Homework 1 Questions

Question 1: You are asked to train a classifier to predict the probability that a patient has cancer, given his cellular images. Which classifier are you most likely to use?

a. kNN

b. SVM

c. GNB

Question 2: Let there be 900 images without cancer and 100 with cancer in the previous example. A given classifier achieves 85% accuracy on the training set. Is this a good classifier?

a. Yes

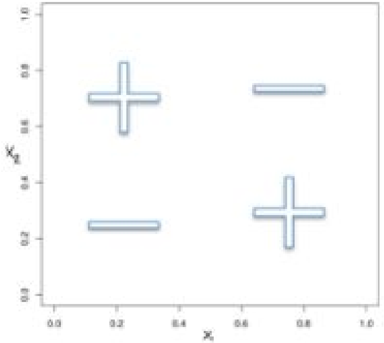
b. No

Question 3: Which of these classifiers will be the least likely to classify the following data points correctly?

a. kNN

b. SVM

c. GNB



Model selection and evaluation of results

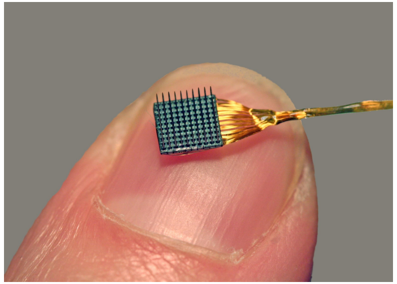
06/10/2016

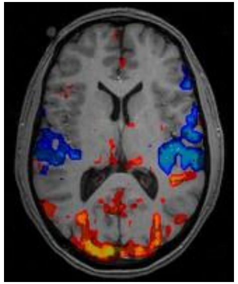
Mariya Toneva
mariya@cmu.edu

Some figures derived from slides by Alona Fyshe

How can ML help neuroscientists?

- ❑ Deal with large number of sensors/recording sites

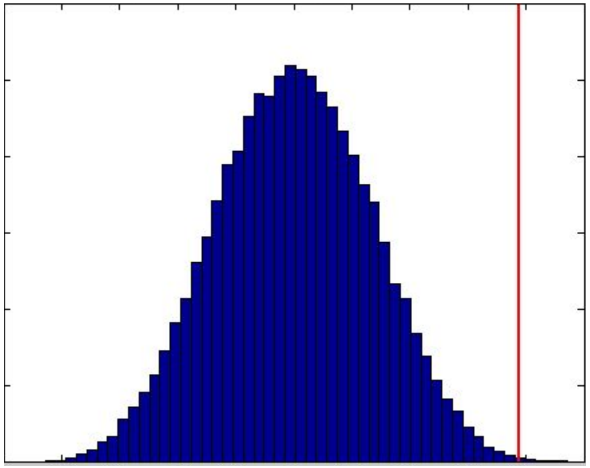




- ❑ investigate high-dimensional representations
  - ❑ classification (what does this high-dimensional data represent?)
  - ❑ regression (how does it represent it? can we predict a different representation?)
  - ❑ **model selection (what model would best describe this high dimensional data?)**

How can ML help neuroscientists?

- **Evaluate results**
  - **nearly assumption-free significance testing (are the results significantly different from chance?)**



Today: model selection and evaluation

- Model selection
  - overfitting
  - cross validation
  - feature selection
  - regularization
- Evaluation of results
  - significance testing
    - permutation test
    - multiple comparison corrections

Goal of most ML methods: generalize well to new data

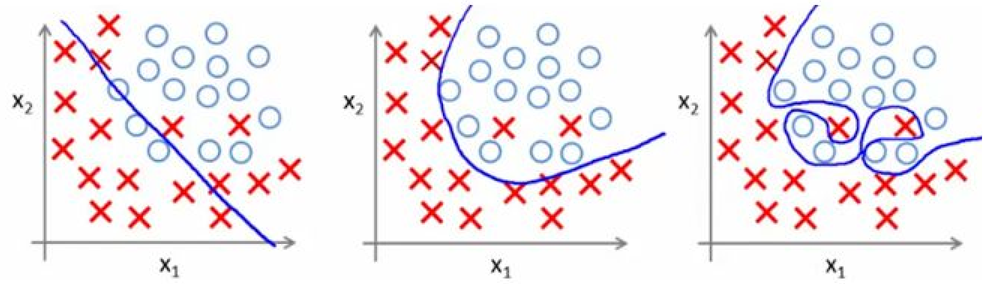
- Recall 3 main steps for most classifiers: assume, train, test

Goal of most ML methods: generalize well to new data

- Recall 3 main steps for most classifiers: assume, train, test
- Goal: train a model that is able to perform well on new test data = generalize from the training data to the test data

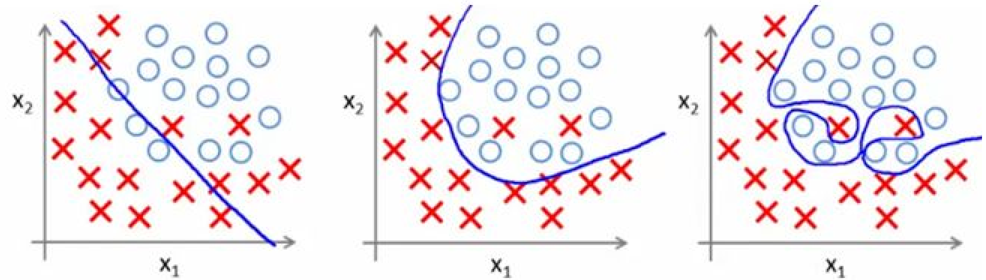
Goal of most ML methods: generalize well to new data

- Recall 3 main steps for most classifiers: assume, train, test
- Goal: train a model that is able to perform well on new test data = generalize from the training data to the test data
- But multiple models are possible for the same data. How do we choose the best one?



Goal of most ML methods: generalize well to new data

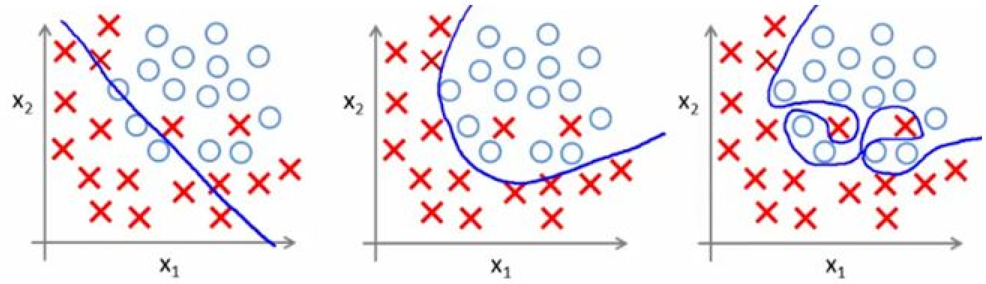
- Recall 3 main steps for most classifiers: assume, train, test
- Goal: train a model that is able to perform well on new test data = generalize from the training data to the test data
- But multiple models are possible for the same data. How do we choose the best one?



- So we want to select models that do not underfit or overfit to the training data

Goal of most ML methods: generalize well to new data

- Recall 3 main steps for most classifiers: assume, train, test
- Goal: train a model that is able to perform well on new test data = generalize from the training data to the test data
- But multiple models are possible for the same data. How do we choose the best one?



- So we want to select models that do not underfit or overfit to the training data
  - underfitting is generally easier to detect than overfitting because it performs poorly on the training set

Problem: overfitting to training data

- Overfitting occurs when $error_{train} < error_{test}$

Problem: overfitting to training data

- Overfitting occurs when $error_{train} < error_{test}$
- Amount of overfitting = $error_{train} - error_{test}$

Problem: overfitting to training data

- Overfitting occurs when $error_{train} < error_{test}$
- Amount of overfitting = $error_{train} - error_{test}$

- What happens when we test on the same data we trained? Is there overfitting?

Problem: overfitting to training data

- Overfitting occurs when $error_{train}$ < $error_{test}$
- Amount of overfitting = $error_{train}$ - $error_{test}$

- What happens when we test on the same data we trained? Is there overfitting?
    - Demo -> split iris data set in half; train on half of the data, test on same half. Then, test on other half, compare accuracy

Problem: overfitting to training data

- Overfitting occurs when $error_{train} < error_{test}$
- Amount of overfitting = $error_{train} - error_{test}$

- What happens when we test on the same data we trained? Is there overfitting?
    - Demo -> split iris data set in half; train on half of the data, test on same half. Then, test on other half, compare accuracy
    - Yes! But we just can’t evaluate how much overfitting there is if we test on the training set. Ignorance is not bliss.

Why does overfitting happen?

- The test data isn't the same as the training data

Why does overfitting happen?

- The test data isn't the same as the training data
  - we evaluate performance on different data (test data) from the one we used to estimate parameters (training data)
    - some overfitting is natural

Why does overfitting happen?

- The test data isn't the same as the training data
  - we evaluate performance on different data (test data) from the one we used to estimate parameters (training data)
    - some overfitting is natural
- Few training examples for certain parameters

Why does overfitting happen?

- The test data isn't the same as the training data
  - we evaluate performance on different data (test data) from the one we used to estimate parameters (training data)
    - some overfitting is natural
- Few training examples for certain parameters
  - model learns to care about specific noise in those training examples, which is not generalizable to new data

Why does overfitting happen?

- The test data isn't the same as the training data
  - we evaluate performance on different data (test data) from the one we used to estimate parameters (training data)
    - some overfitting is natural
- Few training examples for certain parameters
  - model learns to care about specific noise in those training examples, which is not generalizable to new data
    - example: GNB, learning mus, sigmas from 2 repetitions -> overfit to noise vs learning from 12 repetitions

Why does overfitting happen?

- The test data isn't the same as the training data
  - we evaluate performance on different data (test data) from the one we used to estimate parameters (training data)
    - some overfitting is natural
- Few training examples for certain parameters
  - model learns to care about specific noise in those training examples, which is not generalizable to new data
    - example: GNB, learning mus, sigmas from 2 repetitions -> overfit to noise vs learning from 12 repetitions
  - the more complex a model is, the more likely it is to suffer from overfitting

Ways to prevent overfitting

- Select the model that performs best on a third data set that is distinct from the training and testing data sets

Ways to prevent overfitting

- Select the model that performs best on a third data set that is distinct from the training and testing data sets
  - called the “validation data set”

Ways to prevent overfitting

- Select the model that performs best on a third data set that is distinct from the training and testing data sets
  - called the “validation data set”
  - trade off between size of training and validation data sets
    - cross-validation

Ways to prevent overfitting

- Select the model that performs best on a third data set that is distinct from the training and testing data sets
  - called the “validation data set”
  - trade off between size of training and validation data sets
    - cross-validation
- Remove features that are irrelevant for a classification task
  - feature selection

Ways to prevent overfitting

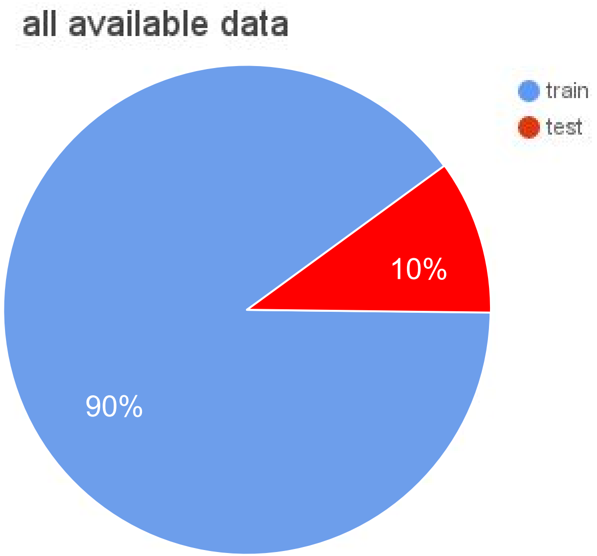
- Select the model that performs best on a third data set that is distinct from the training and testing data sets
  - called the “validation data set”
  - trade off between size of training and validation data sets
    - cross-validation
- Remove features that are irrelevant for a classification task
  - feature selection
- Explicitly penalize complex models because we know they are prone to overfitting
  - regularization

Ways to prevent overfitting

- Select the model that performs best on a third data set that is distinct from the training and testing data sets
  - called the “validation data set”
  - trade off between size of training and validation data sets
    - **cross-validation**
- Remove features that are irrelevant for a classification task
  - feature selection
- Explicitly penalize complex models because we know they are prone to overfitting
  - regularization

Cross-validation as a measure of generalizability

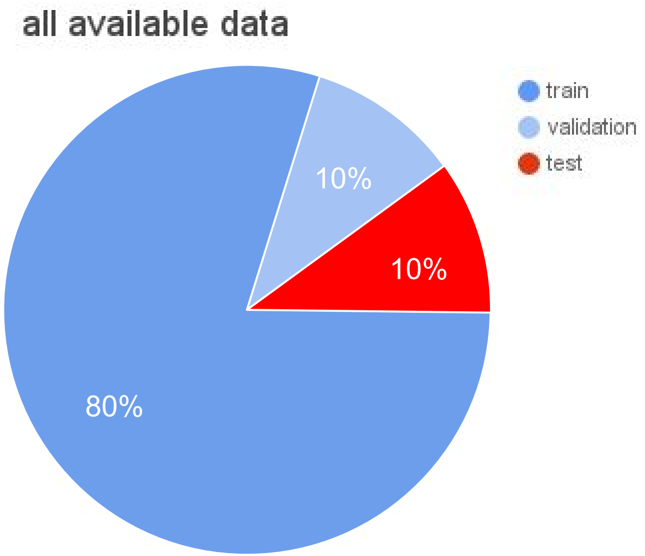
Step 1: split data into a training and testing sets



Cross-validation as a measure of generalizability

Step 1: split data into a training and testing sets

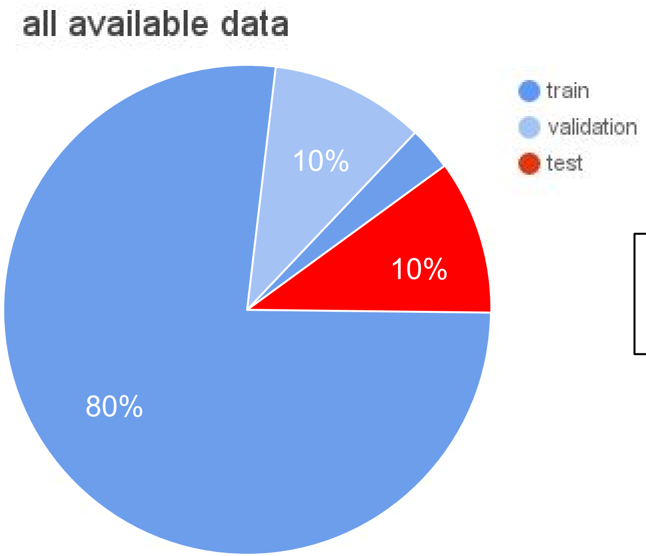
Step 2: run 1 cross-validation fold
- ❑ split original **training data** into a training and validation sets
- ❑ train on training data
- ❑ test on validation data
- ❑ record error



Cross-validation as a measure of generalizability

Step 1: split data into a training and testing sets

Step 2: run 1 cross-validation fold
- ❏ split original **training data** into a training and validation sets
- ❏ train on training data
- ❏ test on validation data
- ❏ record error

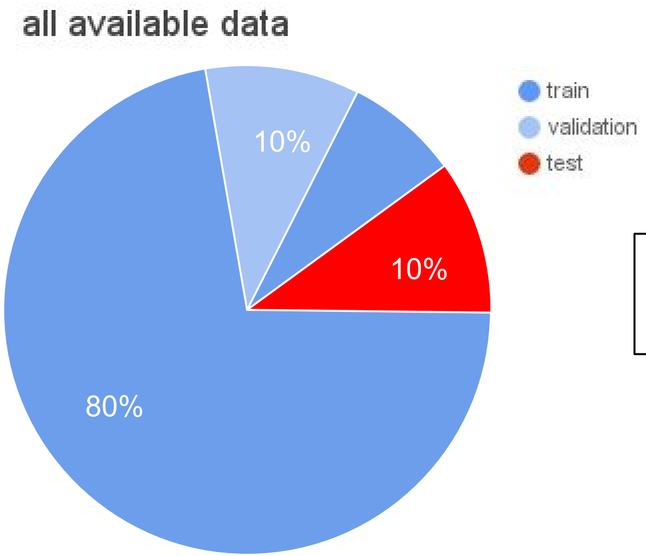


Step 3: repeat step 2 with a different split

Cross-validation as a measure of generalizability

Step 1: split data into a training and testing sets

Step 2: run 1 cross-validation fold
- ❑ split original **training data** into a training and validation sets
- ❑ train on training data
- ❑ test on validation data
- ❑ record error



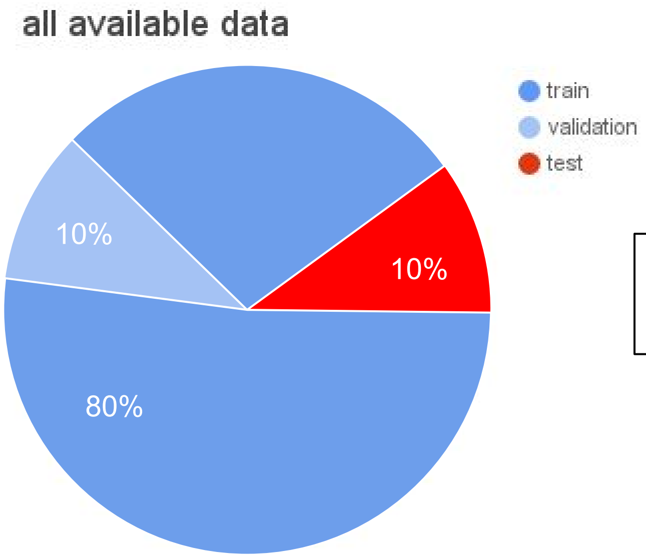
Step 3: repeat step 2 with a different split

Cross-validation as a measure of generalizability

Step 1: split data into a training and testing sets

Step 2: run 1 cross-validation fold

- ❏ split original **training data** into a training and validation sets
- ❏ train on training data
- ❏ test on validation data
- ❏ record error



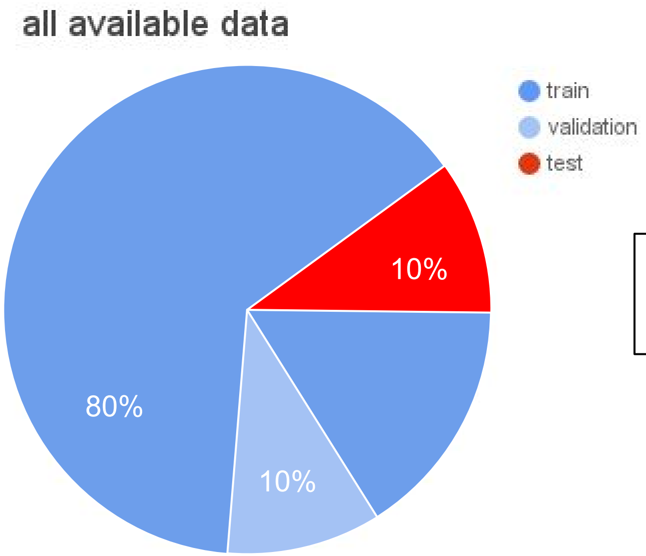
Step 3: repeat step 2 with a different split

Cross-validation as a measure of generalizability

Step 1: split data into a training and testing sets

Step 2: run 1 cross-validation fold

- ❏ split original **training data** into a training and validation sets
- ❏ train on training data
- ❏ test on validation data
- ❏ record error

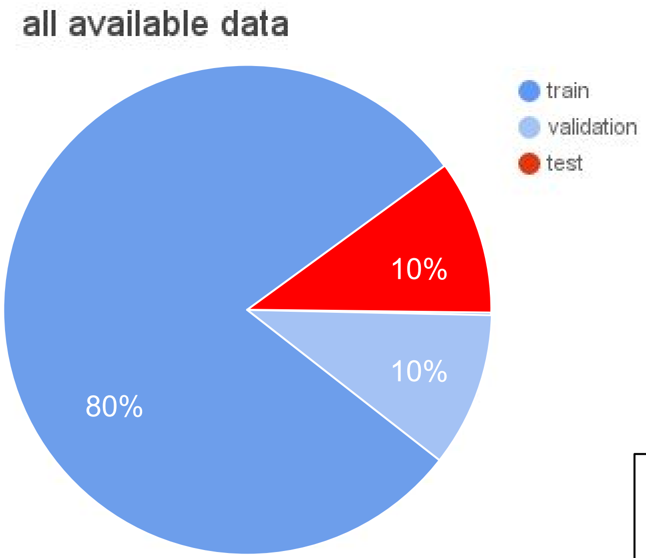


Step 3: repeat step 2 with a different split

Cross-validation as a measure of generalizability

Step 1: split data into a training and testing sets

Step 2: run 1 cross-validation fold
- ❑ split original **training data** into a training and validation sets
- ❑ train on training data
- ❑ test on validation data
- ❑ record error



Step 3: repeat step 2 with a different split

Step 4: average errors from all cross-validation folds = cross-validation (CV)

Cross-validation as a measure of generalizability

- Perform cross-validation for each model that you are considering

Cross-validation as a measure of generalizability

- Perform cross-validation for each model that you are considering
- The model which has the lowest CV error generalizes the best

Cross-validation as a measure of generalizability

- Perform cross-validation for each model that you are considering
- The model which has the lowest CV error generalizes the best
- Select this model for evaluation on the real test set

Cross-validation as a measure of generalizability

- Perform cross-validation for each model that you are considering
- The model which has the lowest CV error generalizes the best
- Select this model for evaluation on the real test set
- Note that we never used the final test set in the model selection stage!

How to split the data?

- Train-test split is less important

How to split the data?

- Train-test split is less important
  - The more training data, the better; but still need enough data to evaluate the selected model

How to split the data?

- Train-test split is less important
  - The more training data, the better; but still need enough data to evaluate the selected model
  - Generally, 80/20 or 90/10 splits are acceptable for train/test

How to split the data?

- Train-test split is less important
  - The more training data, the better; but still need enough data to evaluate the selected model
  - Generally, 80/20 or 90/10 splits are acceptable for train/test
- Many ways to divide the original training data into CV training set and a validation set

How to split the data?

- Train-test split is less important
  - The more training data, the better; but still need enough data to evaluate the selected model
  - Generally, 80/20 or 90/10 splits are acceptable for train/test
- Many ways to divide the original training data into CV training set and a validation set
  - exhaustive CV: all possible splits of train and validation

How to split the data?

- Train-test split is less important
  - The more training data, the better; but still need enough data to evaluate the selected model
  - Generally, 80/20 or 90/10 splits are acceptable for train/test
- Many ways to divide the original training data into CV training set and a validation set
  - exhaustive CV: all possible splits of train and validation
    - Leave-p-out CV; LOOCV is a special case when p = 1

How to split the data?

- Train-test split is less important
  - The more training data, the better; but still need enough data to evaluate the selected model
  - Generally, 80/20 or 90/10 splits are acceptable for train/test
- Many ways to divide the original training data into CV training set and a validation set
  - exhaustive CV: all possible splits of train and validation
    - Leave-p-out CV; LOOCV is a special case when $p = 1$
  - non-exhaustive CV: not all possible splits

How to split the data?

- Train-test split is less important
  - The more training data, the better; but still need enough data to evaluate the selected model
  - Generally, 80/20 or 90/10 splits are acceptable for train/test
- Many ways to divide the original training data into CV training set and a validation set
  - exhaustive CV: all possible splits of train and validation
    - Leave-p-out CV; LOOCV is a special case when p = 1
  - non-exhaustive CV: not all possible splits
    - k-fold-CV = randomly partition training data into k equal sized subsets; run k folds of CV in which each subset is used as validation exactly once

How to split the data?

- Train-test split is less important
  - The more training data, the better; but still need enough data to evaluate the selected model
  - Generally, 80/20 or 90/10 splits are acceptable for train/test
- Many ways to divide the original training data into CV training set and a validation set
  - exhaustive CV: all possible splits of train and validation
    - Leave-p-out CV; LOOCV is a special case when p = 1
  - non-exhaustive CV: not all possible splits
    - k-fold-CV = randomly partition training data into k equal sized subsets; run k folds of CV in which each subset is used as validation exactly once
      - k = 10 commonly used

How to split the data?

- Train-test split is less important
    - The more training data, the better; but still need enough data to evaluate the selected model
    - Generally, 80/20 or 90/10 splits are acceptable for train/test
- Many ways to divide the original training data into CV training set and a validation set
    - exhaustive CV: all possible splits of train and validation
        - Leave-p-out CV; LOOCV is a special case when p = 1
    - non-exhaustive CV: not all possible splits
        - k-fold-CV = randomly partition training data into k equal sized subsets; run k folds of CV in which each subset is used as validation exactly once
            - k = 10 commonly used
            - What happens when k is equal to number of samples in the training data?

How to split the data?

- Train-test split is less important
  - The more training data, the better; but still need enough data to evaluate the selected model
  - Generally, 80/20 or 90/10 splits are acceptable for train/test
- Many ways to divide the original training data into CV training set and a validation set
  - exhaustive CV: all possible splits of train and validation
    - Leave-p-out CV; LOOCV is a special case when p = 1
  - non-exhaustive CV: not all possible splits
    - k-fold-CV = randomly partition training data into k equal sized subsets; run k folds of CV in which each subset is used as validation exactly once
      - k = 10 commonly used
      - What happens when k is equal to number of samples in the training data? LOOCV!

How to split the data?

- Train-test split is less important
  - The more training data, the better; but still need enough data to evaluate the selected model
  - Generally, 80/20 or 90/10 splits are acceptable for train/test
- Many ways to divide the original training data into CV training set and a validation set
  - exhaustive CV: all possible splits of train and validation
    - Leave-p-out CV; LOOCV is a special case when p = 1
  - non-exhaustive CV: not all possible splits
    - k-fold-CV = randomly partition training data into k equal sized subsets; run k folds of CV in which each subset is used as validation exactly once
      - k = 10 commonly used
      - What happens when k is equal to number of samples in the training data? LOOCV!
    - stratified k-fold-CV = split such that each fold has ~same proportions of classes

How to split the data?

- Train-test split is less important
  - The more training data, the better; but still need enough data to evaluate the selected model
  - Generally, 80/20 or 90/10 splits are acceptable for train/test
- Many ways to divide the original training data into CV training set and a validation set
  - exhaustive CV: all possible splits of train and validation
    - Leave-p-out CV; LOOCV is a special case when p = 1
  - non-exhaustive CV: not all possible splits
    - k-fold-CV = randomly partition training data into k equal sized subsets; run k folds of CV in which each subset is used as validation exactly once
      - k = 10 commonly used
      - What happens when k is equal to number of samples in the training data? LOOCV!
    - stratified k-fold-CV = split such that each fold has ~same proportions of classes
  - demo!

Cross-validation takeaways

- CV can be used to select the most generalizable model

Cross-validation takeaways

- CV can be used to select the most generalizable model
- There is a trade-off between speed and accuracy in splitting the training data into validation and CV training sets

Ways to prevent overfitting

- Select the model that performs best on a third data set that is distinct from the training and testing data sets
  - called the “validation data set”
  - trade off between size of training and validation data sets
    - cross-validation
- Remove features that are irrelevant for a classification task
  - **feature selection**
- Explicitly penalize complex models because we know they are prone to overfitting
  - regularization

Feature selection as a way to reduce overfitting in complex models

- In high-dimensional data, the number of features (e.g. voxels, sensors, etc.) is large, but there may only be a small number of features that are “relevant” to the learning task

Feature selection as a way to reduce overfitting in complex models

- In high-dimensional data, the number of features (e.g. voxels, sensors, etc.) is large, but there may only be a small number of features that are “relevant” to the learning task
- The learned model may overfit to the large number of irrelevant features unless the training set is fairly large

Feature selection as a way to reduce overfitting in complex models

- In high-dimensional data, the number of features (e.g. voxels, sensors, etc.) is large, but there may only be a small number of features that are “relevant” to the learning task
- The learned model may overfit to the large number of irrelevant features unless the training set is fairly large
- So we can remove some irrelevant features!

Which features are irrelevant?

- Want to compute a score for each feature x_i that tells us how informative this feature is about the class labels y

Which features are irrelevant?

- Want to compute a score for each feature x_i that tells us how informative this feature is about the class labels y
- Mutual information can give us such a score

Which features are irrelevant?

- Want to compute a score for each feature x_i that tells us how informative this feature is about the class labels y
- Mutual information can give us such a score
  - Score each feature by its relative probability with respect to the class labels

Which features are irrelevant?

- Want to compute a score for each feature x_i that tells us how informative this feature is about the class labels *y*
- Mutual information can give us such a score
  - Score each feature by its relative probability with respect to the class labels
  - Example for binary *x*, *y*:

$$MI(x_i, y) = \sum_{x_i \in \{0,1\}} \sum_{y \in \{0,1\}} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i) p(y)}$$

Which features are irrelevant?

- Want to compute a score for each feature x_i that tells us how informative this feature is about the class labels y
- Mutual information can give us such a score
  - Score each feature by its relative probability with respect to the class labels
  - Example for binary *x*, *y*:

$$MI(x_i, y) = \sum_{x_i \in \{0,1\}} \sum_{y \in \{0,1\}} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)}$$

  - What happens when x_i and y are independent?

Which features are irrelevant?

- Want to compute a score for each feature x_i that tells us how informative this feature is about the class labels y
- Mutual information can give us such a score
  - Score each feature by its relative probability with respect to the class labels
  - Example for binary *x*, *y*:

$$MI(x_i, y) = \sum_{x_i \in \{0,1\}} \sum_{y \in \{0,1\}} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)}$$

  - What happens when x_i and y are independent? $P(x_i,y) = P(x_i)P(y)$

Which features are irrelevant?

- Want to compute a score for each feature x_i that tells us how informative this feature is about the class labels y
- Mutual information can give us such a score
  - Score each feature by its relative probability with respect to the class labels
  - Example for binary *x*, *y*:

$$MI(x_i, y) = \sum_{x_i \in \{0,1\}} \sum_{y \in \{0,1\}} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)}$$

  - What happens when x_i and y are independent? $P(x_i,y) = P(x_i)P(y) \Rightarrow MI(x_i,y) = 0$

Feature selection takeaways

- Feature selection is a way to choose only those features that are relevant for a certain task

Feature selection takeaways

- Feature selection is a way to choose only those features that are relevant for a certain task
- Mutual information is one way to select informative features

Ways to prevent overfitting

- Select the model that performs best on a third data set that is distinct from the training and testing data sets
  - called the “validation data set”
  - trade off between size of training and validation data sets
    - cross-validation
- Remove features that are irrelevant for a classification task
  - feature selection
- Explicitly penalize complex models because we know they are prone to overfitting
  - **regularization**

What is regularization? Recall linear regression

$$W = \underset{W}{\operatorname{argmin}} \sum_{m} (y^m - f(x^m; W))^2$$

where Y = labels, X = data instances, and W = weights

What is regularization? Recall linear regression

$$W = \underset{W}{\operatorname{argmin}} \sum_{m} (y^m - f(x^m; W))^2$$

where Y = labels, X = data instances, and W = weights

- Problem: when the data is very high-dimensional, W is large => since every element of matrix W is a parameter, the model has many parameters to be estimated => complex model => prone to overfitting

What is regularization? Recall linear regression

$$W = \underset{W}{\operatorname{argmin}} \sum_{m} (y^m - f(x^m; W))^2$$

where Y = labels, X = data instances, and W = weights

- Problem: when the data is very high-dimensional, W is large => since every element of matrix W is a parameter, the model has many parameters to be estimated => complex model => prone to overfitting
- Solution: limit the complexity of the model by reducing the expressiveness of W

What is regularization? Recall linear regression

$$W = \underset{W}{\operatorname{argmin}} \sum_m (y^m - f(x^m; W))^2 + \lambda R(W)$$

where Y = labels, X = data instances, and W = weights

- Problem: when the data is very high-dimensional, W is large => since every element of matrix W is a parameter, the model has many parameters to be estimated => complex model => prone to overfitting
- Solution: limit the complexity of the model by reducing the expressiveness of W
  - because we are looking for the W that minimizes a certain task, the addition of some function R of W directly penalizes any big elements of W

What is regularization? Recall linear regression

$$W = \underset{W}{\operatorname{argmin}} \sum_m (y^m - f(x^m; W))^2 + \lambda R(W)$$

where Y = labels, X = data instances, and W = weights

- ❏ Problem: when the data is very high-dimensional, W is large => since every element of matrix W is a parameter, the model has many parameters to be estimated => complex model => prone to overfitting
- ❏ Solution: limit the complexity of the model by reducing the expressiveness of W
  - ❏ because we are looking for the W that minimizes a certain task, the addition of some function R of W directly penalizes any big elements of W
  - ❏ λ is a number that can vary between data sets => determine best value for λ through cross-validation

What is regularization? Recall linear regression

$$W = \underset{W}{\operatorname{argmin}} \sum_m (y^m - f(x^m; W))^2 + \lambda R(W)$$

where Y = labels, X = data instances, and W = weights

- Problem: when the data is very high-dimensional, W is large => since every element of matrix W is a parameter, the model has many parameters to be estimated => complex model => prone to overfitting
- Solution: limit the complexity of the model by reducing the expressiveness of W
  - because we are looking for the W that minimizes a certain task, the addition of some function R of W directly penalizes any big elements of W
  - λ is a number that can vary between data sets => determine best value for λ through cross-validation
  - What happens if λ is negative?

What is regularization? Recall linear regression

$$W = \underset{W}{\operatorname{argmin}} \sum_m (y^m - f(x^m; W))^2 + \lambda R(W)$$

where Y = labels, X = data instances, and W = weights

- Problem: when the data is very high-dimensional, W is large => since every element of matrix W is a parameter, the model has many parameters to be estimated => complex model => prone to overfitting
- Solution: limit the complexity of the model by reducing the expressiveness of W
  - because we are looking for the W that minimizes a certain task, the addition of some function R of W directly penalizes any big elements of W
  - λ is a number that can vary between data sets => determine best value for λ through cross-validation
  - What happens if λ is negative? Now, we're not penalizing, but rewarding large elements of W =>

What functions R of W are used for regularization?

$$W = \underset{W}{\operatorname{argmin}} \sum_m (y^m - f(x^m; W))^2 + \lambda R(W)$$

- Two common ways to penalize complexity of W

What functions R of W are used for regularization?

$$W = \underset{W}{\operatorname{argmin}} \sum_m (y^m - f(x^m; W))^2 + \lambda R(W)$$

- Two common ways to penalize complexity of W
    - require sum of squares of elements of W to be small => L2 penalty

What functions R of W are used for regularization?

$$W = \underset{W}{\operatorname{argmin}} \sum_m (y^m - f(x^m; W))^2 + \lambda R(W)$$

- Two common ways to penalize complexity of W
  - require sum of squares of elements of W to be small => L2 penalty
  - require sum of absolute values of the elements of W to be small => L1 penalty

What functions R of W are used for regularization?

$$W = \underset{W}{\operatorname{argmin}} \sum_m (y^m - f(x^m; W))^2 + \lambda R(W)$$

- Two common ways to penalize complexity of W
  - require sum of squares of elements of W to be small => L2 penalty
  - require sum of absolute values of the elements of W to be small => L1 penalty
- Demo!

L1 vs L2 penalty in regression

- L1 penalty:
  - Pro: increased interpretability

L1 vs L2 penalty in regression

- L1 penalty:
  - Pro: increased interpretability
    - only a few elements of W come out to be non-zero -> interpretable solution because we can easily tell which features (e.g. voxels) are related to which labels (e.g. some vector representation of “chair” or “celery”)

L1 vs L2 penalty in regression

- L1 penalty:
  - Pro: increased interpretability
    - only a few elements of W come out to be non-zero -> interpretable solution because we can easily tell which features (e.g. voxels) are related to which labels (e.g. some vector representation of “chair” or “celery”)
  - Con: randomly chooses one of many correlated features to be non-zero

L1 vs L2 penalty in regression

- L1 penalty:
  - Pro: increased interpretability
    - only a few elements of W come out to be non-zero -> interpretable solution because we can easily tell which features (e.g. voxels) are related to which labels (e.g. some vector representation of “chair” or “celery”)
  - Con: randomly chooses one of many correlated features to be non-zero
    - problematic in the case that some of the correlated features are more important than others

L1 vs L2 penalty in regression

- L1 penalty:
  - Pro: increased interpretability
    - only a few elements of W come out to be non-zero -> interpretable solution because we can easily tell which features (e.g. voxels) are related to which labels (e.g. some vector representation of “chair” or “celery”)
  - Con: randomly chooses one of many correlated features to be non-zero
    - problematic in the case that some of the correlated features are more important than others
- L2 penalty:
  - Pro: no random choice of some correlated features over others

L1 vs L2 penalty in regression

- L1 penalty:
  - Pro: increased interpretability
    - only a few elements of W come out to be non-zero -> interpretable solution because we can easily tell which features (e.g. voxels) are related to which labels (e.g. some vector representation of “chair” or “celery”)
  - Con: randomly chooses one of many correlated features to be non-zero
    - problematic in the case that some of the correlated features are more important than others
- L2 penalty:
  - Pro: no random choice of some correlated features over others
  - Con: reduced interpretability because all features have weights

Regularization takeaways

- Regularization directly penalizes the complexity of model parameters

Regularization takeaways

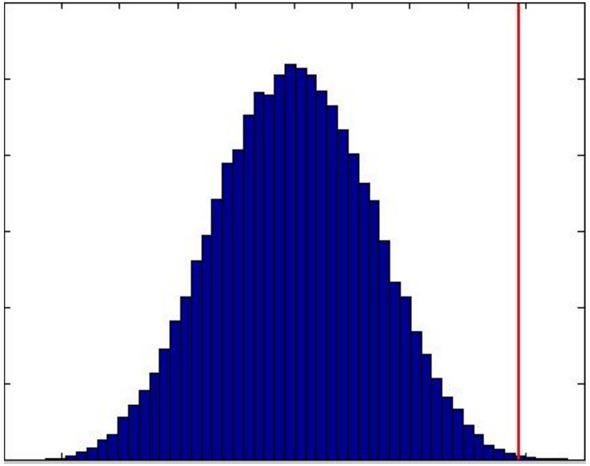
- Regularization directly penalizes the complexity of model parameters
- There are several functions of the parameters that can be used for regularization

Regularization takeaways

- Regularization directly penalizes the complexity of model parameters
- There are several functions of the parameters that can be used for regularization
- The strength of the regularization (λ) can be determined through cross-validation

How can ML help neuroscientists?

- **Evaluate results**
  - **nearly assumption-free significance testing (are the results significantly different from chance?)**



Evaluating significance of results: 4 steps

- Formulate null hypothesis (e.g. results are due to chance) and the alternative hypothesis (e.g. results are due to a real effect)

Evaluating significance of results: 4 steps

- Formulate null hypothesis (e.g. results are due to chance) and the alternative hypothesis (e.g. results are due to a real effect)
- Choose a test statistic to evaluate whether the null hypothesis is true

Evaluating significance of results: 4 steps

- Formulate null hypothesis (e.g. results are due to chance) and the alternative hypothesis (e.g. results are due to a real effect)
- Choose a test statistic to evaluate whether the null hypothesis is true
- Compute a p-value

Evaluating significance of results: 4 steps

- Formulate null hypothesis (e.g. results are due to chance) and the alternative hypothesis (e.g. results are due to a real effect)
- Choose a test statistic to evaluate whether the null hypothesis is true
- Compute a p-value
  - P-value = given that the null hypothesis is true, what is the probability of observing a test statistic that is at least as significant as the one we observe

Evaluating significance of results: 4 steps

- Formulate null hypothesis (e.g. results are due to chance) and the alternative hypothesis (e.g. results are due to a real effect)
- Choose a test statistic to evaluate whether the null hypothesis is true
- Compute a p-value
  - P-value = given that the null hypothesis is true, what is the probability of observing a test statistic that is at least as significant as the one we observe
- Compare the computed p-value to some pre-determined significance α value

Evaluating significance of results: 4 steps

- Formulate null hypothesis (e.g. results are due to chance) and the alternative hypothesis (e.g. results are due to a real effect)
- Choose a test statistic to evaluate whether the null hypothesis is true
- Compute a p-value
  - P-value = given that the null hypothesis is true, what is the probability of observing a test statistic that is at least as significant as the one we observe
- Compare the computed p-value to some pre-determined significance α value
  - commonly α = 0.01 or 0.05

Evaluating significance of results: 4 steps

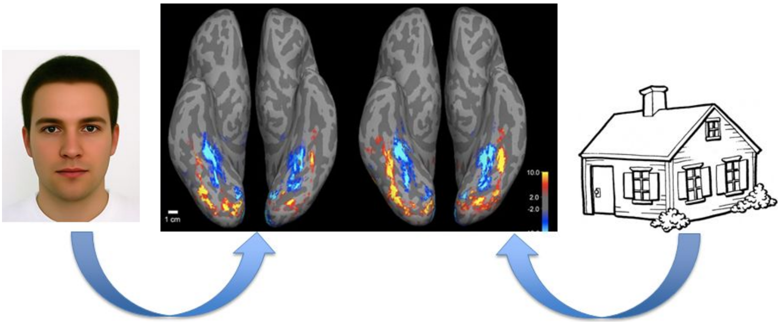
- Formulate null hypothesis (e.g. results are due to chance) and the alternative hypothesis (e.g. results are due to a real effect)
- Choose a test statistic to evaluate whether the null hypothesis is true
- Compute a p-value
  - P-value = given that the null hypothesis is true, what is the probability of observing a test statistic that is at least as significant as the one we observe
- Compare the computed p-value to some pre-determined significance α value
  - commonly α = 0.01 or 0.05
  - if p-value ≤ α, then our results are significant and we reject the null hypothesis and accept the alternative

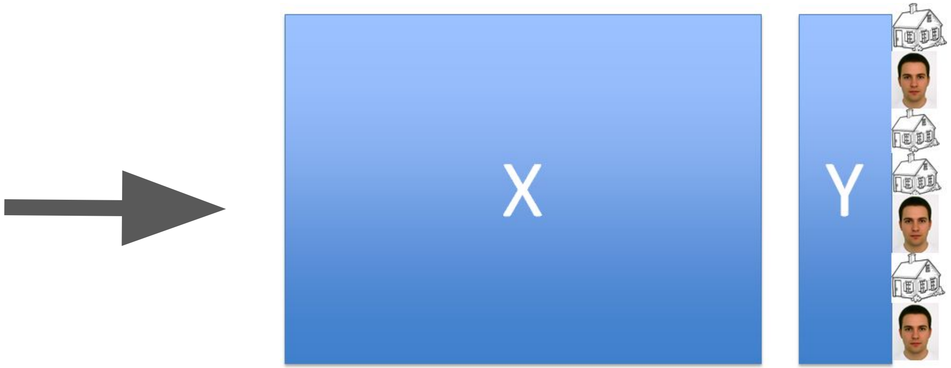
ML offers a test statistic that does not make assumptions about the data distribution

- Formulate null hypothesis (e.g. results are due to chance) and the alternative hypothesis (e.g. results are due to a real effect)
- **Choose a test statistic to evaluate whether the null hypothesis is true**
- Compute a p-value
    - P-value = given that the null hypothesis is true, what is the probability of observing a test statistic that is at least as significant as the one we observe
- Compare the computed p-value to some pre-determined significance α value
    - commonly α = 0.01 or 0.05
    - if p-value ≤ α, then our results are significant and we reject the null hypothesis and accept the alternative

Evaluating significance: example

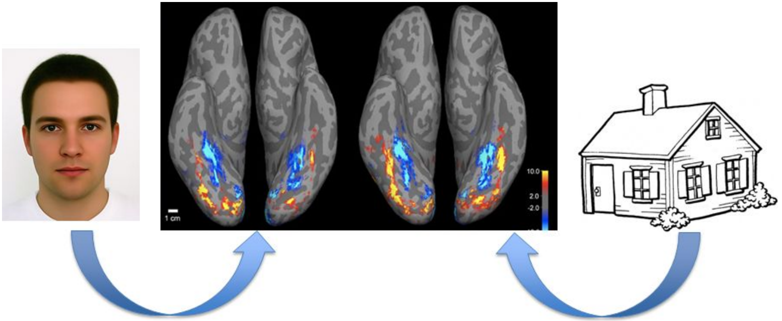
- Experiment to determine whether places and faces have different representations in the brain

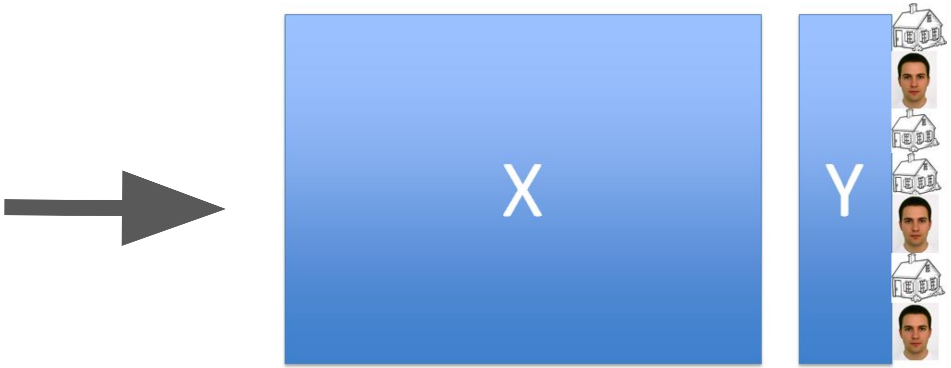




Evaluating significance: example

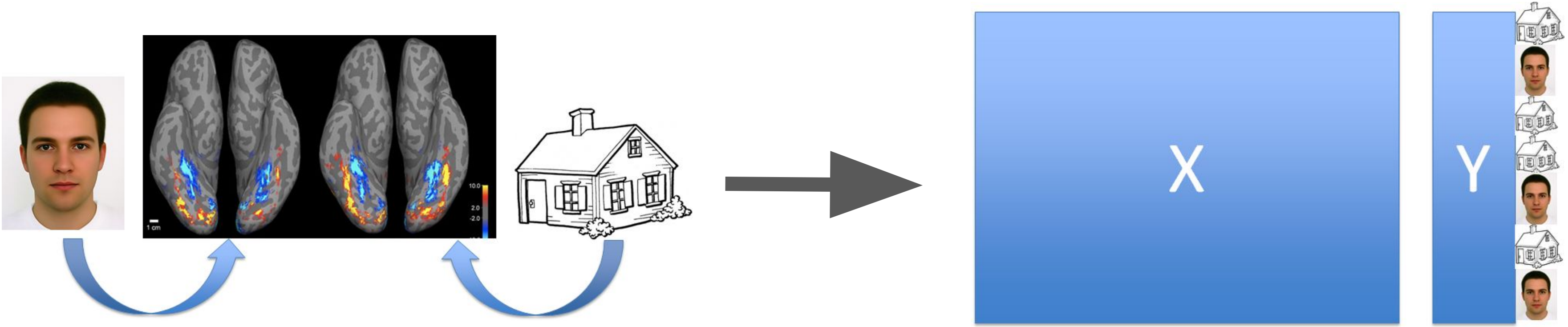
- Experiment to determine whether places and faces have different representations in the brain
- Run a classifier, obtain 80% accuracy





Evaluating significance: example

- Experiment to determine whether places and faces have different representations in the brain
- Run a classifier, obtain 80% accuracy
- What is the null hypothesis? No difference between the representations



Assumption-free test statistic: permutation test

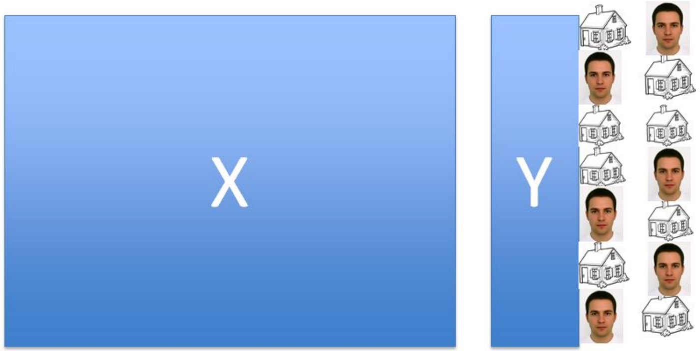
- If the null hypothesis is true (there is no difference between the representations), it shouldn’t matter whether a data instance is recorded during a presentation of a face or a place

Assumption-free test statistic: permutation test

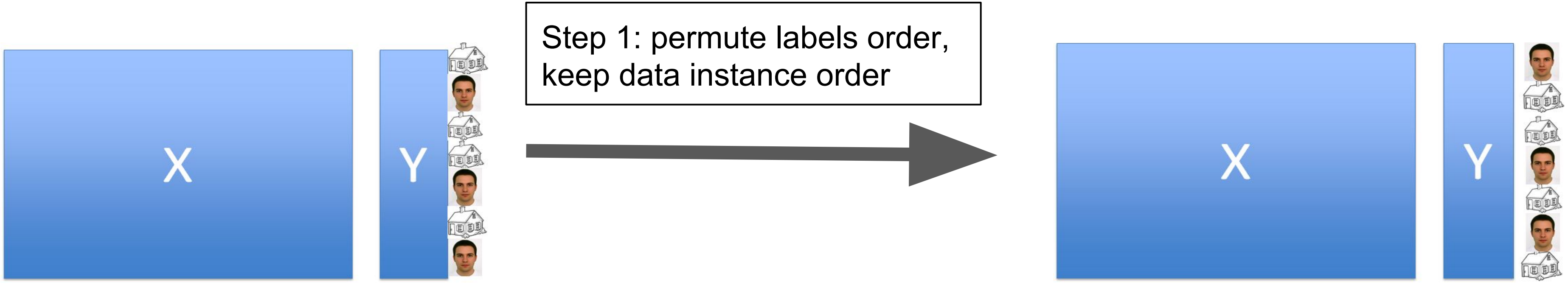
- If the null hypothesis is true (there is no difference between the representations), it shouldn’t matter whether a data instance is recorded during a presentation of a face or a place
  - we should be able to produce ~80% accuracy with random assignment of labels (“face”, “place”) to our data instances

Assumption-free test statistic: permutation test

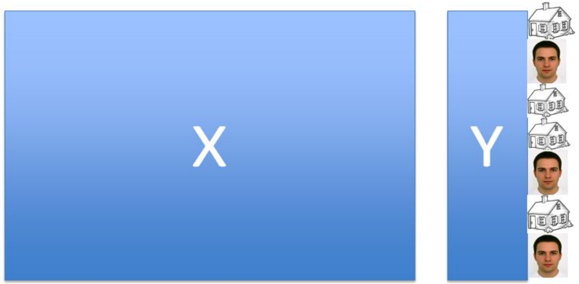
- If the null hypothesis is true (there is no difference between the representations), it shouldn't matter whether a data instance is recorded during a presentation of a face or a place
  - we should be able to produce ~80% accuracy with random assignment of labels ("face", "place") to our data instances
- Shuffle (permute) the order of the labels in the data set, while keeping the order of the data instances the same => recalculate results



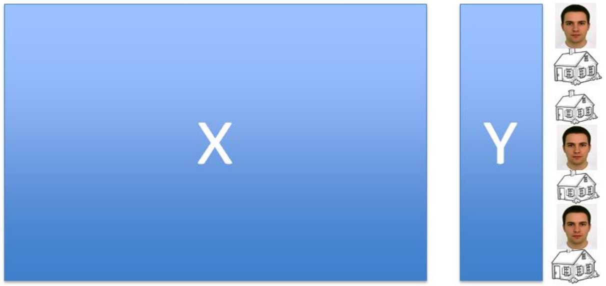
Permutation test steps



Permutation test steps



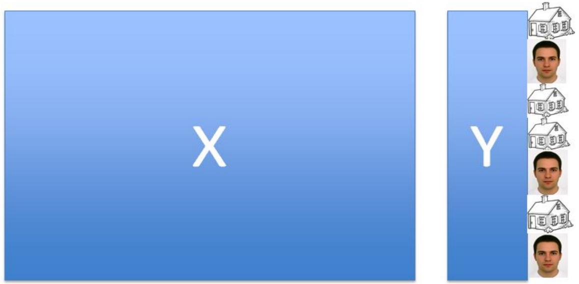
Step 1: permute labels order, keep data instance order



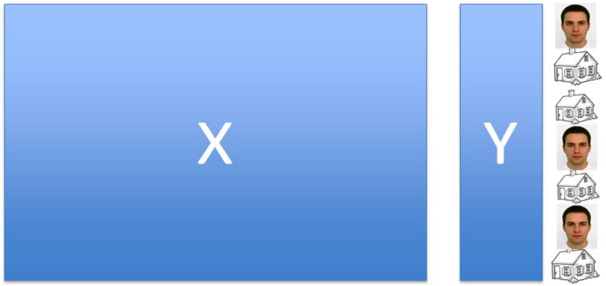
Step 2: run the same analysis as on the unpermuted data

45% classification accuracy

Permutation test steps



Step 1: permute labels order, keep data instance order



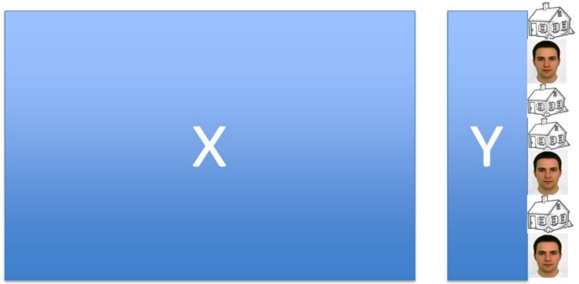
Step 2: run the same analysis as on the unpermuted data

45% classification accuracy

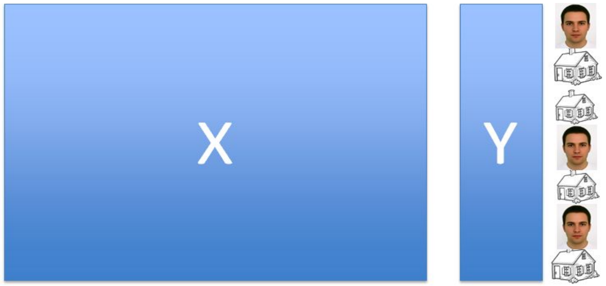
Step 3: save the computed classification accuracy

[45]

Permutation test steps



Step 1: permute labels order, keep data instance order



Step 2: run the same analysis as on the unpermuted data

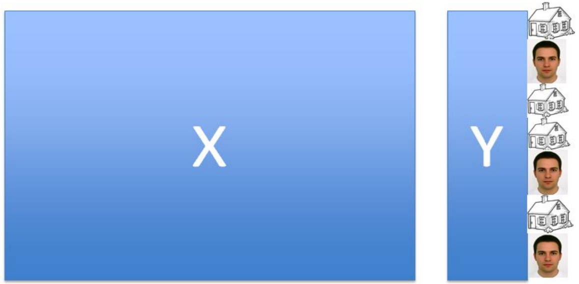
45% classification accuracy

Step 3: save the computed classification accuracy

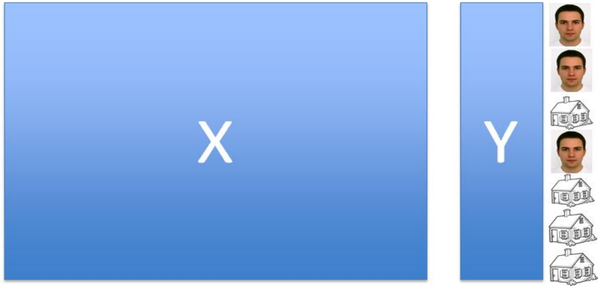
Step 4: repeat steps 1 through 3 many times (100s to 1000s)

[45]

Permutation test steps



Step 1: permute labels order, keep data instance order



Step 2: run the same analysis as on the unpermuted data

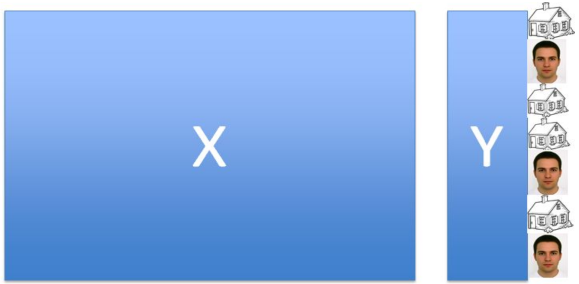
57% classification accuracy

Step 3: save the computed classification accuracy

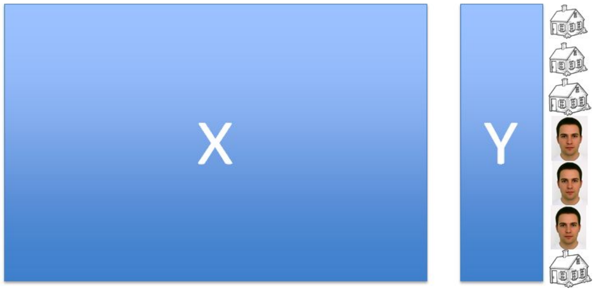
Step 4: repeat steps 1 through 3 many times (100s to 1000s)

[45, 57]

Permutation test steps



Step 1: permute labels order,
keep data instance order



Step 2: run the same analysis as on the unpermuted data

53% classification accuracy

Step 3: save the computed classification accuracy

[45, 57,...,53]

Step 4: repeat steps 1 through 3 many times (100s to 1000s)

Getting a p-value from the permutation test

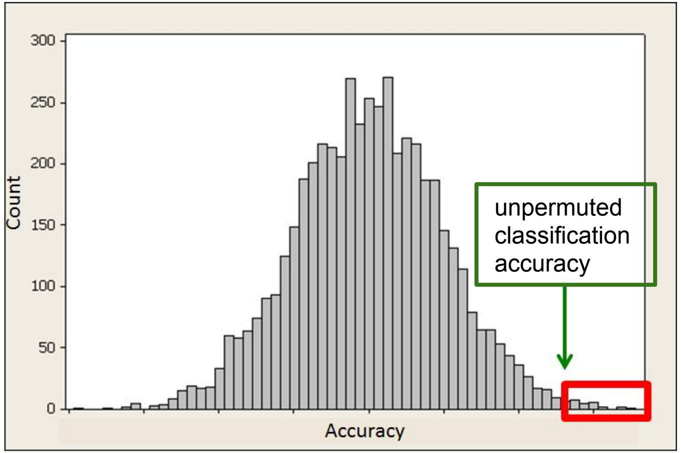
- At the end we have an array of all classification accuracies from permuted data: [45, 57,...,53]

Getting a p-value from the permutation test

- At the end we have an array of all classification accuracies from permuted data: [45, 57,...,53]
- Calculate how many total times the obtained accuracy is greater or equal to 80%

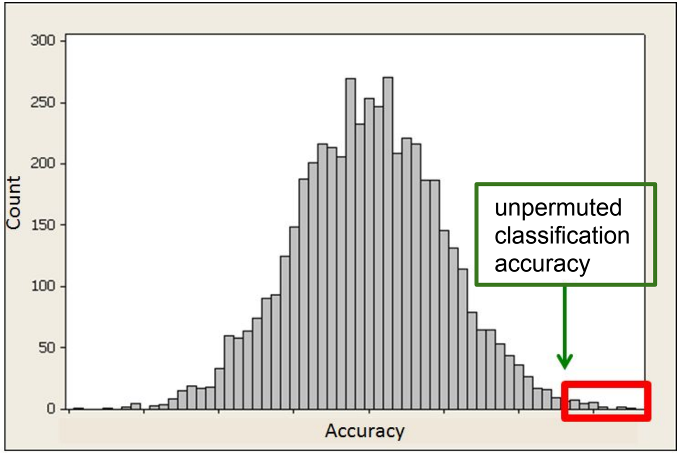
Getting a p-value from the permutation test

- At the end we have an array of all classification accuracies from permuted data: [45, 57,...,53]
- Calculate how many total times the obtained accuracy is greater or equal to 80%
- The proportion of these better-performing runs out of all permuted runs is the p-value



Getting a p-value from the permutation test

- At the end we have an array of all classification accuracies from permuted data: [45, 57,...,53]
- Calculate how many total times the obtained accuracy is greater or equal to 80%
- The proportion of these better-performing runs out of all permuted runs is the p-value



- the unpermuted accuracy is 80%
- let us run 500 permutations
- 5 permutations above 80%
- p-value = 5/500 = 0.01

How many permutations should we run?

- What do you think? 100s or 1000s and why?

How many permutations should we run?

- What do you think? 100s or 1000s and why?
- Trade-off between resolution and computation time

Permutation test takeaways

- Makes no assumptions about the distribution of the data

Permutation test takeaways

- Makes no assumptions about the distribution of the data
- Requires permuting the order of labels

Permutation test takeaways

- Makes no assumptions about the distribution of the data
- Requires permuting the order of labels
- Computationally expensive because need to run the same analysis many times on the different permutations of the data

Permutation test takeaways

- Makes no assumptions about the distribution of the data
- Requires permuting the order of labels
- Computationally expensive because need to run the same analysis many times on the different permutations of the data
    - but computation can be parallelized!

Evaluating significance when testing multiple different models

- Examples
  - want to see how the representations of faces changes over time so we test a different predictor for one of many time windows

Evaluating significance when testing multiple different models

- Examples
  - want to see how the representations of faces changes over time so we test a different predictor for one of many time windows
  - want to see how the representations of faces changes over brain regions so we test a different predictor for one of many regions of interest

Evaluating significance when testing multiple different models

- Examples
  - want to see how the representations of faces changes over time so we test a different predictor for one of many time windows
  - want to see how the representations of faces changes over brain regions so we test a different predictor for one of many regions of interest
- We can find the corresponding p-value for each of the models using a permutation test

Evaluating significance when testing multiple different models

- Examples
  - want to see how the representations of faces changes over time so we test a different predictor for one of many time windows
  - want to see how the representations of faces changes over brain regions so we test a different predictor for one of many regions of interest
- We can find the corresponding p-value for each of the models using a permutation test
- But if we're using $\alpha = 0.05$, there is a 5% chance of incorrectly rejecting the null hypothesis, so what happens when we test multiple hypotheses?

Problem with multiple comparisons: example

- Let there be 100 null hypothesis (e.g. ROI 1 does not represent faces,..., ROI 100 does not represent face).

Problem with multiple comparisons: example

- Let there be 100 null hypothesis (e.g. ROI 1 does not represent faces,..., ROI 100 does not represent face).
- We perform 100 independent permutation tests to test whether each null hypothesis is true.

Problem with multiple comparisons: example

- Let there be 100 null hypothesis (e.g. ROI 1 does not represent faces,..., ROI 100 does not represent face).
- We perform 100 independent permutation tests to test whether each null hypothesis is true.
- What is the probability that if all null hypotheses are true, there will be at least one incorrect rejection?

Problem with multiple comparisons: example

- Let there be 100 null hypothesis (e.g. ROI 1 does not represent faces,..., ROI 100 does not represent face).
- We perform 100 independent permutation tests to test whether each null hypothesis is true.
- What is the probability that if all null hypotheses are true, there will be at least one incorrect rejection?
  - 1 - probability of no incorrect rejections

Problem with multiple comparisons: example

- Let there be 100 null hypothesis (e.g. ROI 1 does not represent faces,..., ROI 100 does not represent face).
- We perform 100 independent permutation tests to test whether each null hypothesis is true.
- What is the probability that if all null hypotheses are true, there will be at least one incorrect rejection?
  - 1 - probability of no incorrect rejections
  - What is the probability that one hypothesis is correctly rejected? 1 - 0.05 = 0.95

Problem with multiple comparisons: example

- Let there be 100 null hypothesis (e.g. ROI 1 does not represent faces,..., ROI 100 does not represent face).
- We perform 100 independent permutation tests to test whether each null hypothesis is true.
- What is the probability that if all null hypotheses are true, there will be at least one incorrect rejection?
  - 1 - probability of no incorrect rejections
  - What is the probability that one hypothesis is correctly rejected? 1 - 0.05 = 0.95
  - So what is the probability that 100 hypotheses are correctly rejected? 0.95^{100}

Problem with multiple comparisons: example

- Let there be 100 null hypothesis (e.g. ROI 1 does not represent faces,..., ROI 100 does not represent face).
- We perform 100 independent permutation tests to test whether each null hypothesis is true.
- What is the probability that if all null hypotheses are true, there will be at least one incorrect rejection?
  - 1 - probability of no incorrect rejections
  - What is the probability that one hypothesis is correctly rejected? $1 - 0.05 = 0.95$
  - So what is the probability that 100 hypotheses are correctly rejected? 0.95^{100}
  - $1 - 0.95^{100} \cong 0.994$

Solution: multiple comparisons correction

- Goal: come up with a way to threshold individual p-values to control probability of having false positives after all tests

Solution: multiple comparisons correction

- Goal: come up with a way to threshold individual p-values to control probability of having false positives after all tests
- Bonferroni correction
  - keeps the probability of having at least one false positive after all tests under α

Solution: multiple comparisons correction

- Goal: come up with a way to threshold individual p-values to control probability of having false positives after all tests
- Bonferroni correction
  - keeps the probability of having at least one false positive after all tests under α
  - for m p-values, threshold $p_i \leq \frac{\alpha}{m}$

Solution: multiple comparisons correction

- Goal: come up with a way to threshold individual p-values to control probability of having false positives after all tests
- Bonferroni correction
  - keeps the probability of having at least one false positive after all tests under α
  - for m p-values, threshold $p_i \leq \frac{\alpha}{m}$
  - very conservative when there is large number of tests m, not necessarily good

Solution: multiple comparisons correction

- Goal: come up with a way to threshold individual p-values to control probability of having false positives after all tests
- Bonferroni correction
  - keeps the probability of having at least one false positive after all tests under α
  - for m p-values, threshold $p_i \leq \frac{\alpha}{m}$
  - very conservative when there is large number of tests m, not necessarily good
- FDR (False discovery rate) correction
  - control the expected proportion of incorrect rejections of the null (false discoveries/positives)

Solution: multiple comparisons correction

- Goal: come up with a way to threshold individual p-values to control probability of having false positives after all tests
- Bonferroni correction
  - keeps the probability of having at least one false positive after all tests under α
  - for m p-values, threshold $p_i \leq \frac{\alpha}{m}$
  - very conservative when there is large number of tests m, not necessarily good
- FDR (False discovery rate) correction
  - control the expected proportion of incorrect rejections of the null (false discoveries/positives)
  - less stringent than Bonferroni, but greater power (i.e. P(reject null when alternate is true))

Solution: multiple comparisons correction

- Goal: come up with a way to threshold individual p-values to control probability of having false positives after all tests
- Bonferroni correction
  - keeps the probability of having at least one false positive after all tests under α
  - for m p-values, threshold $p_i \leq \frac{\alpha}{m}$
  - very conservative when there is large number of tests m, not necessarily good
- FDR (False discovery rate) correction
  - control the expected proportion of incorrect rejections of the null (false discoveries/positives)
  - less stringent than Bonferroni, but greater power (i.e. P(reject null when alternate is true))
  - many different variants

Solution: multiple comparisons correction

- Goal: come up with a way to threshold individual p-values to control probability of having false positives after all tests
- Bonferroni correction
  - keeps the probability of having at least one false positive after all tests under α
  - for m p-values, threshold $p_i \leq \frac{\alpha}{m}$
  - very conservative when there is large number of tests m, not necessarily good
- FDR (False discovery rate) correction
  - control the expected proportion of incorrect rejections of the null (false discoveries/positives)
  - less stringent than Bonferroni, but greater power (i.e. P(reject null when alternate is true))
  - many different variants
    - Benjamini-Hochberg procedure:

Solution: multiple comparisons correction

- Goal: come up with a way to threshold individual p-values to control probability of having false positives after all tests
- Bonferroni correction
  - keeps the probability of having at least one false positive after all tests under α
  - for m p-values, threshold $p_i \leq \frac{\alpha}{m}$
  - very conservative when there is large number of tests m, not necessarily good
- FDR (False discovery rate) correction
  - control the expected proportion of incorrect rejections of the null (false discoveries/positives)
  - less stringent than Bonferroni, but greater power (i.e. P(reject null when alternate is true))
  - many different variants
    - Benjamini-Hochberg procedure:
      - sort p-values in increasing order, find largest k such that $p_k \leq \frac{\alpha}{m} k$

Solution: multiple comparisons correction

- Goal: come up with a way to threshold individual p-values to control probability of having false positives after all tests
- Bonferroni correction
  - keeps the probability of having at least one false positive after all tests under α
  - for m p-values, threshold $p_i \leq \frac{\alpha}{m}$
  - very conservative when there is large number of tests m, not necessarily good
- FDR (False discovery rate) correction
  - control the expected proportion of incorrect rejections of the null (false discoveries/positives)
  - less stringent than Bonferroni, but greater power (i.e. P(reject null when alternate is true))
  - many different variants
    - Benjamini-Hochberg procedure:
      - sort p-values in increasing order, find largest k such that $p_k \leq \frac{\alpha}{m} k$
      - reject null hypothesis for all tests i = 1,..k

Solution: multiple comparisons correction

- Goal: come up with a way to threshold individual p-values to control probability of having false positives after all tests
- Bonferroni correction
  - keeps the probability of having at least one false positive after all tests under α
  - for m p-values, threshold $p_i \leq \frac{\alpha}{m}$
  - very conservative when there is large number of tests m, not necessarily good
- FDR (False discovery rate) correction
  - control the expected proportion of incorrect rejections of the null (false discoveries/positives)
  - less stringent than Bonferroni, but greater power (i.e. P(reject null when alternate is true))
  - many different variants
    - Benjamini-Hochberg procedure:
      - sort p-values in increasing order, find largest k such that $p_k \leq \frac{\alpha}{m} k$
      - reject null hypothesis for all tests i = 1,..k
      - first test equivalent to Bonferroni correction, others slightly less stringent

Multiple comparison corrections takeaways

- Often, we must test multiple hypotheses and produce multiple p-values

Multiple comparison corrections takeaways

- Often, we must test multiple hypotheses and produce multiple p-values
- Thresholding each p-value independently at α, results in much greater false positive rate over all tests

Multiple comparison corrections takeaways

- Often, we must test multiple hypotheses and produce multiple p-values
- Thresholding each p-value independently at α, results in much greater false positive rate over all tests
- Multiple comparison corrections aim to establish thresholds for individual p-values such that the overall false positive rate is controlled

Multiple comparison corrections takeaways

- Often, we must test multiple hypotheses and produce multiple p-values
- Thresholding each p-value independently at α, results in much greater false positive rate over all tests
- Multiple comparison corrections aim to establish thresholds for individual p-values such that the overall false positive rate is controlled
- The most common multiple comparison correction is FDR, and there are many types of FDR procedures

Main takeaways: model selection & significance testing

- Multiple models exist, but want to choose the simplest model (prevent overfitting) that learns well (prevent underfitting)

Main takeaways: model selection & significance testing

- Multiple models exist, but want to choose the simplest model (prevent overfitting) that learns well (prevent underfitting)
- Cross-validation, feature selection, and regularization can all be used individually or in groups to perform model selection

Main takeaways: model selection & significance testing

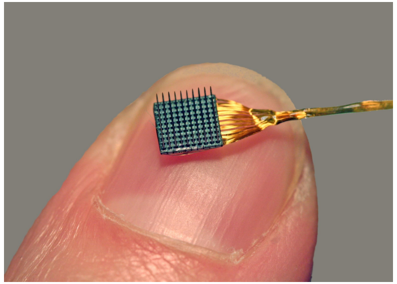
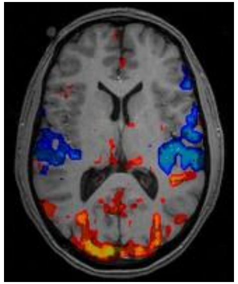
- Multiple models exist, but want to choose the simplest model (prevent overfitting) that learns well (prevent underfitting)
- Cross-validation, feature selection, and regularization can all be used individually or in groups to perform model selection
- Permutation test is one significance test that does not make assumptions about the data distribution

Main takeaways: model selection & significance testing

- Multiple models exist, but want to choose the simplest model (prevent overfitting) that learns well (prevent underfitting)
- Cross-validation, feature selection, and regularization can all be used individually or in groups to perform model selection
- Permutation test is one significance test that does not make assumptions about the data distribution
- When we wish to evaluate several hypotheses, we must correct for the multiple comparison in order to control the rate of incorrectly rejecting the null hypothesis

Next time: dimensionality reduction & clustering

- ❏ Deal with large number of sensors/recording sites

- ❏ investigate high-dimensional representations
  - ❏ classification (what does this high-dimensional data represent?)
  - ❏ regression (how does it represent it? can we predict a different representation?)
  - ❏ model selection (what model would best describe this high dimensional data?)
- ❏ **uncover few underlying processes that interact in complex ways**
  - ❏ **dimensionality reduction techniques**