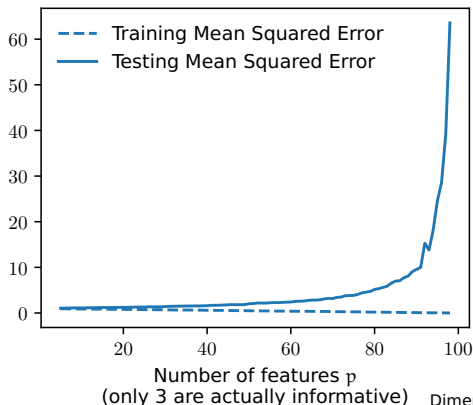
See [sklearn.datasets.make\\_regression](#) for generating data



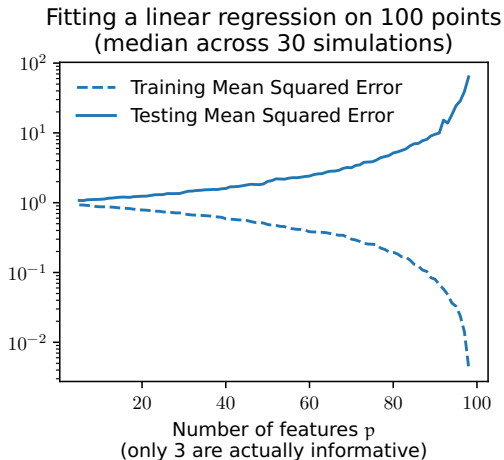
# Model complexity: overfitting

- Model complexity increases with dimension.
- Example: a linear model in dimension  $p$  can fit exactly (0 training error) any set of  $p + 1$  points.
- Risk of overfitting: fitting exactly training data but failing on test data

Fitting a linear regression on 100 points  
(median across 30 simulations)

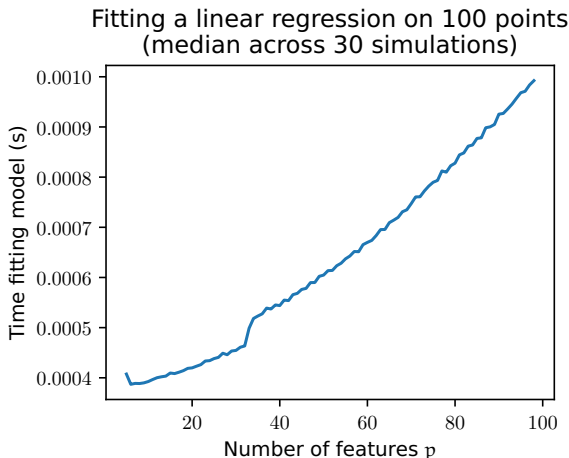


# Same plot in log scale



# Cost of fitting many parameters

- Many algorithms require polynomial time in  $p$
- Implementations often make copies of the design matrix (e.g. for centering & rescaling)





# Solution 1: univariate feature selection

- a.k.a. feature screening, filtering ...
- Check features (columns of  $X$ ) one by one for association with the output  $y$
- Keep only a fixed number or percentage of the features

# Solution 1: univariate feature selection

- a.k.a. feature screening, filtering ...
- Check features (columns of  $X$ ) one by one for association with the output  $y$
- Keep only a fixed number or percentage of the features

## Simple (linear) association criteria

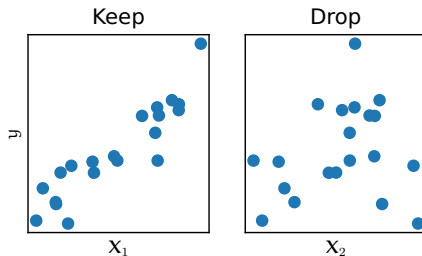
- for regression: correlation
- for classification: ANalysis Of VAriance

Read more in the scikit-learn user guide

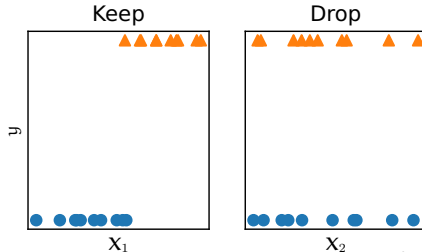
[https://scikit-learn.org/stable/modules/feature\\_selection.html#feature-selection](https://scikit-learn.org/stable/modules/feature_selection.html#feature-selection)

# Simple selection criteria

- Regression: correlation

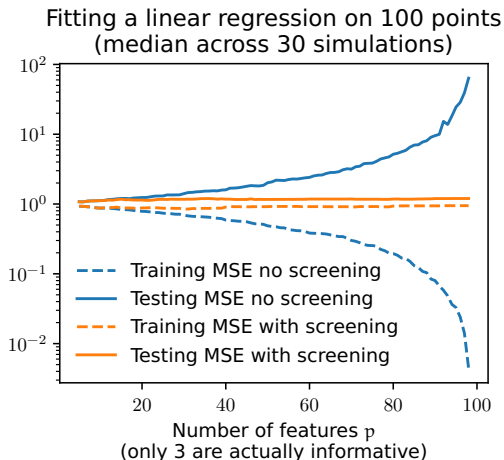


- Classification: ANOVA

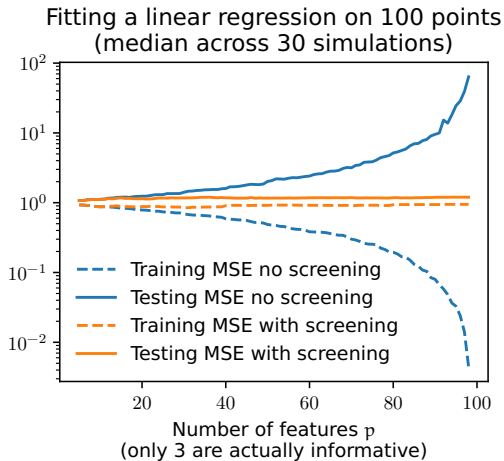


# Univariate feature selection

Keeping only the 10 best features (most correlated with  $y$ )

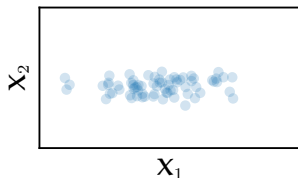


# Same plot in log scale

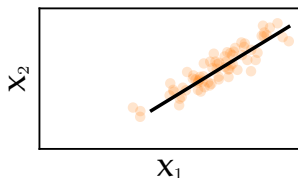


## Solution 2: linear decomposition methods

Maybe OK to drop  $X_2$ :



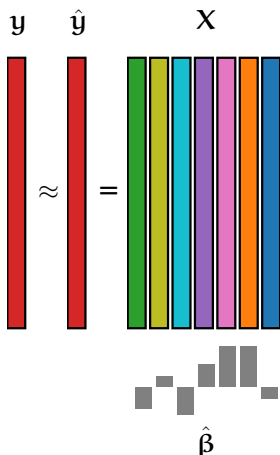
Data low-dimensional but no feature can be dropped:



Find a better referential in which to represent the data

# Linear regression: projection on the column space of $X$

$$\hat{y} = X\hat{\beta} \quad (7)$$

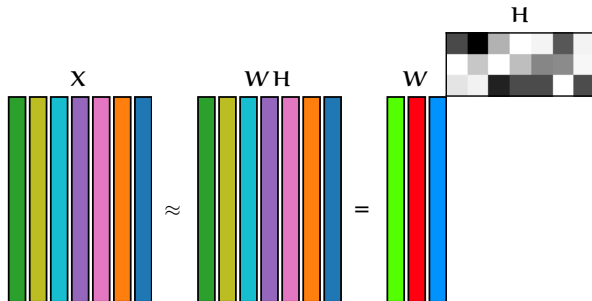


- Too many features: high variance & unstable solution
- Feature selection: drop some columns of  $X$
- Other ways to build a family of  $k$  vectors on which to regress  $y$ ?

# Linear decomposition: low-rank approximation of $\mathbf{X}$

Minimize

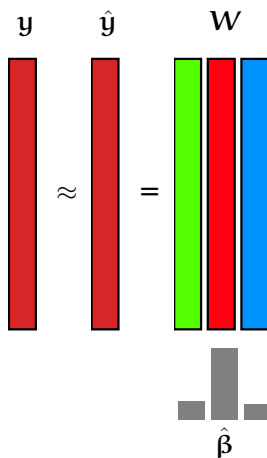
$$\|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2 = \sum_{i,j} (\mathbf{X}_{i,j} - (\mathbf{W}\mathbf{H})_{i,j})^2 \quad (8)$$





# Linear regression after dimensionality reduction

$$\hat{y} = W \hat{\beta} \quad (9)$$



## Prediction for a new data point $\mathbf{x} \in \mathbb{R}^p$

- Find the combination of rows of  $\mathbf{H}$  that is closest to  $\mathbf{x}$ : regress  $\mathbf{x}$  on  $\mathbf{H}^T$
- Multiply by  $\hat{\boldsymbol{\beta}}$

$$\mathbf{x} \in \mathbb{R}^p \rightarrow \text{projection} \rightarrow \mathbf{w} \in \mathbb{R}^k \rightarrow \langle \cdot, \hat{\boldsymbol{\beta}} \rangle \rightarrow \hat{y} \in \mathbb{R} \quad (10)$$

# Principal Component Analysis

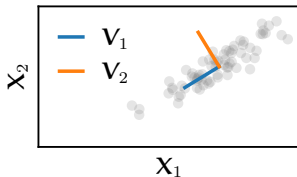
- Singular Value Decomposition of  $X$ :

$$X = U S V^T \quad (11)$$

with  $X \in \mathbb{R}^{n \times p}$ ,  $U \in \mathbb{R}^{n \times r}$ ,  $S \in \mathbb{R}^{r \times r}$ ,  $V \in \mathbb{R}^{r \times p}$

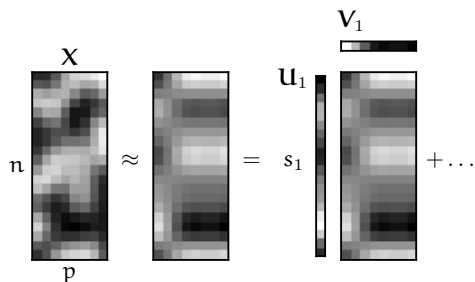
- $r = \min(n, p)$
- $S \succeq 0$  diagonal with decreasing values  $s_j$  along the diagonal
- $U^T U = I_r$
- $V^T V = I_r$

Truncating the SVD to keep only the first  $k$  components gives the best rank- $k$  approximation of  $X$



# Singular Value Decomposition

$$\mathbf{X} = \mathbf{U} \mathbf{S} \mathbf{V}^T \quad (12)$$



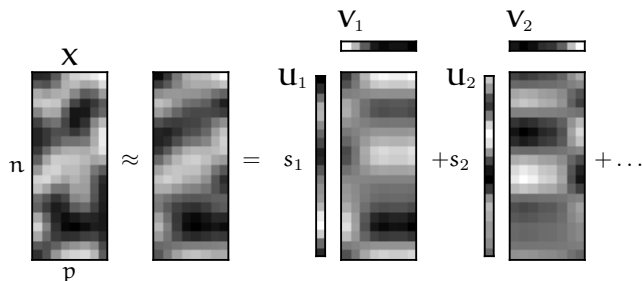
Explained variance: 0.53

$$\mathbf{U}^T \mathbf{U} = \mathbf{I}_p \quad (13)$$

$$\mathbf{V}^T \mathbf{V} = \mathbf{I}_p \quad (14)$$

# Singular Value Decomposition

$$\mathbf{X} = \mathbf{U} \mathbf{S} \mathbf{V}^T \quad (15)$$



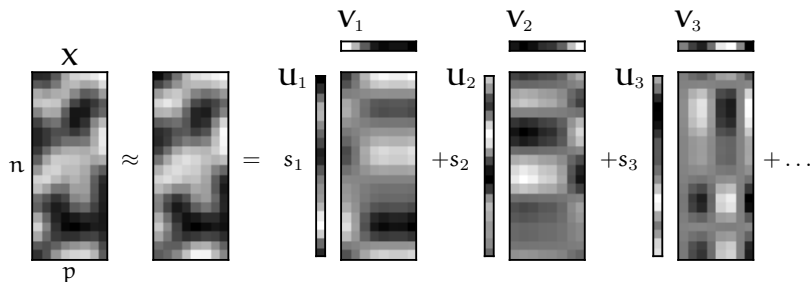
Explained variance: 0.84

$$\mathbf{U}^T \mathbf{U} = \mathbf{I}_p \quad (16)$$

$$\mathbf{V}^T \mathbf{V} = \mathbf{I}_p \quad (17)$$

# Singular Value Decomposition

$$\mathbf{X} = \mathbf{U} \mathbf{S} \mathbf{V}^T \quad (18)$$



Explained variance: 0.97

$$\mathbf{U}^T \mathbf{U} = \mathbf{I}_p \quad (19)$$

$$\mathbf{V}^T \mathbf{V} = \mathbf{I}_p \quad (20)$$

# Other decomposition methods

Many other methods use the same objective (sum of squared reconstruction errors), but add penalties or constraints on the factors

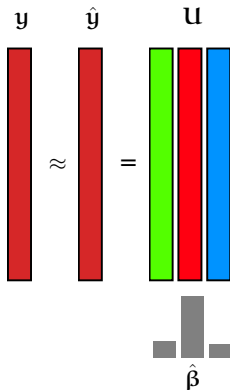
- Dictionary Learning
- Non-negative Matrix Factorization
- K-means clustering
- ...

What about  $y$ ?

- PCA is an example of *unsupervised* learning: it does not use  $y$
- Some other methods take it into account: e.g. Partial Least Squares

# Ridge regression and PCA

- Both ridge regression and PC regression compute the coordinates of  $y$  in the basis given by the SVD of  $X$
- Ridge shrinks the coordinate along  $U_j$  by a factor  $s_j^2/(s_j^2 + \lambda)$
- PC regression sets the coordinates to 0 except for those corresponding to the  $k$  largest  $s_j$ : shrinks by a factor  $1_{\{j \leq k\}}$





# Setting hyperparameters

How can we choose:

- Number of features or PCA components  $k$ ?
- The ridge hyperparameter  $\lambda$ ?

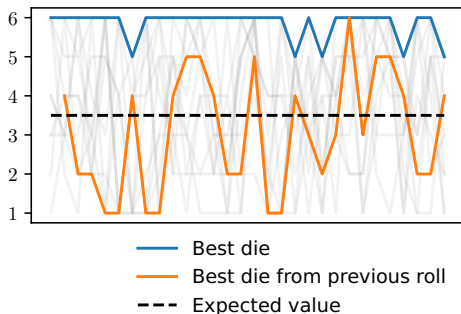
Try a few and pick the best one...

But measure its performance on separate data!

# Need for fresh test data

When you hear "best", "maximum", "select", ... think "bias"

- I have 4 dice and want to find one that rolls high numbers
- I roll them all once and select the die that gives the highest number
- The selected die rolled a 5. Is 5 a good estimate of that die's average result? What if I had 1,000 dice?
- I need to roll it again to get an unbiased estimate



# Nested cross-validation

When you hear "best", "maximum", "select", ... think "bias"

# Nested cross-validation

When you hear "best", "maximum", "select", ... think "bias"

## Setting the parameters

- **Select**  $\beta$  that gives the **best** prediction on training data
- The prediction score for  $\hat{\beta}$  is biased: compute a new score on unseen test data.

# Nested cross-validation

When you hear "best", "maximum", "select", ... think "bias"

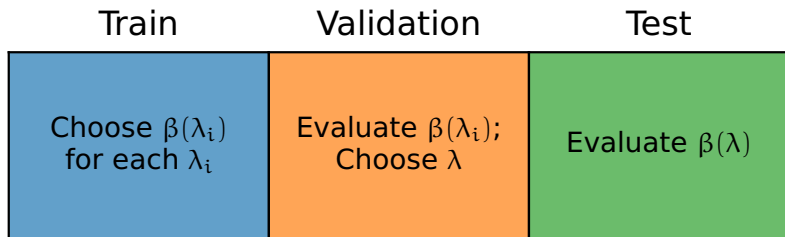
## Setting the parameters

- **Select**  $\beta$  that gives the **best** prediction on training data
- The prediction score for  $\hat{\beta}$  is biased: compute a new score on unseen test data.

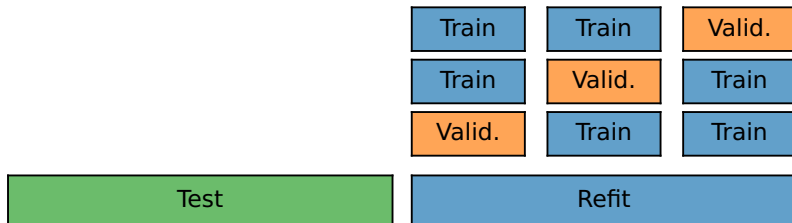
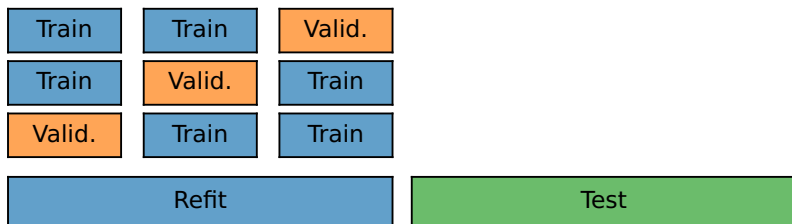
## Setting the hyperparameters

- Repeat step 1 for a few values of  $\lambda$ ,  $k$ , etc. ., fitting and testing several models
- **Select** the hyperparameter that obtains the **best** prediction on test data
- The prediction score of that model on *test* data is biased: evaluate it again on unseen data

# One split



# Nested cross-validation



see `sklearn.model_selection.GridSearchCV`

# Some common pitfalls with cross-validation

Fitting part of the pipeline on the whole dataset

- e.g. fit PCA on all data, then do cross-validation on dim-reduced dataset
- use `sklearn.pipeline.Pipeline`



# Some common pitfalls with cross-validation

## Fitting part of the pipeline on the whole dataset

- e.g. fit PCA on all data, then do cross-validation on dim-reduced dataset
- use `sklearn.pipeline.Pipeline`

## Ignoring dependencies between samples

- Multiple datapoints per participant
- Time series

# Some common pitfalls with cross-validation

## Fitting part of the pipeline on the whole dataset

- e.g. fit PCA on all data, then do cross-validation on dim-reduced dataset
- use `sklearn.pipeline.Pipeline`

## Ignoring dependencies between samples

- Multiple datapoints per participant
- Time series

## Ignoring dependencies between CV scores

- Training sets overlap: cross-validation scores of different splits are not independent

# Some common pitfalls with cross-validation

## Fitting part of the pipeline on the whole dataset

- e.g. fit PCA on all data, then do cross-validation on dim-reduced dataset
- use `sklearn.pipeline.Pipeline`

## Ignoring dependencies between samples

- Multiple datapoints per participant
- Time series

## Ignoring dependencies between CV scores

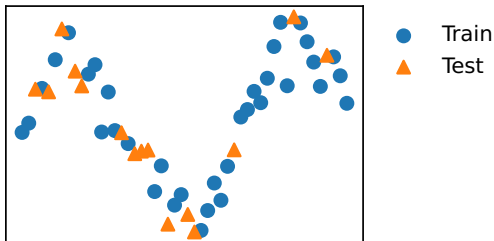
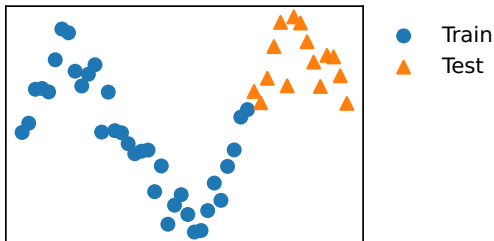
- Training sets overlap: cross-validation scores of different splits are not independent

## Over-interpreting good CV scores

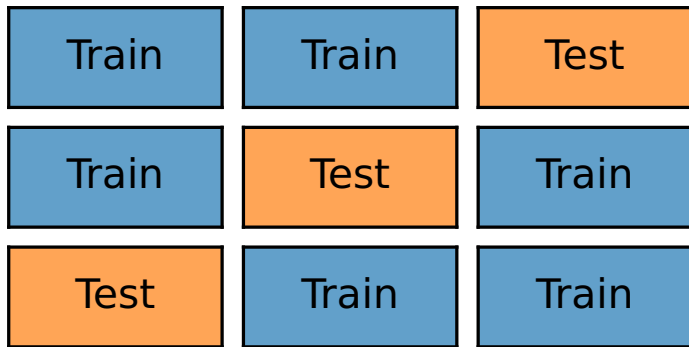
- Good CV scores on one dataset do not mean the model will always perform well on a new dataset

# Split choice example: time series

Which is easier?



Remember that CV training sets overlap



So the scores are not independent! Their variance can be underestimated.

# Supervised learning with fMRI

- Predict in which site / with which scanner a resting-state fMRI sequence was acquired

# The prediction pipeline

- Masking: extracting voxels that are inside the brain
- Connectivity: measuring correlations between brain regions to build a feature vector for each participant
- Univariate feature selection with ANalysis Of VAriance
- Classifier: logistic regression

# Implementation: in class