Decoding analyses

Neuroimaging: Pattern Analysis 2017

Scope

- Conceptual explanation of decoding/machine learning (ML) and related concepts;
- **Not** about mathematical basis of ML algorithms!

Learning goals

- Understand the difference (and surprising overlap) between statistics and machine learning;
- Recognize the major different "flavors" of decoding analyses;
- Understand the concepts of overfitting and cross-validation
- Know the most important steps in a decoding analysis pipeline;

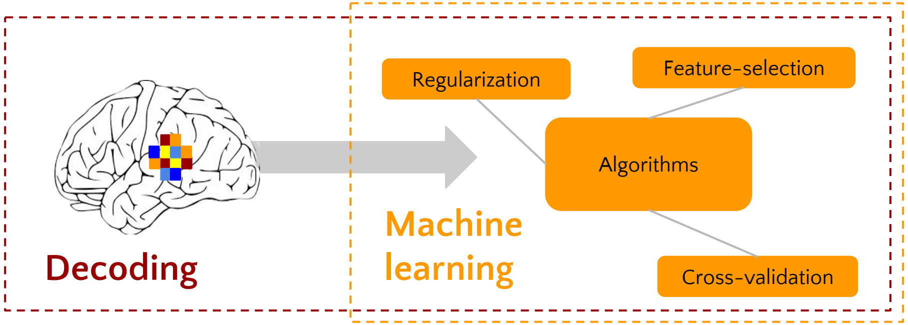
Contents

- **PART 1**: what is machine learning?
  - ... and how does it differ from statistics?
  - What "flavors" of decoding analyses are there?
  - Important ML concepts: cross-validation, overfitting
- **PART 2**: how to set up a ML pipeline?
  - From partitioning to significance testing

PART 1:
WHAT IS MACHINE LEARNING?

Terminology

- Decoding ≈ machine learning
  - Decoding is the neuroimaging-specific application of machine learning algorithms and techniques;



Machine learning = statistics?

- Defining **machine learning** is (in a way) trivial:
  - Algorithms that learn how to model the data without an explicit instruction how to do so;
- But how it that different from traditional statistics?
  - Like the familiar GLM (linear regression, t-tests)?
  - Similar, but different …

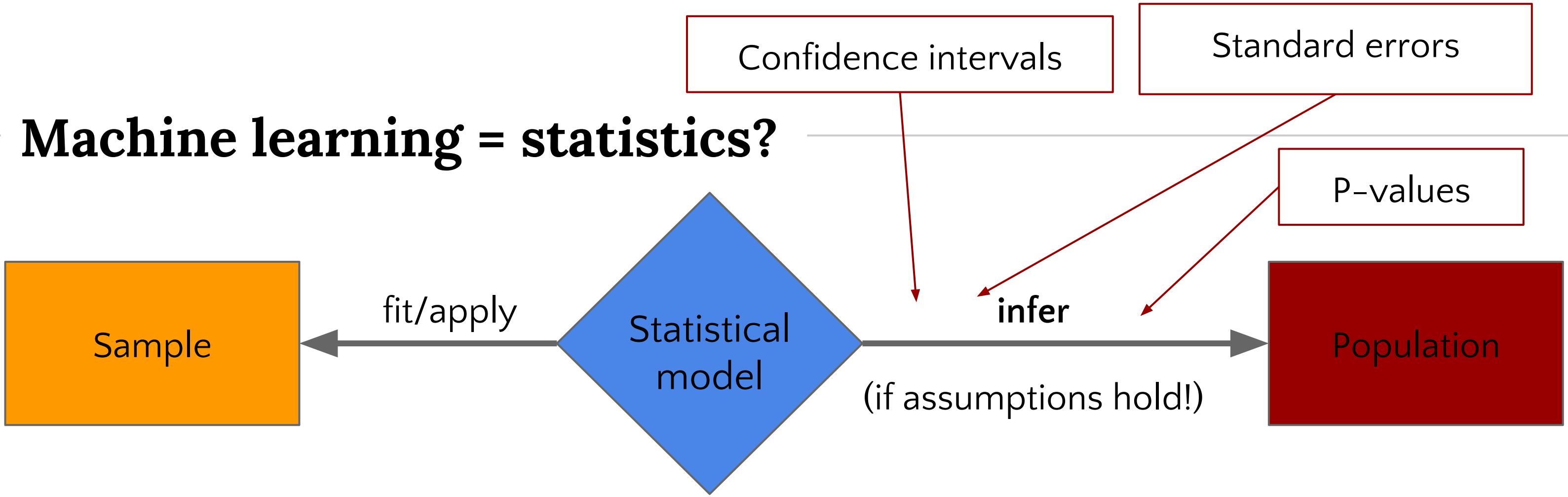
Machine learning = statistics?

- They have different **origins**:
  - Statistics is a subfield from mathematics
  - Machine learning is a subfield from computer science
  - "Science vs. engineering"
- They have a different **goal**:
  - Statistical models aim for **inference** about the population based on a limited sample
  - Machine learning models aim for accurate **prediction** of new samples

Machine learning = statistics?

- Psychologists (you included) are taught **statistics**: how to make inferences about general psychological "laws" based on a limited sample:
  - "I've tested the reaction time of 40 people (**sample**) before and after drinking coffee …
  - … and based on the significant results of a paired sample t-test, $t(39)$, $p < 0.05$ (**statistical test**) …
  - … I conclude that caffeine improves reaction times (statement about **population**)

Machine learning = statistics?

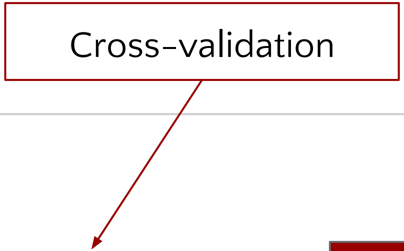


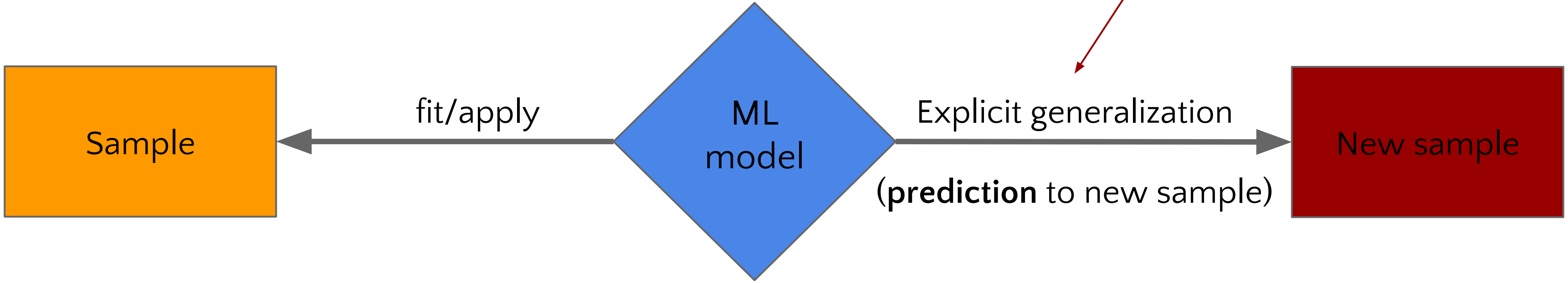
- Crucially: we are quite certain that our findings in the sample will generalize to the population, *if and only if assumptions of the model hold (and sample = truly random)*
- Uses concepts like standard errors, confidence intervals, and p-values to generalize the model to the population

Machine learning = statistics?

- ML models do not aim for **inference**, but aim for **prediction**
  - Instead of assuming findings will generalize to the population, ML analyses in fact ***literally check*** whether it generalizes;
  - It's like they're saying: "I don't give a shit about assumptions – if it works, it works."

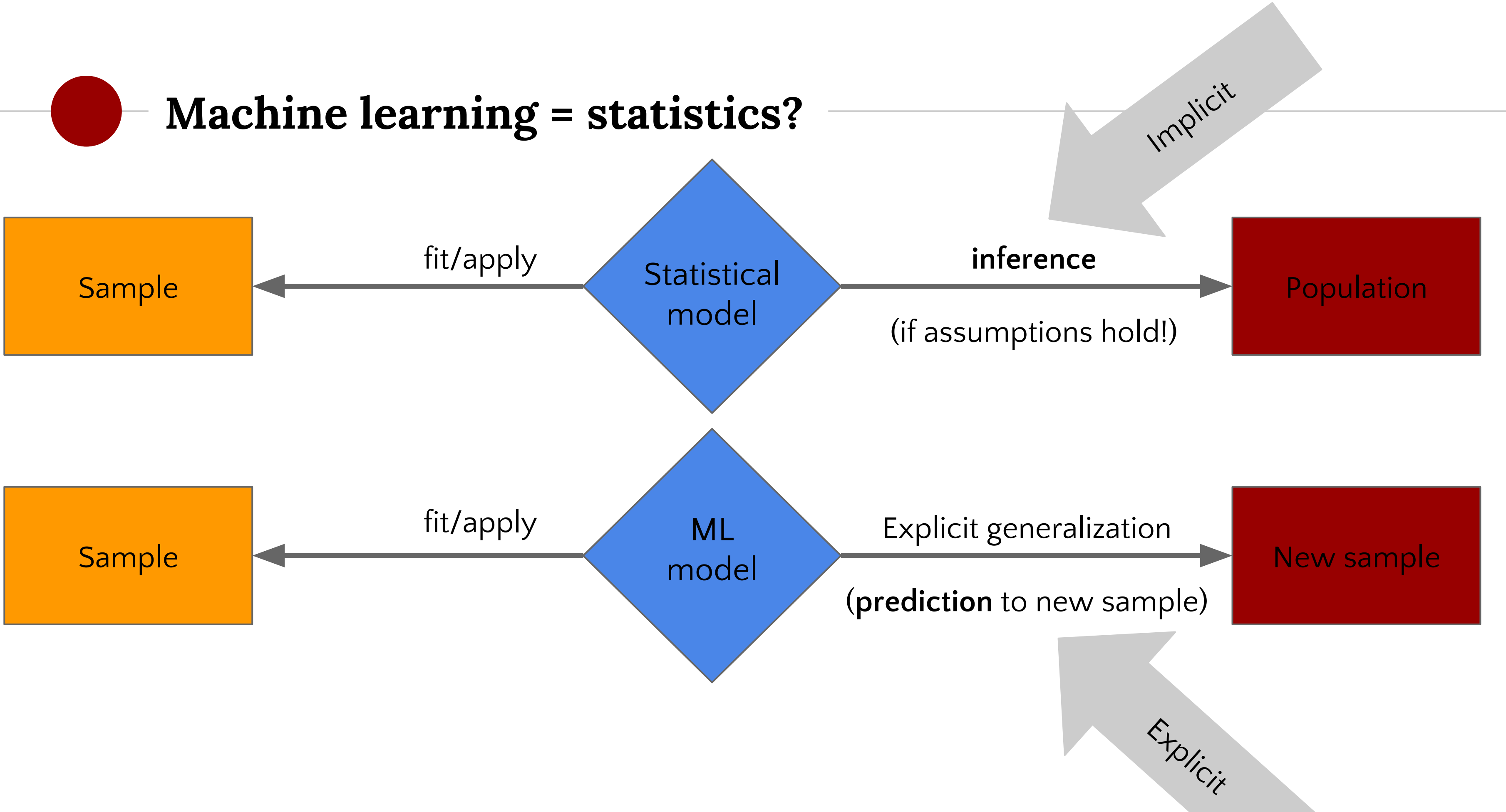
Machine learning = statistics?





- Instead of assuming that the findings from the model will generalize beyond the sample, ML tests this explicitly by applying this to ("predicting") a new sample
- New sample is concretely part of your dataset! (not like "the population")

Machine learning = statistics?



Machine learning = statistics?

- While having a different goal, in the end ***both types of models simply try to model the data***;
- Take for example linear regression:
  - It has a **statistics** 'version' (as defined in the GLM) ...
  - ... and an **ML** 'version' (using a mathematical technique called gradient descent to find the optimal βs)

So …?

- As said, statistics and ML are the **same, yet different**;
  - Both aim to model the data (with different techniques) …
  - … but have a different way to generalize findings
- "But why do we have to learn a whole new paradigm (ML), then?", you might ask …

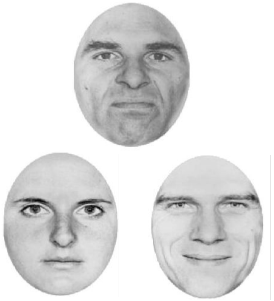
Good question!

- Traditional statistical models do not fare well with **high-dimensional problems**
  - In decoding: dimensionality = amount of voxels
- Neuroimaging data likely **violates** many assumptions of traditional statistical models …
- Sometimes, decoding analyses actually need prediction: e.g. predict clinical treatment outcome

Back to the brain

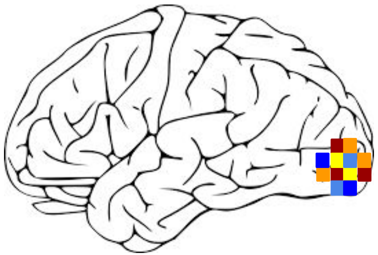
"Features in the **world**"

"Features in the **brain**"



y = model(x)

DECODING



What happens in that arrow?

Machine learning model

- Machine learning algorithms try to model the **features** (X) such that they approximate the **target** (y)
- In fMRI (decoding): can the **voxel activities** (X) be weighted such that they approximate the **feature-of-interest** (y)?

Machine learning model

Samples-by-features matrix (BRAIN)

$$\hat{y} = f(X)$$

The predicted target variable, numerically coded (WORLD)

ML algorithm that specifies how to weigh X

Support vector machines

(Logistic) regression

Decision trees

f(X) is simply weighting!

$$\hat{y} = f(X) = \boxed{\beta X}$$

β denotes the weighting parameters (or just "parameters" or "coefficients") for our features (in X)

βX denotes the (matrix) product of the weighting parameters and X – which means that y is approximated as a weighted linear combination of features

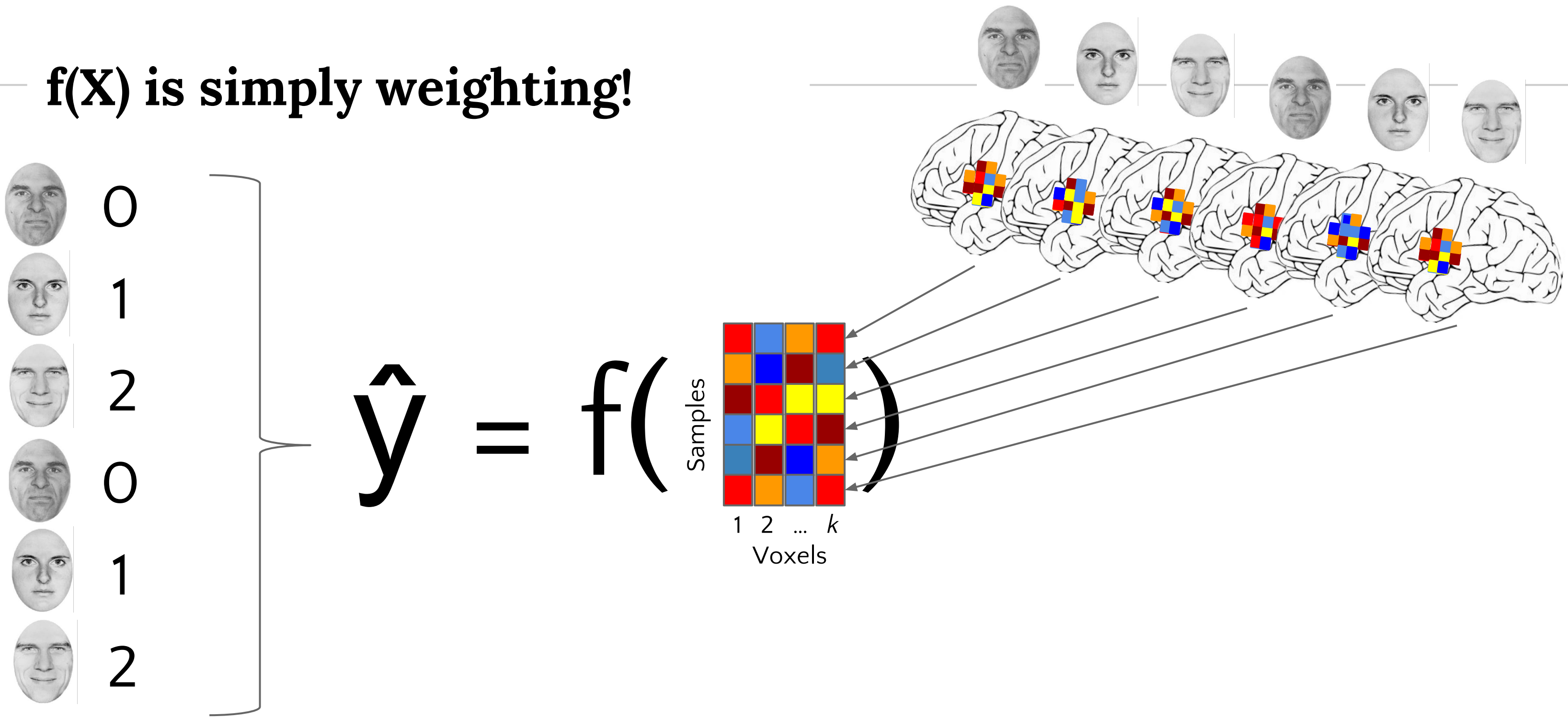
Linear vs. non-linear

- Disclaimer: the "βX" term implies that it is a **linear** model (i.e. a linear weighting of features)
- Decoding analyses almost always use linear models, because **most world-brain relations are probably linear** (cf. Naselaris & Kay)
- Non-linear models exist, but are **rarely used**
  - Also because they often perform worse than linear models

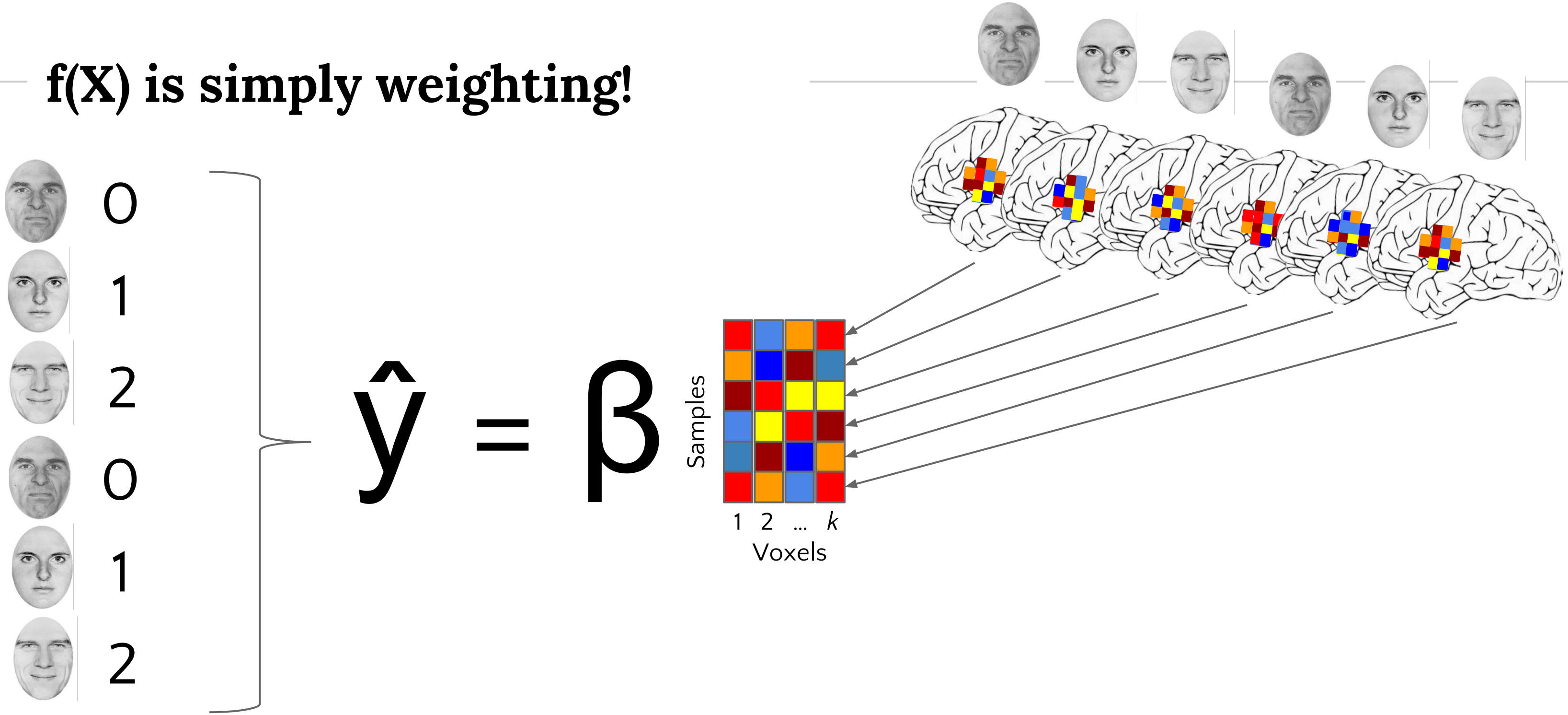
f(X) is weighting!

$$\hat{y} = f(\,X\,)$$

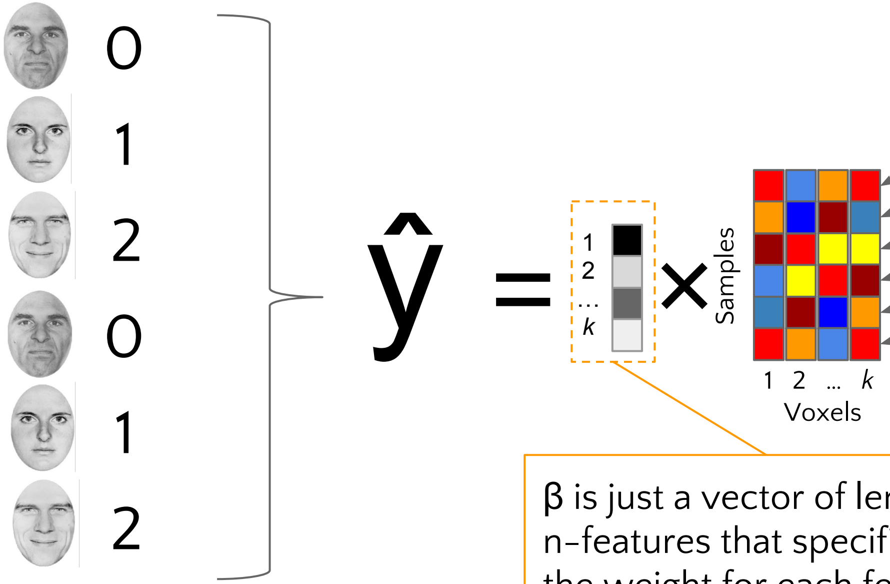
f(X) is simply weighting!

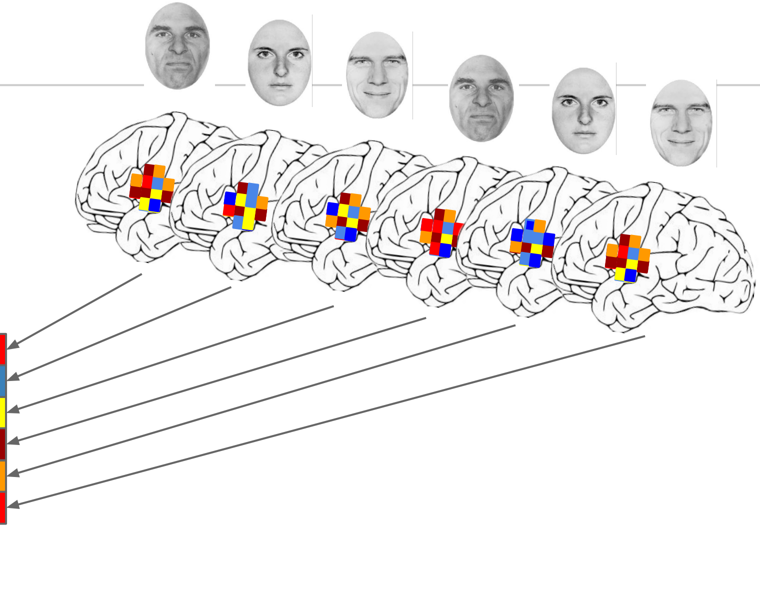


f(X) is simply weighting!



f(X) is simply weighting!





β is just a vector of length n-features that specifies the weight for each feature to optimally approximate y

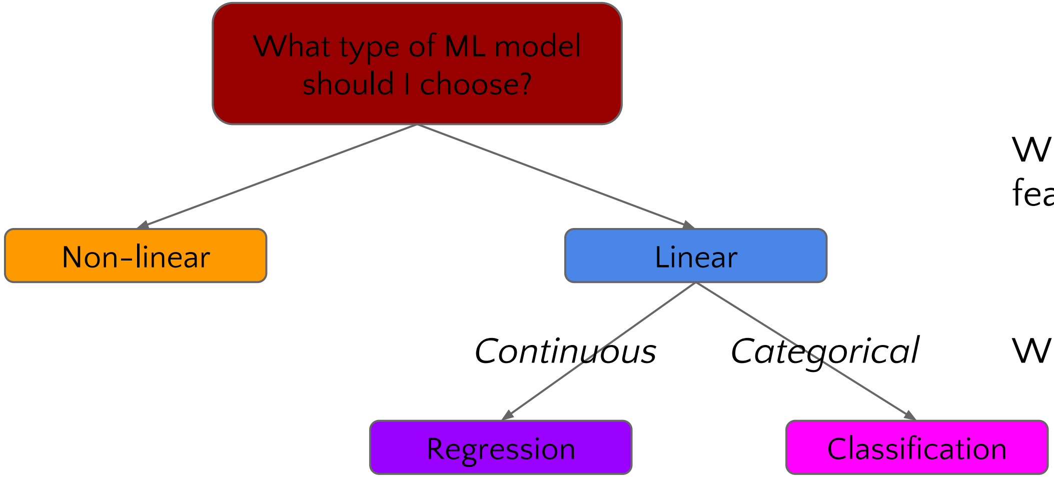
Summary

- ML models (f) find **parameters** (β) that weigh **features** (X) such that they optimally approximate the **target** (y)
- Applied to fMRI: we decode from the **brain** (X) to the **world** (y) by **weighing** voxels!

Questions so far?

Flavors of ML algorithms

- ML algorithms – f() – exist in different 'flavors'



What do you assume is the relation between features and the target? (Probably linear ...)

What is the nature of your target feature?

Regression vs. classification

Regression

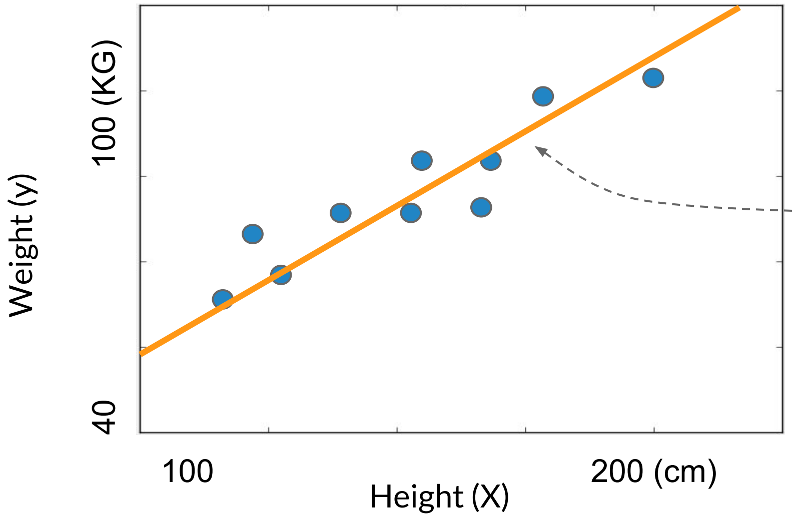
- Target (y) is continuous
- "Predict someone's weight based on someone's height"
- Example models: linear regression, ridge regression, LASSO

Classification

- Target (y) is categorical
- "Predict someone's gender based on someone's height"
- Example models: support vector machine (SVM), logistic regression, decision trees,

Regression

- Regression models a continuous variable



$$\hat{y}_{weight} = 0.5 * X_{height} = \beta_{height} X_{height}$$

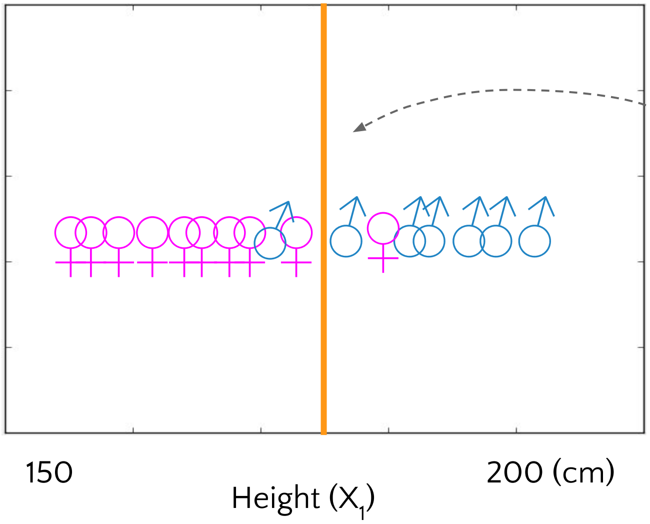
This is exactly like 'univariate' regression models, but instead of modelling in the encoding direction, this is in the decoding direction!

* We are ignoring the intercept here for simplicity

Regression vs. classification

- Classification models a categorical variable

y = {♀,♂}



"squash()" is a function that forces the output of βX to be categorical

$\hat{y} = \boxed{\text{squash}}(\beta 0.5 X_{height\ height}))$

"If βX > 5, then ŷ = ♂
otherwise ŷ = ♀

Test your knowledge!

Subjects perform a memory task in which they have to give responses. Their responses can be either correct or incorrect.

I want to analyze whether the patterns in parietal cortex are predictive of whether someone is going to respond (in)correctly.

Classification ? Regression?

Test your knowledge!

During fMRI acquisition, subject see a set of images of varying emotional valence (from 0, very negative, to 100, very positive).

I want to decode stimulus valence from the bilateral insula.

Classification ? Regression?

Test your knowledge!

Subjects perform an attentional blink task in the scanner (during which we measure fMRI).

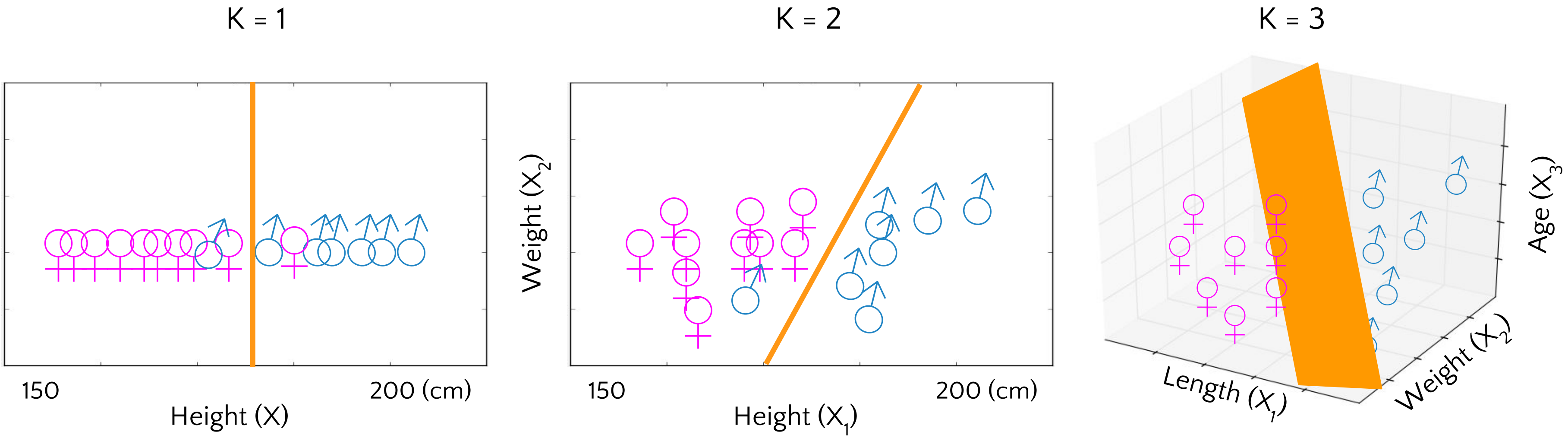
I want to predict whether someone has a relatively high IQ (>100) or low IQ (<100) based upon the patterns in dorsolateral PFC during the attentional blink task.

Classification ? Regression?

Regression & classification

- Both types of model try to approximate the target by '**weighting**' the features (X)
  - Additionally, classification algorithms need a "squash" function to convert the outputs of βX to a categorical value
- The examples were simplistic (K = 1); usually, ML models operate on high-dimensional data!

Dimensionality



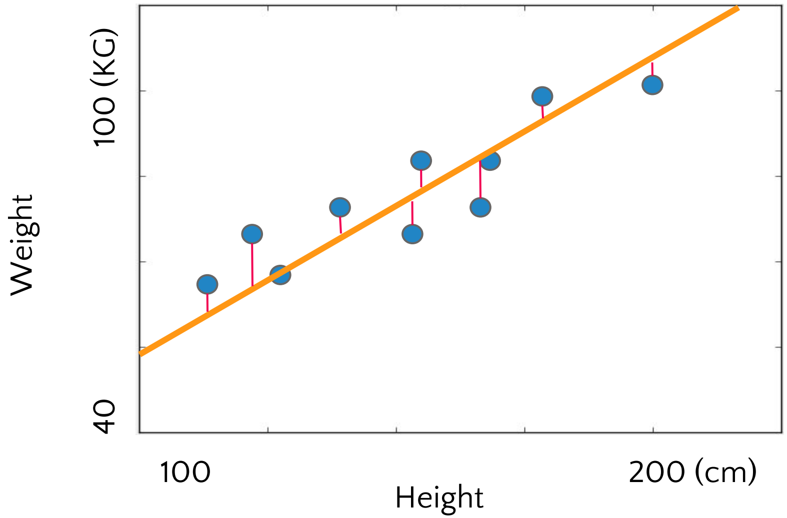
K = >3? → Difficult to visualize, but process is the same: weighting features such that a multidimensional plane is able to separate classes as well as possible in K-dimensional space

Model performance

- We know what ML models do (find weighting parameters β to approximate y), but how do we evaluate the model?
- In other words, what is the model performance?

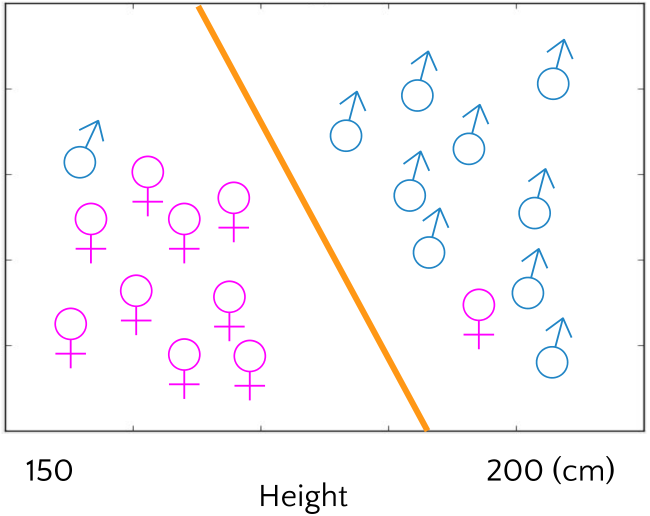
Model performance

Regression



R^2 = 0.92 [explained variance]
MSE = 8.2 [mean deviation from prediction]

Classification

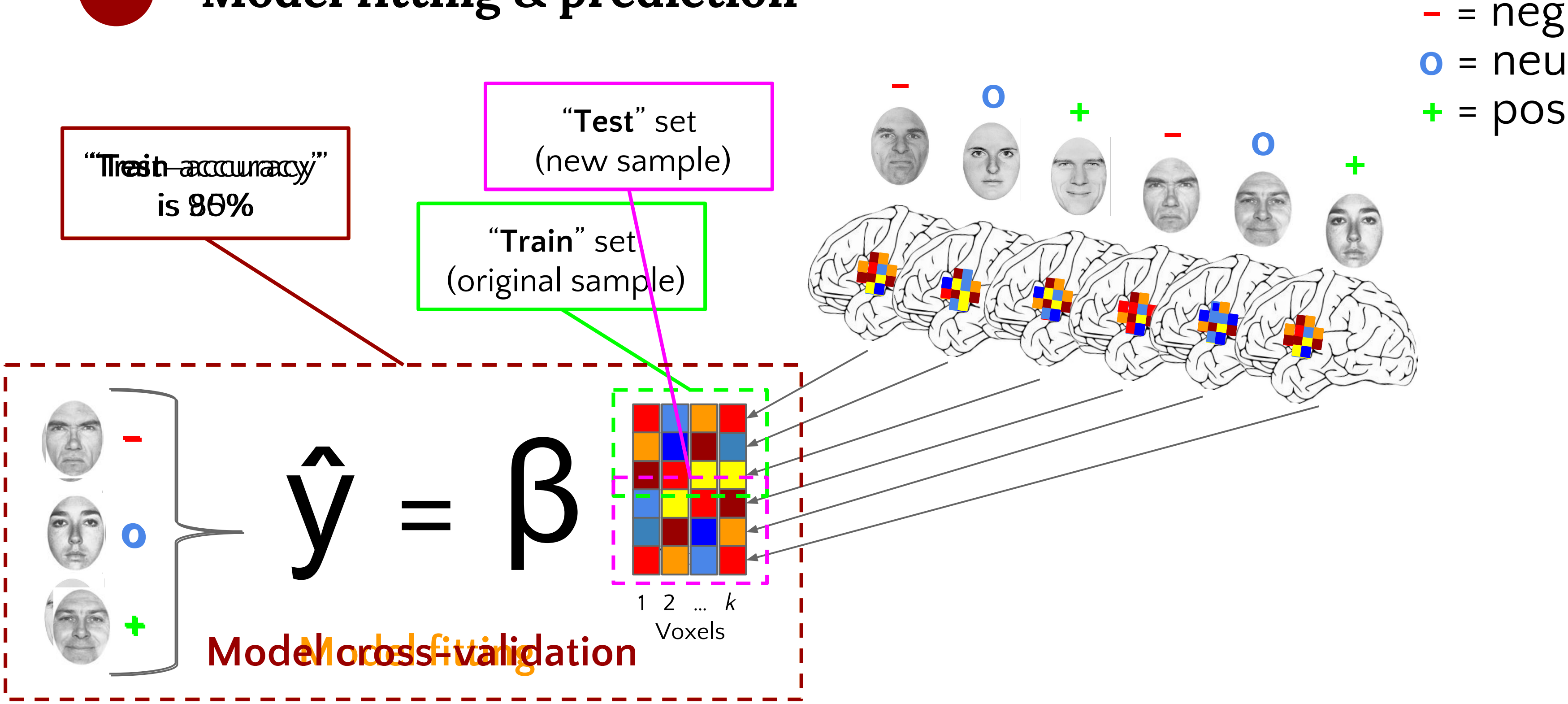


Accuracy = 18 / 20 = 90% [percent correct]

Model performance

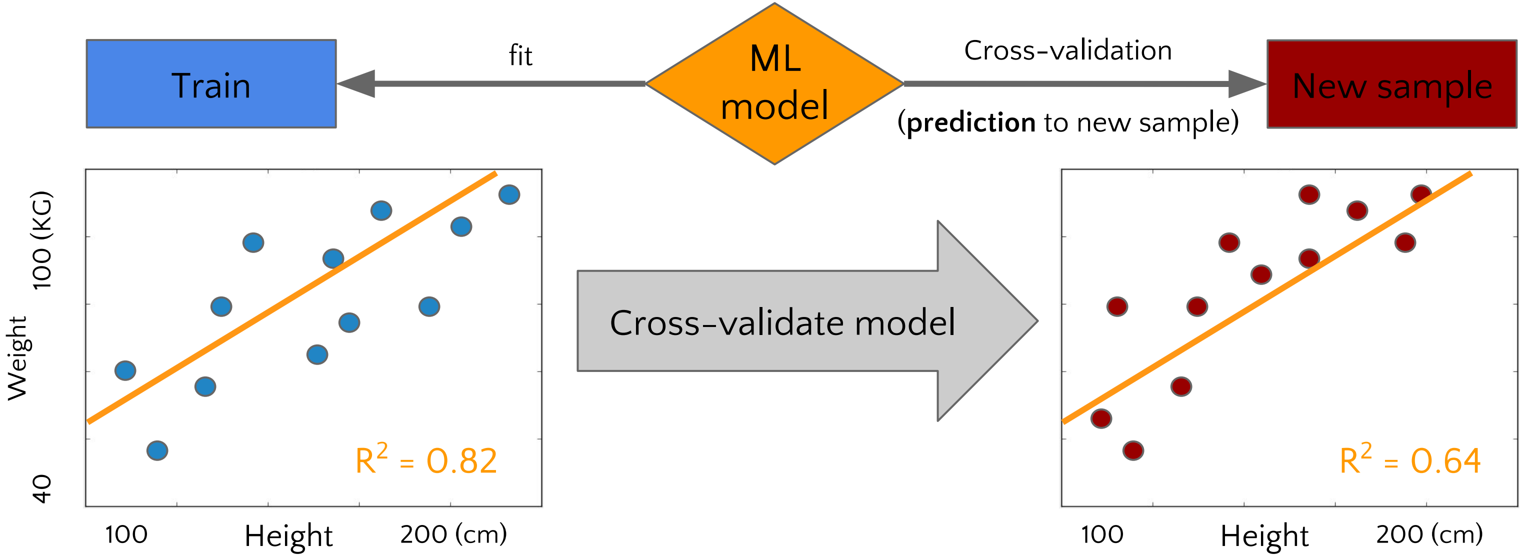
- Performance is often evaluated not only on the original sample, but also on a **“new sample”**
- This process is called **“cross-validation”**

Model fitting & prediction



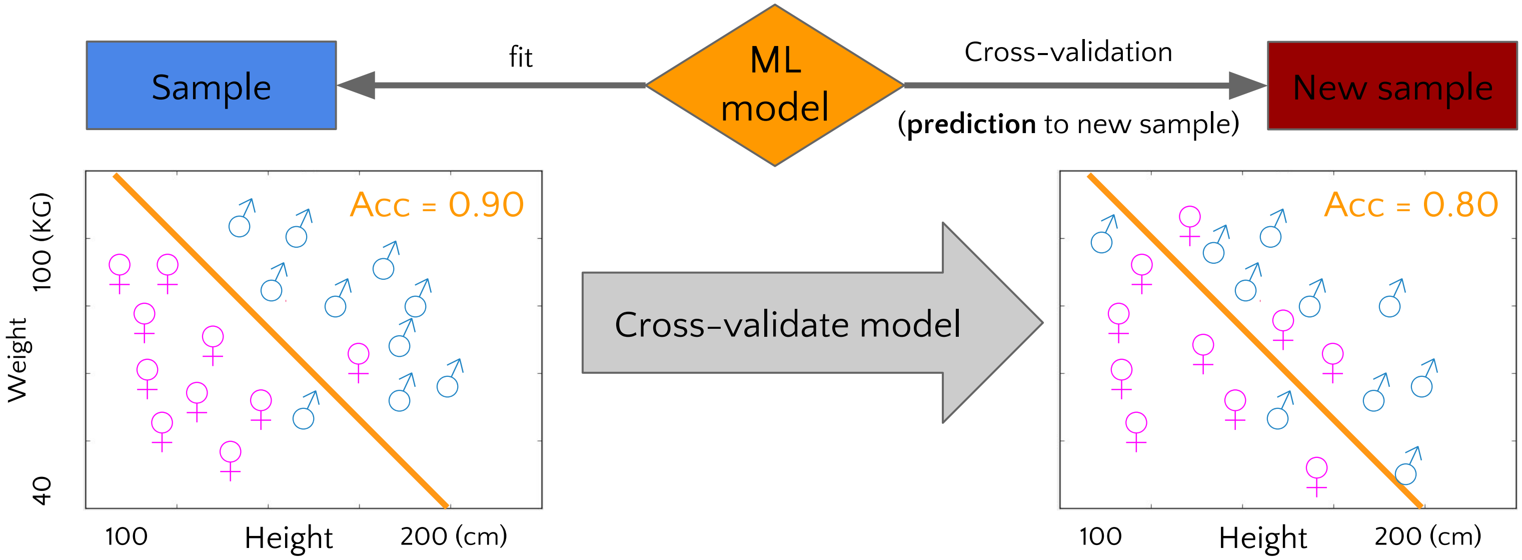
Model performance

- Model performance is often evaluated on a new ("unseen") sample: **cross-validation**



Model performance

- Model performance is often evaluated on a new ("unseen") sample: **cross-validation**



Why do we want (need) to do cross-validation?

Model fit ≠ good prediction

Overfitting

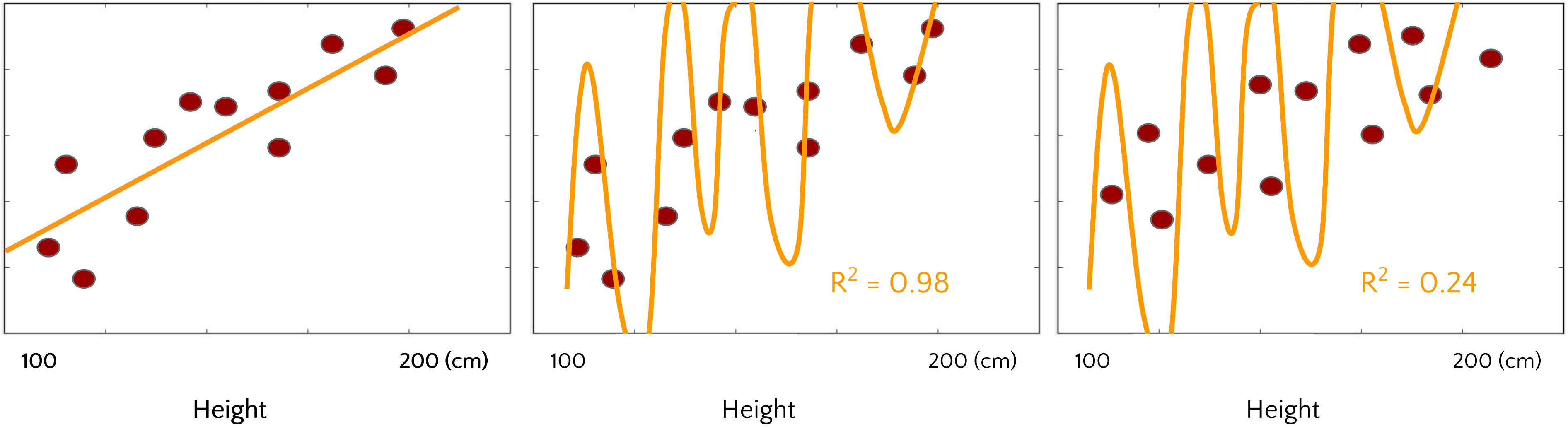
- When your fit on your **train-set** is better than on your **test-set**, you're **overfitting**
- Overfitting means you're modelling **noise** instead of **signal**

Overfitting

Overfitting

True model + noise

(over)Fitted model > New sample



Overfitting

- Overfitting = **modeling noise**
- Noise = **random** (uncorrelated from sample to sample)
- Therefore, a model based on noise will not **generalize**

What causes overfitting?

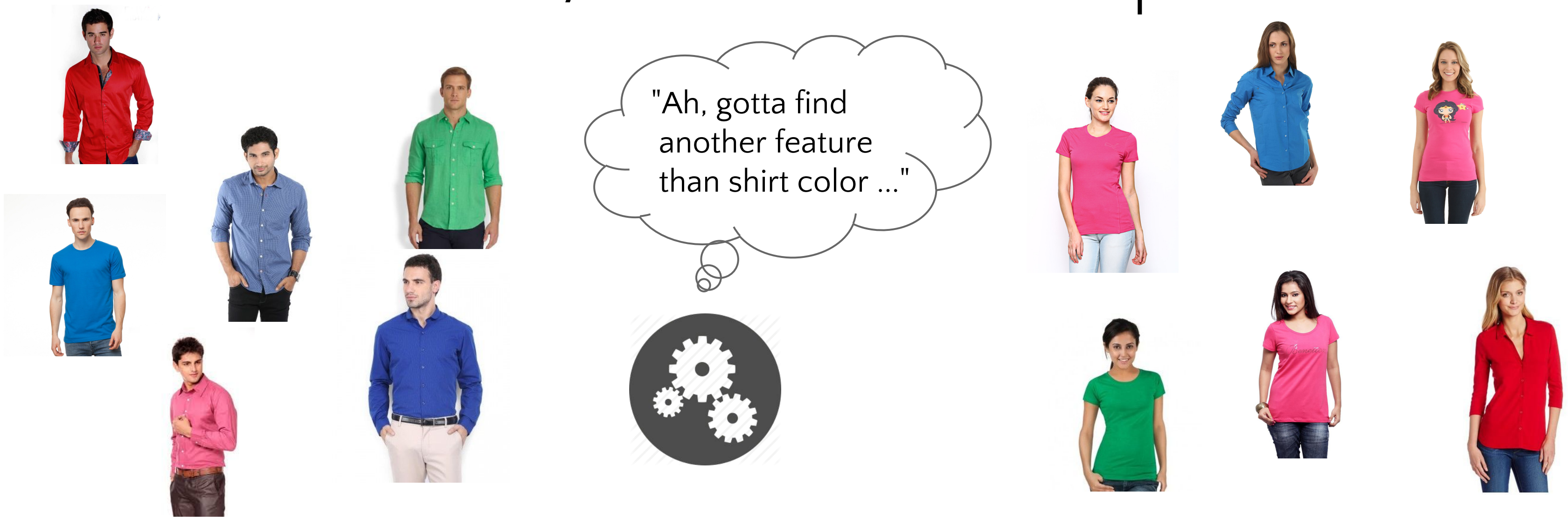
- A small **sample/feature-ratio** often causes overfitting
- When there are few samples *or* many features, models may fit on random/accidental relationships

What causes overfitting?

- When there are few samples, models may fit on random/accidental relationships

What causes overfitting?

- When there are few samples, models may fit on random/accidental relationships



What causes overfitting?

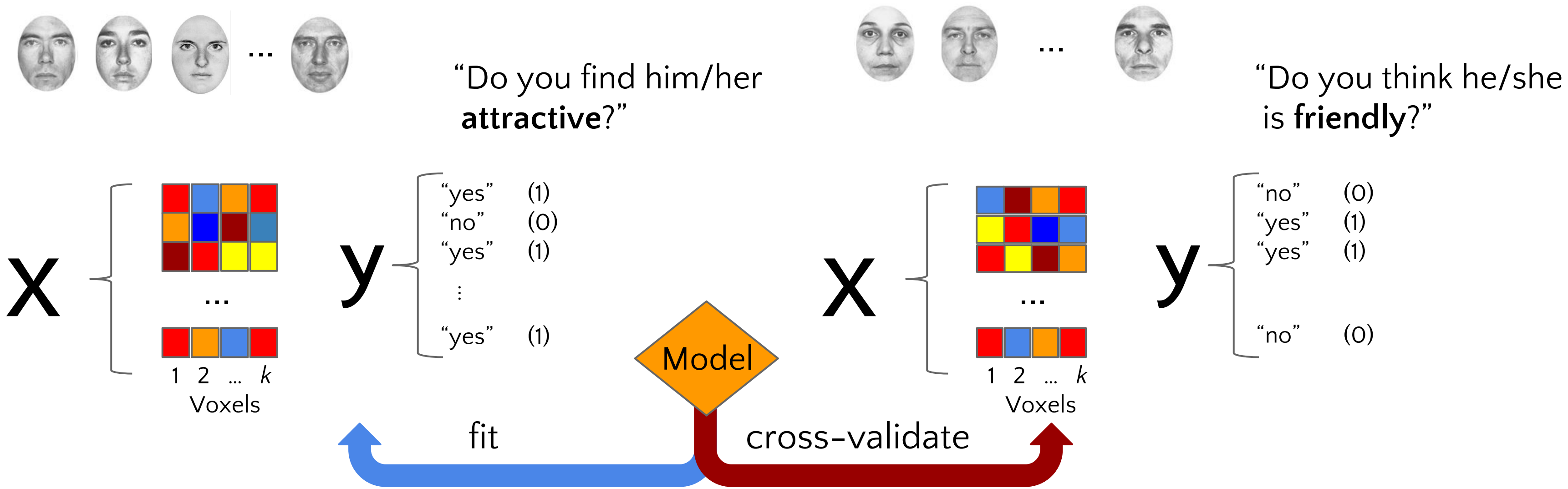
- Two options:
  - Gather **more data** (not always feasible)
  - **Reduce** the amount of **features** (more about this later)
  - [**Regularization** – beyond the scope of this course!]
- Feature selection/extraction is an often-used technique in decoding analyses

Cross-decoding!

- Sometimes, decoding analyses use a specific form of cross-validation to perform **cross-decoding**
- In cross-decoding, you aim to show “informational overlap” between two type of representations

Cross-decoding!

- For example: suppose you have the hypothesis attractiveness drives the perception of friendliness



Summary

- ML models find weights to approximate a continuous (regression) or categorical (classification) dependent variable (y)
- Good fit ≠ good generalization …
- … therefore, cross-validate the model!
- Optimize the sample/feature ratio to reduce overfitting (spurious feature-DV correlations)
- Cross-decoding uses cross-validation to uncover shared representations ('informational overlap')

PART 2:
BUILDING A DECODING PIPELINE

A typical decoding pipeline

0. Pattern extraction & preparation
1. Partitioning train/test
2. Feature selection/extraction
3. Model fitting (TRAIN)
4. Model generalization (TEST)
5. Statistical test of performance
6. Optional: plot weights

A typical decoding pipeline

0. Pattern extraction & preparation
1. Partitioning train/test
2. Feature selection/extraction
3. Model fitting (TRAIN)
4. Model generalization (TEST)
5. Statistical test of performance
6. Optional: plot weights

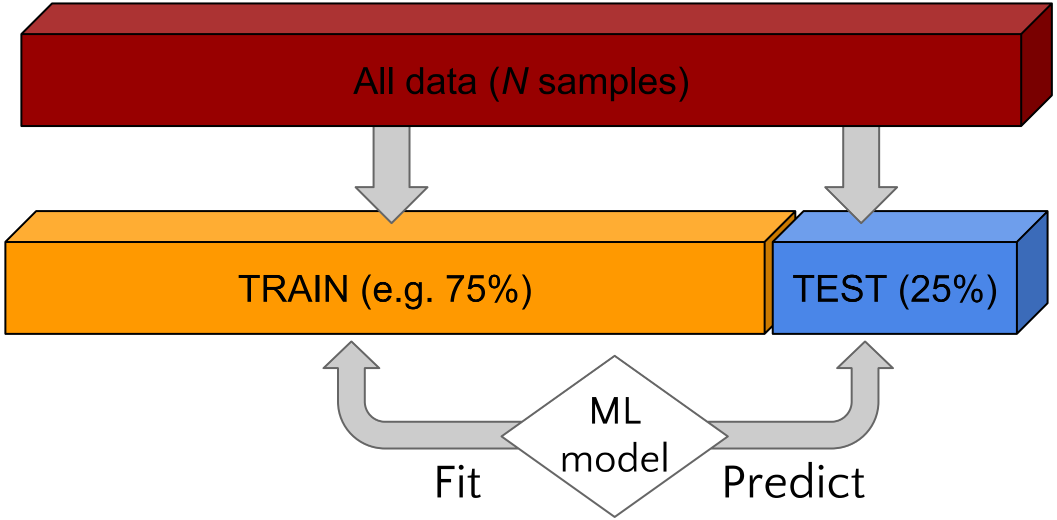
Week 1!

A typical decoding pipeline

0. Pattern extraction & preparation
1. **Partitioning train/test**
2. Feature selection/extraction
3. Model fitting (TRAIN)
4. Model generalization (TEST)
5. Statistical test of performance
6. Optional: plot weights

Step 1: Partitioning

- Hold-out CV

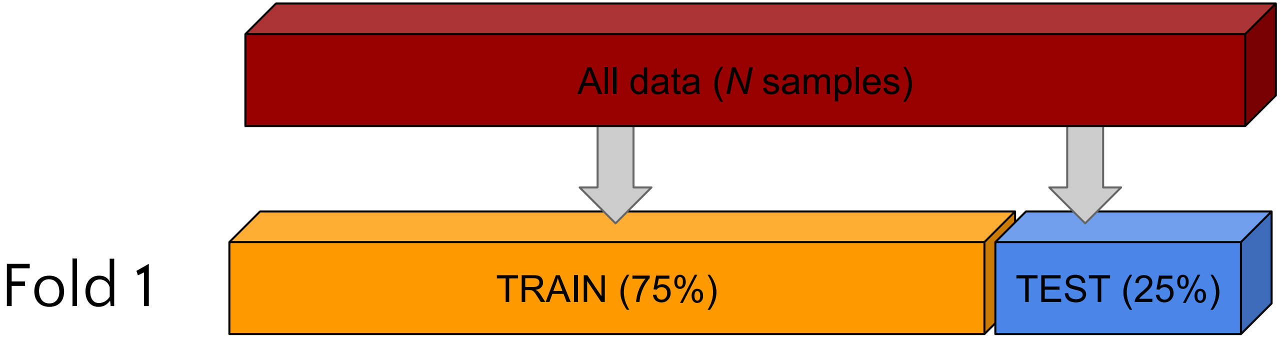


Only cross-validate once!

Kind of a "waste" of data … Reuse data?

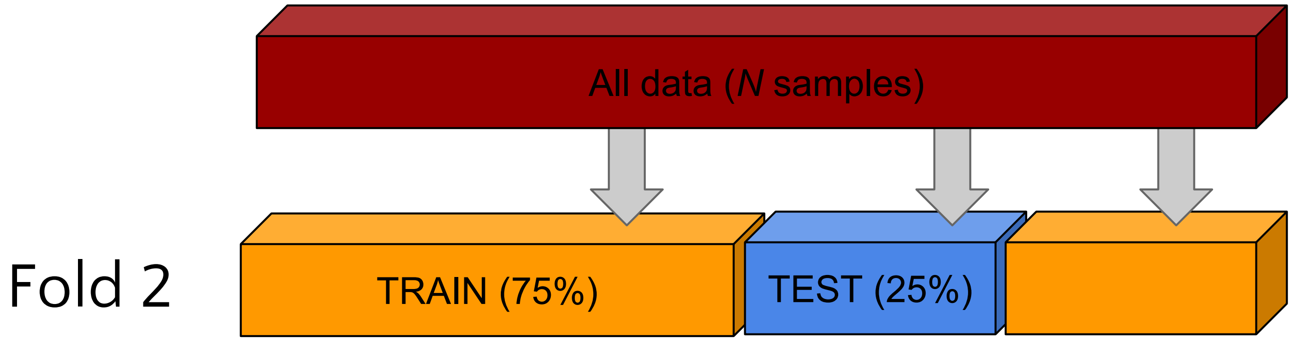
Step 1: Partitioning

- K-fold CV, e.g. 4-fold



Step 1: Partitioning

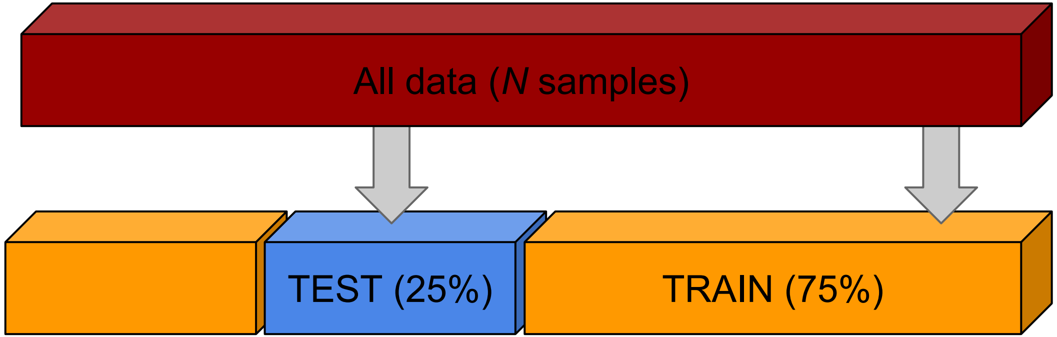
- K–fold CV, e.g. 4–fold



Step 1: Partitioning

- K-fold CV, e.g. 4-fold

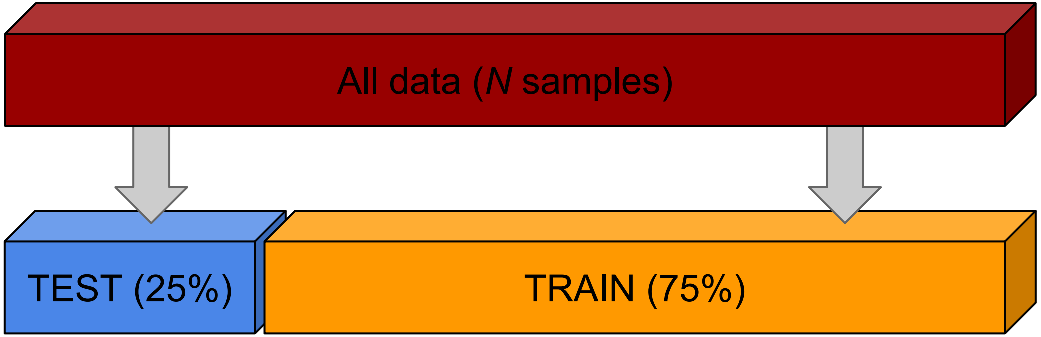
Fold 3



Step 1: Partitioning

- K-fold CV, e.g. 4-fold

Fold 4



A typical decoding pipeline

0. Pattern extraction & preparation
1. Partitioning train/test
2. **Feature selection/extraction**
3. Model fitting (TRAIN)
4. Model generalization (TEST)
5. Statistical test of performance
6. Optional: plot weights

Feature selection

- Goal: high sample/feature ratio!
- MRI: often **many features** (voxels), **few samples** (trials/instances/subjects)
- What to do???

Feature selection vs. extraction

- Reducing the dimensionality of patterns:
  - Select a subset of features (feature **selection**)
  - Transform features in lower-dimensional components (feature **extraction**)

Feature selection vs. extraction

- Ideas for feature **selection**?
  - ROI-based (e.g. only hippocampus);
  - (Independent!) functional mapper;
  - Data-driven selection: univariate feature selection

Feature selection vs. extraction

- Ideas for feature **selection**?
  - ROI-based (e.g. only hippocampus);
  - (Independent!) functional mapper;
  - **Data-driven selection: univariate feature selection**

Univariate feature selection

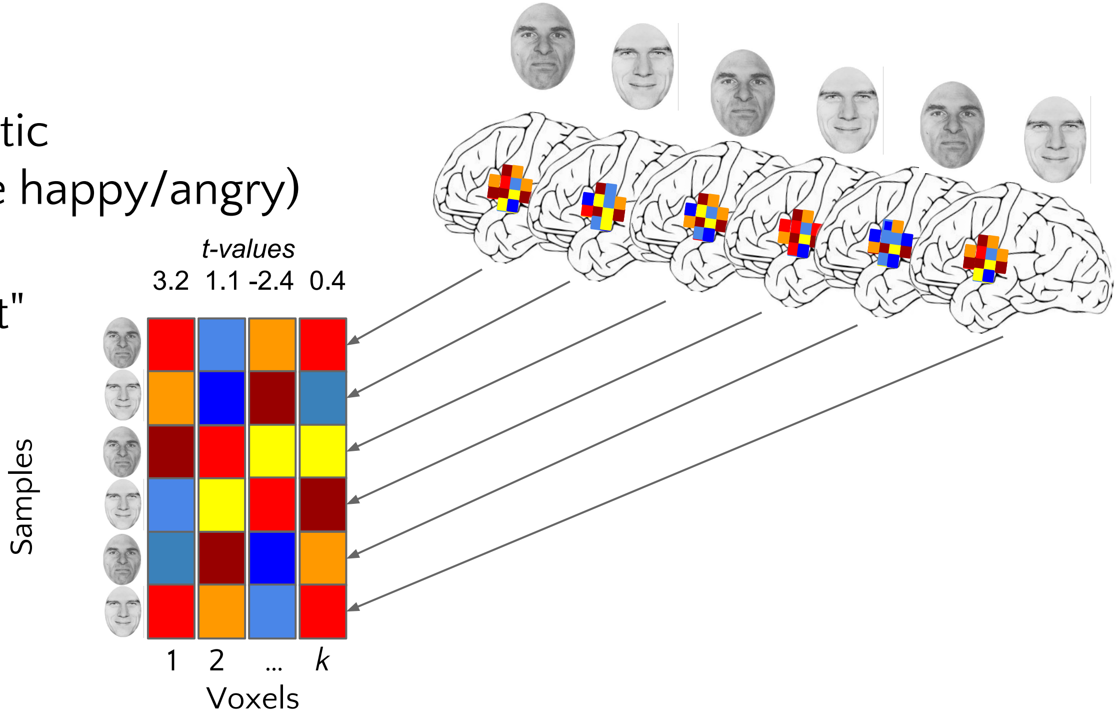
- Use univariate difference scores (e.g. t-value/F-value) to select features
- Only select a subset of the voxels with the highest scores

Univariate feature selection

Steps:

1. Calculate test-statistic
   (t-test for difference happy/angry)

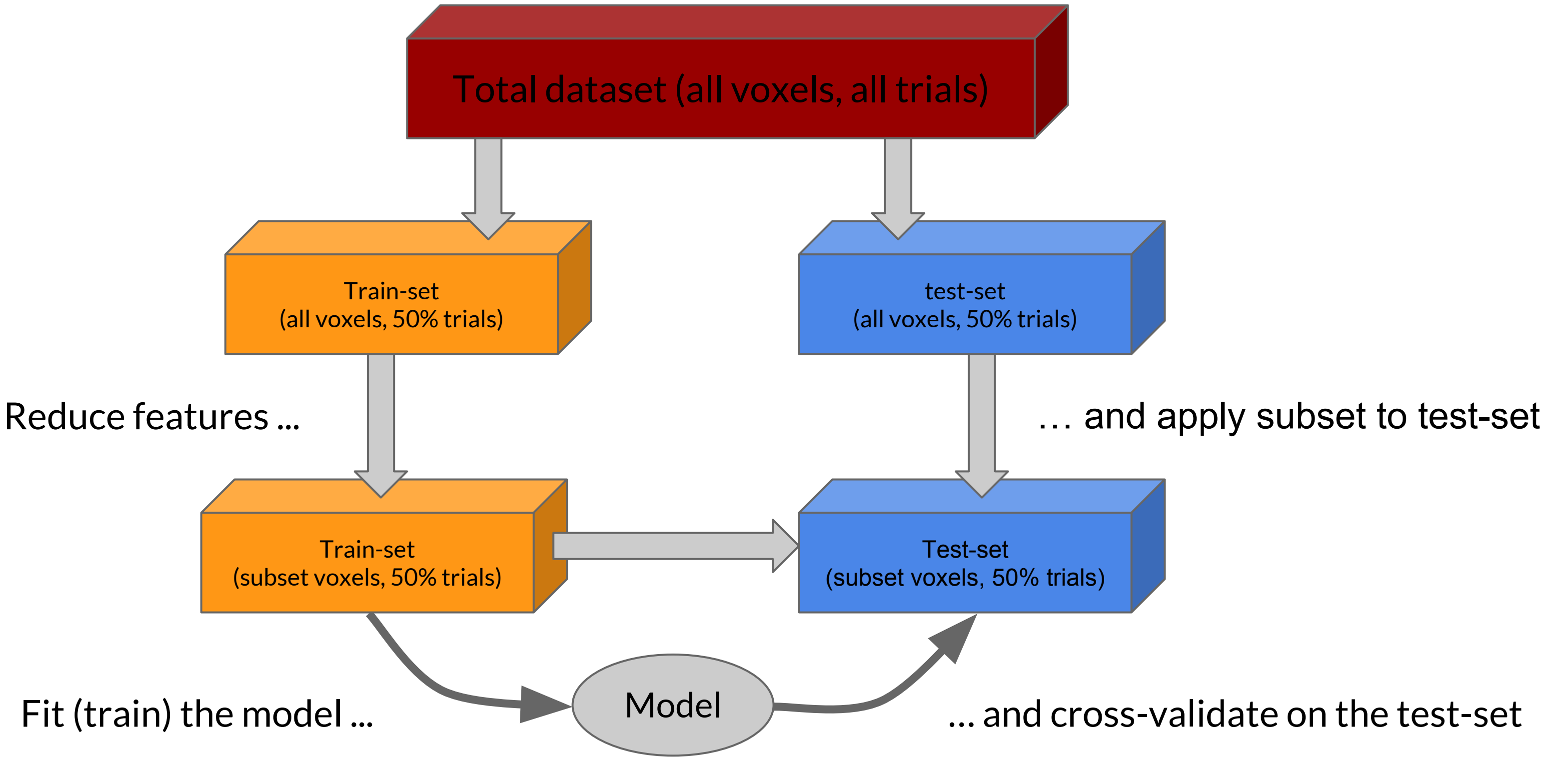
2. Select only the "best" 100 voxels (or a percentage)



Cross-validation in FS

- Importantly, data-driven feature selection needs to be **cross-validated**!
  - Performed on train-set **only**!

Cross-validation in FS

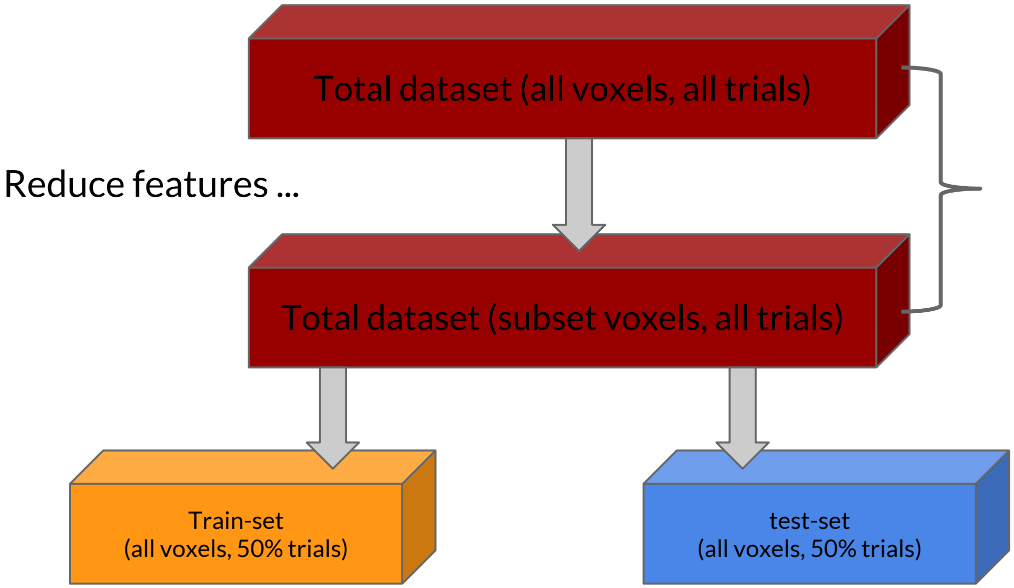


ToThink:

Why do you need to cross-validate your feature selection?

Isn't cross-validating the model-fit enough?

ToThink:
Why do you need to cross-validate your feature selection?

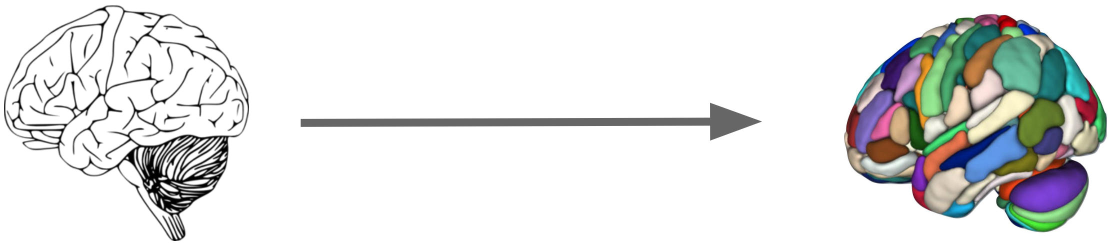


Here, you are using "information" from the test-set already!

Train & test-set are not independent anymore!

Feature selection vs. extraction

- Ideas for feature extraction?
  - PCA
  - Averaging within regions ("downsampling")



~60,000 features (voxels)

~110 features (brain regions)

Feature selection vs. extraction

- Ideas for feature **extraction**?
  - PCA
  - Averaging within regions ("downsampling")
- Feature extraction (often) does not need to be cross-validated

A typical decoding pipeline

0. Pattern extraction & preparation
1. Partitioning train/test
2. Feature selection/extraction
3. **Model fitting (TRAIN)**
4. **Model generalization (TEST)**
5. Statistical test of performance
6. Optional: plot weights

A typical decoding pipeline

0. Pattern extraction & preparation
1. Partitioning train/test
2. Feature selection/extraction
3. Model fitting (TRAIN)
4. Model generalization (TEST)
5. **Statistical test of performance**
6. Optional: plot weights

Statistical test of performance

- Remember I said ML does not use any statistical tests for **inference**?
  - I lied.

- Decoding analyses do use significance tests to infer whether decoding performance (R^2 or % accuracy) would **generalize to the population**

Statistical test of performance

- Remember, if you want to statistically test something, you need to have a null- and alternative hypothesis:
  - H_0: performance = chance level
  - H_a: performance > chance level

Statistical test of performance

- What is chance level?
- R^2?
  - Cross-validated R^2_{null}: 0
- Accuracy?
  - 1 / number of classes
  - Decode negative/positive/neutral? Chance = 33%

y = {  ,  ,  }

Statistical test of performance

- Observed performance is measured as the average cross-validated performance (R^2 / accuracy)
- Slightly different for within/between subject analyses:
  - Within: average performance **across subjects**
  - Between: average performance **across folds**

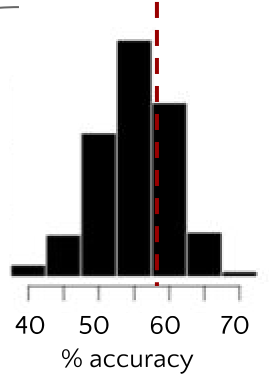
Statistics: within-subject

- Take e.g. a 3-fold cross-validation setup

Point-estimate of performance

Observed performance

58.5% = average



75% = average(90% 70% 65%) 1

53% = average(55% 45% 60%) 2

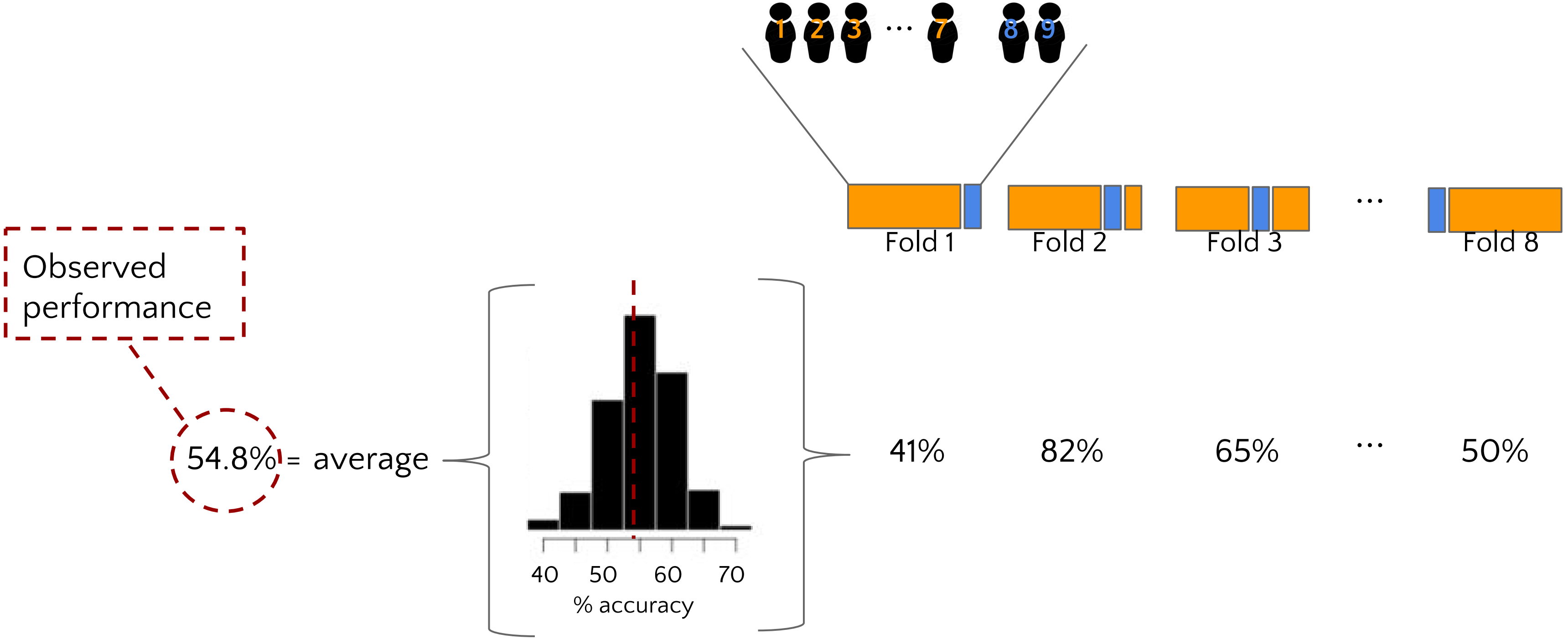
⋮

42% = average(90% 70% 65%) *N*

Statistics: within-subject

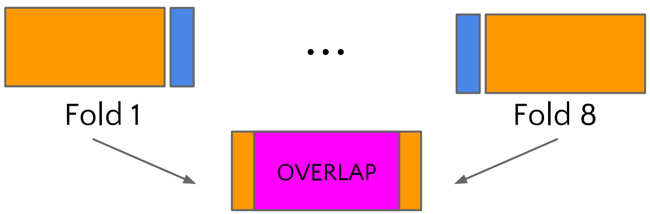
- H_a: 58.5 > 50
- Subject-wise average performance estimates represent the data points
- Assuming independence between subjects, we can use simple parametric statistics
  - t-test(n-subjects – 1) of observed performance (i.e. 58.5%) against the null performance (i.e. 50%)

Statistics: between-subject

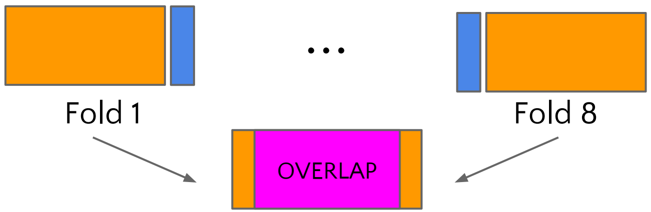


Statistics: between-subject

- H_a: 54.8 > 50
- Fold-wise performance estimates represent data points
- Problem: different folds contain the same subjects



Statistics: between-subject

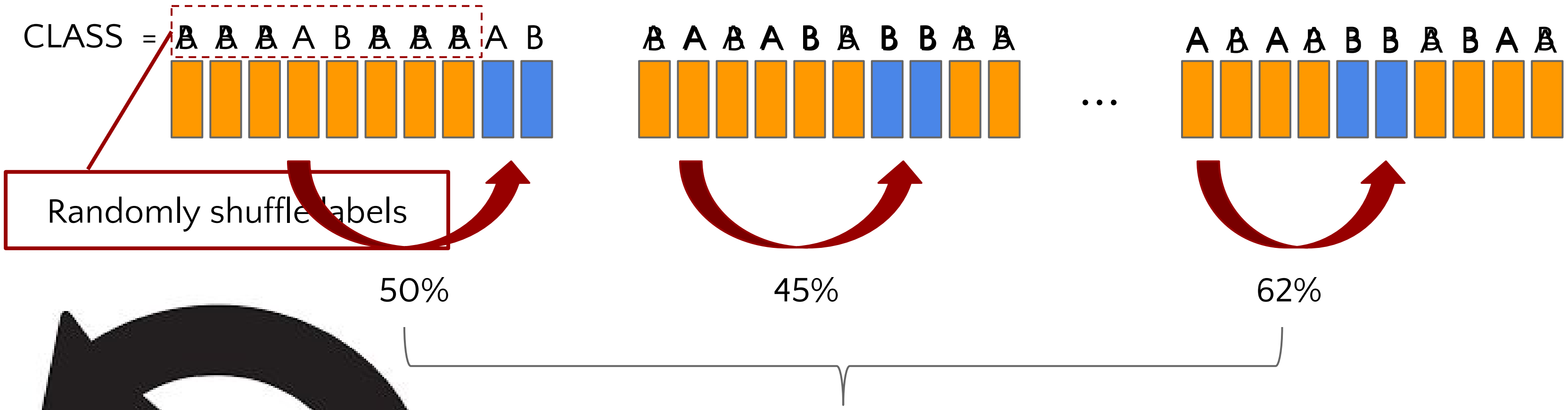


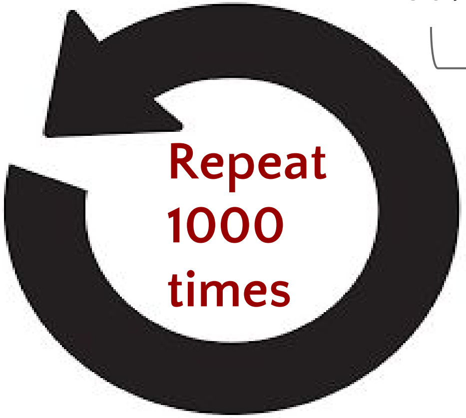
- Problem: different folds contain the same subjects
- Consequence: **dependence** between data points
  - Violates assumptions of many parametric statistical tests
- Solution: non-parametric (**permutation**) test

Statistics: between-subject

- Permutation tests do not assume (the shape of) a null-distribution, but "simulate" them
- To simulate the null-distribution (results expected when H_0 is true), **permutation tests literally simulate "performance at chance"**

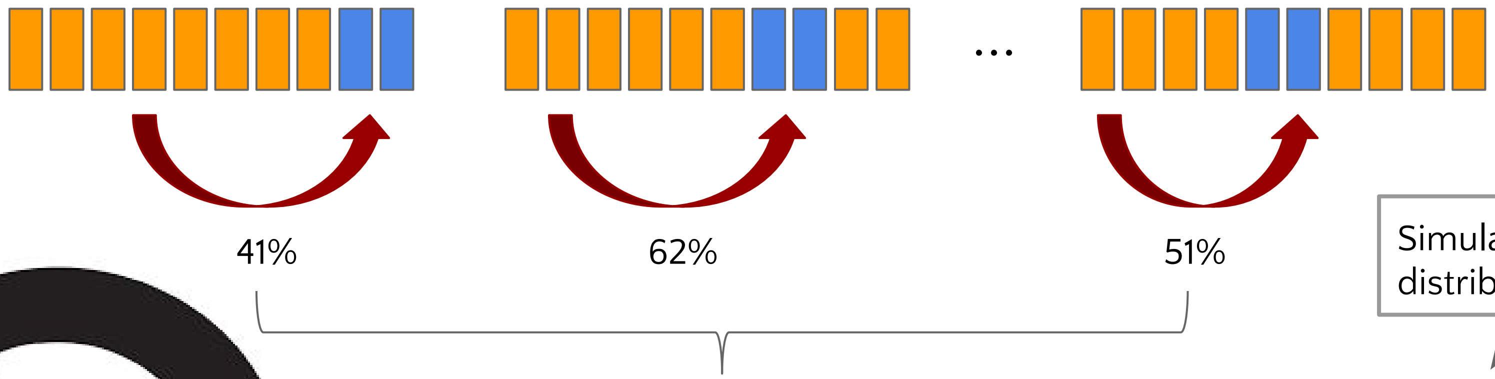
Statistics: between-subject

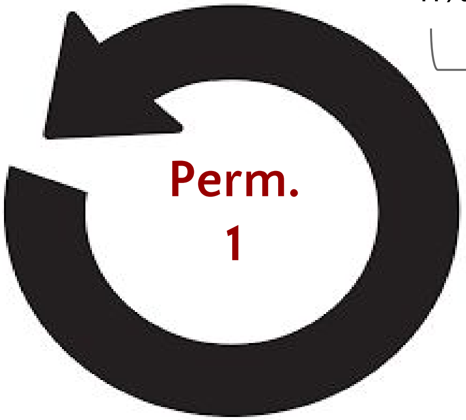




Permuted performance (averaged over folds):
52%

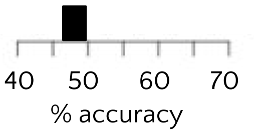
Statistics: between-subject



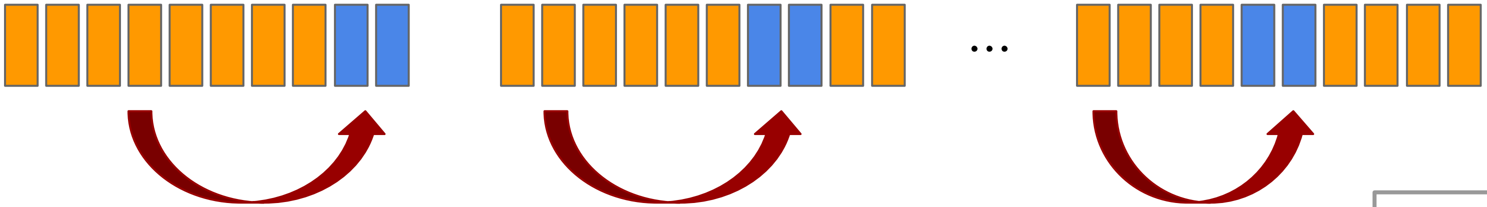


Permuted performance (averaged over folds):
48%

Simulated null-distribution!



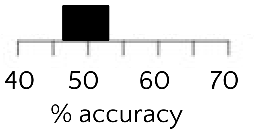
Statistics: between-subject



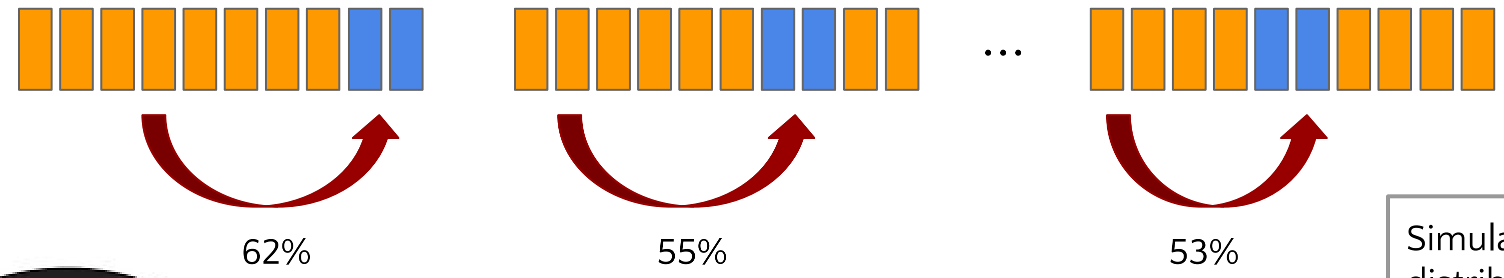
51% 53% 49%

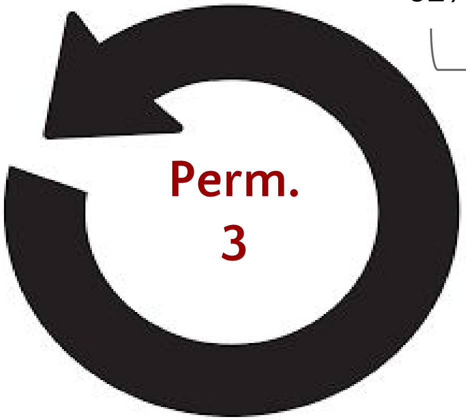
Simulated null-distribution!

Permuted performance (averaged over folds):
52%



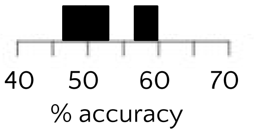
Statistics: between-subject



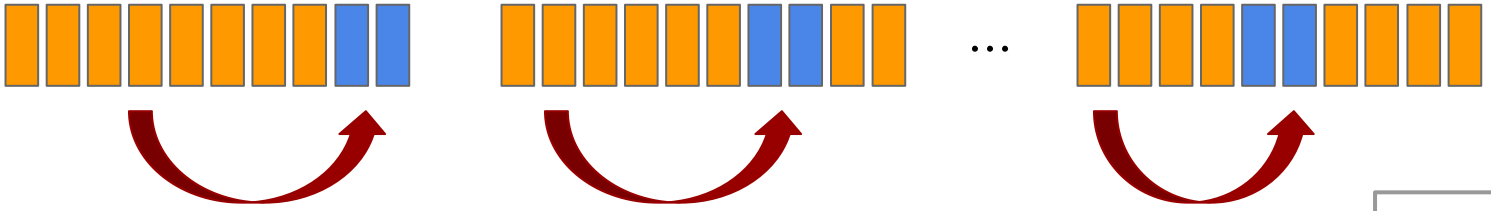


Permuted performance (averaged over folds):
57%

Simulated null-distribution!



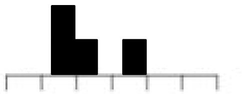
Statistics: between-subject



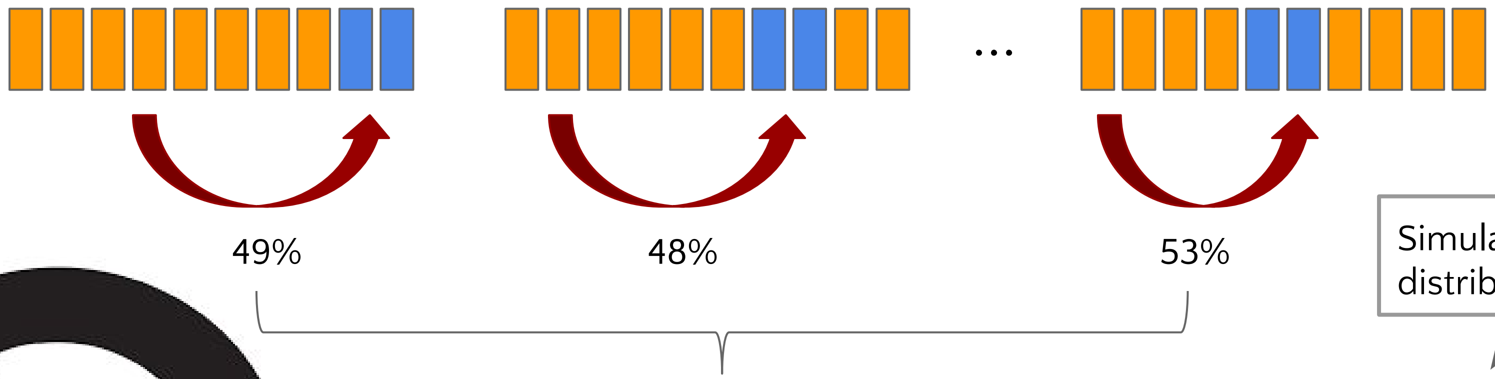
51% 42% 47%

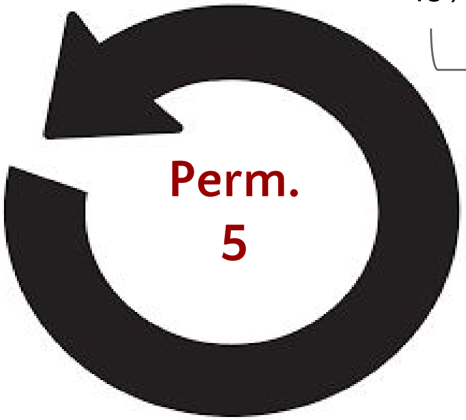
Permuted performance (averaged over folds):
48%

Simulated null-distribution!



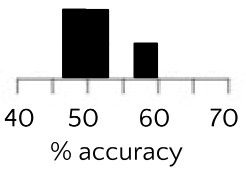
Statistics: between-subject



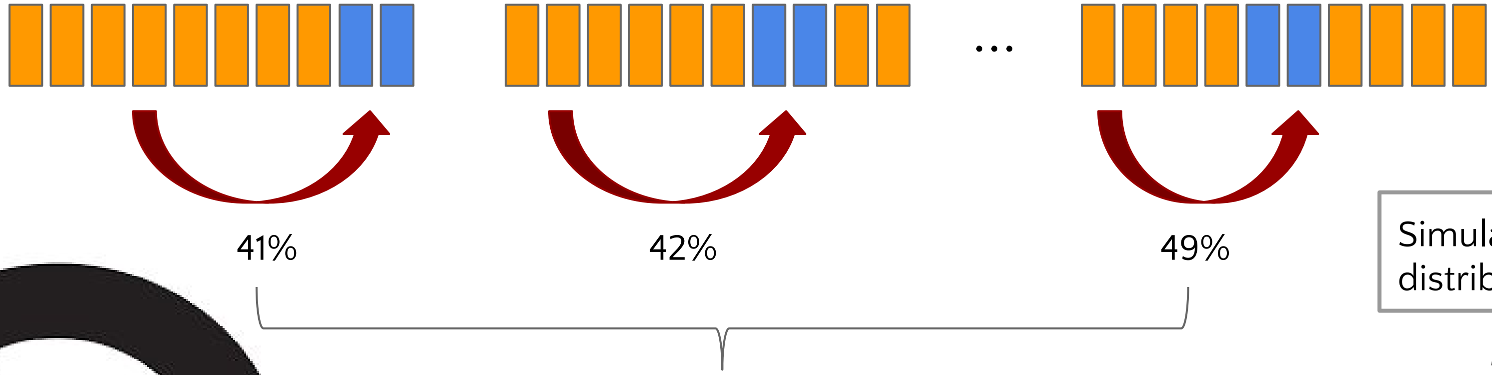


Permuted performance (averaged over folds):
50%

Simulated null-distribution!

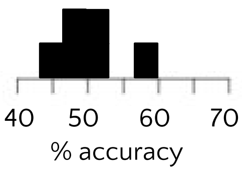


Statistics: between-subject

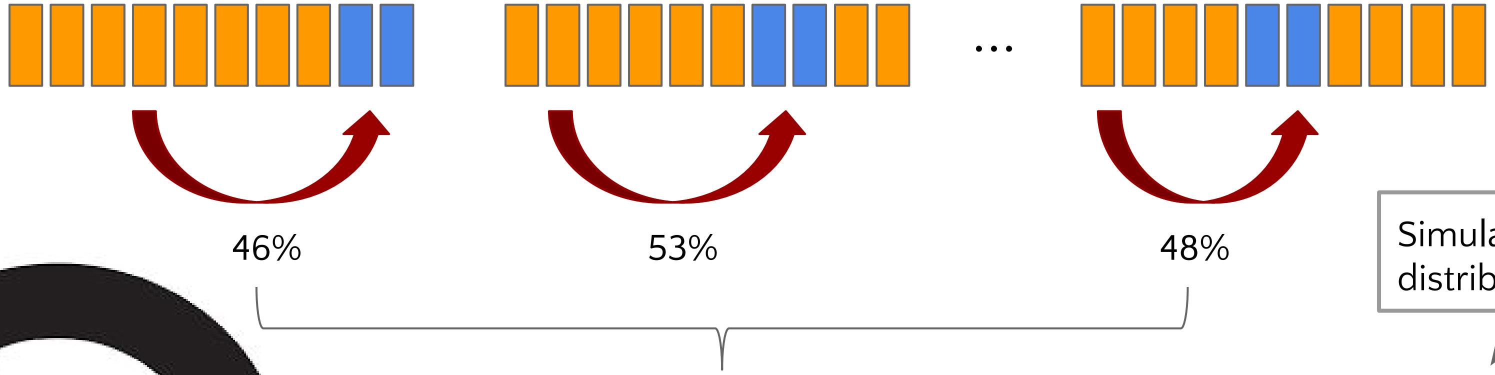


Permuted performance (averaged over folds):
45%

Simulated null-distribution!

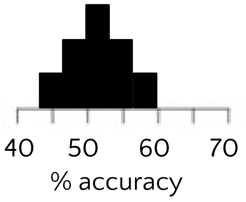


Statistics: between-subject

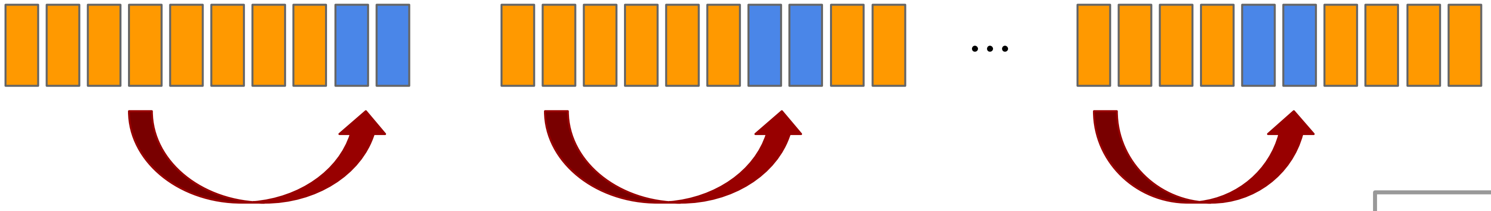


Permuted performance (averaged over folds):
48%

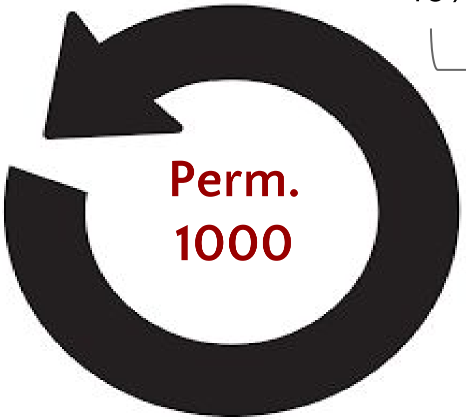
Simulated null-distribution!



Statistics: between-subject

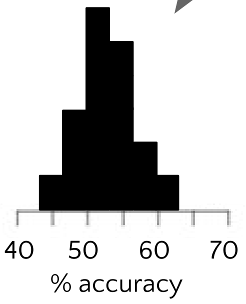


40% 39% 52%



Permuted performance (averaged over folds):
43%

Simulated null-distribution!

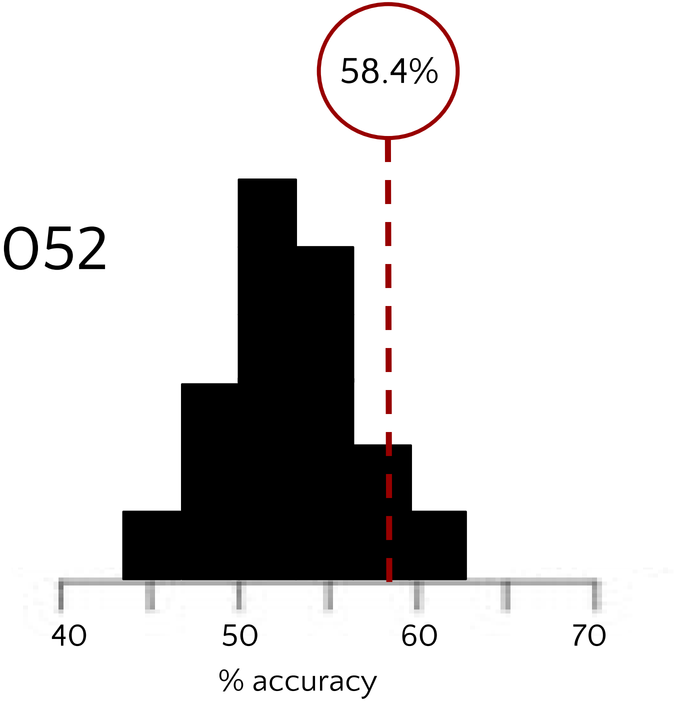


Statistics: between-subject

"Non-parametric" p-value =

$$\frac{\sum(\text{null-scores} > \text{observed-scores})}{\text{Number of permutations}} = 0.052$$

$$\frac{52}{1000}$$



A typical decoding pipeline

0. Pattern extraction & preparation
1. Partitioning train/test
2. Feature selection/extraction
3. Model fitting (TRAIN)
4. Model generalization (TEST)
5. Statistical test of performance
6. Optional: plot weights

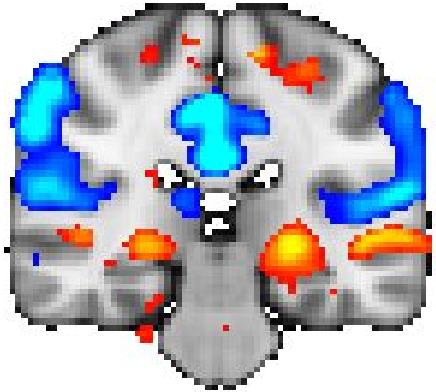
If there is time left!

Weight mapping

- Often, researchers (read: reviewers) ask:

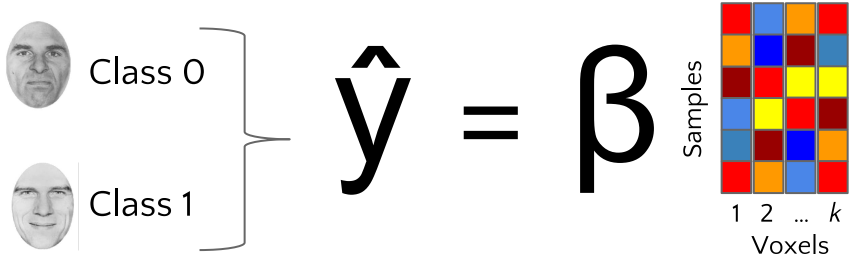
  "Which features are important
  for which class?"

- What they want:
- Intrinsic need for blobs?



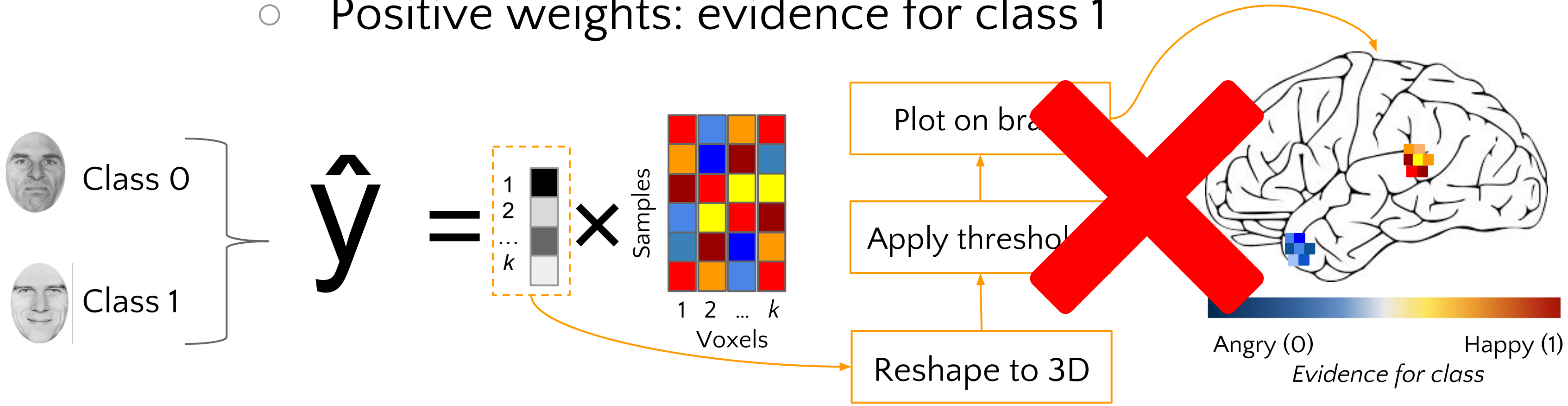
Weight mapping

- Technically, weights (β) in *linear* models are interpretable: higher = more important
  - Negative weights: evidence for class 0
  - Positive weights: evidence for class 1



Weight mapping

- Technically, weights (β) in *linear* models are interpretable: higher = more important
  - Negative weights: evidence for class 0
  - Positive weights: evidence for class 1



Weight mapping

- Haufe et al. (2014, *NeuroImage*) showed that **high weights ≠ class importance**

… *A widespread misconception about multivariate classifier weight vectors is that (the brain regions corresponding to) measurement channels* with large weights are strongly related to the experimental condition.* In fact, such conclusions can be unjustified.

*voxels

Weight mapping

- Haufe et al. (2014, *NeuroImage*) showed that **high weights ≠ class importance**
- Features (voxels) may function like (class-independent) "**filters**"
  - E.g. reflect physiological noise
- Inherent problem for decoding analyses (**brain** → **world**)

Weight mapping

- Solution: mathematical trick to convert a **decoding model** to an **encoding model**
  - Weights from encoding models *are* interpretable
- Activation patterns = $(X'X)\beta$
- But: very similar to traditional activation-based analysis ...
- Conclusion: (like always) choose the analysis **best suited for your question**!

Summary

0. Pattern extraction & preparation
1. Partitioning train/test
2. Feature selection/extraction
3. Model fitting (TRAIN)
4. Model prediction (TEST)
5. Statistical test of performance
6. Optional: plot weights

Estimate and extract patterns such that X = N-samples by N-features (voxels)

Summary

0. Pattern extraction & preparation
1. **Partitioning train/test**
2. Feature selection/extraction
3. Model fitting (TRAIN)
4. Model prediction (TEST)
5. Statistical test of performance
6. Optional: plot weights

Use hold-out or K-fold partitioning

Summary

0. Pattern extraction & preparation
1. Partitioning train/test
2. **Feature selection/extraction**
3. Model fitting (TRAIN)
4. Model prediction (TEST)
5. Statistical test of performance
6. Optional: plot weights

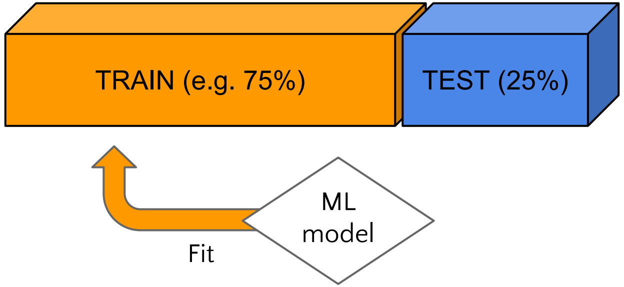
Reduce the amount of features

Feature selection
(voxel subset

Feature extraction
(voxels → components)

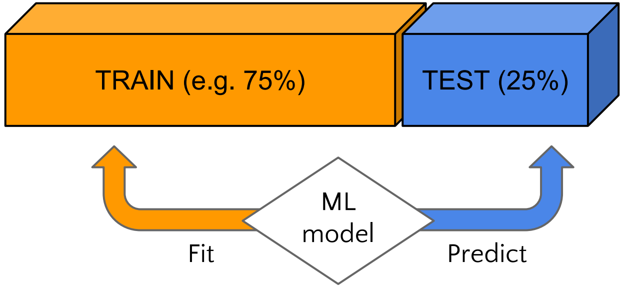
Summary

0. Pattern extraction & preparation
1. Partitioning train/test
2. Feature selection/extraction
3. **Model fitting (TRAIN)**
4. Model prediction (TEST)
5. Statistical test of performance
6. Optional: plot weights



Summary

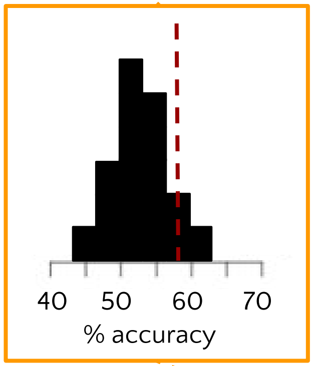
0. Pattern extraction & preparation
1. Partitioning train/test
2. Feature selection/extraction
3. **Model fitting (TRAIN)**
4. **Model prediction (TEST)**
5. Statistical test of performance
6. Optional: plot weights



Summary

0. Pattern extraction & preparation
1. Partitioning train/test
2. Feature selection/extraction
3. Model fitting (TRAIN)
4. Model prediction (TEST)
5. **Statistical test of performance**
6. Optional: plot weights

Parametric test against chance (within-subject)



Permutation test against simulated null (between- subject)

Summary

0. Pattern extraction & preparation
1. Partitioning train/test
2. Feature selection/extraction
3. Model fitting (TRAIN)
4. Model prediction (TEST)
5. Statistical test of performance
6. **Optional: plot weights**

Do not plot weights! A (complementary) univariate analysis would suffice

If you insist, plot the corresponding encoding model – $(X'X)\beta$

Literature

- Abraham et al: about the scikit-learn package for decoding analyses (read before tomorrow if possible!)
- Pereira et al: tutorial-style paper about decoding analyses