University of Amsterdam

SPINOZA

Spinoza Centre for Neuroimaging

Multivariate pattern analysis

Spinoza fMRI Course

lukas-snoek.com

@LukasSnoek

About me

- **PhD-student** at the University of Amsterdam, lab of Steven Scholte
- Interested in **emotion** (**perception**), face perception, and (fMRI) **methodology**/ computational modelling
- **Teaching** two courses on fMRI analysis

About this presentation

- Teach you the **basics** of "multivariate pattern analysis", MVPA (the how, what, and why)
  - Enough to get you started
- Mostly about "**decoding**" (machine learning)
- For **beginners**!

About this presentation

- Meant to be **interactive** – ask questions!
- **Conceptual**, rather than mathematical
- Focuses on f MRI (but applicable to EEG/MEG, structural MRI, etc.)
- **Not hands-on**
  - But I'll be here tomorrow to help with analyses!

About this presentation

- Introduction
- Why?
  - Why look at patterns instead of single voxels?
- What?
  - Decoding and related concepts
- How?
  - How to set up a complete decoding pipeline

About this presentation

- Introduction
- Why?
- What?
- How?

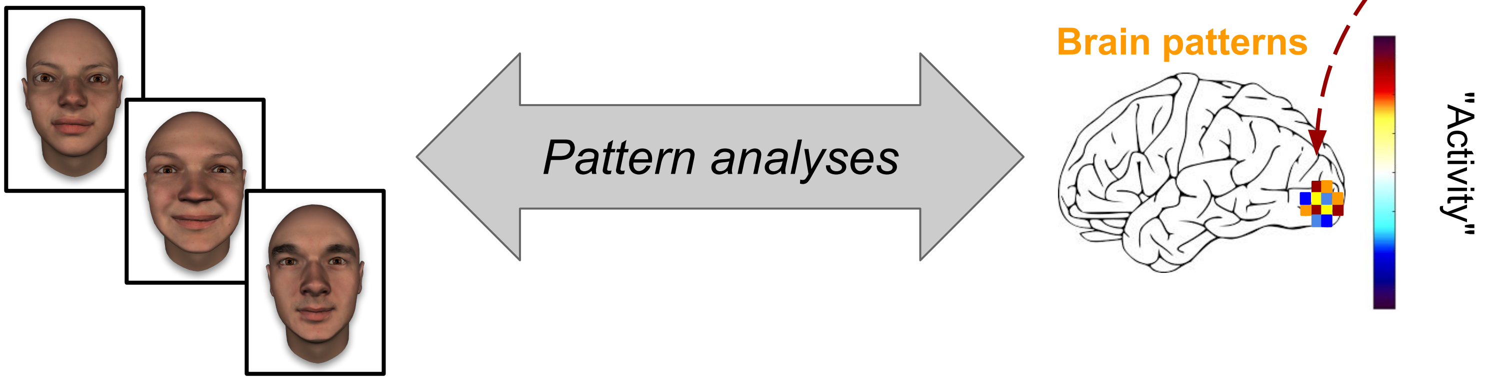
About this presentation

- Introduction
- Why?
- What?
- How?

“Multivariate” (sometimes “multivoxel”) refers to the brain patterns (>1 voxel)

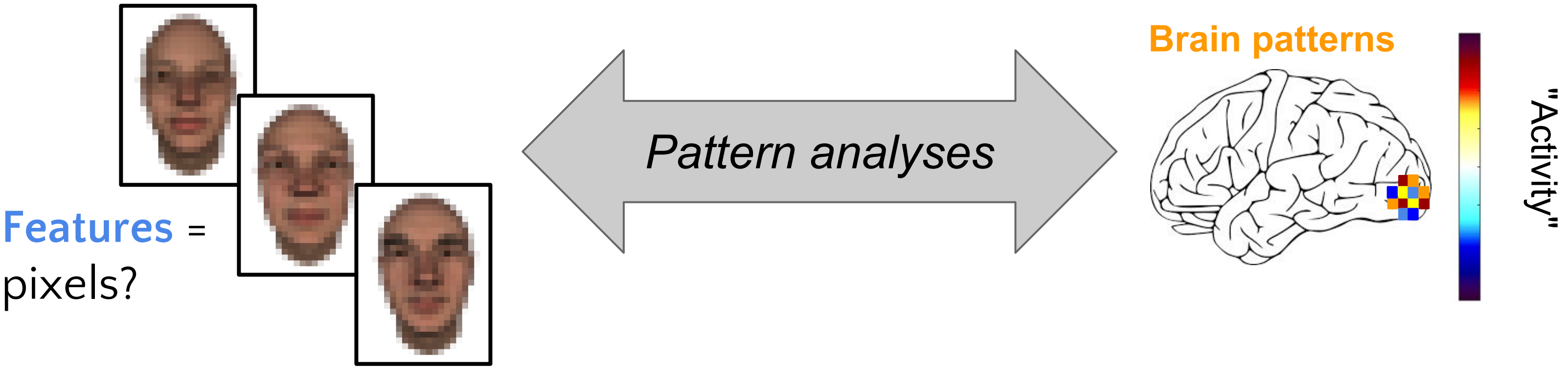
What is MVPA?

- Multivariate pattern analysis (MVPA) relates **patterns** of neuroimaging data (voxels/sensors) to (stimulus/task/subject) ‘**features**’



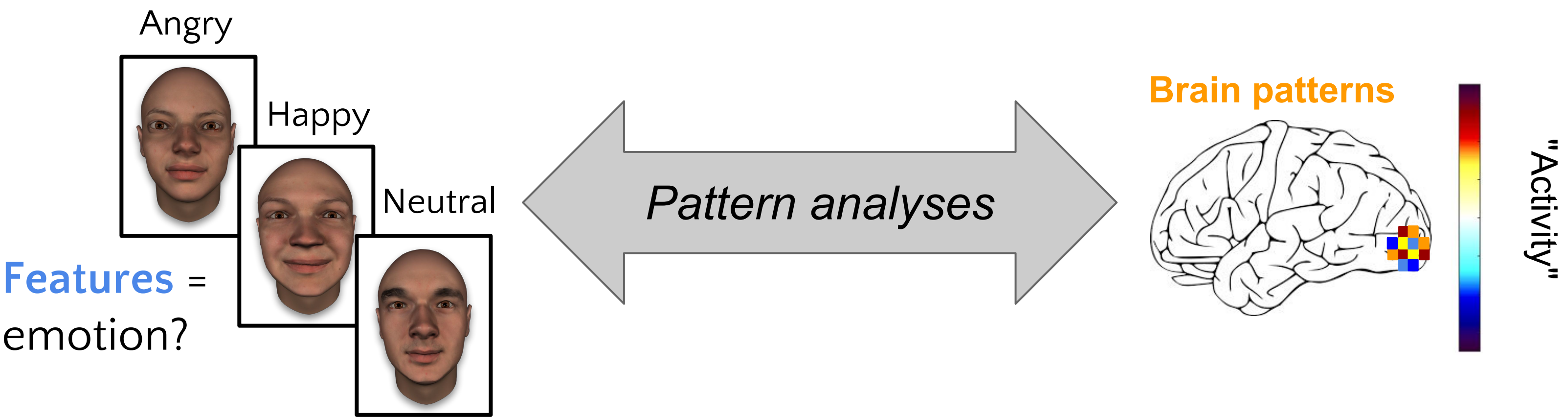
What is MVPA?

- Multivariate pattern analysis (MVPA) relates **patterns** of neuroimaging data (voxels/sensors) to (stimulus/task/subject) '**features**'



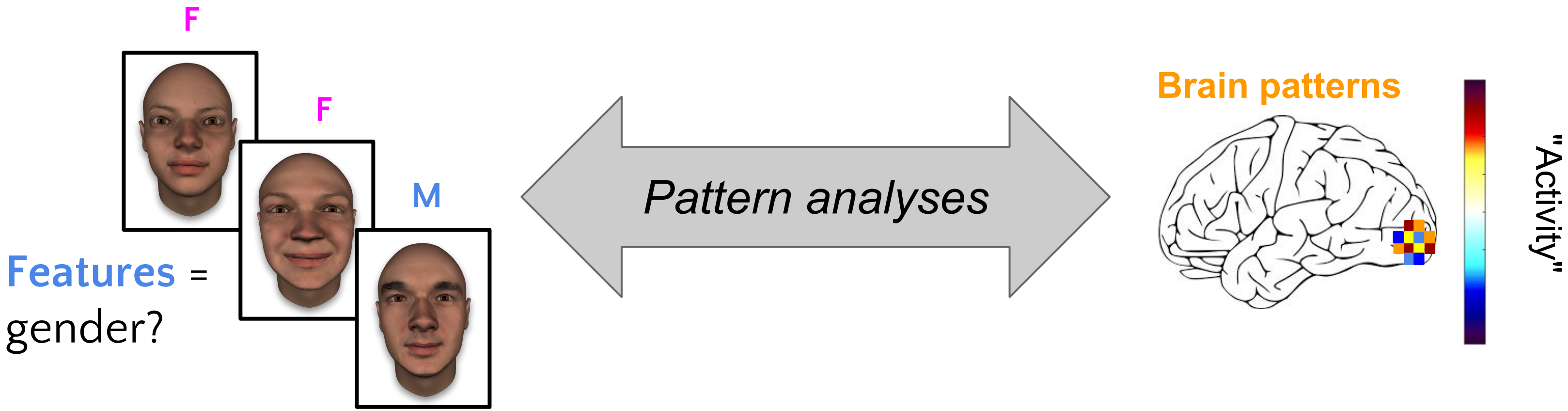
What is MVPA?

- Multivariate pattern analysis (MVPA) relates **patterns** of neuroimaging data (voxels/sensors) to (stimulus/task/subject) '**features**'



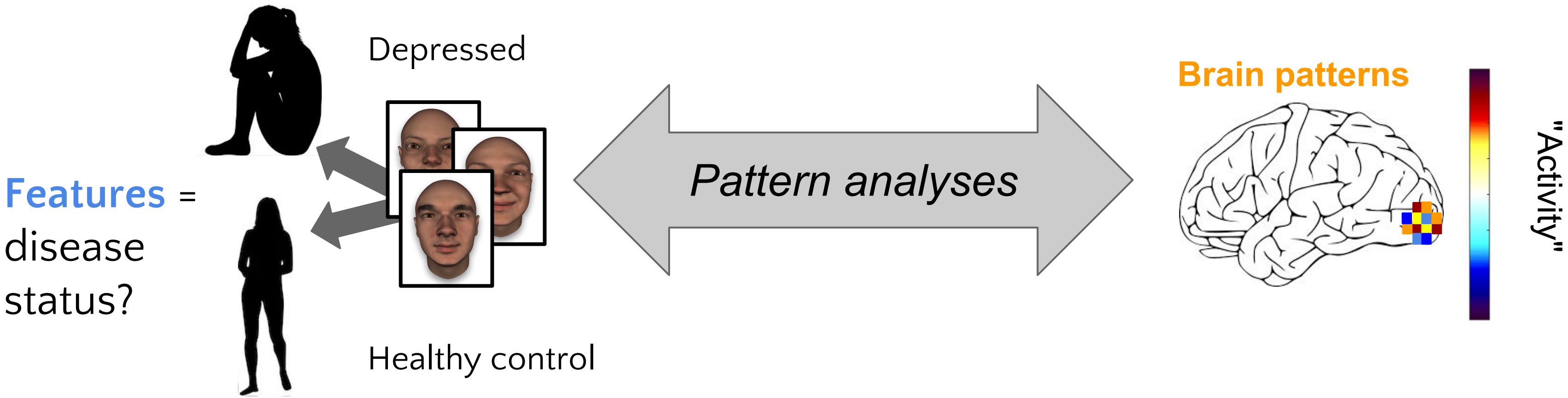
What is MVPA?

- Multivariate pattern analysis (MVPA) relates patterns of neuroimaging data (voxels/sensors) to (stimulus/task/subject) 'features'



What is MVPA?

- Multivariate pattern analysis (MVPA) relates **patterns** of neuroimaging data (voxels/sensors) to (stimulus/task/subject) '**features**'

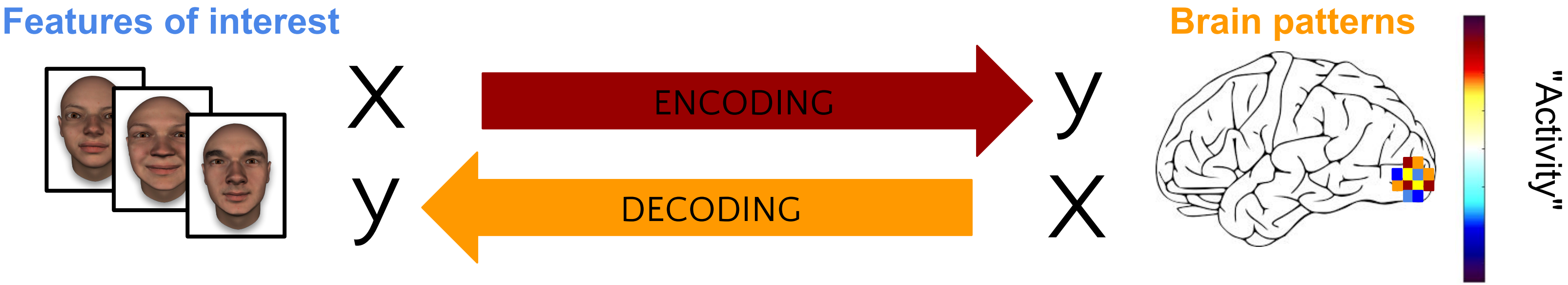


What is MVPA?

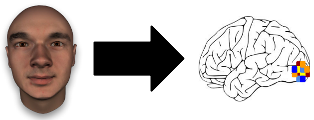
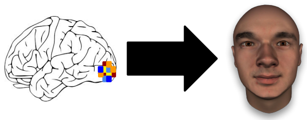
- There are **many different types** of MVPA!
  - "Decoding" (machine-learning)
  - Representational similarity analysis (RSA)
  - Pattern component modeling
  - Cross-validated MANOVA
- **Common ground**: they operate on patterns of voxels

MVPA vs. univariate analysis

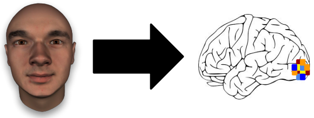
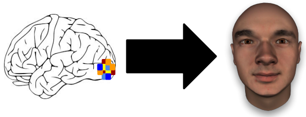
- "Decoding" models (topic of today) are a specific type of MVPA
- Relative to "encoding" models, decoding models differ in the "direction of analysis"



Analysis landscape

	Encoding: 	Decoding: 
Univariate:	**S**tatistical **P**arametric **M**apping (mass-univariate) pRF models	???
Multivariate:	MANOVA Pattern component modelling	Machine learning (“decoding”) Inverted encoding models

Analysis landscape

	Encoding: 	Decoding: 
Univariate:	**S**tatistical **P**arametric **M**apping (mass-univariate) pRF models	???
Multivariate:	MANOVA Pattern component modelling	Machine learning ("decoding") Inverted encoding models

Test your understanding!

Tomas measures the gray-matter density of a 100 subjects.

He then wants to investigate whether the gray-matter density in the hypothalamus is predictive of whether someone is male or female.

Encoding?

?

Decoding?

Test your understanding!

Steven shows TV-commercials which are either boring, funny, or neutral.

He then wants to investigate which brain regions respond more to funny than to boring commercials.

Encoding?

?

Decoding?

Test your understanding!

Noor show subjects images of different complexity ('visual clutter').

She then wants to analyze whether these complexity parameters can explain the voxel patterns in early visual cortex.

Encoding?

?

Decoding?

Contents

- Introduction
- Why?
- What?
- How?

Why pattern analyses?

- Why look at **patterns** instead of **single voxels**?
- Three reasons:
  - One practical
  - One theoretical
  - One instrumental

Practical reason

- "**It's more sensitive**" than univariate analyses
- Effect sizes are often larger in MVPA
- (If you ask me: that's comparing apples and oranges)
- But there are also theoretical reasons for analyzing patterns

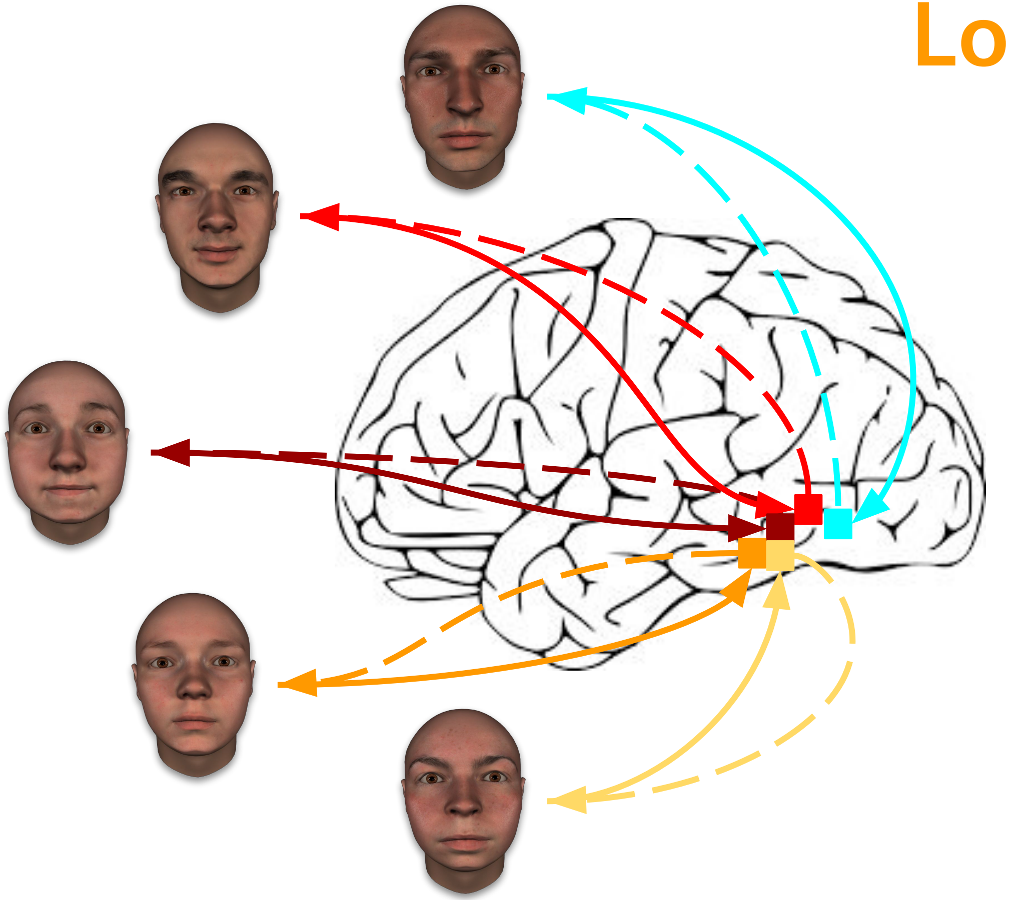
Neural correlations, population coding and computation

Bruno B. Averbeck*, Peter E. Latham† and Alexandre Pouget*

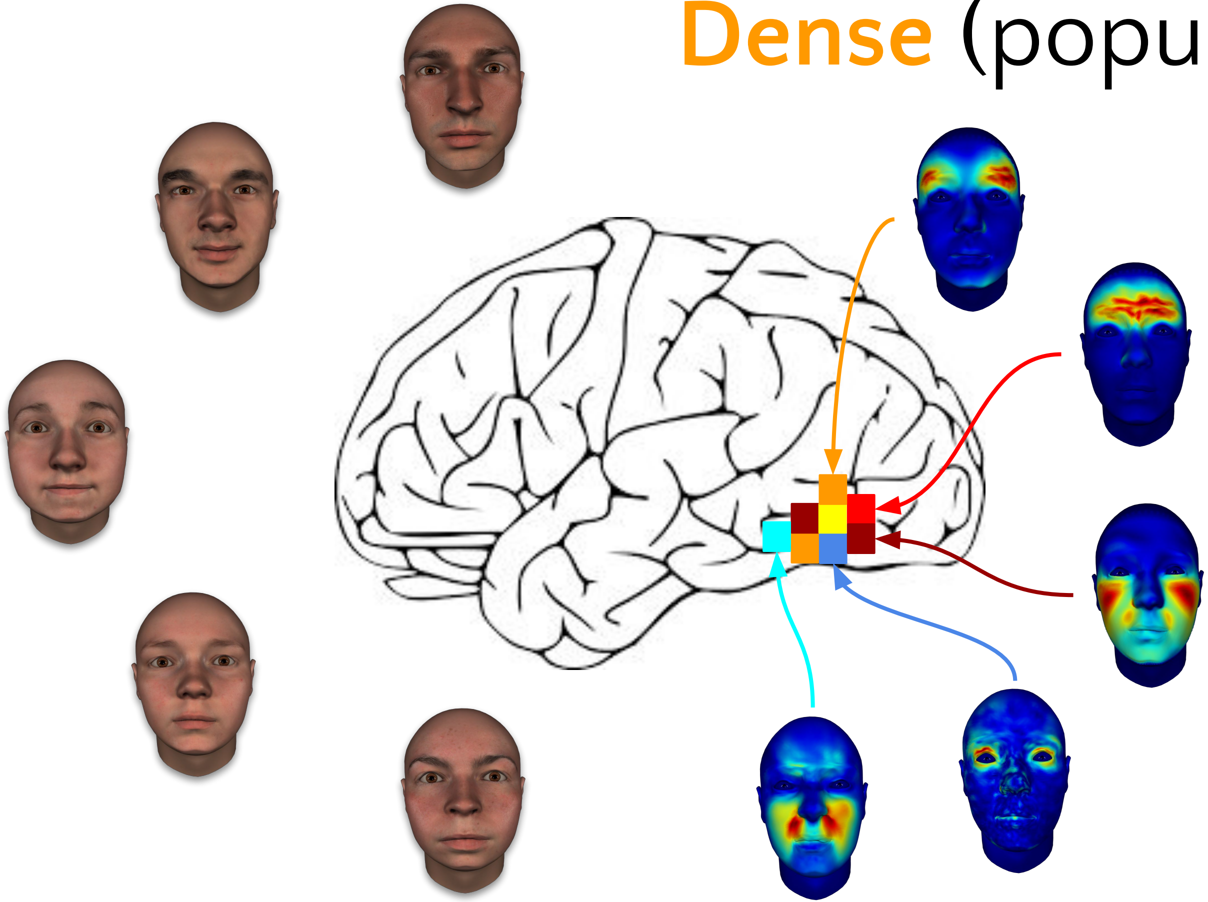
Theoretical reasons

- MVPA fits well with the notion of **distributed/dense (population) coding**
- Information is not encoded in **single or few neurons** (or even voxels), but in distributed **populations of neurons** (voxels)

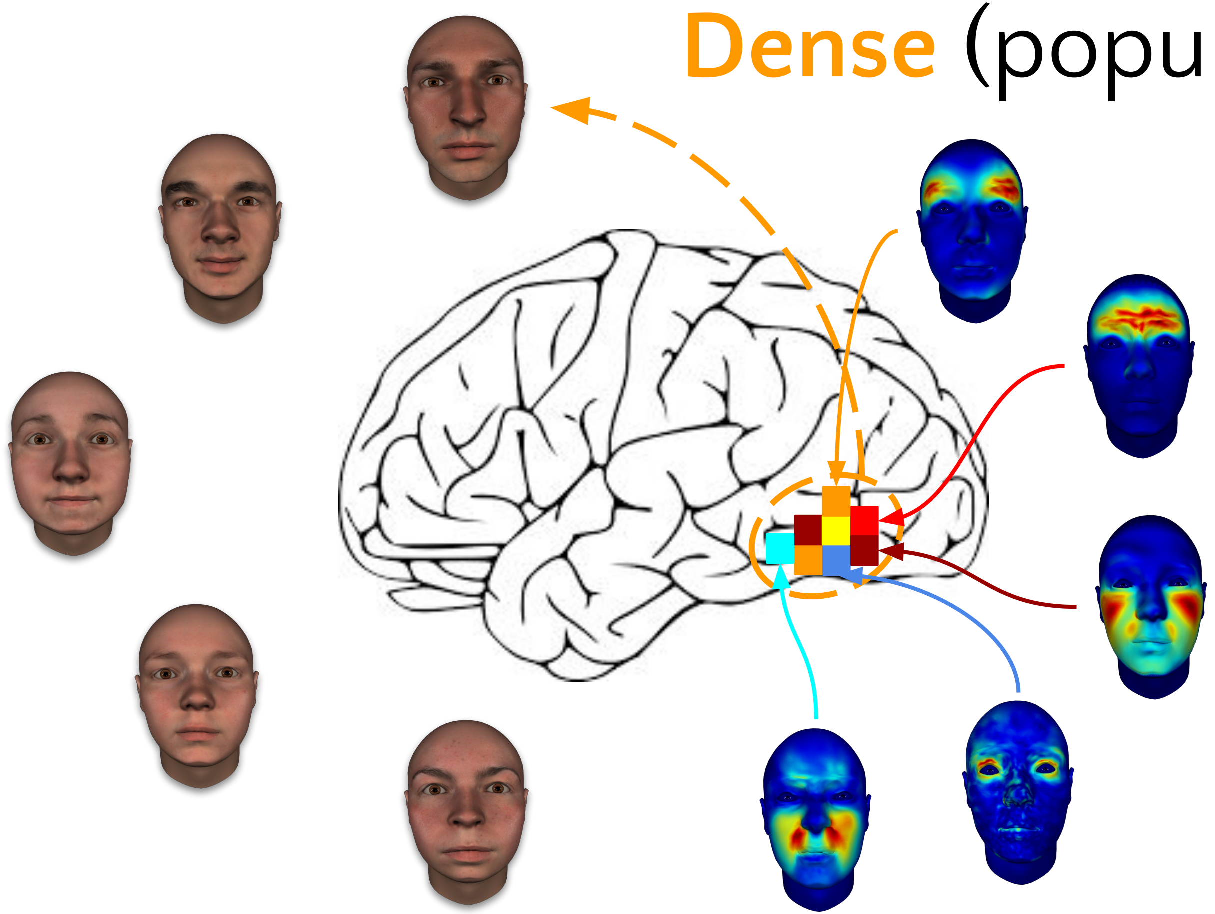
Local coding



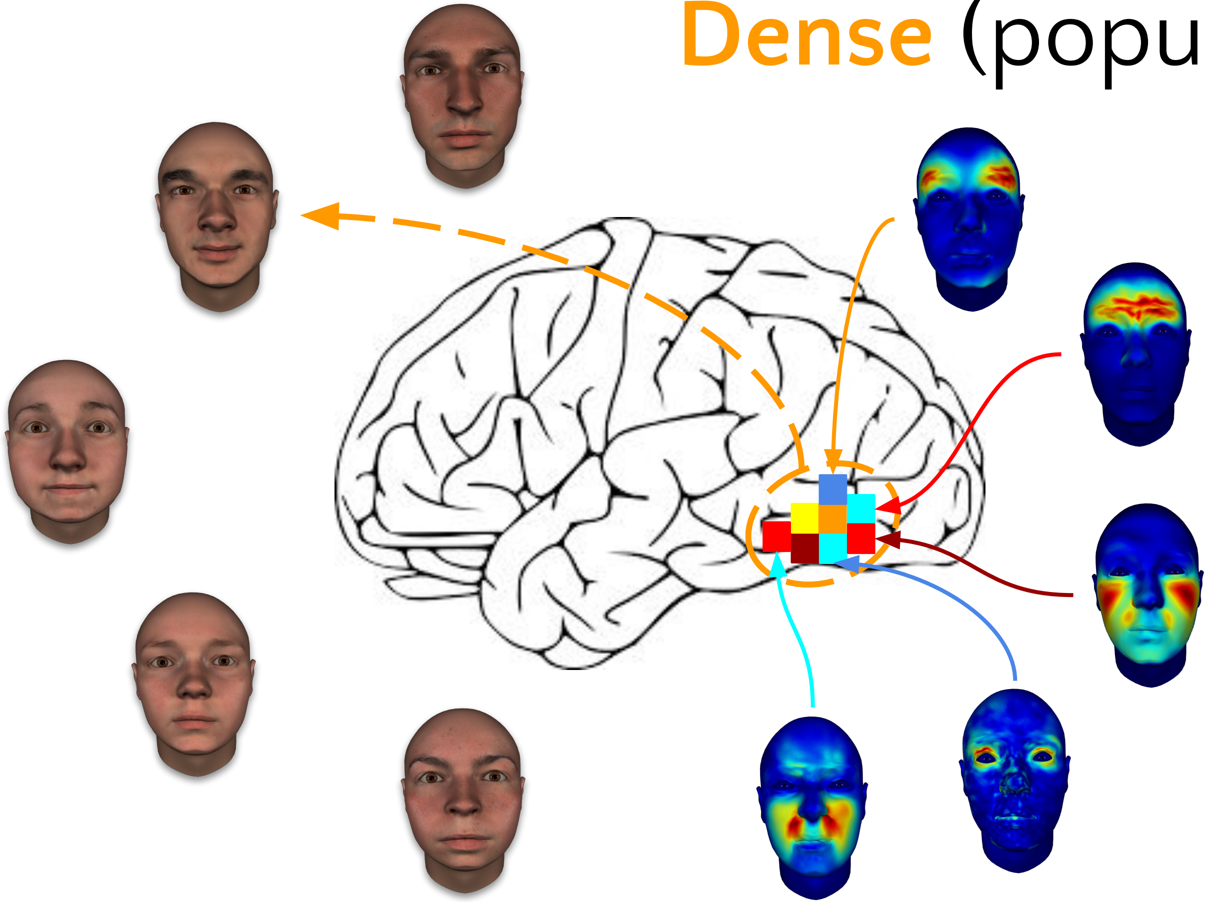
Dense (population) coding



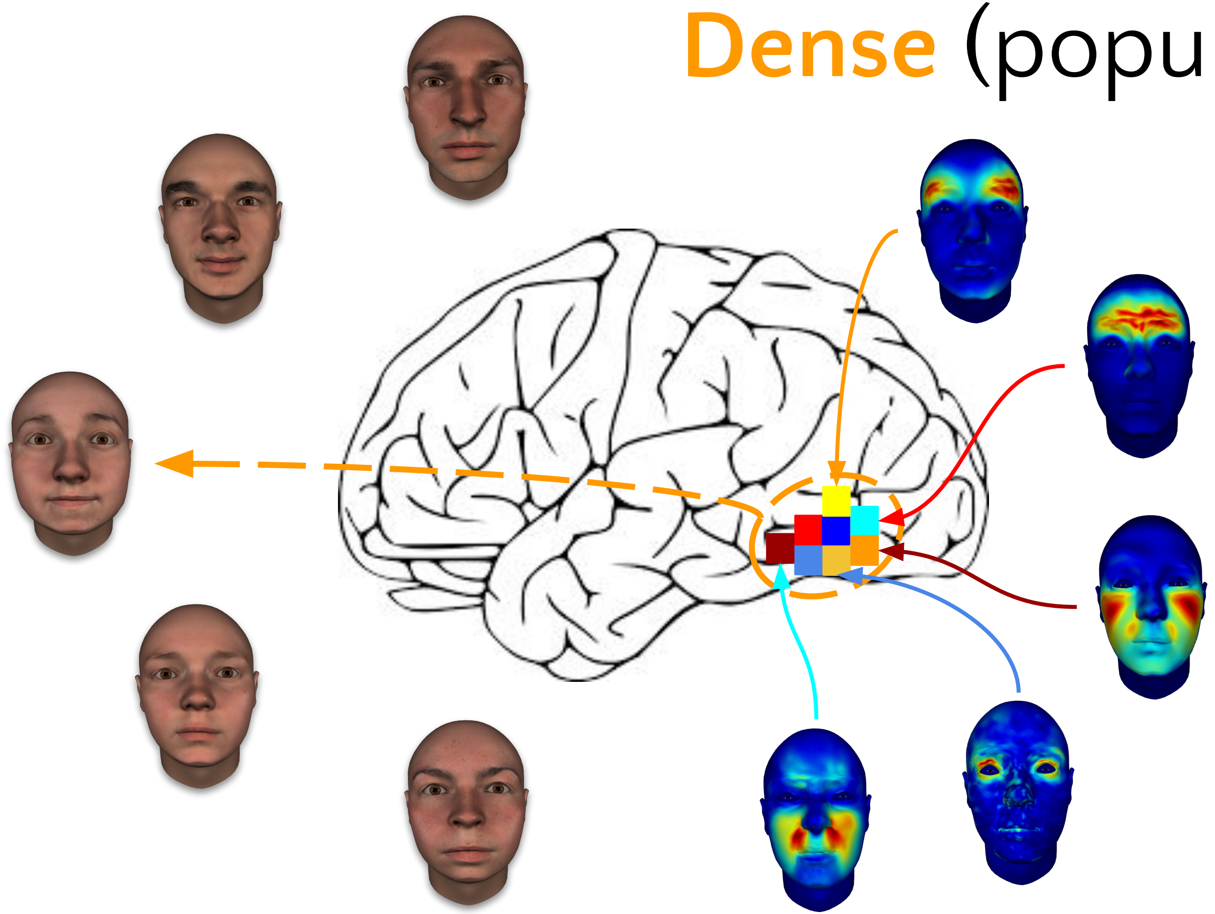
Dense (population) coding



Dense (population) coding



Dense (population) coding



Instrumental reasons

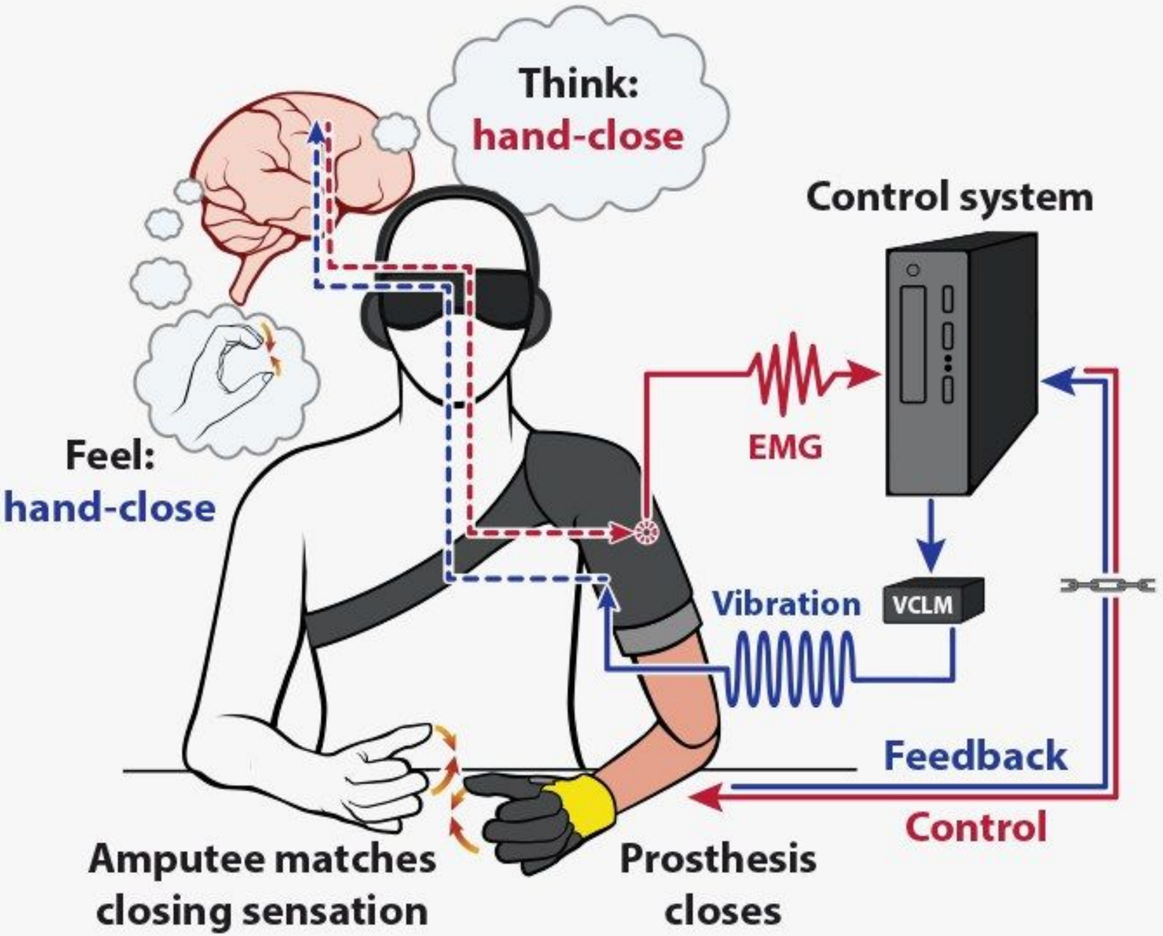
- As opposed to many other methods, decoding models are **predictive**
- They can predict **new** ("out of sample") ...
  - Stimuli (brain reading / reconstruction)

Presented clip

Clip reconstructed from brain activity

Instrumental reasons

- As opposed to many other methods, decoding models are **predictive**
- They can predict **new** (“out of sample”) …
  - Stimuli (brain reading / reconstruction)
  - Motor actions


Think:
hand-close
Control system
Feel:
hand-close
EMG
Vibration
VCLM
Feedback
Control
Amputee matches
closing sensation
Prosthesis
closes

Instrumental reasons

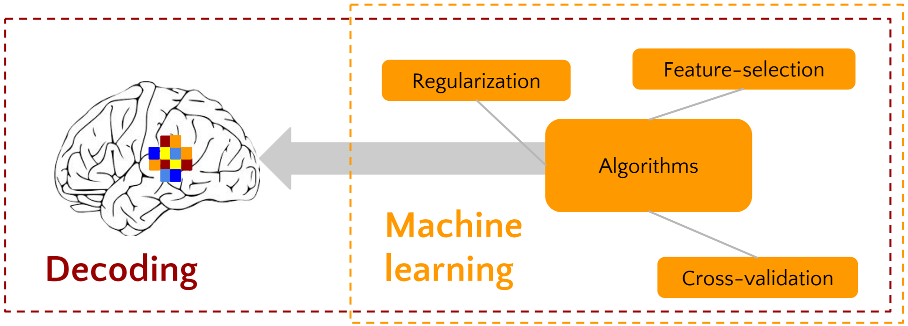
- As opposed to many other methods, decoding models are **predictive**
- They can predict **new** ("out of sample") ...
  - Stimuli (brain reading / reconstruction)
  - Motor actions (brain-machine interfaces)
  - Disease development (biomarkers; next talk!)

Contents

- Introduction
- Why?
- What?
- How?

Terminology

- **Decoding** ≈ machine learning
  - Generic name for brain → stimulus/task analyses;
  - Also: neuroimaging-specific name for 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 ...

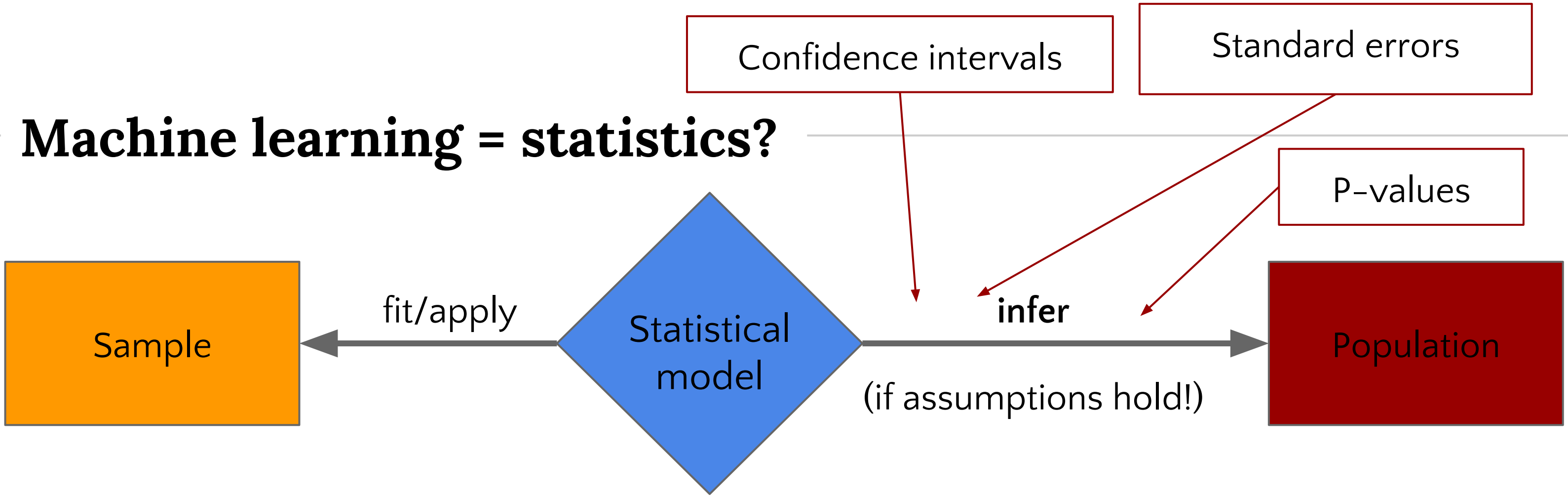
Machine learning = statistics?

- 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 traditional **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?

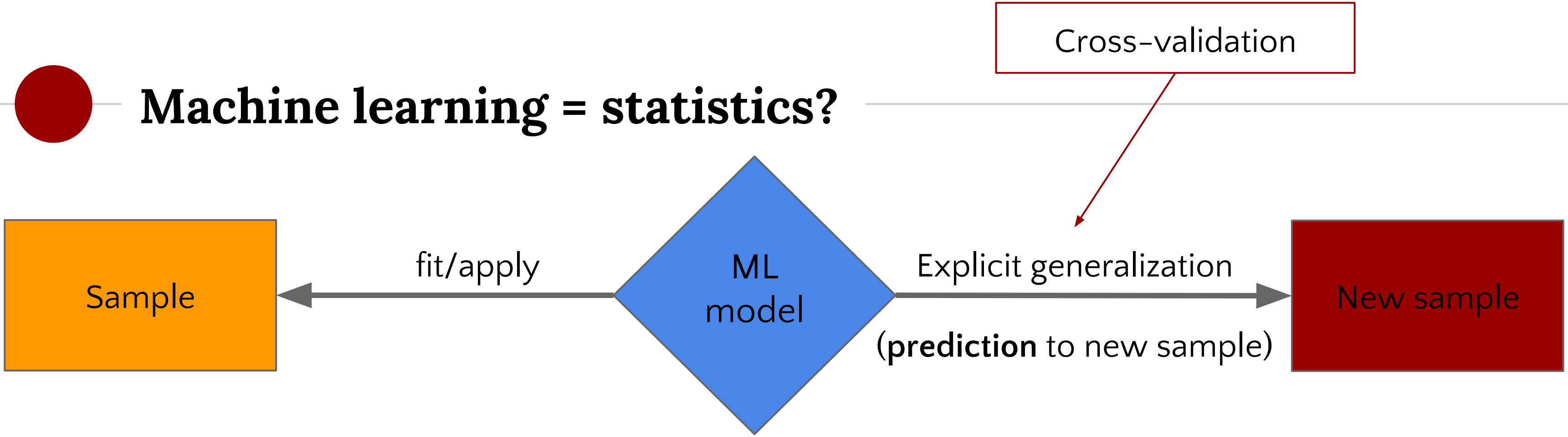


- Crucially: we are quite certain that our findings in the sample will **generalize to the population**, *if and only if assumptions of the model hold* (and sample = truly <u>random</u>)
- Uses concepts like standard errors, confidence intervals, and p-values to generalize the model to the population

Machine learning = statistics?

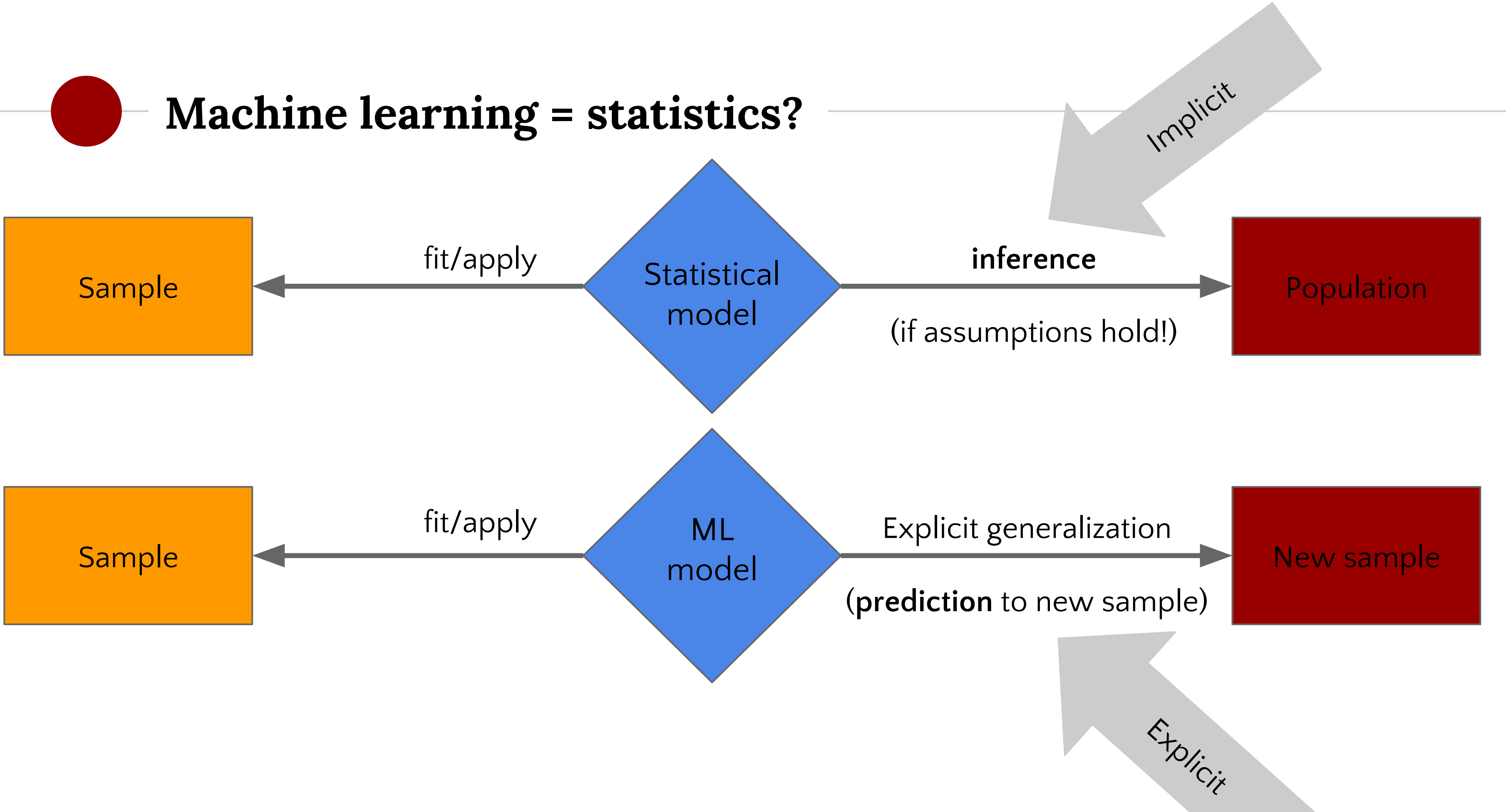
- ML models do not aim for **inference**, but aim for **prediction**
  - Instead of assuming findings will generalize to the population, ML analyses in fact ***literally check*** whether it generalizes;
  - It's like they're saying: "I don't give a shit about assumptions – if it works, it works."

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 explain 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 …?

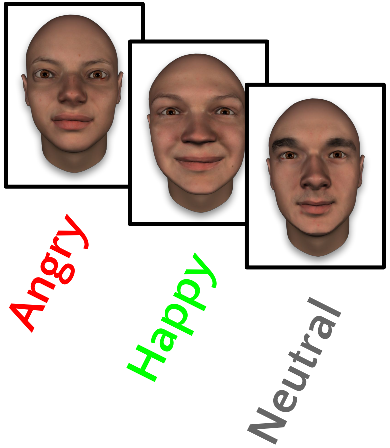
- As said, statistics and ML are the **same, yet different**;
  - Both aim to model the data (with different techniques) …
  - … but have a different way to generalize findings
- "But why do we have to learn a whole new paradigm (ML), then?", you might ask …

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** specifically: e.g., predict clinical treatment outcome

What is decoding?

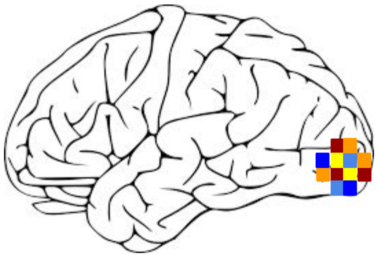
“feature space”



$y = \text{model}(X)$

DECODING

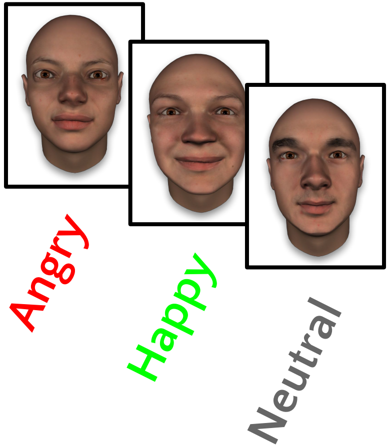
“brain space”



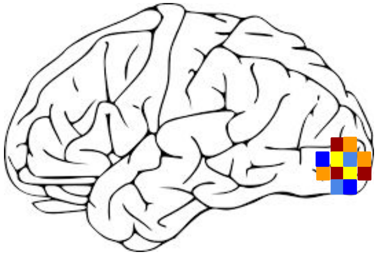
What happens in that arrow?

Back to the brain

“feature space” “brain space”



$$y = f(X)$$



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 linearly combined ("weighted") such that they approximate the **feature-of-interest** (y)?

Machine learning model

Samples-by-features matrix (brain space)

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

The predicted target(s) variable, numerically coded (feature space)

ML algorithm that specifies how to weigh X

Support vector machines

(Logistic) regression

Decision trees

f(X) is simply weighting!

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

$\boldsymbol{\beta}$ denotes the weighting parameters (or just "parameters" or "coefficients") for our features (in X)

$X\boldsymbol{\beta}$ denotes the (matrix) product of the weighting parameters and X – which means that y is approximated as a weighted linear combination of features

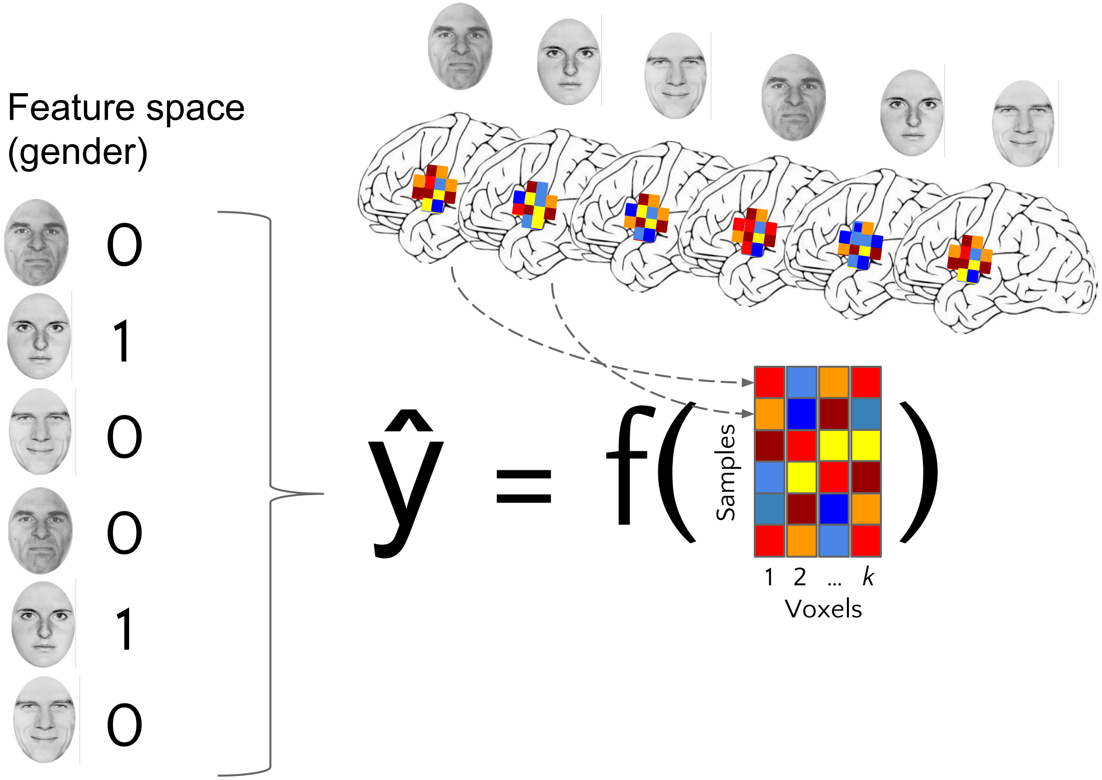
Linear vs. non-linear

- Disclaimer: the "$X\boldsymbol{\beta}$" 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 et al., 2011)
- Non-linear models exist, but are **rarely used**
    - Also because they often perform worse than linear models

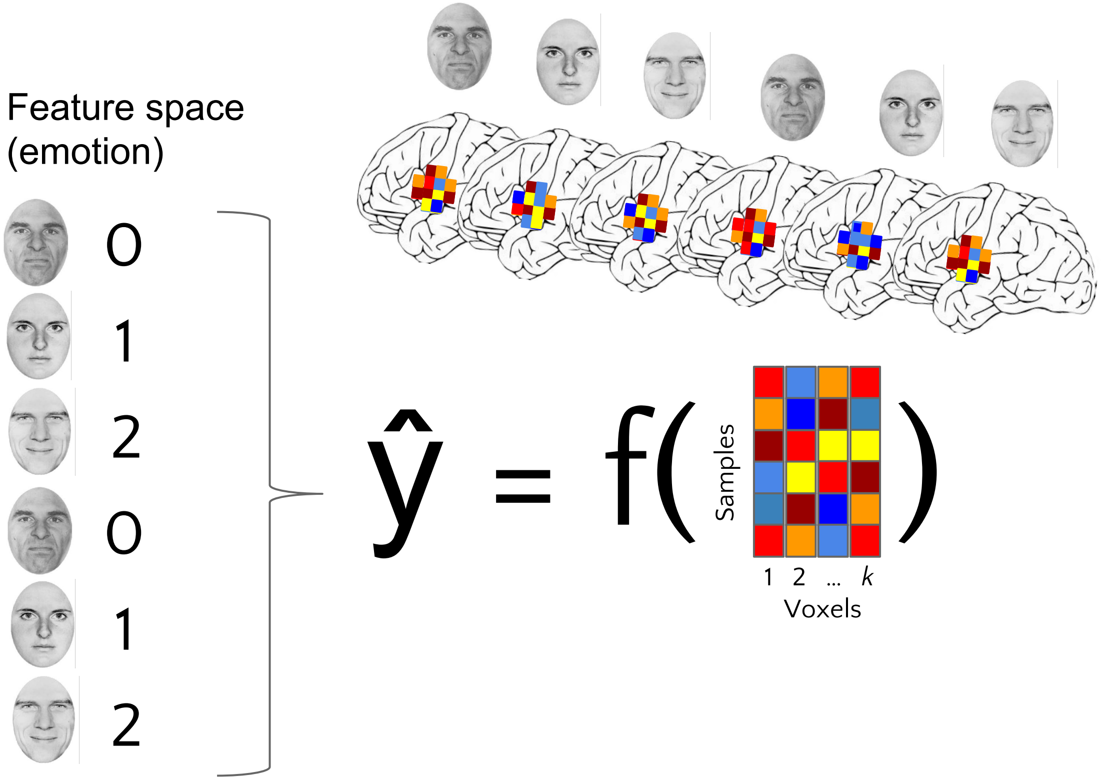
f(X) is simply weighting!

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

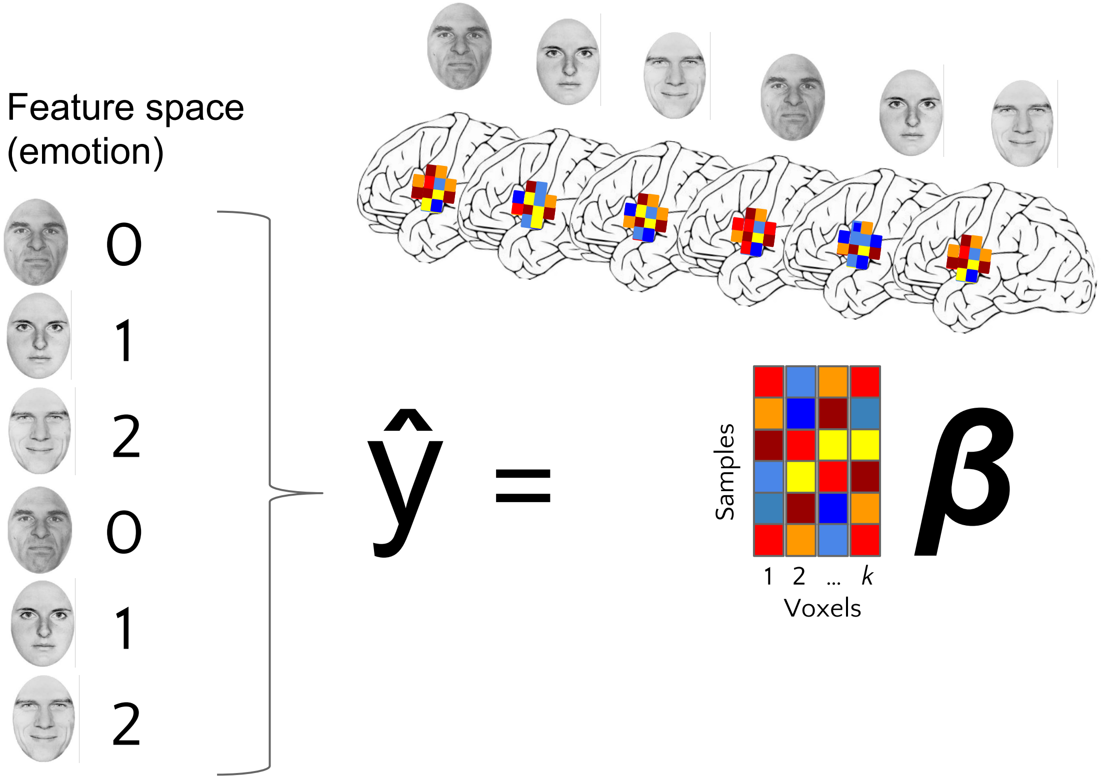
f(X) is simply weighting!



f(X) is simply weighting!

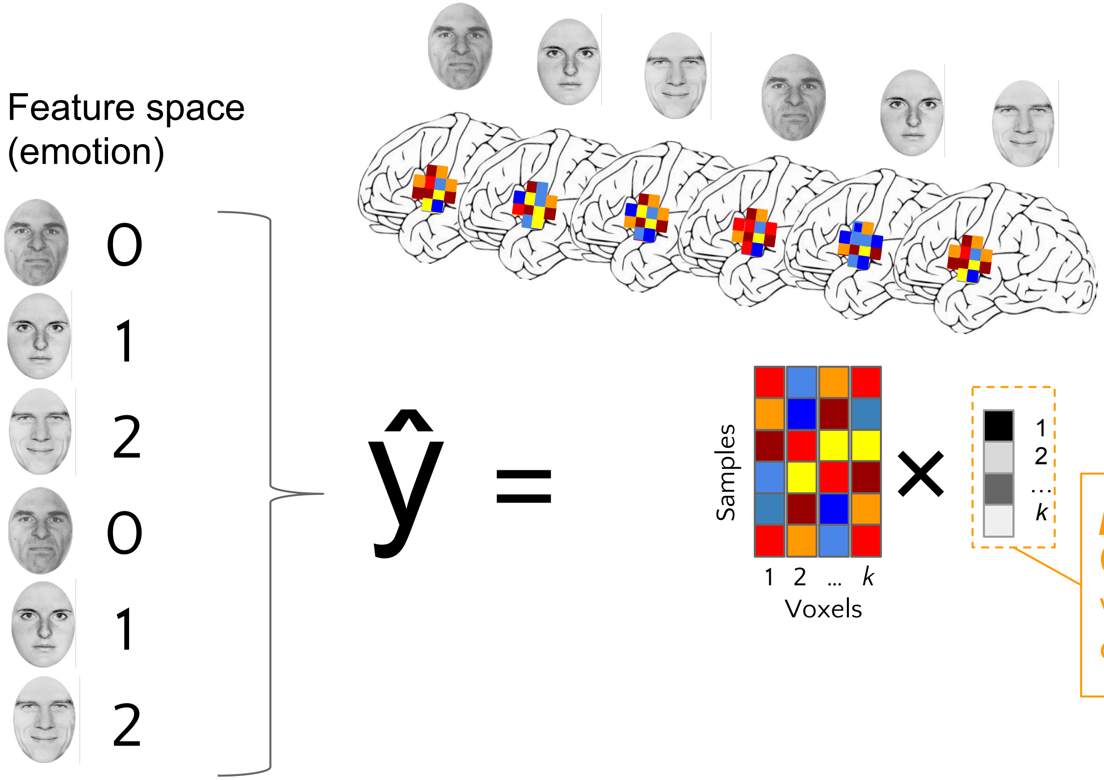


f(X) is simply weighting!



f(X) is simply weighting!

Feature space (emotion)

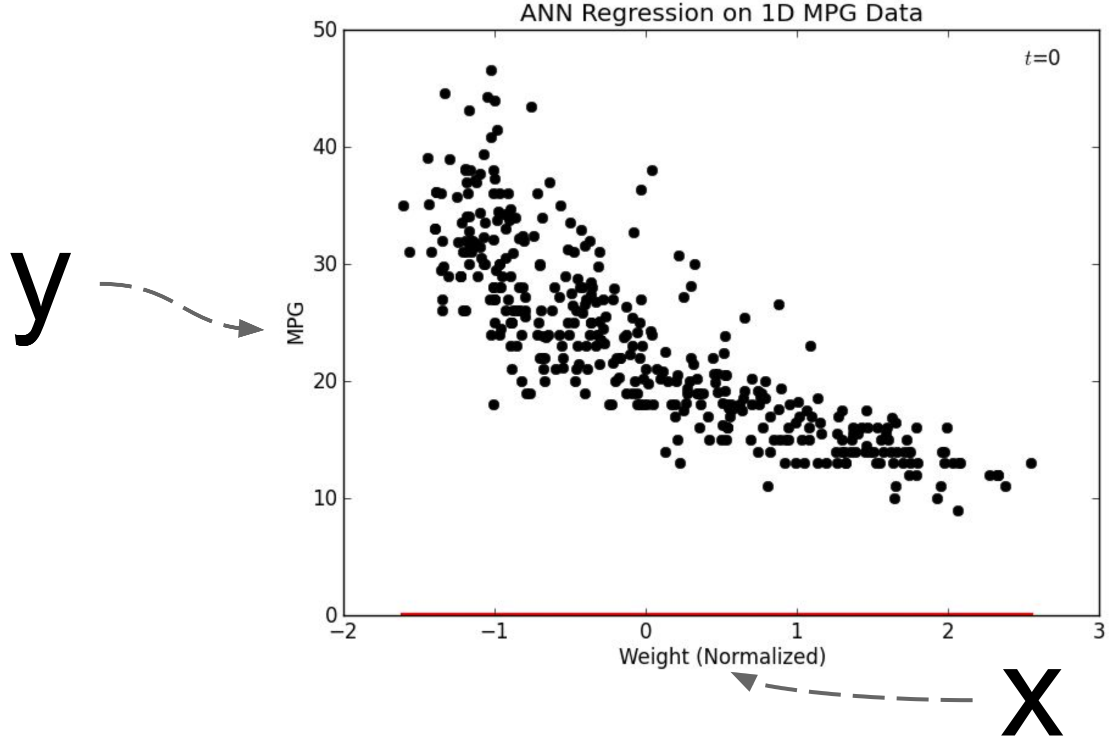


β is just a vector of length K (features) that specifies the weight for each feature to optimally approximate y

f(X) is simply weighting!

- But **how** does f(X) find the "optimal" weights ***β***?
- Depends on the specific (linear) ML algorithm!
  - Usually by (intelligently) **trying out different values for *β*** until it doesn't improve anymore ("gradient descent")
- Not part of this lecture!

Gradient descent



Model: $y = \beta_0 + X\beta_1$

- Start with $\beta_0 = 0$ and $\beta_1 = 0$
- Update $\boldsymbol{\beta}$ in ‘the right direction’
- Repeat until fit doesn’t get (much) better (“convergence”)

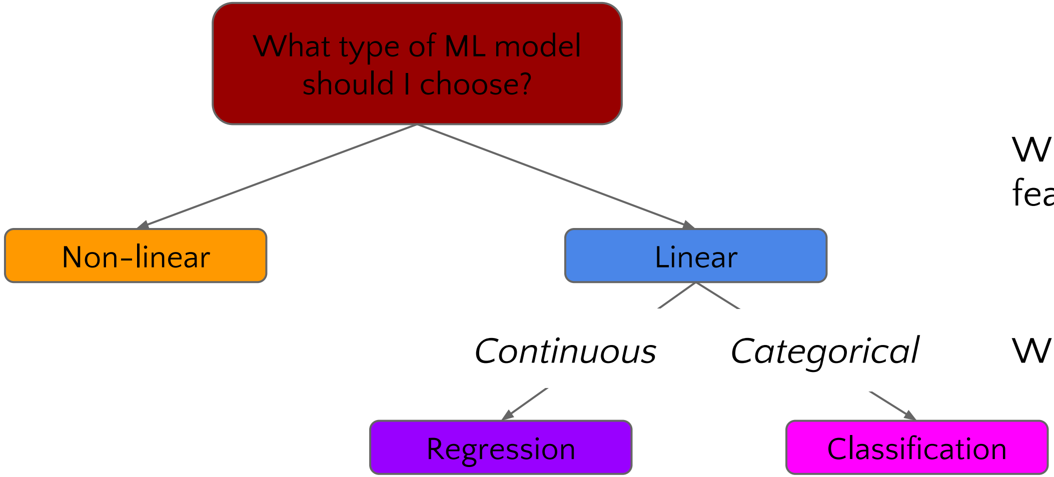
Summary

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

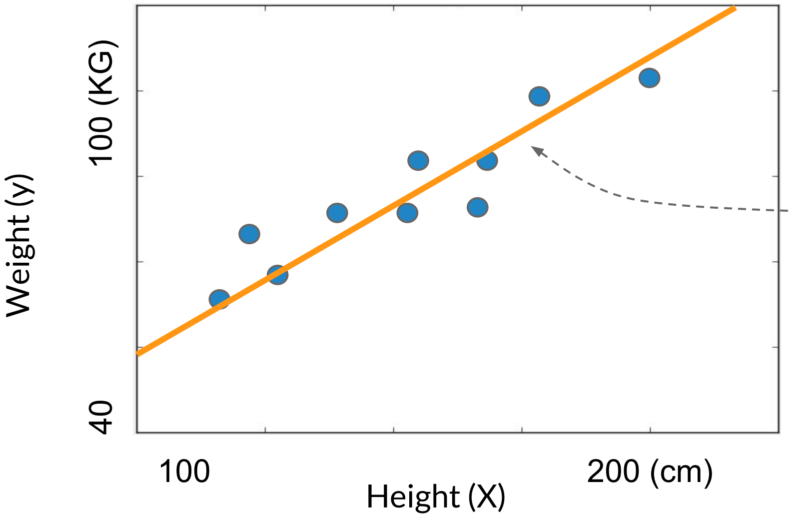
- Target (y) is continuous
- "Predict someone's weight based on someone's height"
- Example models: linear regression, ridge regression, LASSO

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



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

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

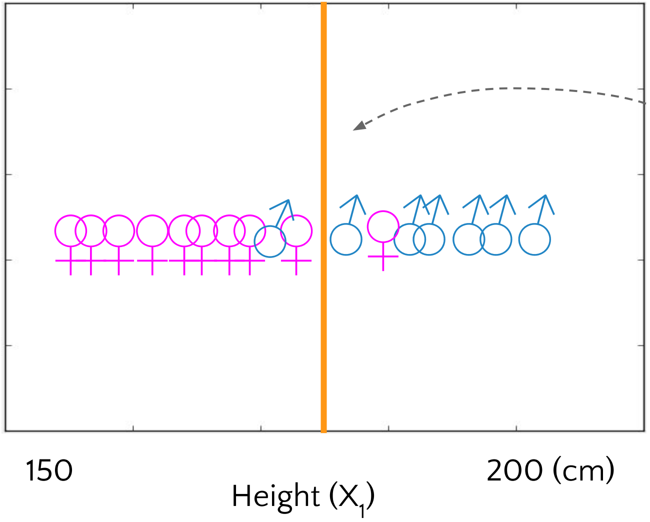
This is exactly like 'univariate' regression models, but instead of modelling in the encoding direction, this is in the decoding direction!

* 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 $X\boldsymbol{\beta}$ to be categorical

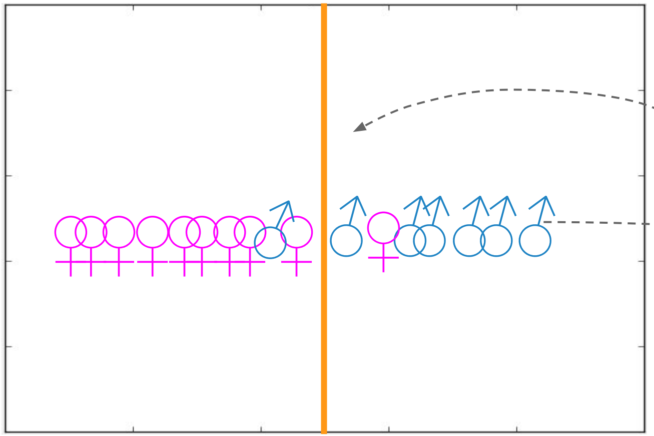
$\hat{y}$ = squash(X0.5Xβ height height))

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

Regression vs. classification

- Classification models a categorical variable

y = {♀,♂ }



$\hat{y}$ = squash(20.5X_{height})

$\hat{y}$ = squash(20.5*200)

$\hat{y}$ = 1 (predicted as 'male')

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, subjects 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).

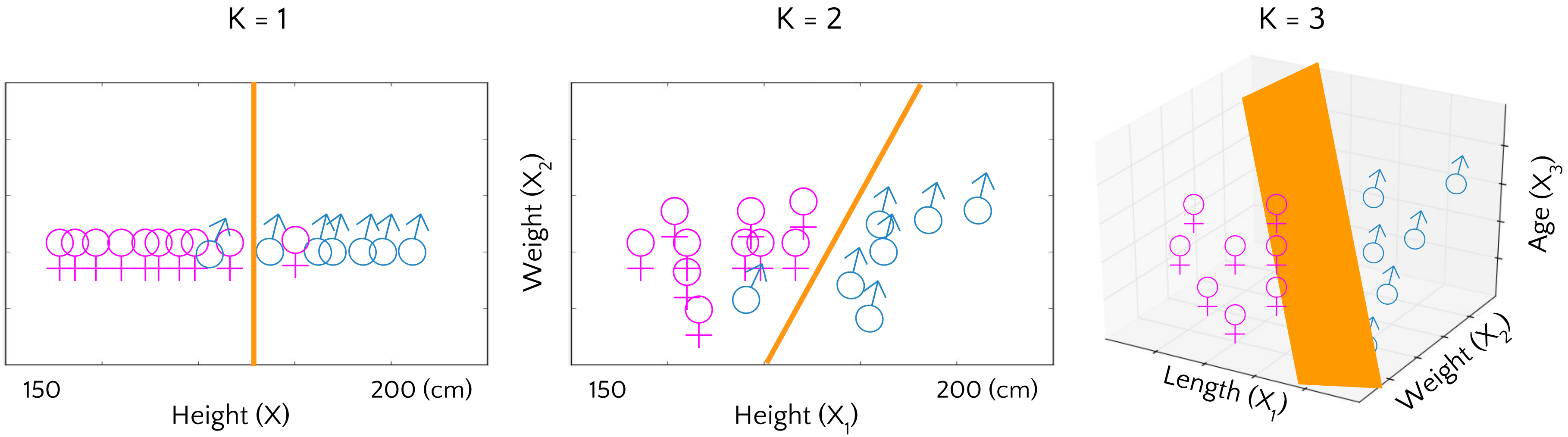
I want to predict whether someone has a relatively high IQ (>100) or low IQ (<100) based upon the patterns in dorsolateral PFC during the attentional blink task.

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 **$X\beta$** to a categorical value
- The examples were simplistic ($K = 1$); usually, ML models operate on **high-dimensional data** ($K > 200{,}000$)!

Dimensionality



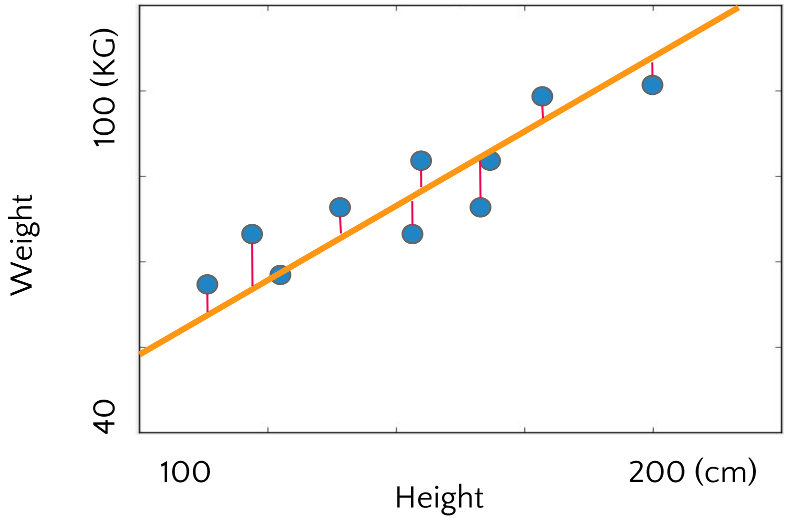
K = >3? → Difficult to visualize, but process is the same: weighting features such that a multidimensional plane is able to separate classes as well as possible in K-dimensional space

Model performance

- We know what ML models do (find weighting parameters $\boldsymbol{\beta}$ to approximate y), but how do we **evaluate** the model?
- In other words, what is the **model performance ("fit")?**

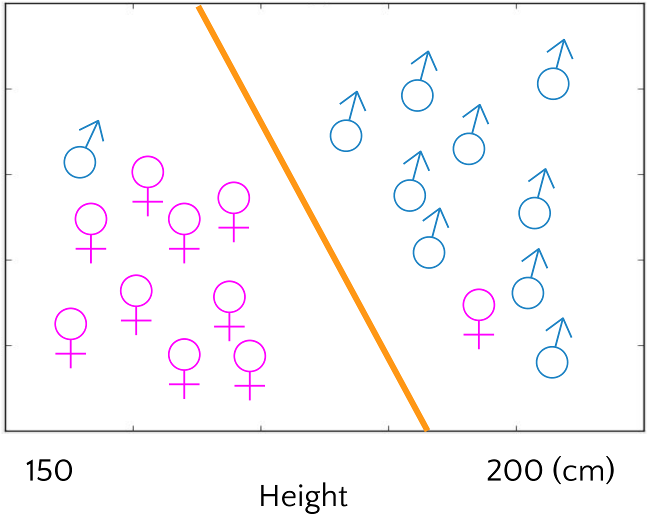
Model performance

Regression



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

Classification

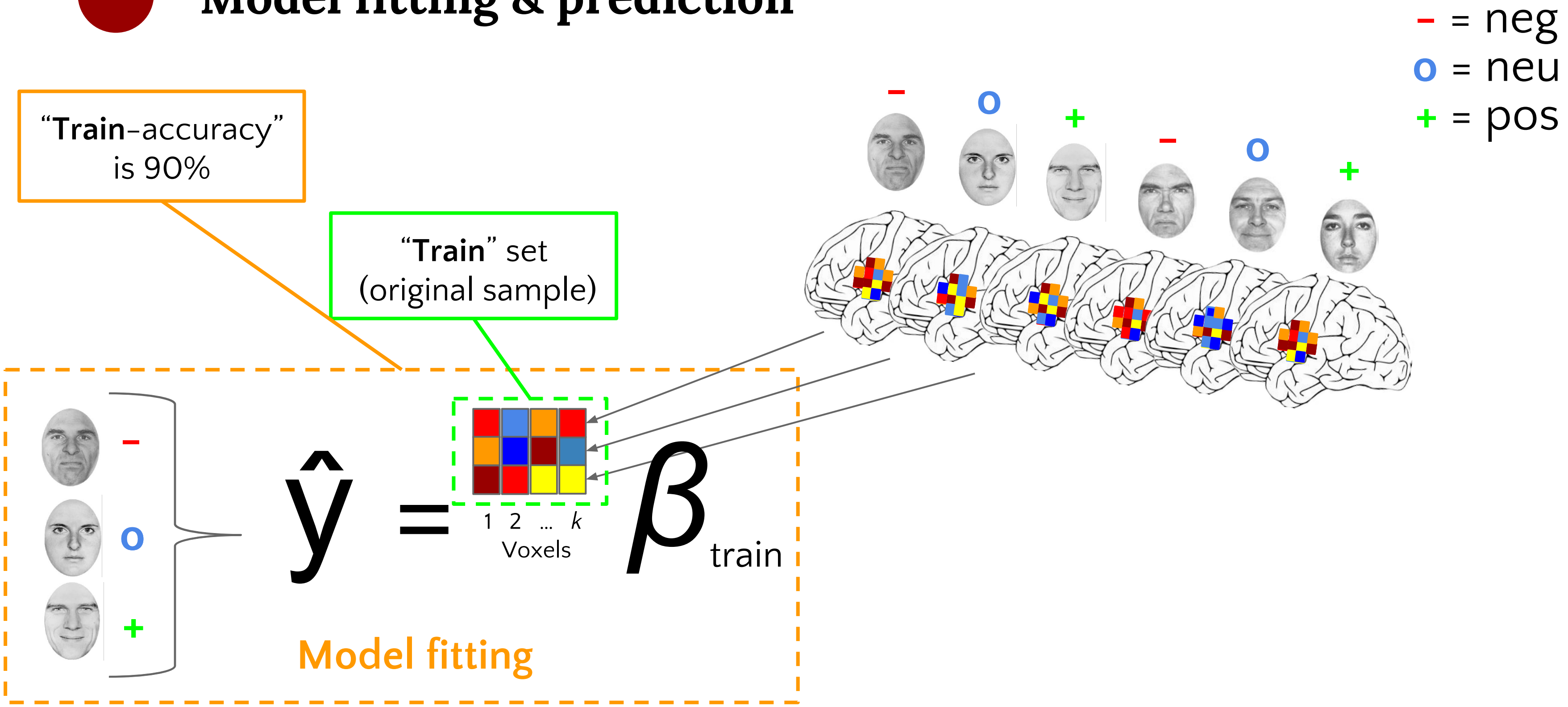


Accuracy = 18 / 20 = 90% [percent correct]

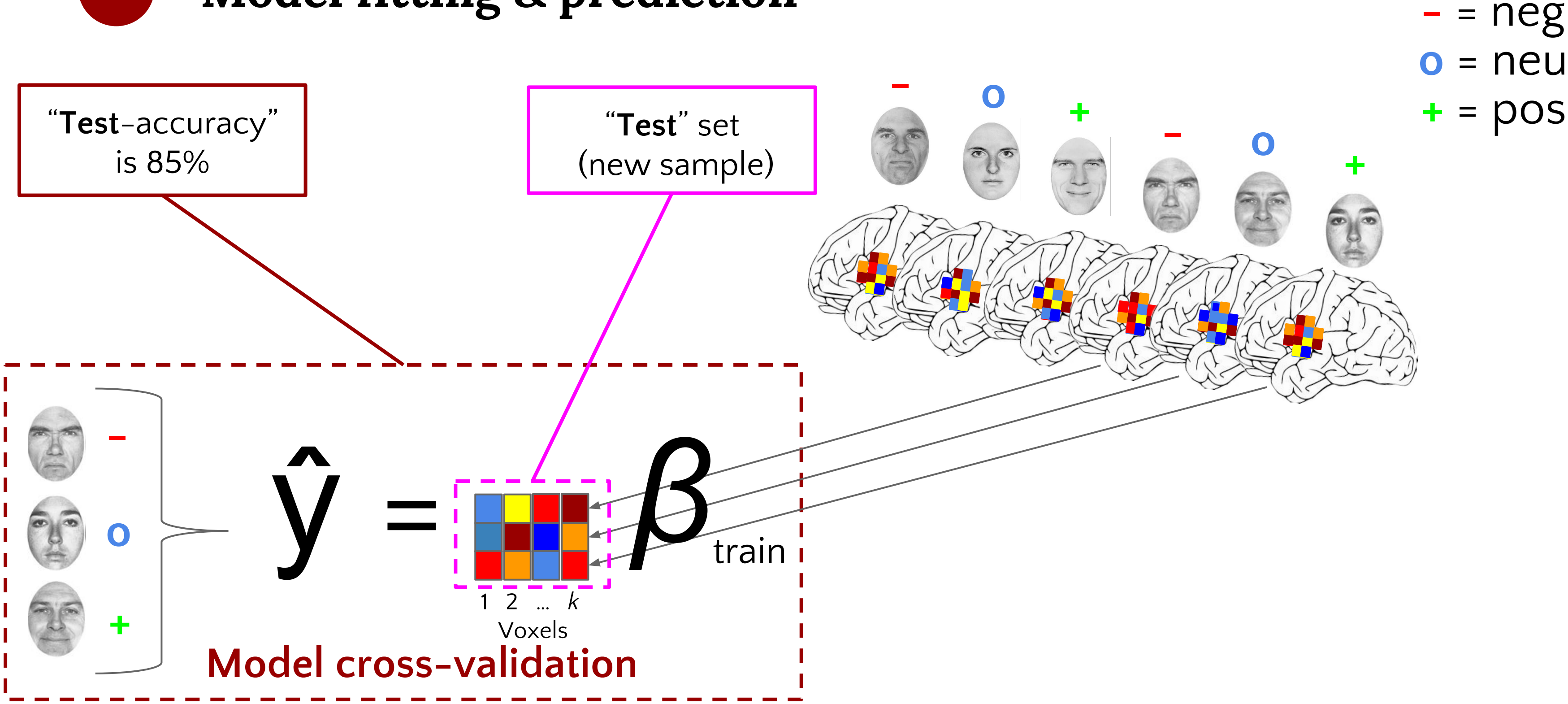
Model performance

- 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

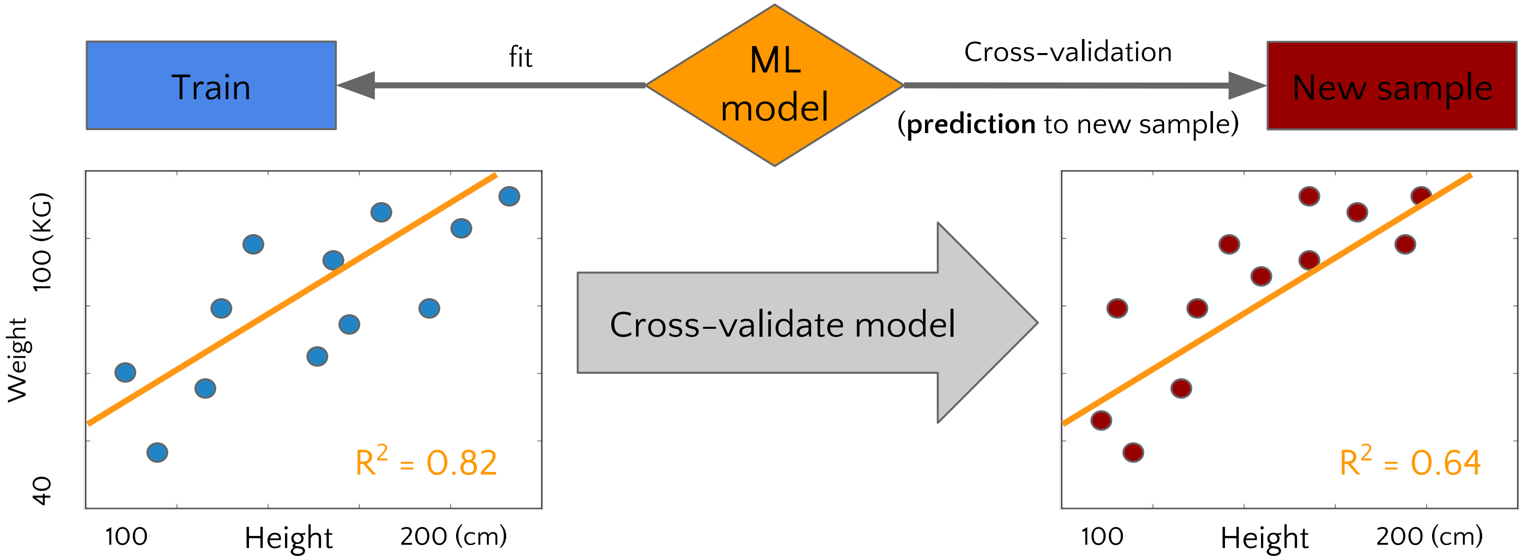


Model fitting & prediction



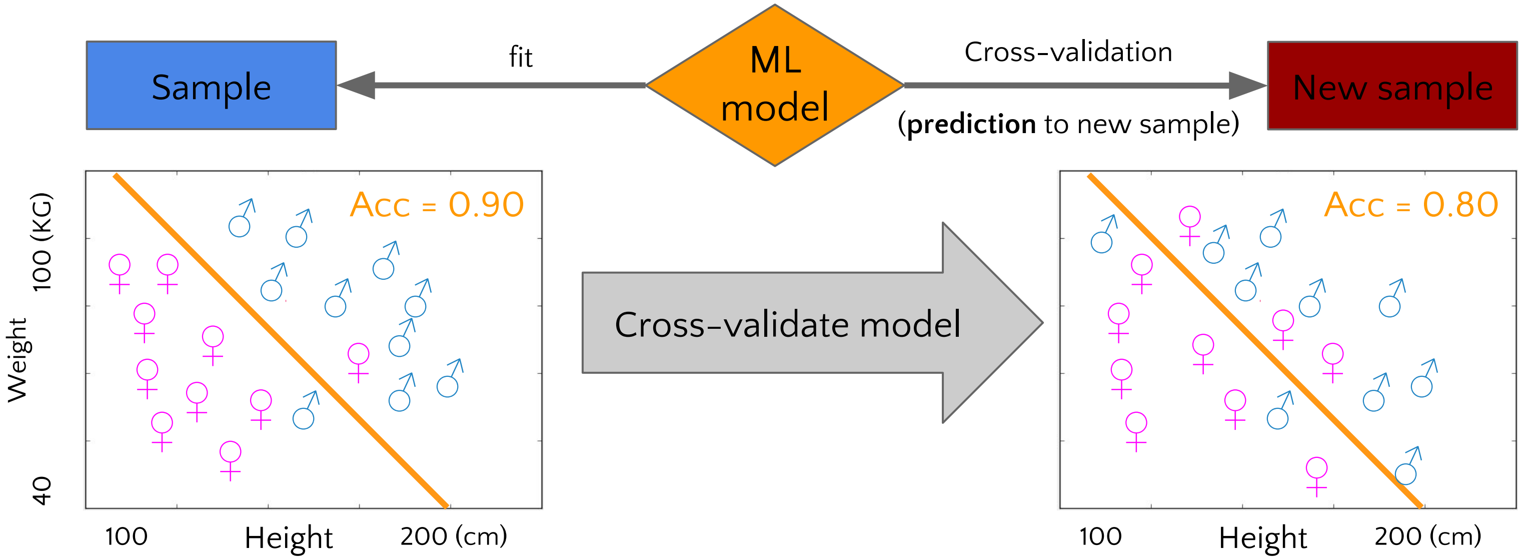
Model performance

- 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

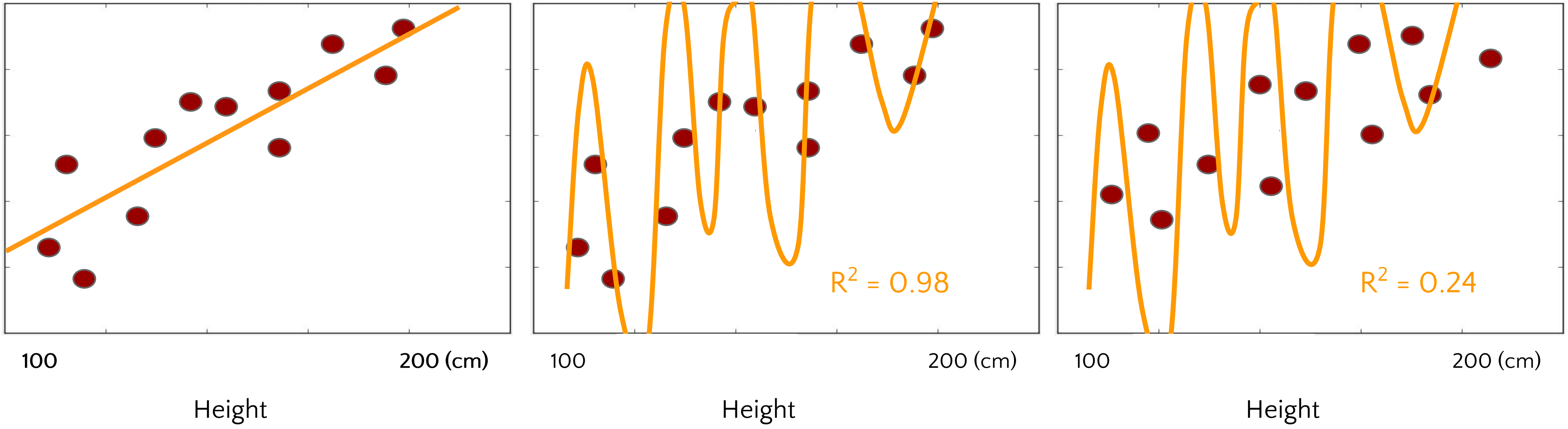
- When your fit on your **train-set** is better than on your **test-set**, you're **overfitting**
- Overfitting means you're modelling **noise** instead of **signal**

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?

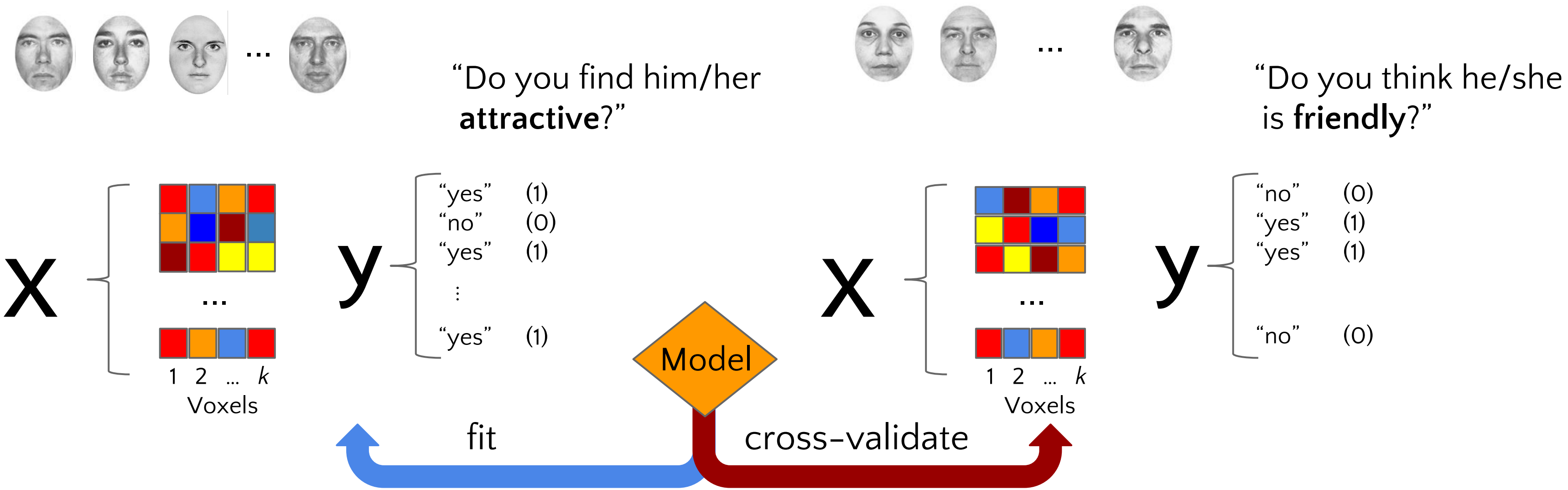
- Two options:
  - Gather **more data** (not always feasible)
  - **Reduce** the amount of **features** (more about this later)
  - [**Regularization** – beyond the scope of this course!]
- Feature selection/extraction is an often-used technique in decoding analyses
  - Discussed later ("**How?**")

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 types of representations

Cross-decoding!

- For example: suppose you have the hypothesis that 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')

Contents

- Introduction
- Why?
- What?
- How?

Slides online

- If we're out of time (very likely), you can check the rest of the slides online:

  https://tinyurl.com/MVPA-SPINOZA

- I'll be here tomorrow to help with (multivariate) analyses!

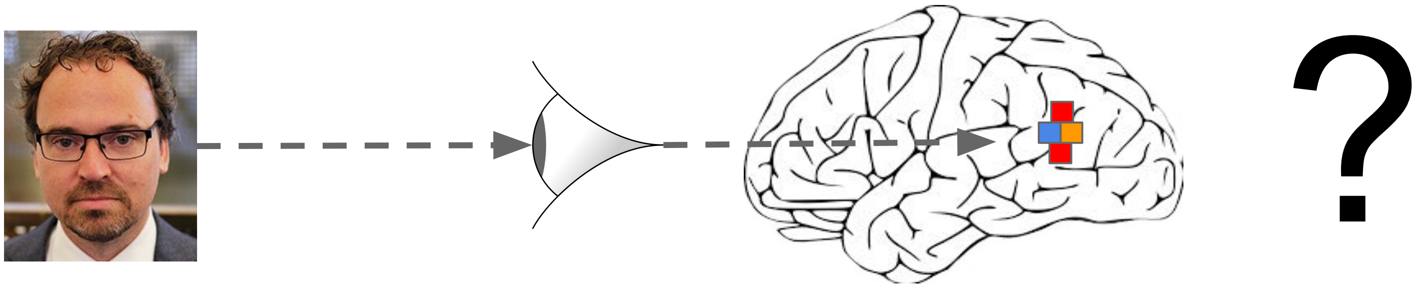
A typical decoding pipeline

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

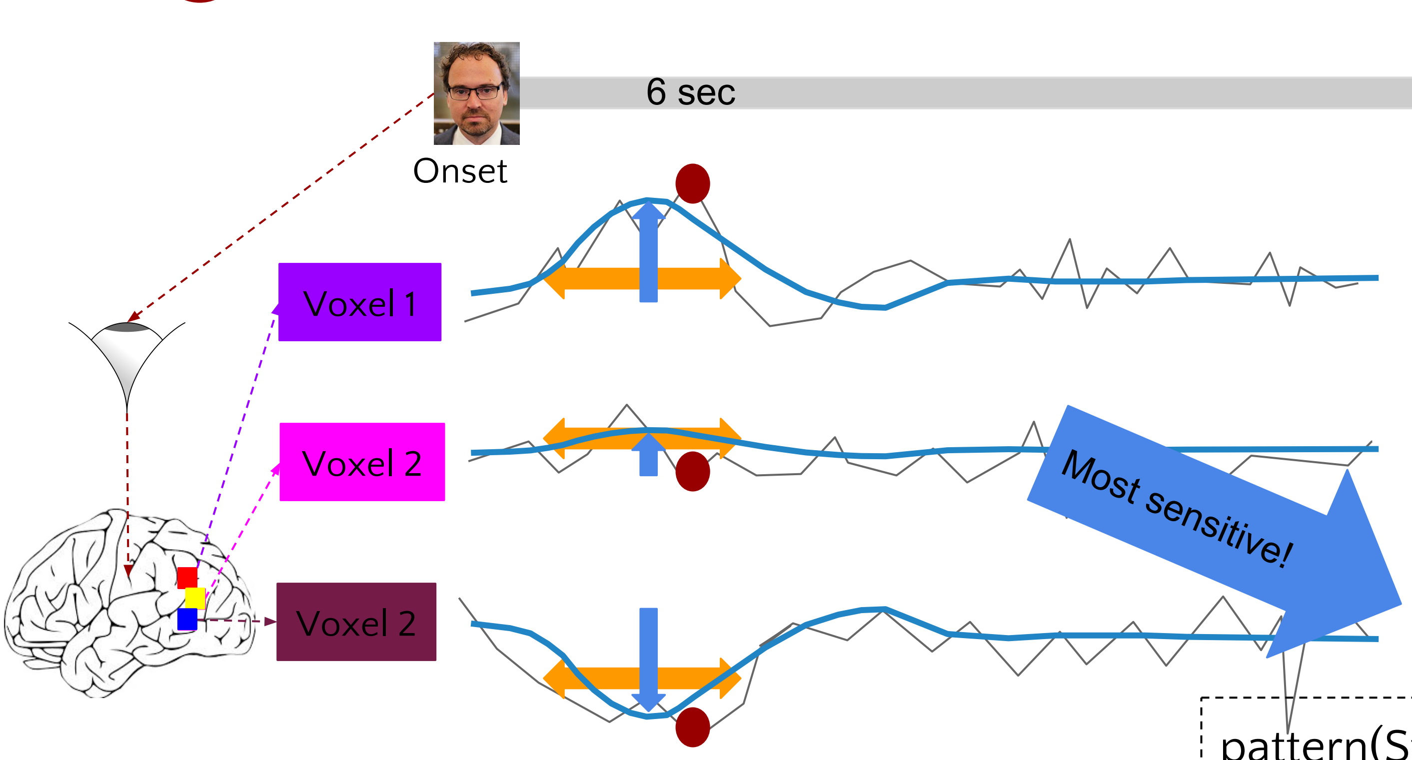
A typical decoding pipeline

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

How do we get here?



Estimating patterns: within



Three ways to extract patterns:

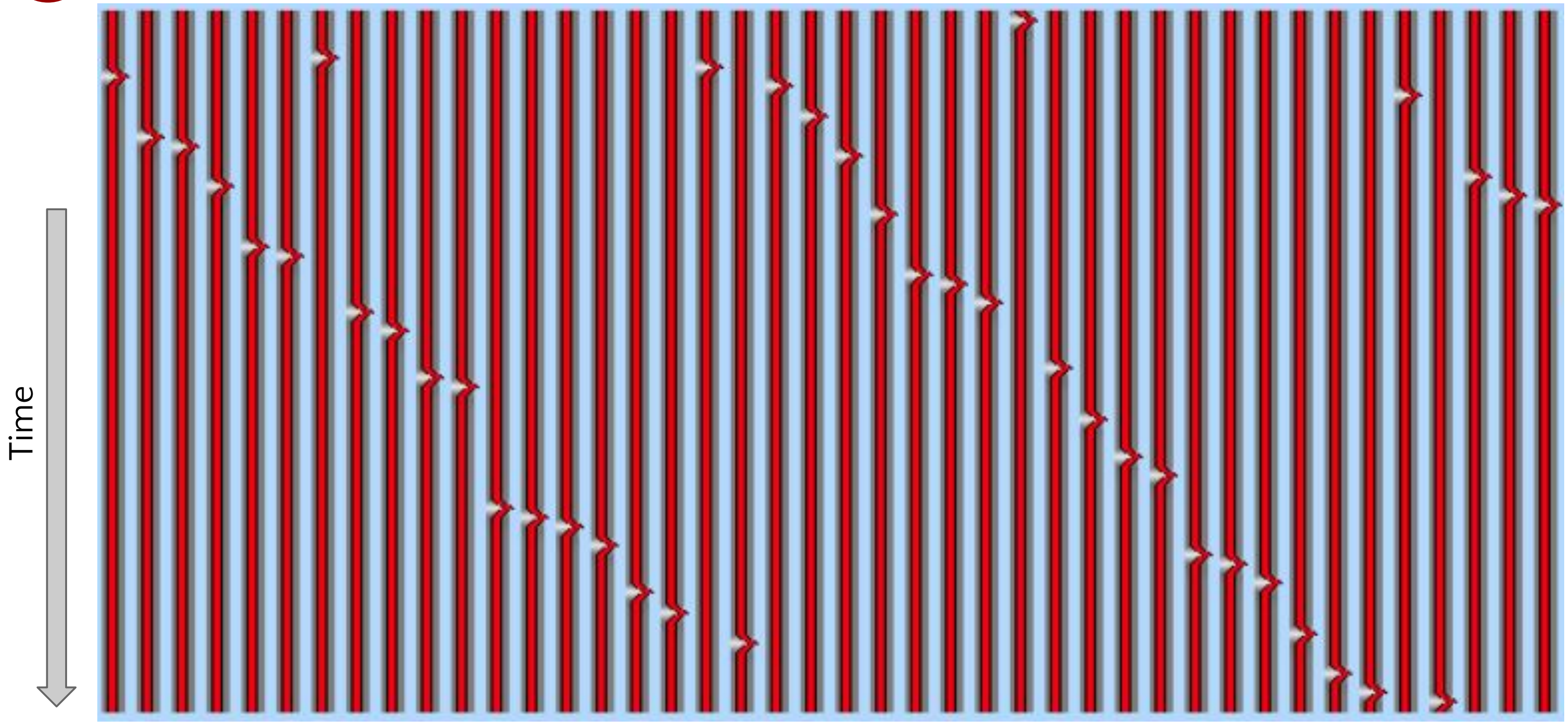
1. **Extract datapoint at a prespecified timepoint;**
2. **Extract mean of range of timepoints;**
3. **Fit HRF and extract ß or t-value;**

pattern(Steven) = [3.1, 0.4, -4.2]

Estimating patterns: within

- The design(-matrix) used in within-subject pattern analyses is often called a **single-trial design**
- Thus, you estimate a pattern **for each instance of your feature-of-interest** ("trial")

Estimating patterns: within



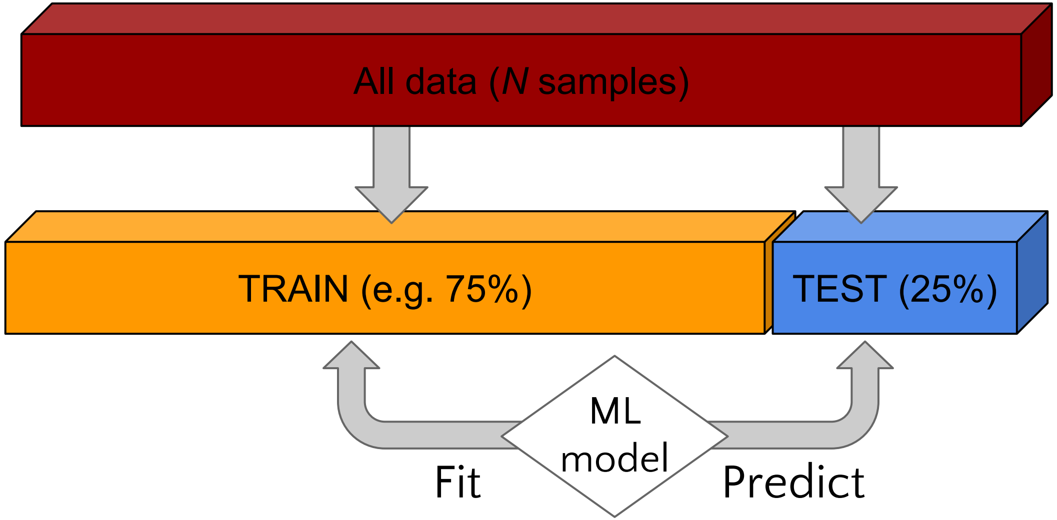
A typical decoding pipeline

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?

Leads to high "**variance**" of the model (little confidence) ...

Step 1: Partitioning

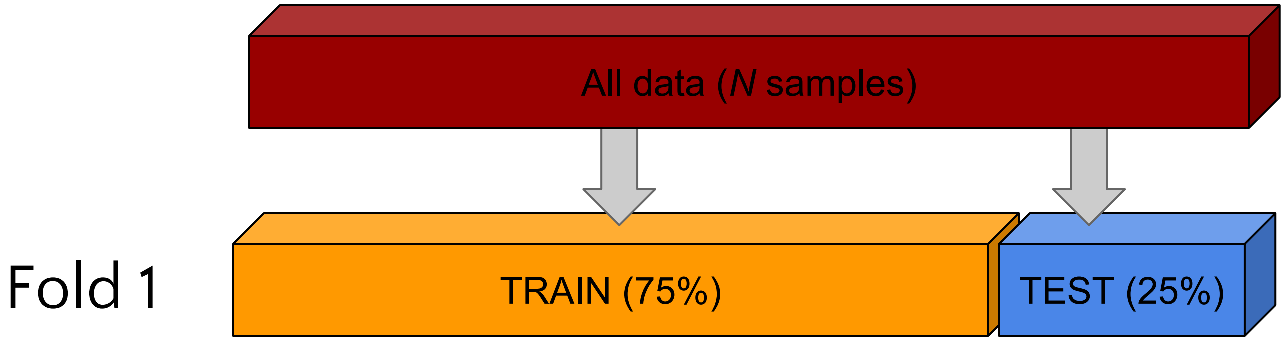
- Hold-out CV
- Advantage: computationally efficient (only fit model once!)
  - Recommended for very large datasets ($N > 1000$)
- Disadvantage: not very "robust" (especially for small datasets
  - If your test-set has only 10 samples, your estimated cross-validated accuracy may be spurious!

Step 1: Partitioning

- How to reduce variance? (increase robustness)
- Iteratively create partitions!
  - Into different folds: K-fold cross-validation

Step 1: Partitioning

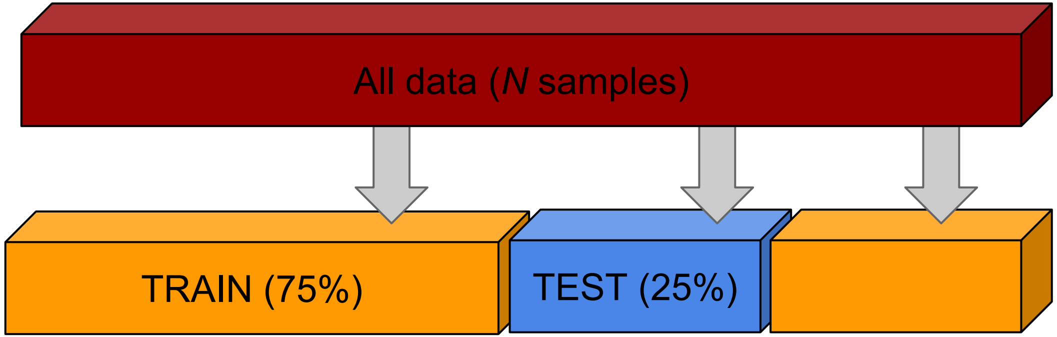
- K-fold CV, e.g. 4-fold



Step 1: Partitioning

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

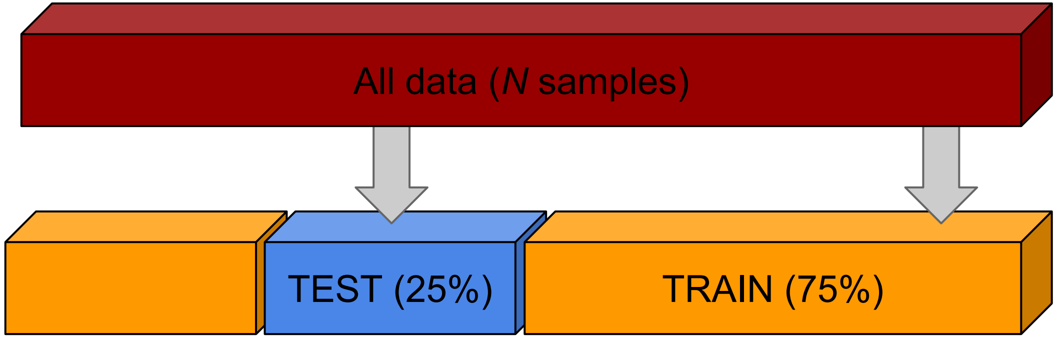
Fold 2



Step 1: Partitioning

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

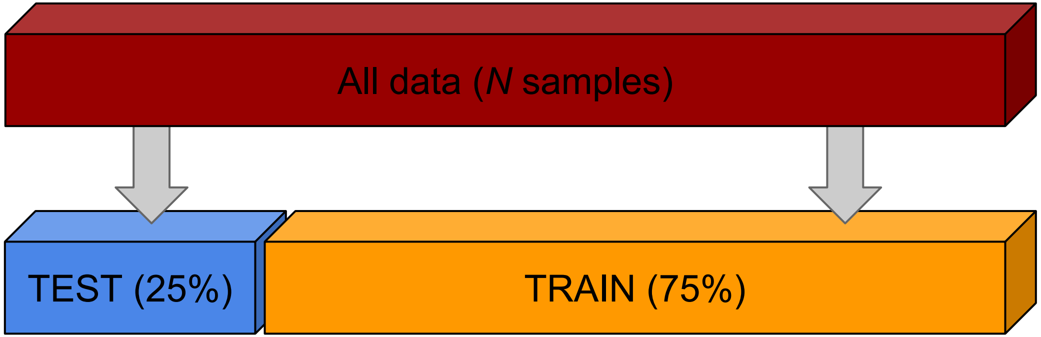
Fold 3



Step 1: Partitioning

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

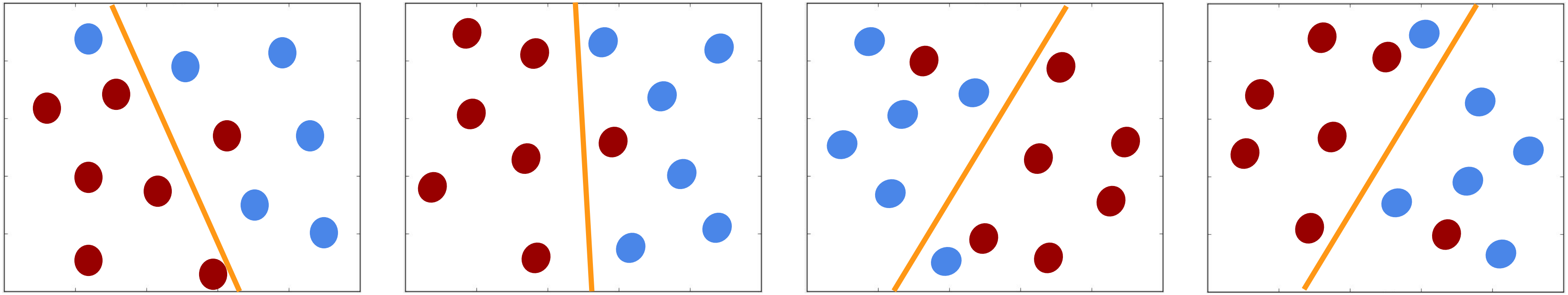
Fold 4



Step 1: Partitioning

- Often-heard question: “but now we fit 4 potential completely different models!”

FOLD 1 FOLD 2 FOLD 3 FOLD 4



Step 1: Partitioning

- Often-heard question: "but now we fit 4 potential completely different models!"
- True! Conclusion should be: "**a model with these features** (X) is able to predict with x% accuracy"
- Not: "this *specific* model ($X\beta$) is able to predict with x% accuracy"

Step 1: Partitioning

- How to reduce variance? (increase robustness)
- Iteratively create partitions!
  - Into different folds: K-fold cross-validation
- Average of fold-wise performance is more robust than just a single performance estimate (i.e., more robust)!

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

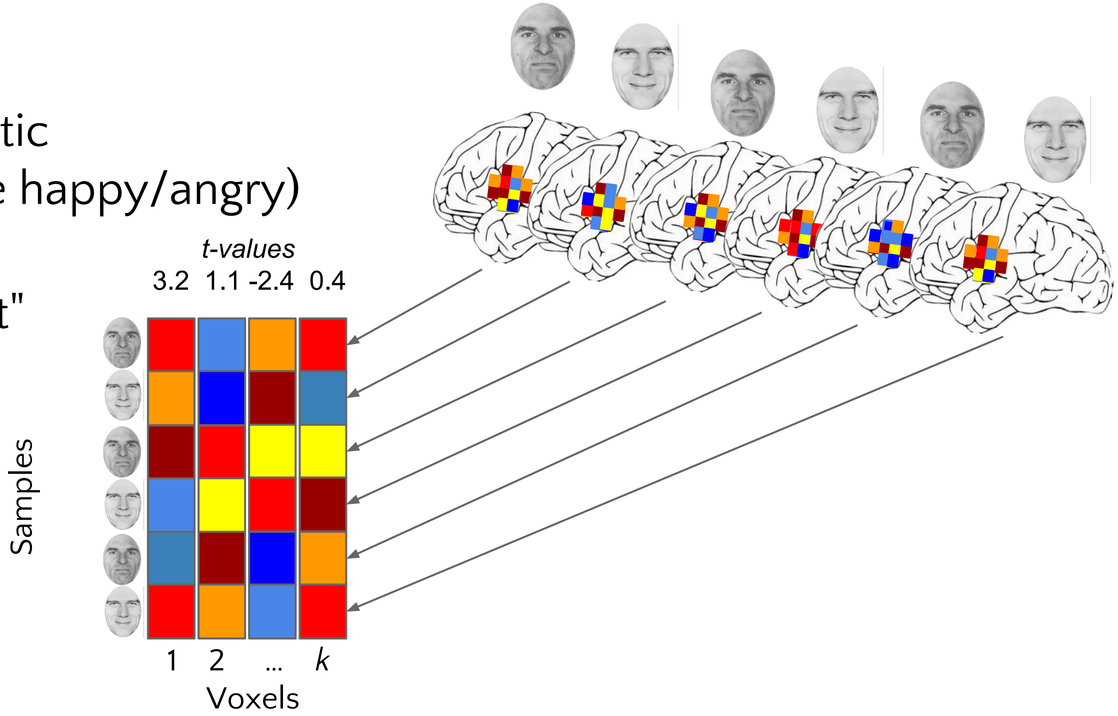
Univariate feature selection

- 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:

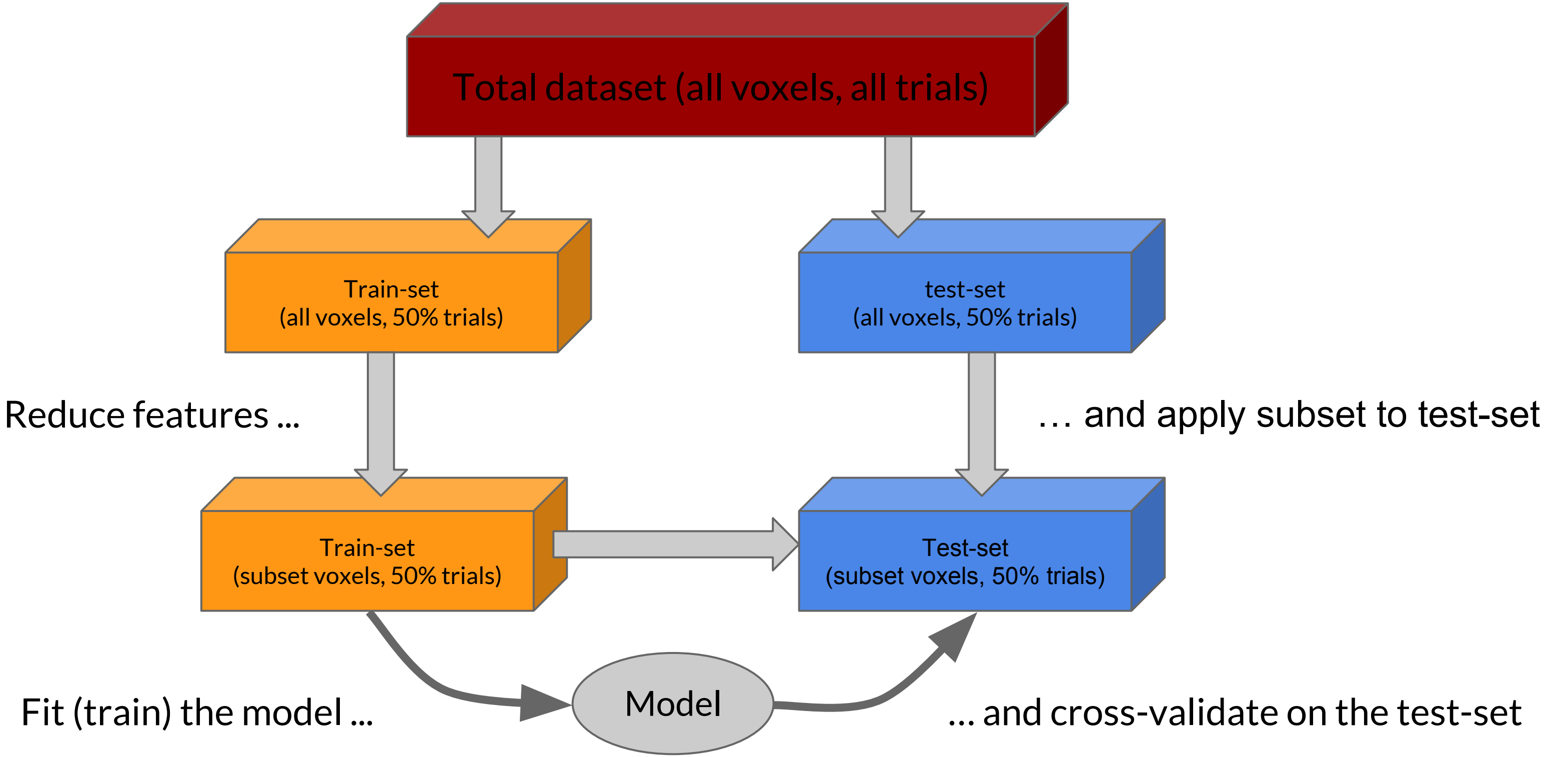
1. Calculate test-statistic (t-test for difference happy/angry)
2. Select only the "best" 100 voxels (or a percentage)



Cross-validation in FS

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

Cross-validation in FS



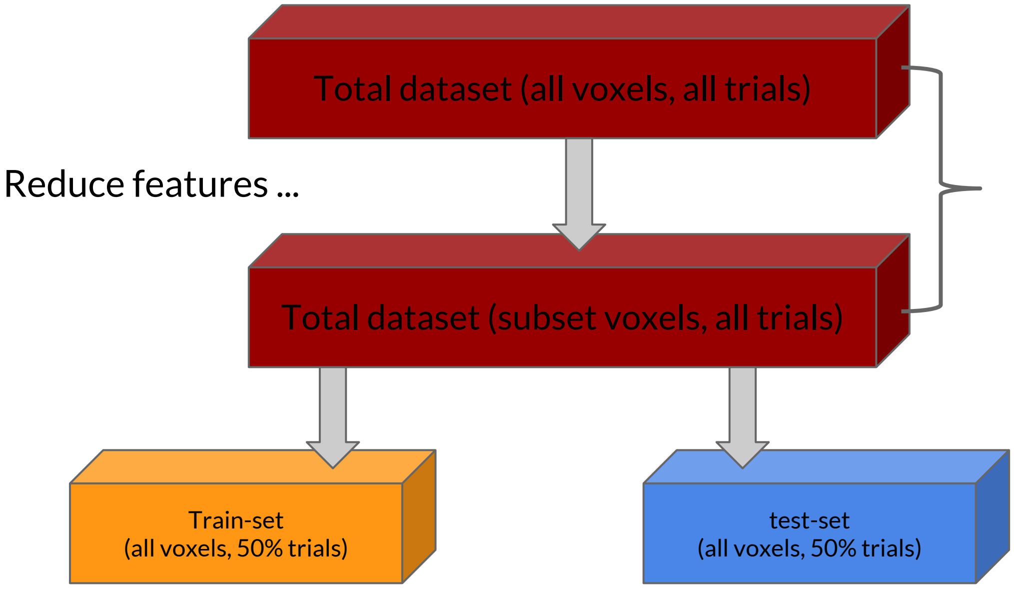
ToThink:

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?

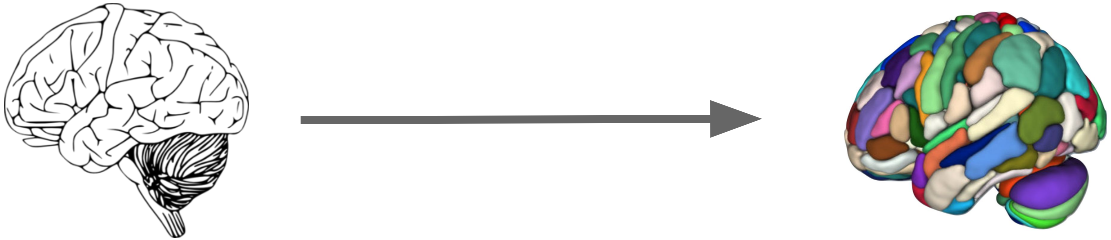


Here, you are using "information" from the test-set already!

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)

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

- When is a particular model performance "**good enough**"?
- Decoding analyses 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%

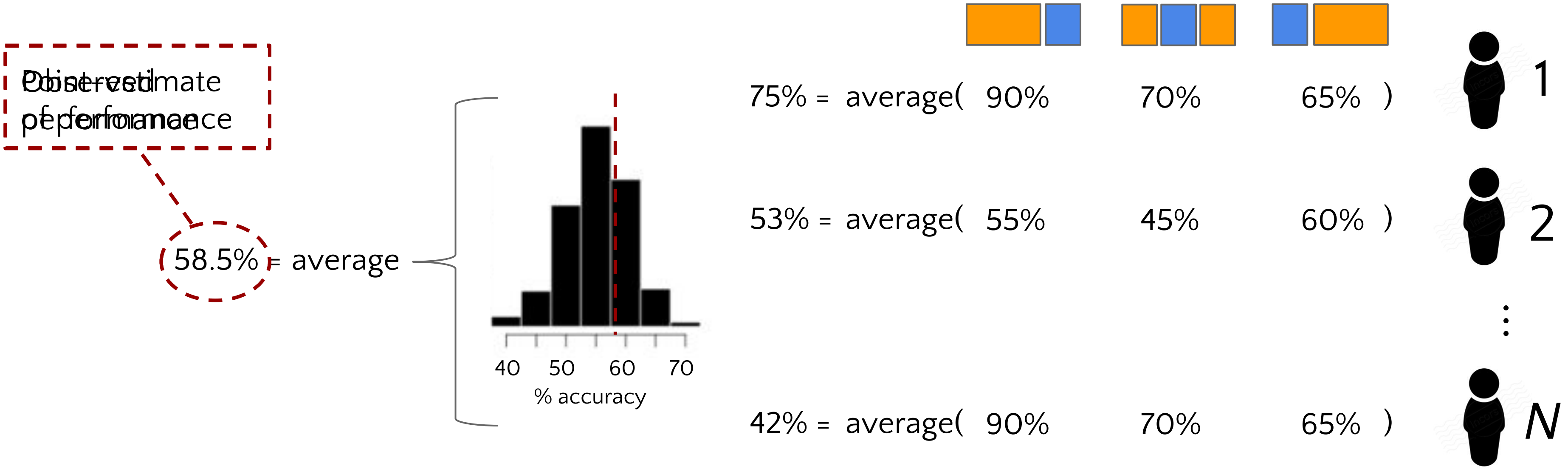
y = {  ,  ,  }

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



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 – 1) of **observed performance** (i.e. 58.5%) against the **null performance** (i.e. 50%)

Prevalence inference

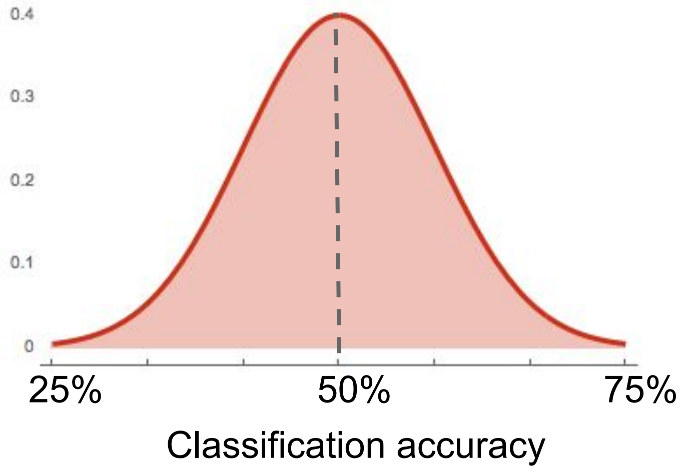
- Assuming independence between subjects, we can use simple parametric statistics*
- *Actually: not really ...

Valid population inference for information-based imaging: From the second-level *t*-test to prevalence inference

Carsten Allefeld[a,*], Kai Görgen[a,1], John-Dylan Haynes[a,b,1]

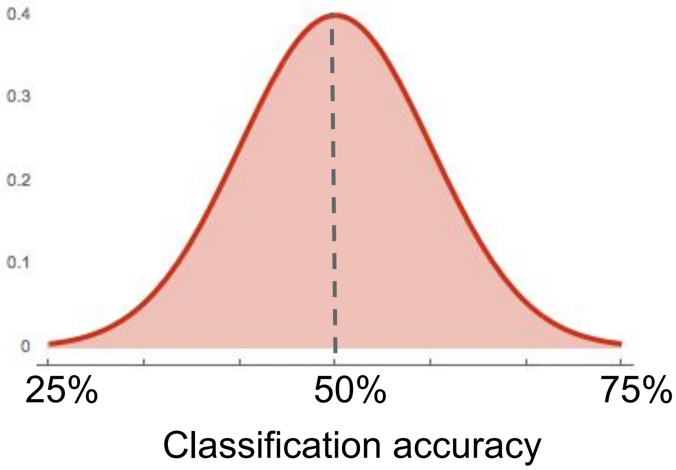
Prevalence inference

- Parametric statistics (in decoding) assume that you're testing against a symmetric null distribution



Prevalence inference

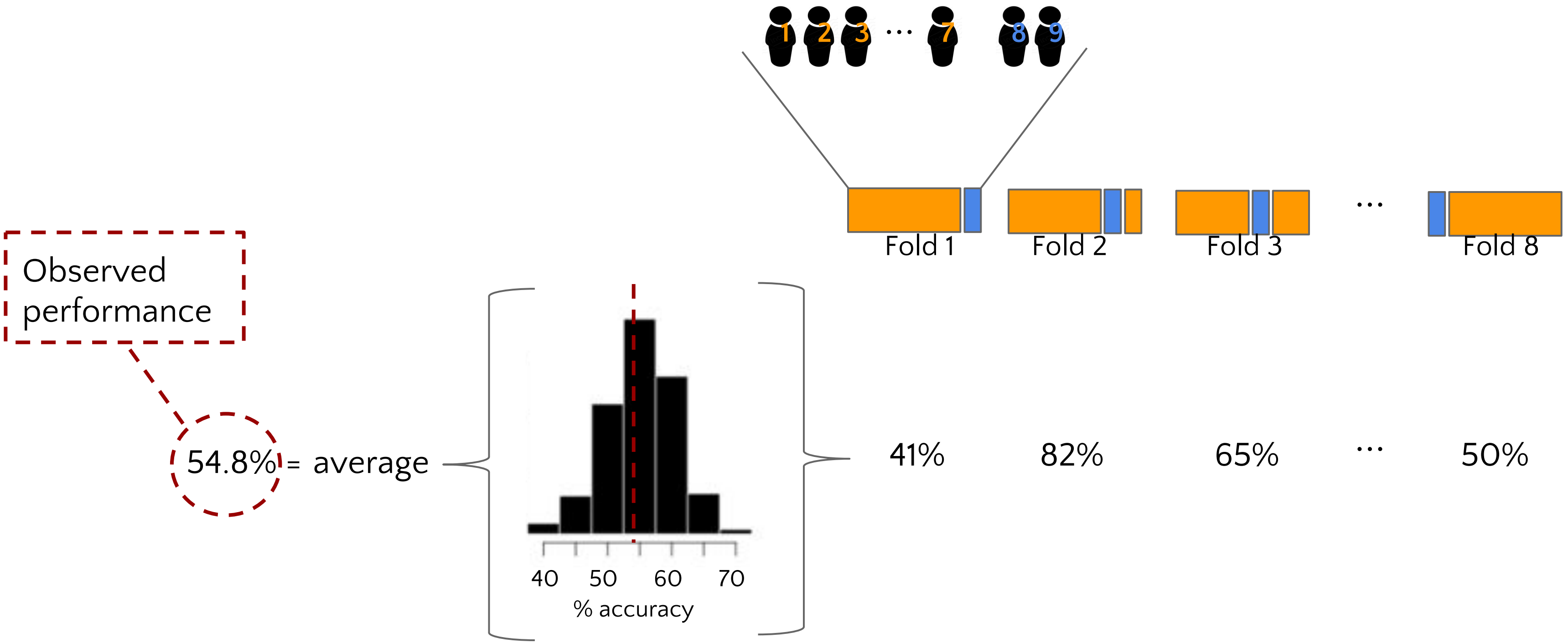
- Parametric statistics (in decoding) assume that you're testing against a symmetric null distribution
  - Assumes below-chance accuracy is possible
  - But this is not possible on a **population** level!
  - It's like testing the average travel time from house → work against $H_0 = 0$
    - Negative time???



Prevence inference

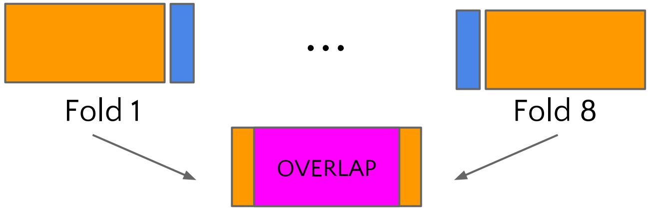
- Solution: test the **prevalence** of an effect (≈ how *many* subjects show an above-chance performance)
- Beyond the scope of this course …
    - But a possible topic for your final project!

Statistics: between-subject

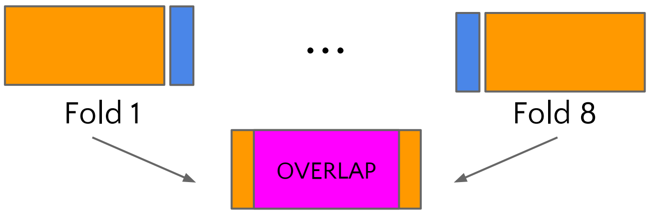


Statistics: between-subject

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



Statistics: between-subject

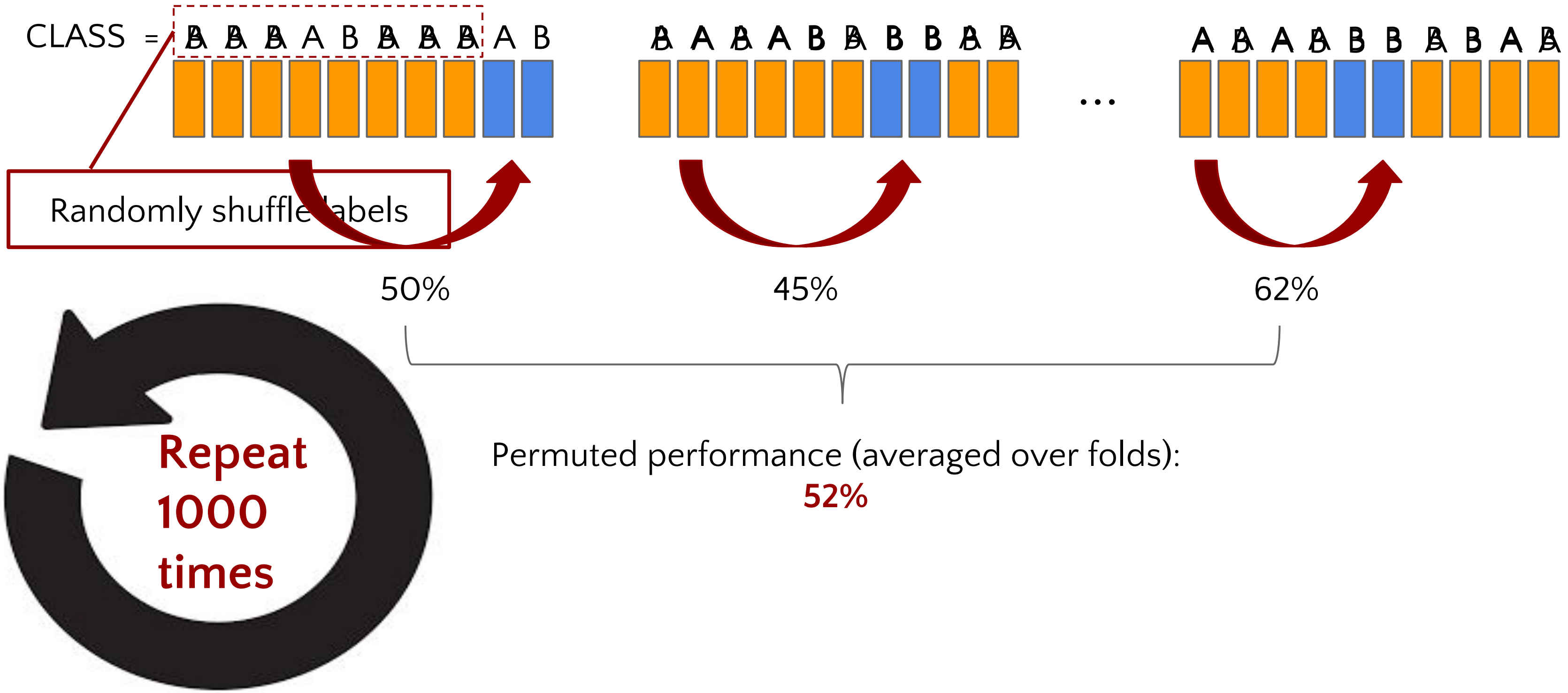


- Problem: different folds contain the same subjects
- Consequence: dependence between data points
    - Violates assumptions of many parametric statistical tests
- Solution: non-parametric (**permutation**) test

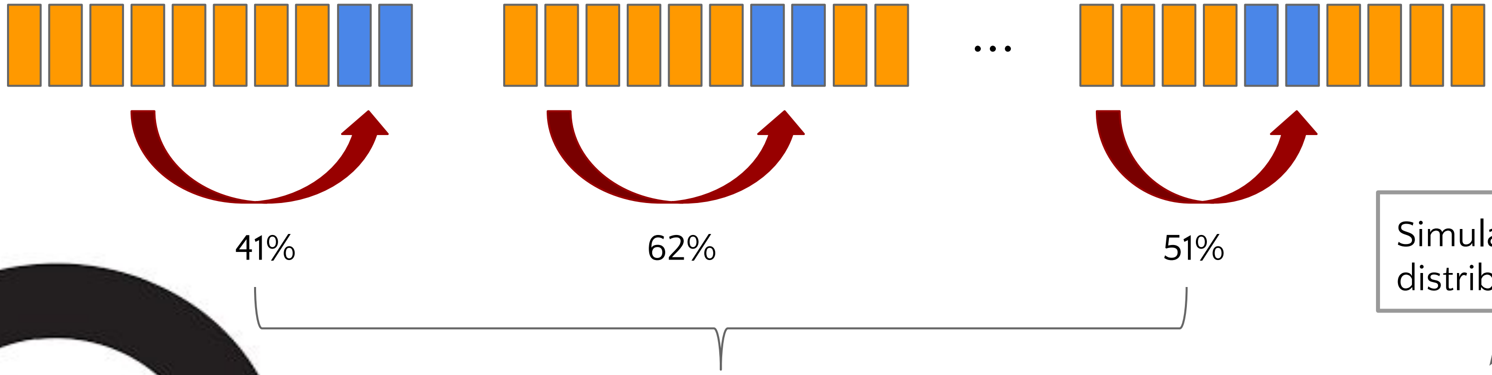
Statistics: between-subject

- 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"**

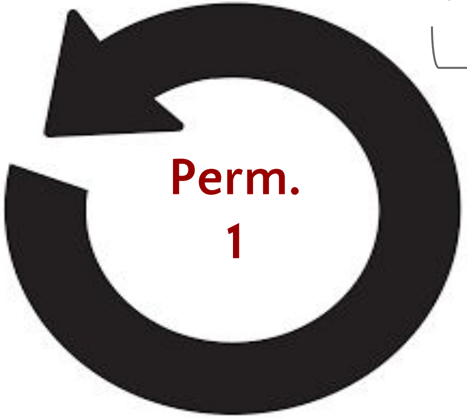
Statistics: between-subject



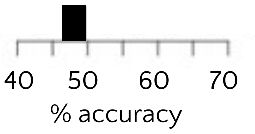
Statistics: between-subject



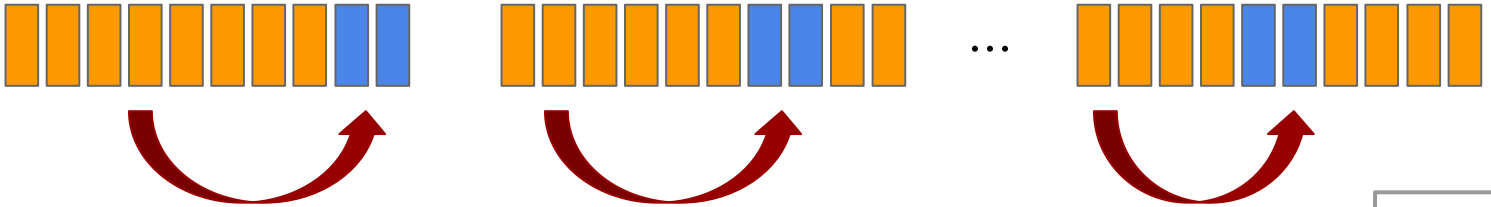
Simulated null-distribution!



Permuted performance (averaged over folds):
48%



Statistics: between-subject

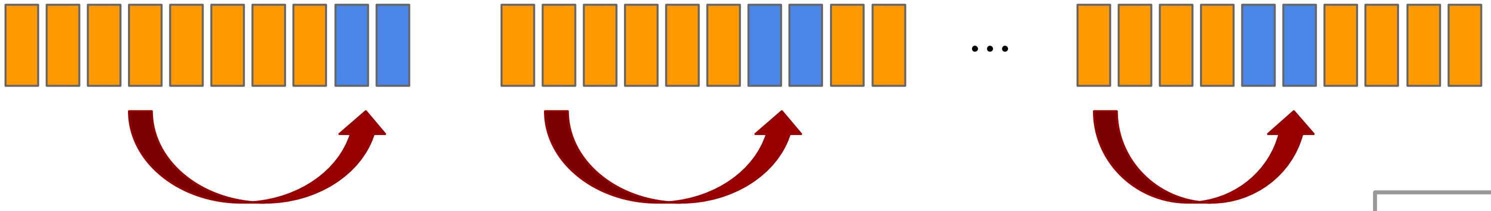


51% 53% 49%

Permuted performance (averaged over folds):
52%

Simulated null-distribution!

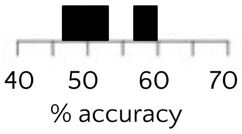
Statistics: between-subject



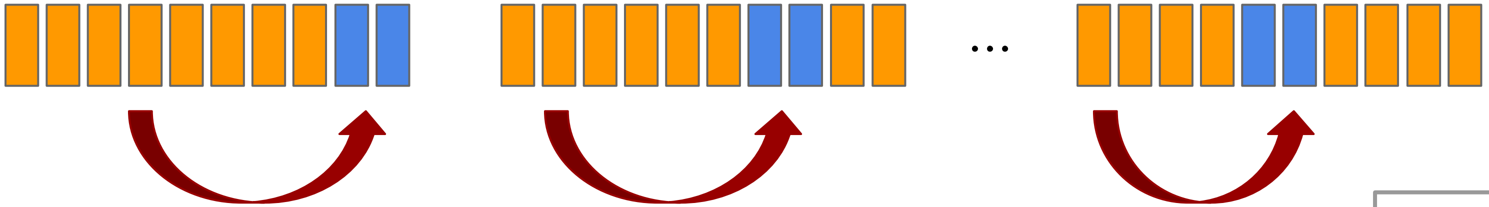
62%   55%   53%

Simulated null-distribution!

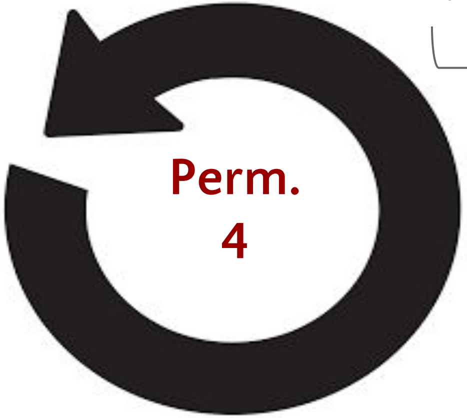
Permuted performance (averaged over folds):
57%



Statistics: between-subject

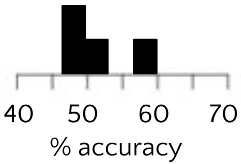


51% 42% 47%

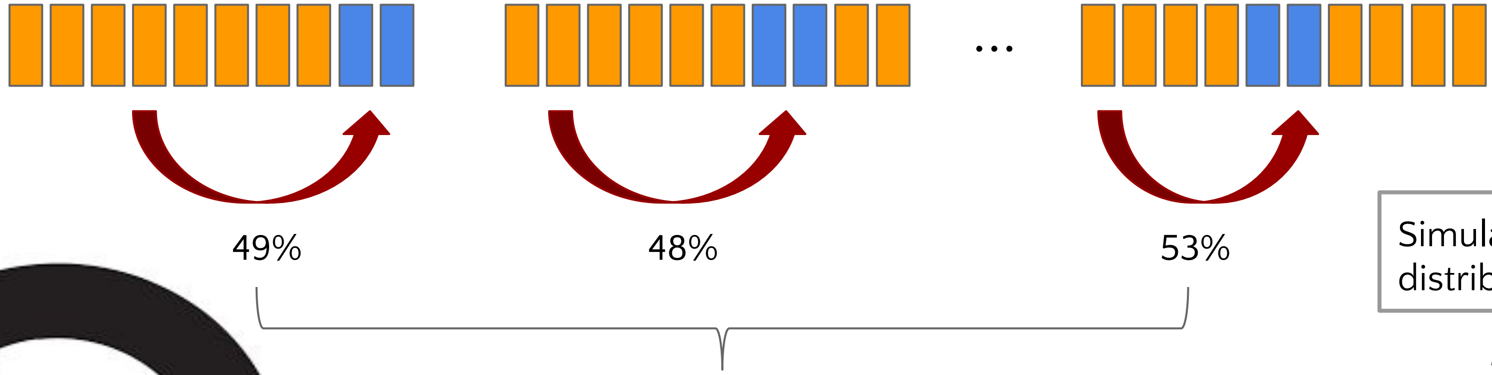


Simulated null-distribution!

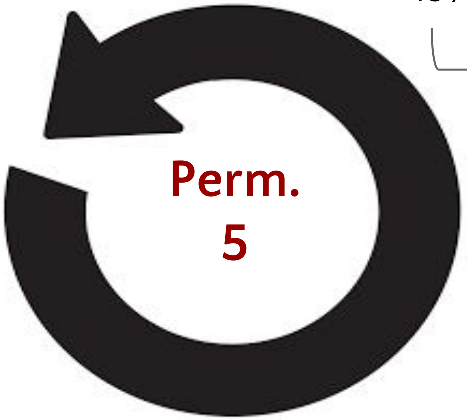
Permuted performance (averaged over folds):
48%



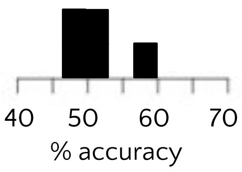
Statistics: between-subject



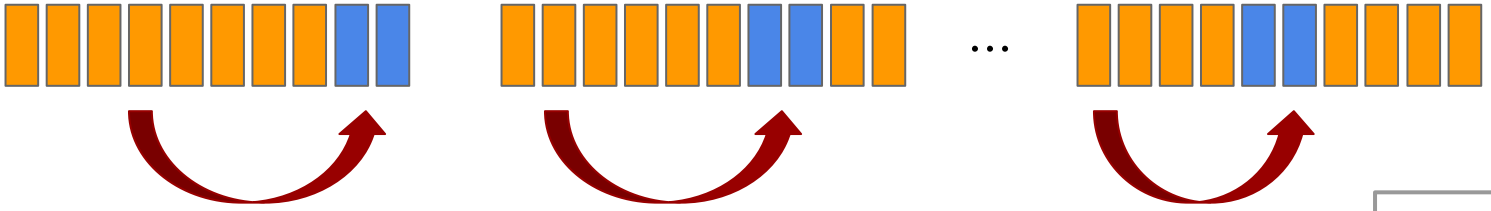
Simulated null-distribution!



Permuted performance (averaged over folds):
50%



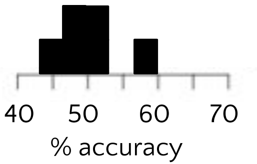
Statistics: between-subject



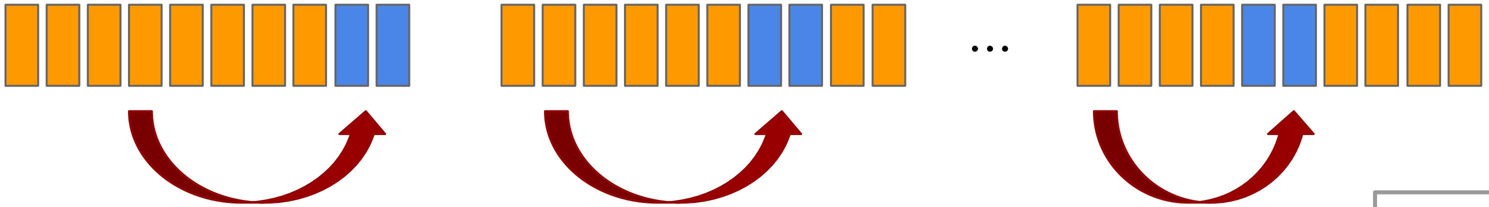
41% 42% … 49%

Simulated null-distribution!

Permuted performance (averaged over folds):
45%



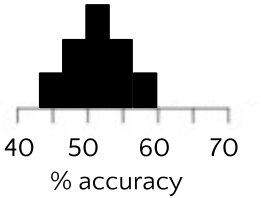
Statistics: between-subject



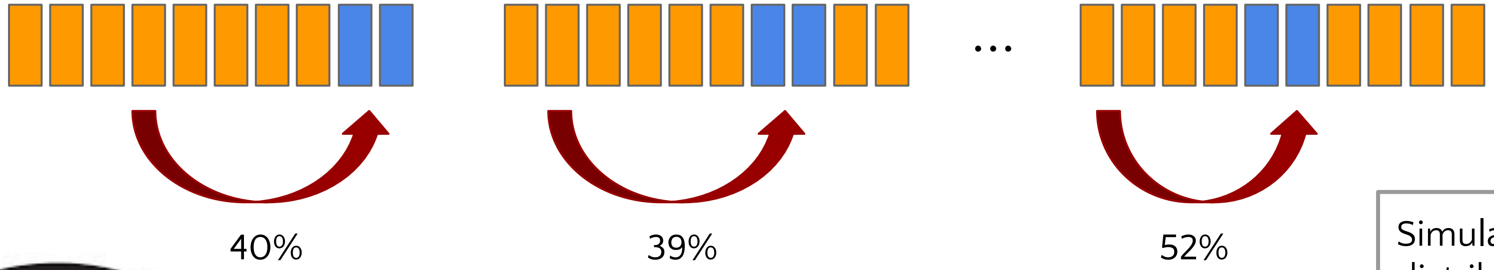
46% 53% ... 48%

Permuted performance (averaged over folds):
48%

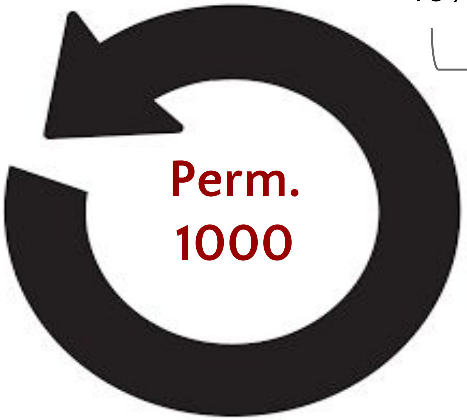
Simulated null-distribution!



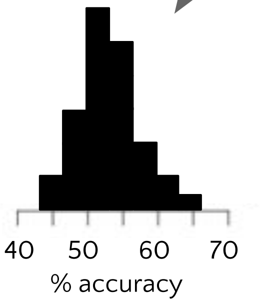
Statistics: between-subject



Permuted performance (averaged over folds):
43%



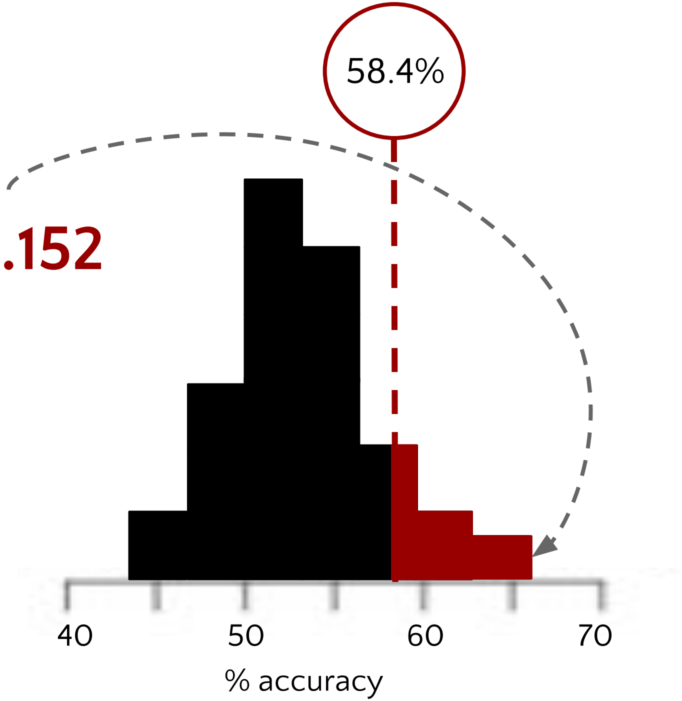
Simulated null-distribution!



Statistics: between-subject

"Non-parametric" **p-value** =

$$\frac{\sum(\text{null-scores} > \text{observed-score})}{\text{Number of permutations}} = \frac{152}{1000} = \mathbf{0.152}$$



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

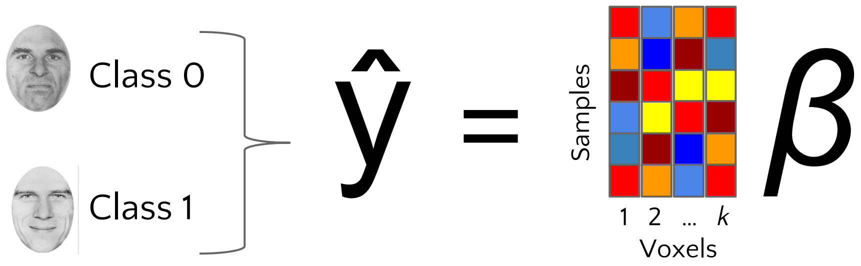
- Often, researchers (read: reviewers) ask:

  "Which features are important
  for which class?"

- What they want:
- Intrinsic need for blobs?

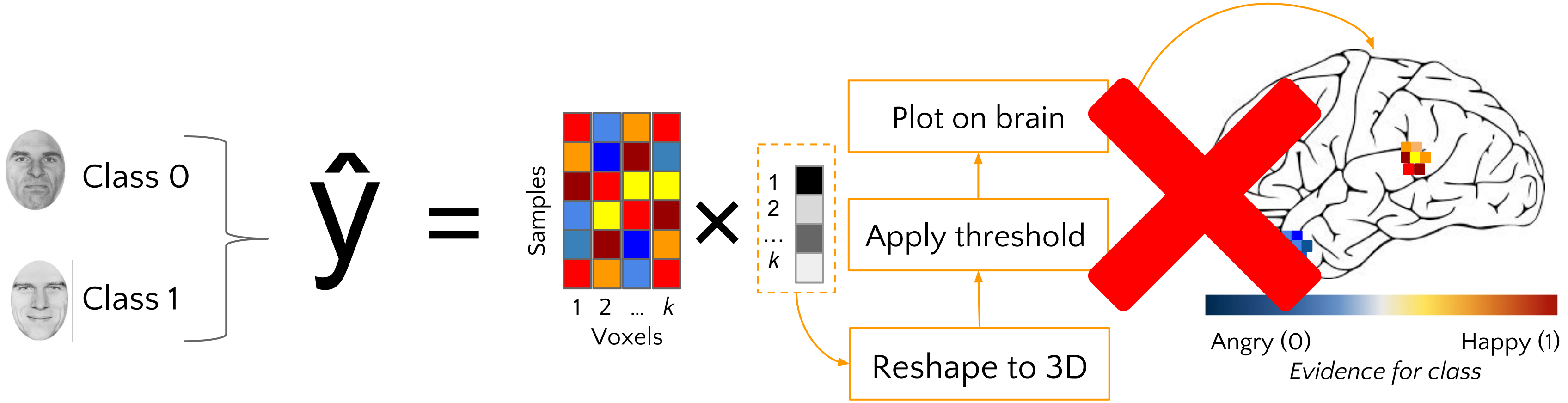
Weight mapping

- <u>Technically</u>, weights (β) in *linear* models are interpretable: higher = more important
  - Also, the following is assumed (but wrong):
    - 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



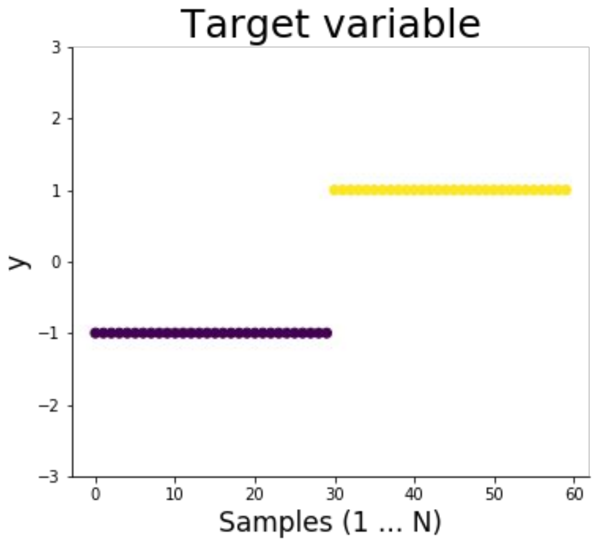
Weight mapping

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

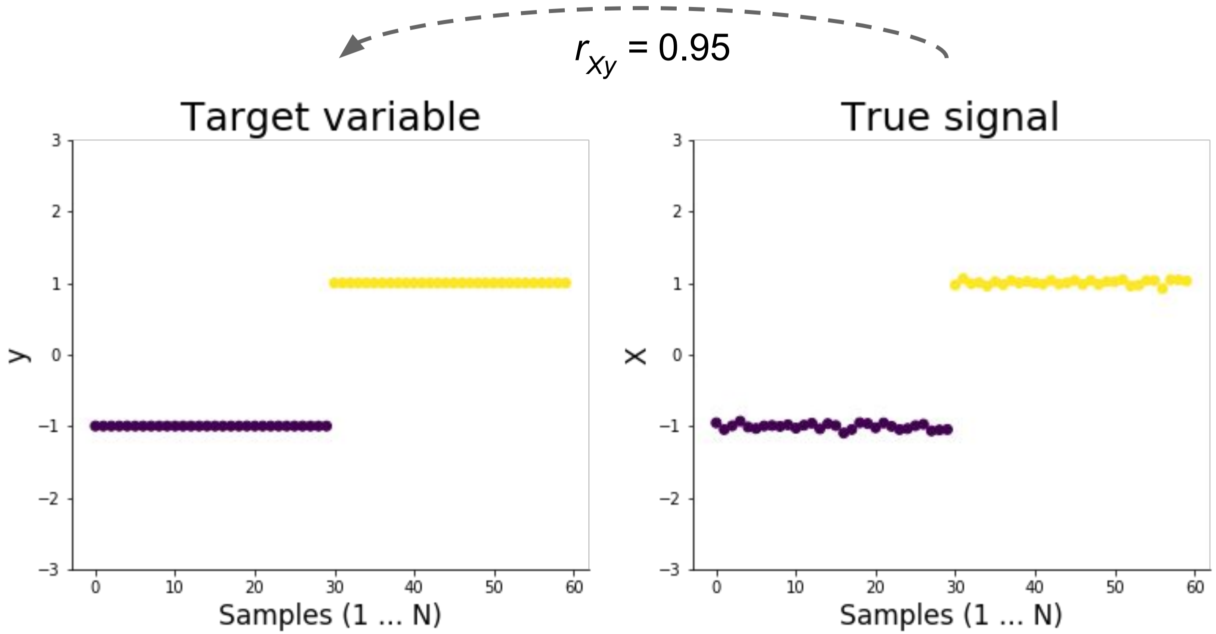
… *A widespread misconception about multivariate classifier weight vectors is that (the brain regions corresponding to) measurement channels* with large weights are strongly related to the experimental condition.* In fact, such conclusions can be unjustified.

*voxels

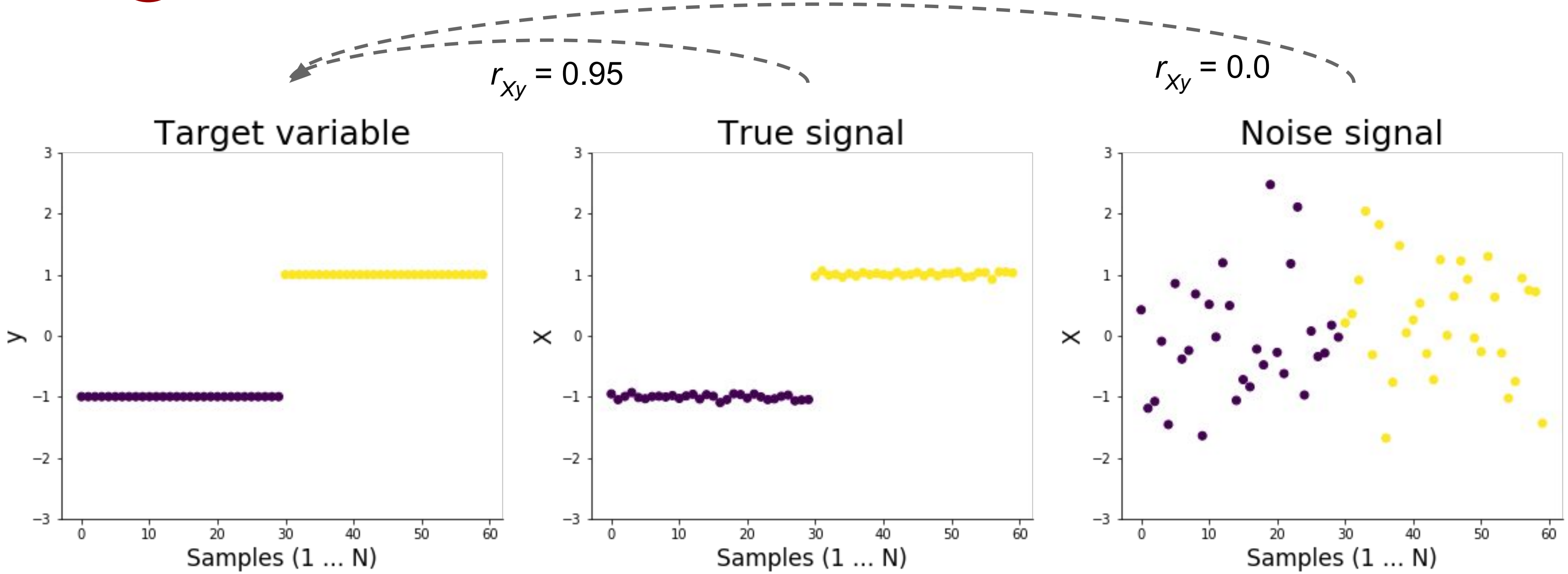
Weight mapping



Weight mapping



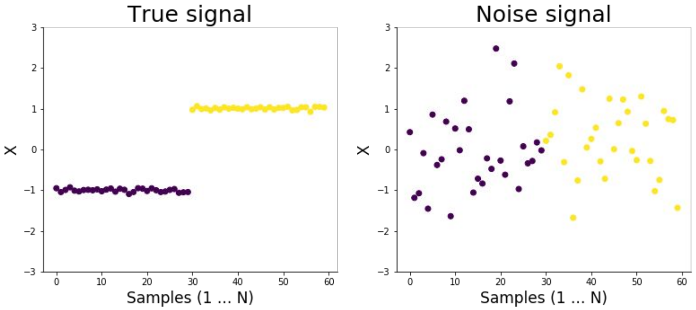
Weight mapping

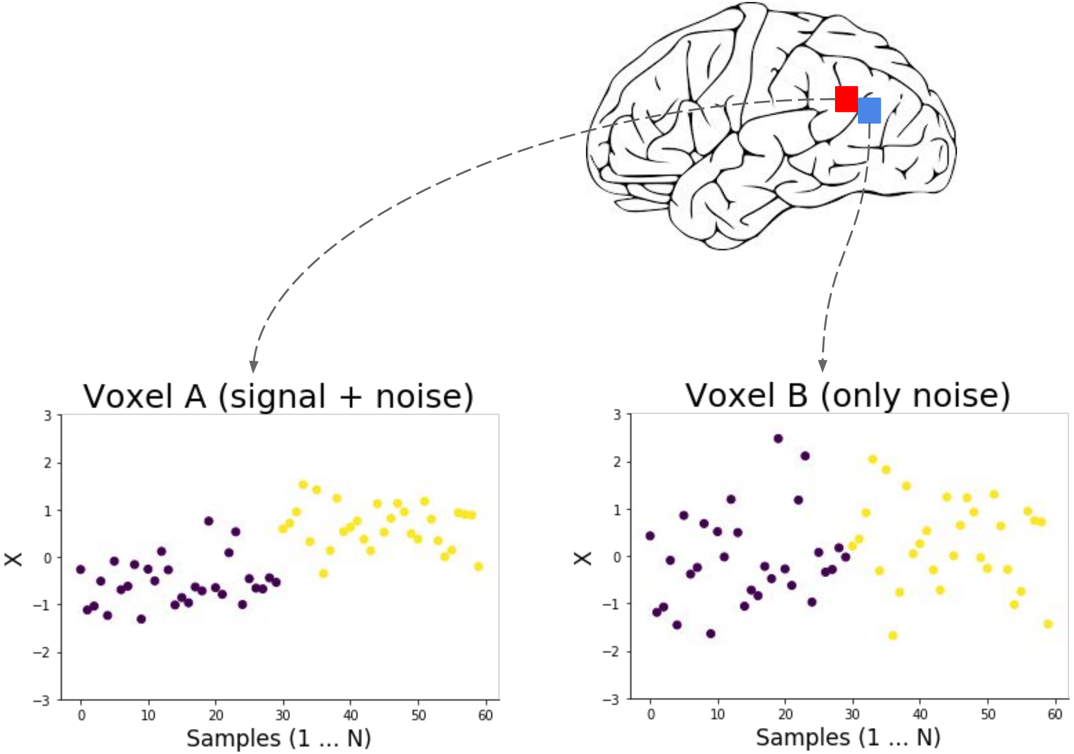


What could you do instead if somebody asks for pretty brain pictures related to your decoding analysis?

$$\hat{y} = X_A\beta_A + X_B\beta_B$$

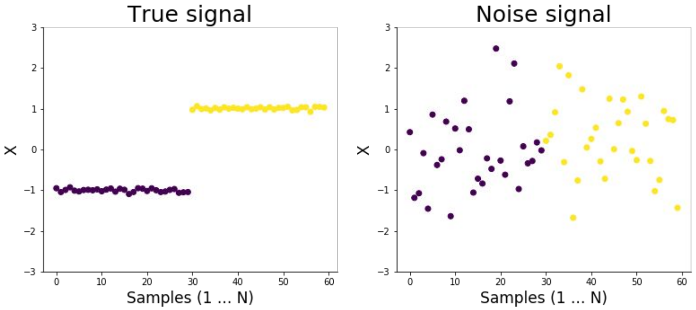
Weight mapping

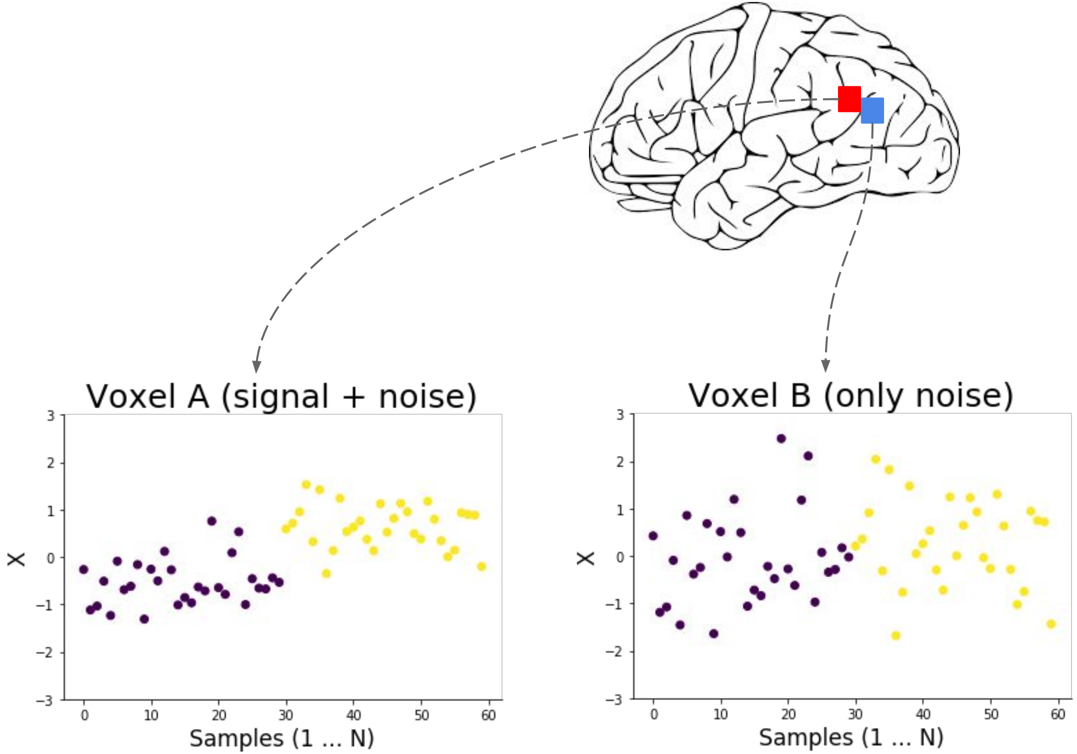




$$\hat{y} = X_A \cdot 1 + X_B \cdot -1$$

Weight mapping





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
- Features with high weights may be ***completely* unrelated** to the target (y)!

Weight mapping

- Just do a mass-univariate analysis!
  - MVPA has a different goal
- 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

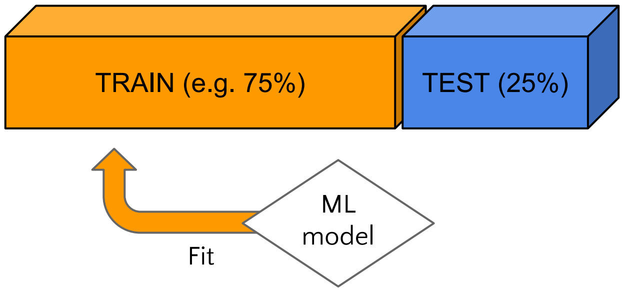
Feature selection (voxel subset)

Reduce the amount of features

Feature extraction (voxels → components)

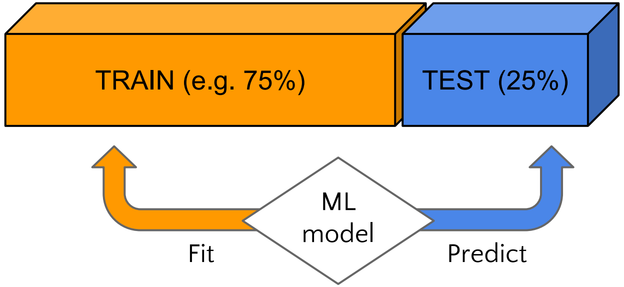
Summary

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

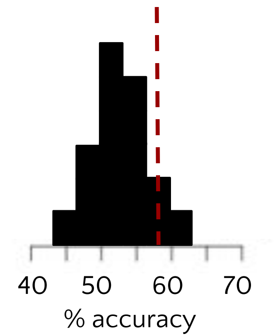
0. Pattern extraction & preparation
1. Partitioning train/test
2. Feature selection/extraction
3. **Model fitting (TRAIN)**
4. **Model prediction (TEST)**
5. Statistical test of performance
6. Optional: plot weights



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

Recommended literature

- Abraham et al. (2014): about the scikit-learn package for decoding analyses
- Pereira et al. (2009): tutorial-style paper about decoding analyses
- Ritchie et al. (2017): what are you actually measuring with decoding?
    - Highly recommended!

Thanks!