Case Study 3
Feature Engineering

ML & AI for Data Scientists
(2022)

Sebastian Stein

drsstein.github.io

@ssteinuk

Scan to access these slides on Google Drive or go to https://tinyurl.com/sp7ky24w

Overview

1. Case Study 3
2. What is Feature Engineering?
3. Feature Selection
    - 3.1. Filtering Methods
    - 3.2. Wrapper Methods
    - 3.3. Embedding Methods
4. Feature Extraction

Overview

1. Case Study 3: Predicting Central Neuropathic Pain
2. What is Feature Engineering?
3. Feature Selection
    - 3.1. Filtering Methods
    - 3.2. Wrapper Methods
    - 3.3. Embedding Methods
4. Feature Extraction

Predicting Central Neuropathic Pain

Approximately 50% of people with Spinal Cord Injury (SCI) have Central Neuropathic Pain (CNP).

- Pain in response to non-painful stimuli, episodic (electric shock), “pins and needles”, numbness
- There is currently no treatment, only prevention
- Preventative medications have strong side-effects

Predicting Central Neuropathic Pain

Approximately 50% of people with Spinal Cord Injury (SCI) have Central Neuropathic Pain (CNP).

- Pain in response to non-painful stimuli, episodic (electric shock), “pins and needles”, numbness
- There is currently no treatment, only prevention
- Preventative medications have strong side-effects

Predicting whether a patient is likely to develop pain is useful for selective treatment

- Manual assessment is time-consuming, error-prone and somewhat subjective
- There is some evidence that brain Electroencephalogram (EEG) data has characteristic markers
- We have a (small) dataset with EEG from SCI patients, of which some later developed CNP
- The data is extremely high-dimensional, so it is very hard for a classifier to tell them apart

Predicting Central Neuropathic Pain

Approximately 50% of people with Spinal Cord Injury (SCI) have Central Neuropathic Pain (CNP).

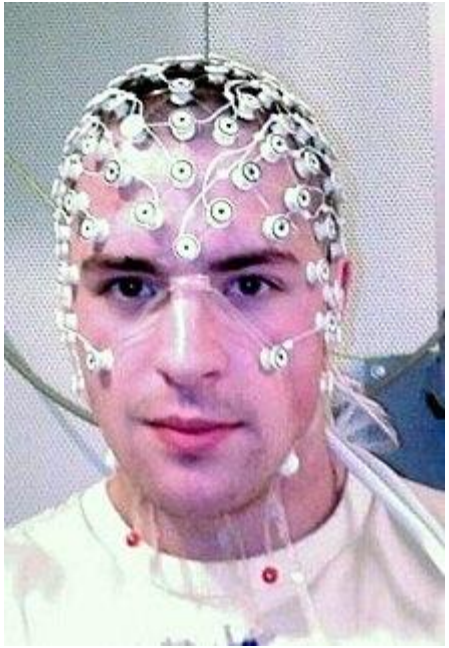
- Pain in response to non-painful stimuli, episodic (electric shock), “pins and needles”, numbness
- There is currently no treatment, only prevention
- Preventative medications have strong side-effects

Predicting whether a patient is likely to develop pain is useful for selective treatment

- Manual assessment is time-consuming, error-prone and somewhat subjective
- There is some evidence that brain Electroencephalogram (EEG) data has characteristic markers
- We have a (small) dataset with EEG from SCI patients, of which some later developed CNP
- The data is extremely high-dimensional, so it is very hard for a classifier to tell them apart.

Can feature engineering help to predict - better than random guessing - who later develops CNP?

What is Electroencephalogram (EEG)



Case Study 3: Dataset

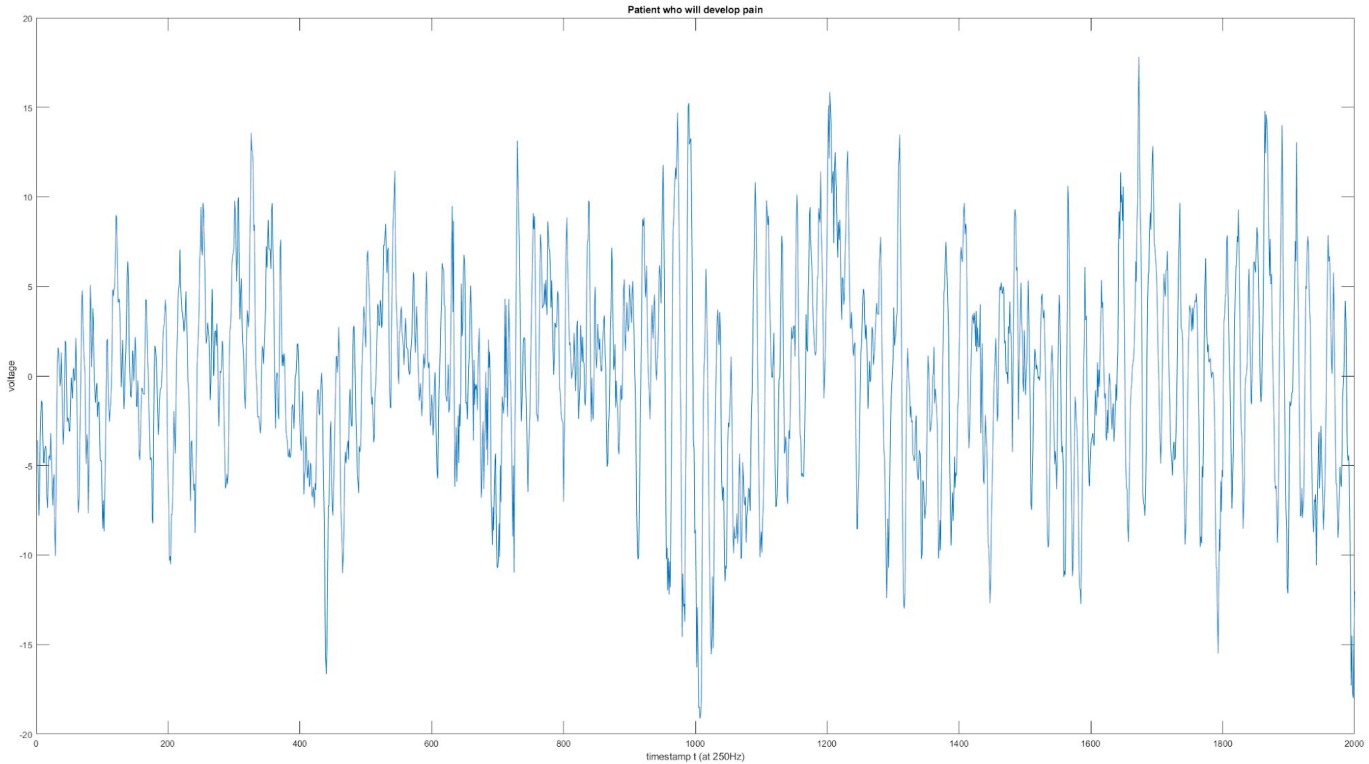
Participants: N=18 participants with SCI

- 8 participants did not develop CNP within 6 months after data collection (PNP or ‘negative’)
- 10 participants developed CNP within 6 months after data collection (PDP or ‘positive’)

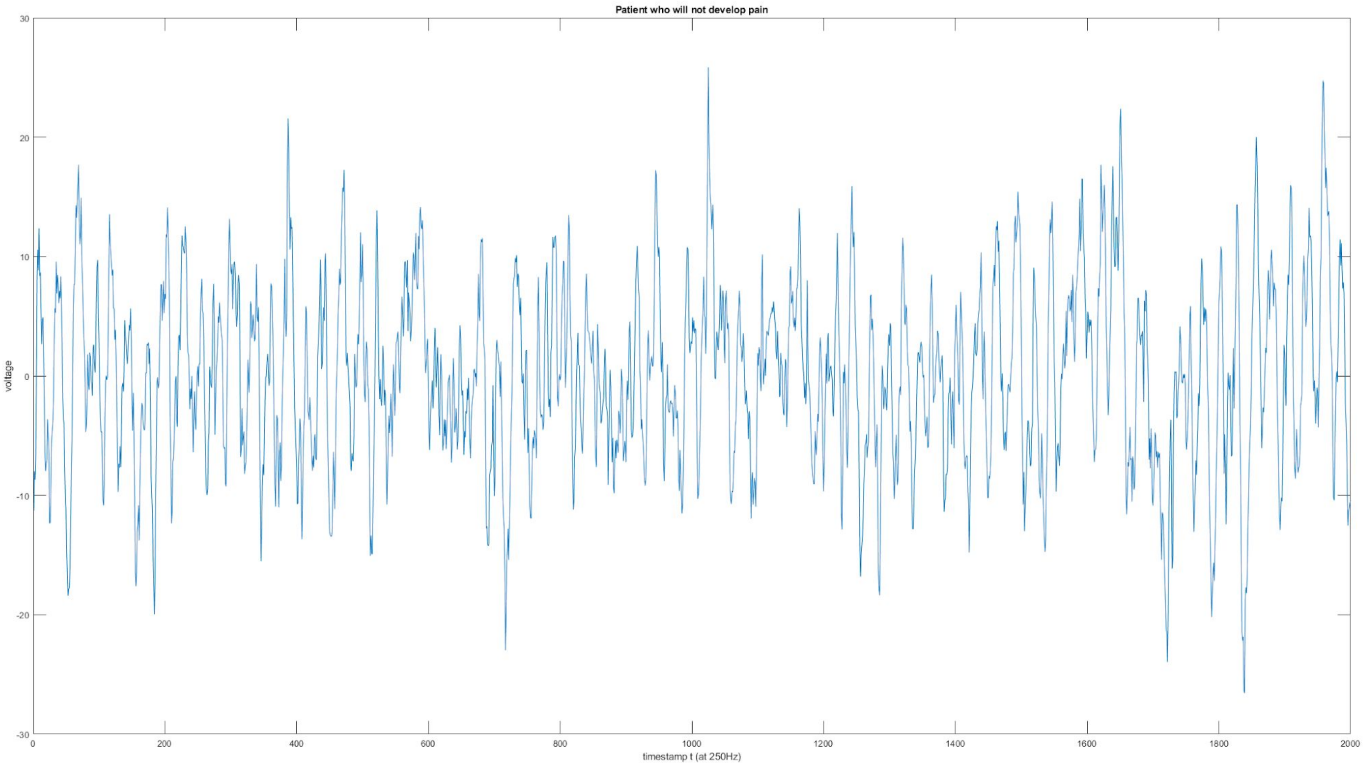
Data Collection

- 48 electrode EEG, recording electrical activity of the brain at 250 Hz
- participants were asked to relax with eyes closed (EC) and eyes opened (EO)
- Segments of data with 5 second length were recorded with 10 "repetitions” per participant
- 180 labeled data points (18 participants with 10 repetitions each) are available in total

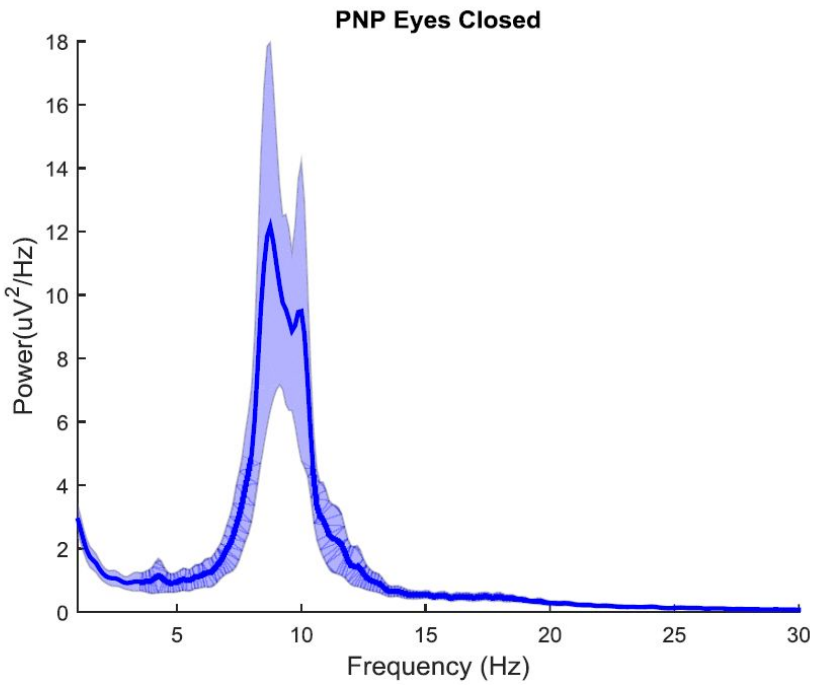
Raw Data - Single Electrode

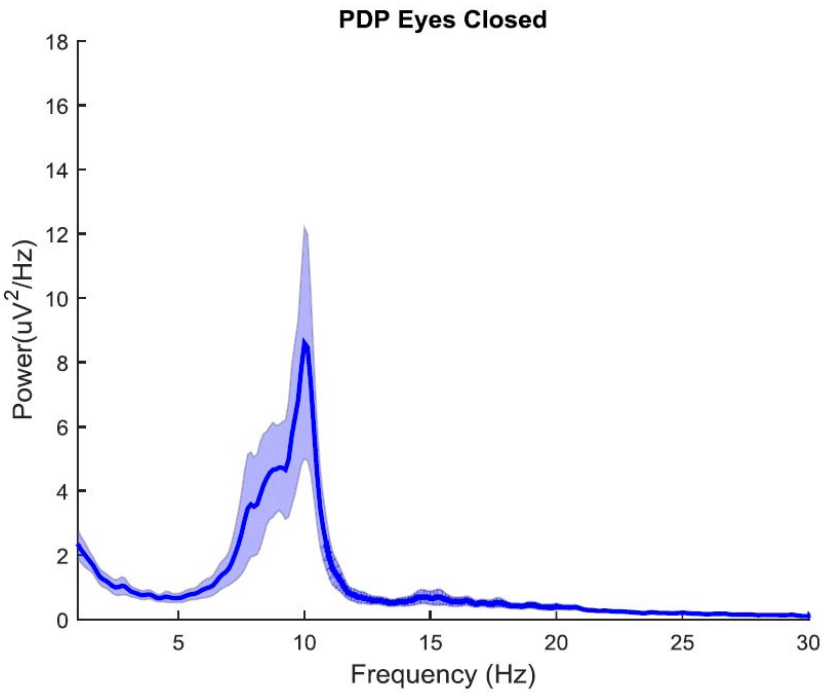


Raw Data - Single Electrode (2)

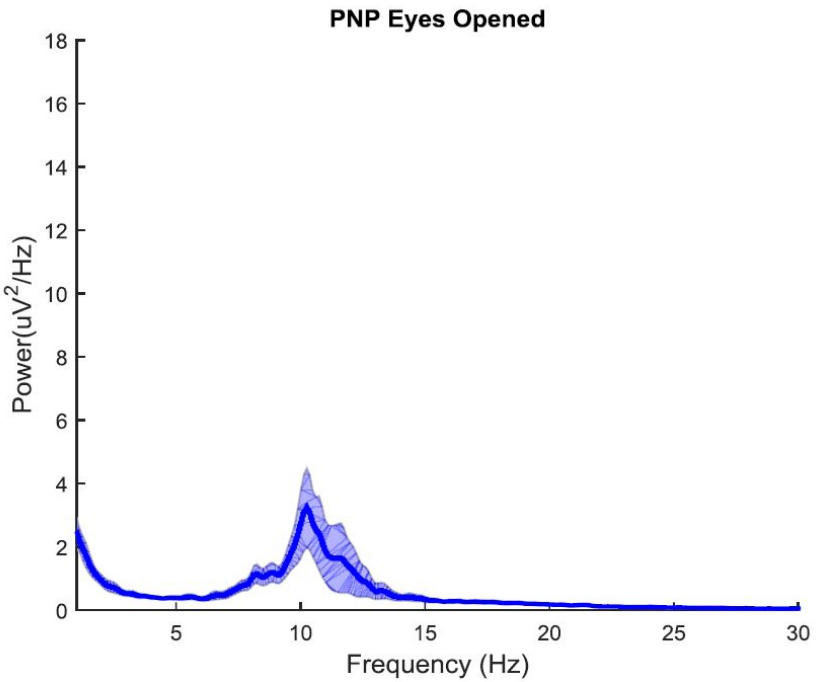


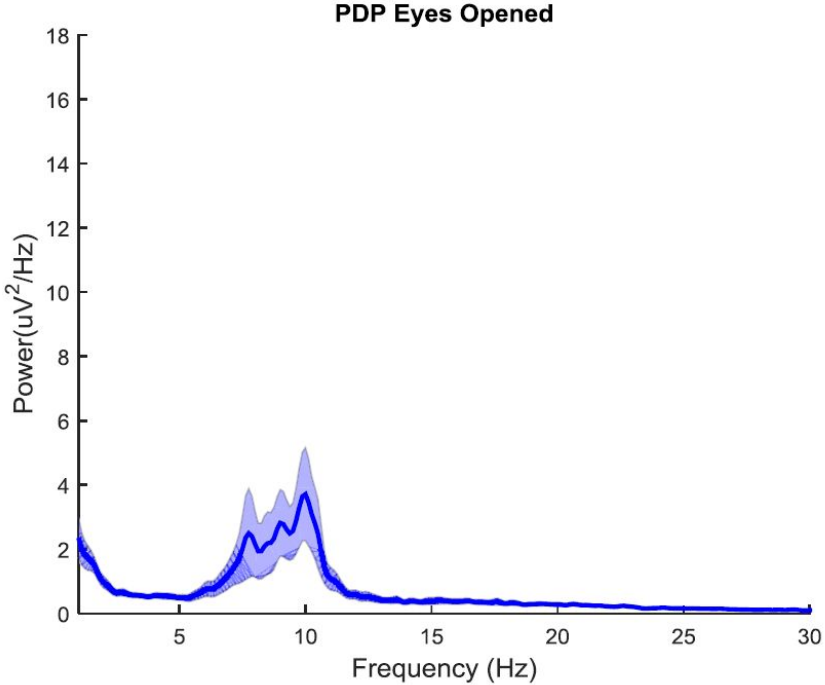
Frequency Spectrum



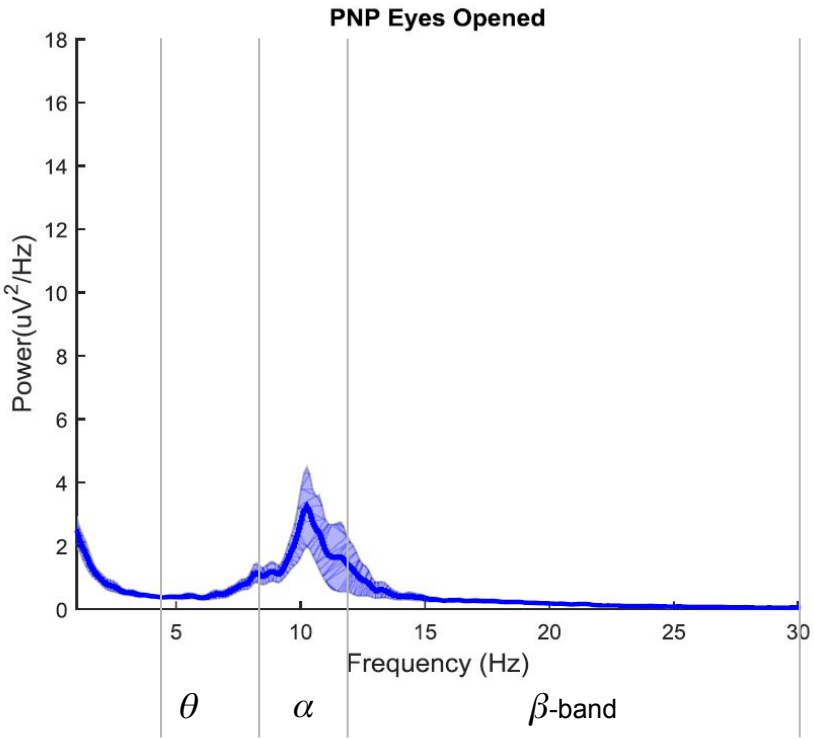


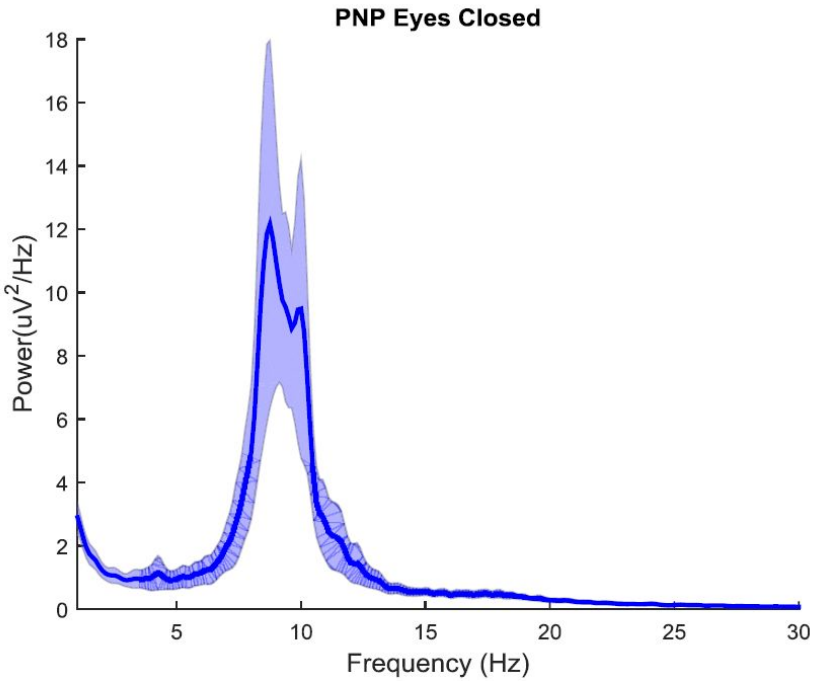
Frequency Spectrum (2)



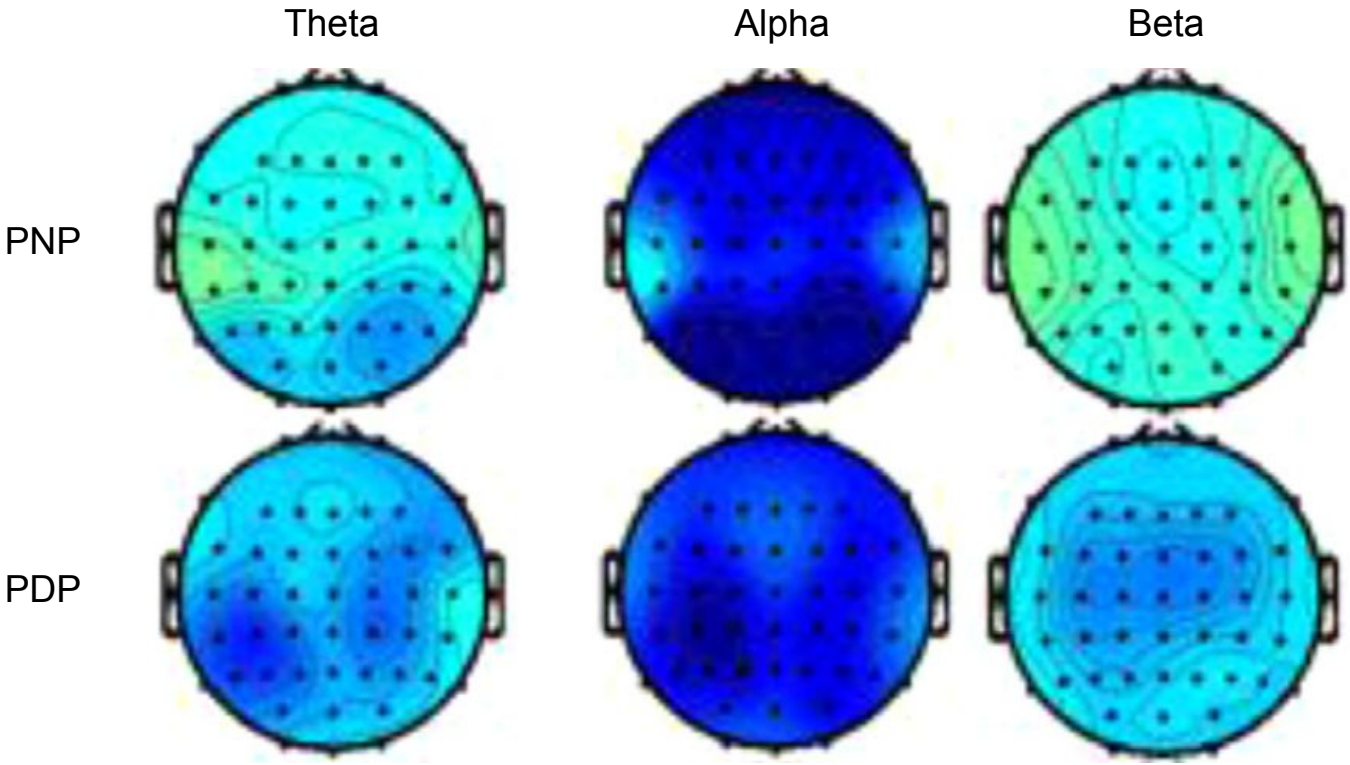


Frequency Band Power





Case Study 3: Bandpower by Electrode Position



Case Study 3: Task

Your task is to devise a feature engineering strategy.

Data

- preprocessing already applied
  - signal denoising and normalization
  - temporal segmentation
  - frequency band power estimation
- 180 rows (18 subjects x 10 repetitions) x 432 columns (9 features x 48 electrodes)

Objective Measure

Leave one subject out cross-validation accuracy, sensitivity and specificity

Report

Describe your feature engineering strategy and give evidence for why your strategy is better than others.

Case Study 3: Requirements

Compare at least 2 feature selection methods (groups of 5-6: at least 4)

- filtering methods (at least one scoring function)
- wrapper methods (at least one, e.g., forward feature selection or backward feature elimination)
- embedding methods (at least one, e.g., L1 regularisation)

Combine each method with at least 2 classifiers (justify your choice of classifiers in the report)

- SVM
- KNN
- ...

Perform Leave-one-group-out cross-validation, where all 10 samples from each subject are in one group

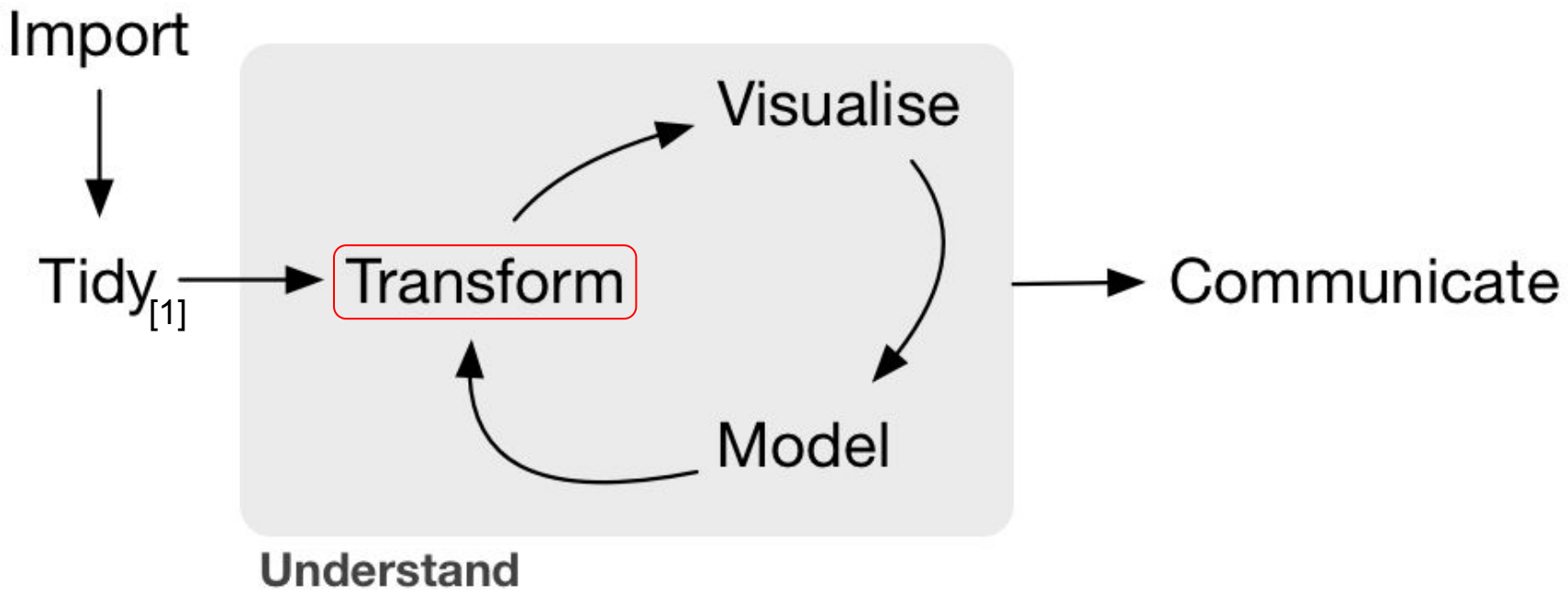
Use cross-validation to optimise hyper-parameter values

As a baseline for comparison, evaluate each of your classifiers trained on the full set of features (trained without feature selection)

Overview

1. Case Study 3
2. What is Feature Engineering?
3. Feature Selection
    - 3.1. Filtering Methods
    - 3.2. Wrapper Methods
    - 3.3. Embedding Methods
4. Feature Extraction

What is Feature Engineering?



Source: Garrett Grolemund and Hadley Wickham, *R for Data Science*, https://r4ds.had.co.nz/

[1] https://vita.had.co.nz/papers/tidy-data.html

What is Feature Engineering?

Feature engineering is the process of transforming raw data into something that better represents the learning problem to the predictive model, resulting in improved generalization to unseen data.

“... some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used.” [Pedro Domingos]

What do we mean by features?

A feature is an individual measurable property or characteristic of a phenomenon being observed.

Training data	$\{(x_i, y_i)\}_N$
Features	$\Phi(x_i)$
Model	$f(\Phi(x_i), \theta)$
Learning problem	$\text{argmin}_\theta L(y_i, f(\Phi(x_i), \theta))$

What is Feature Engineering?

- Encoding prior knowledge we have about the data domain and the problem
- Guiding questions
  - Which information can be ignored?
  - Which information should be retained?
  - How should retained information be represented?
- Answers to these questions depend on
  - Data domain
  - Problem to be solved
  - Predictive model
- Some feature engineering methods are specific to the data and problem domains.

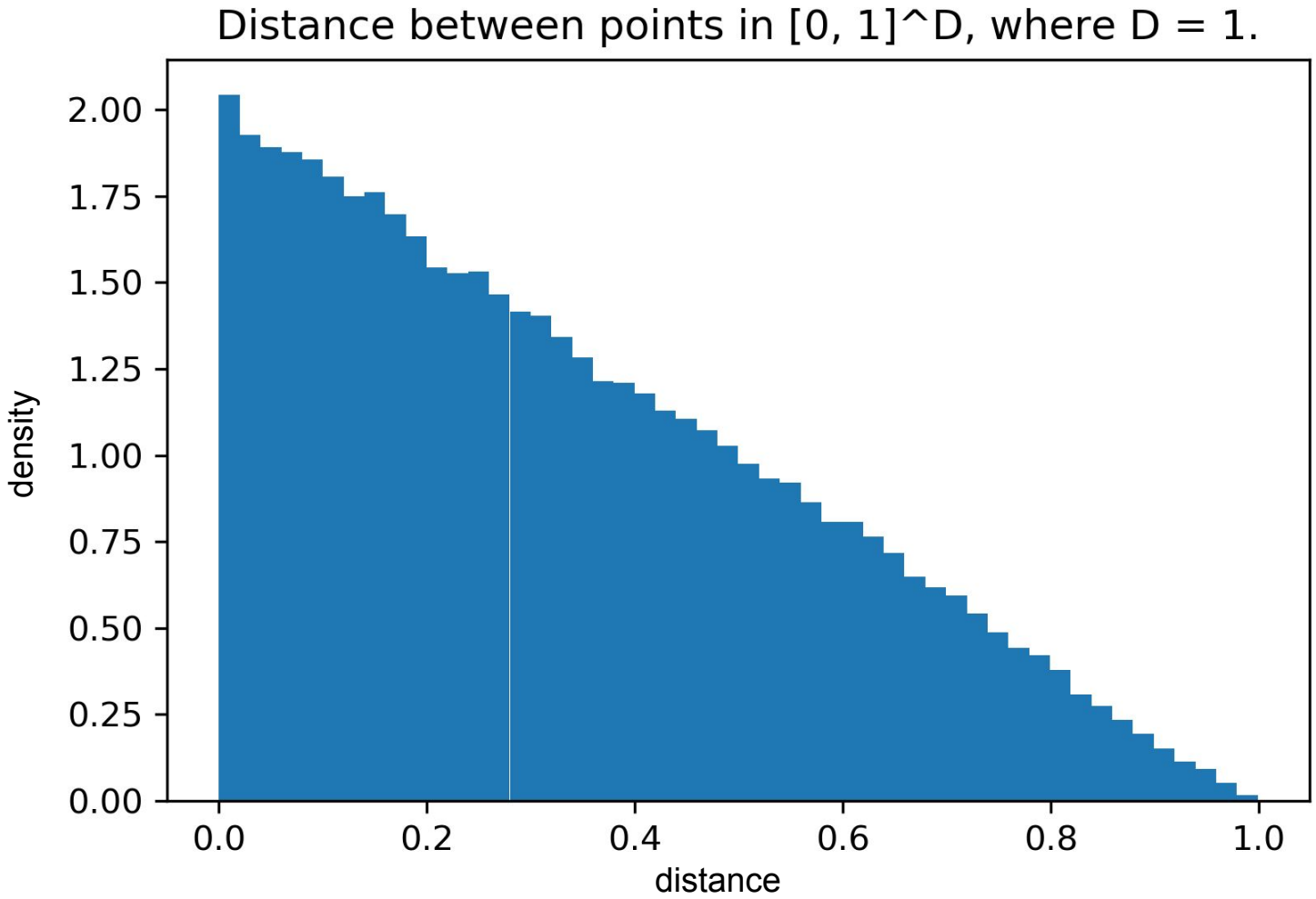
But there are common techniques!

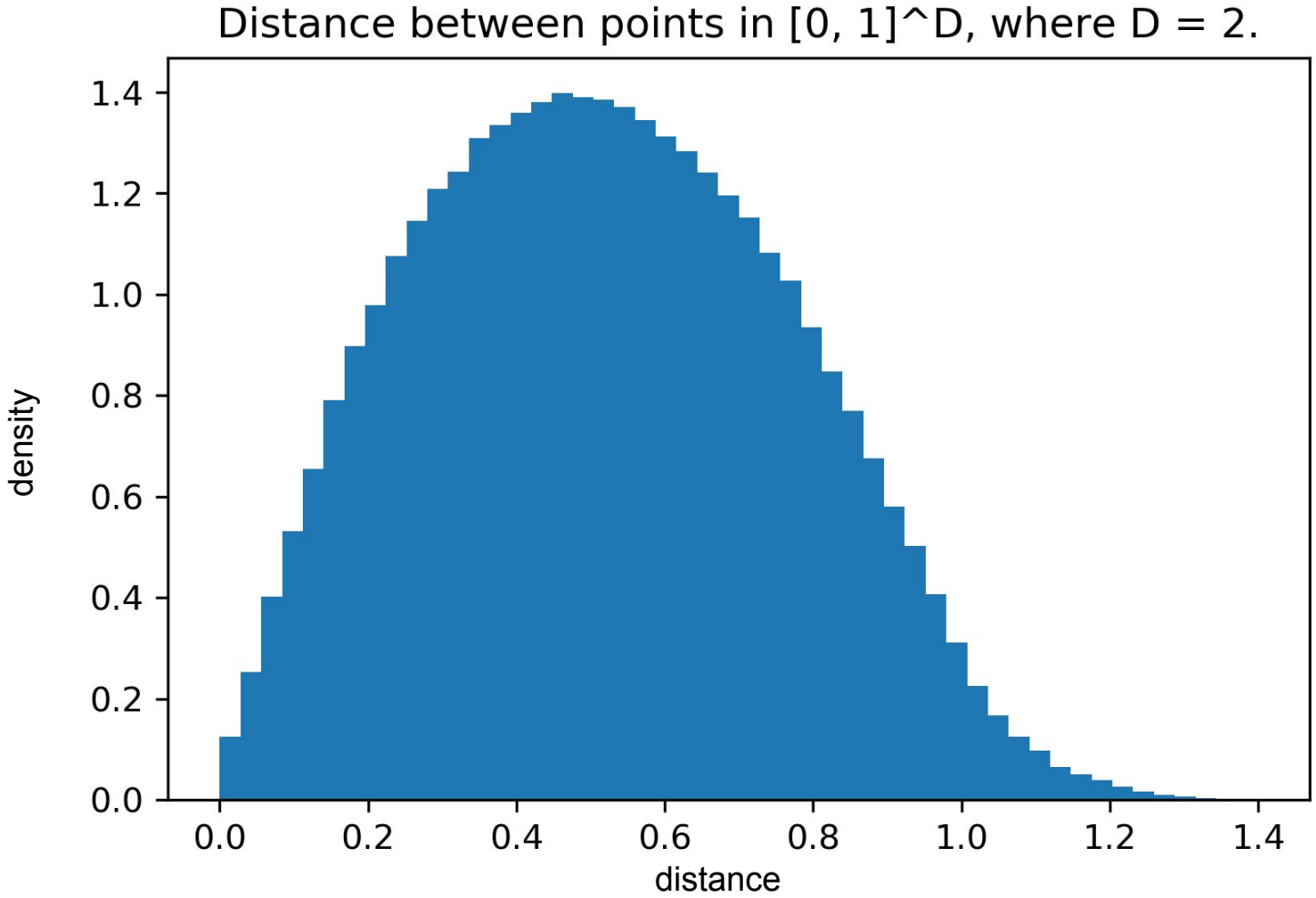
Why don’t we just keep everything?

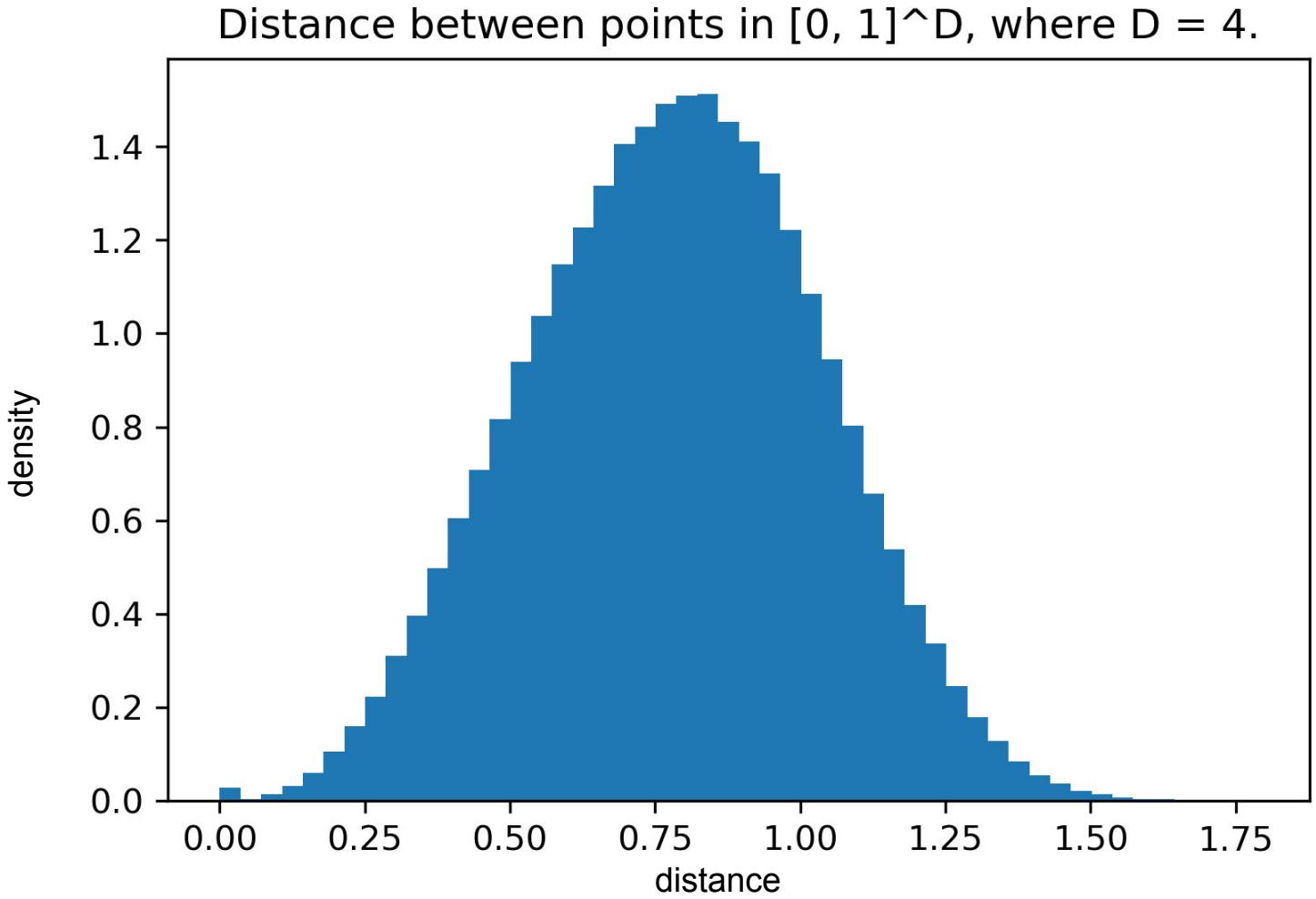
High-dimensional spaces are challenging to work with.

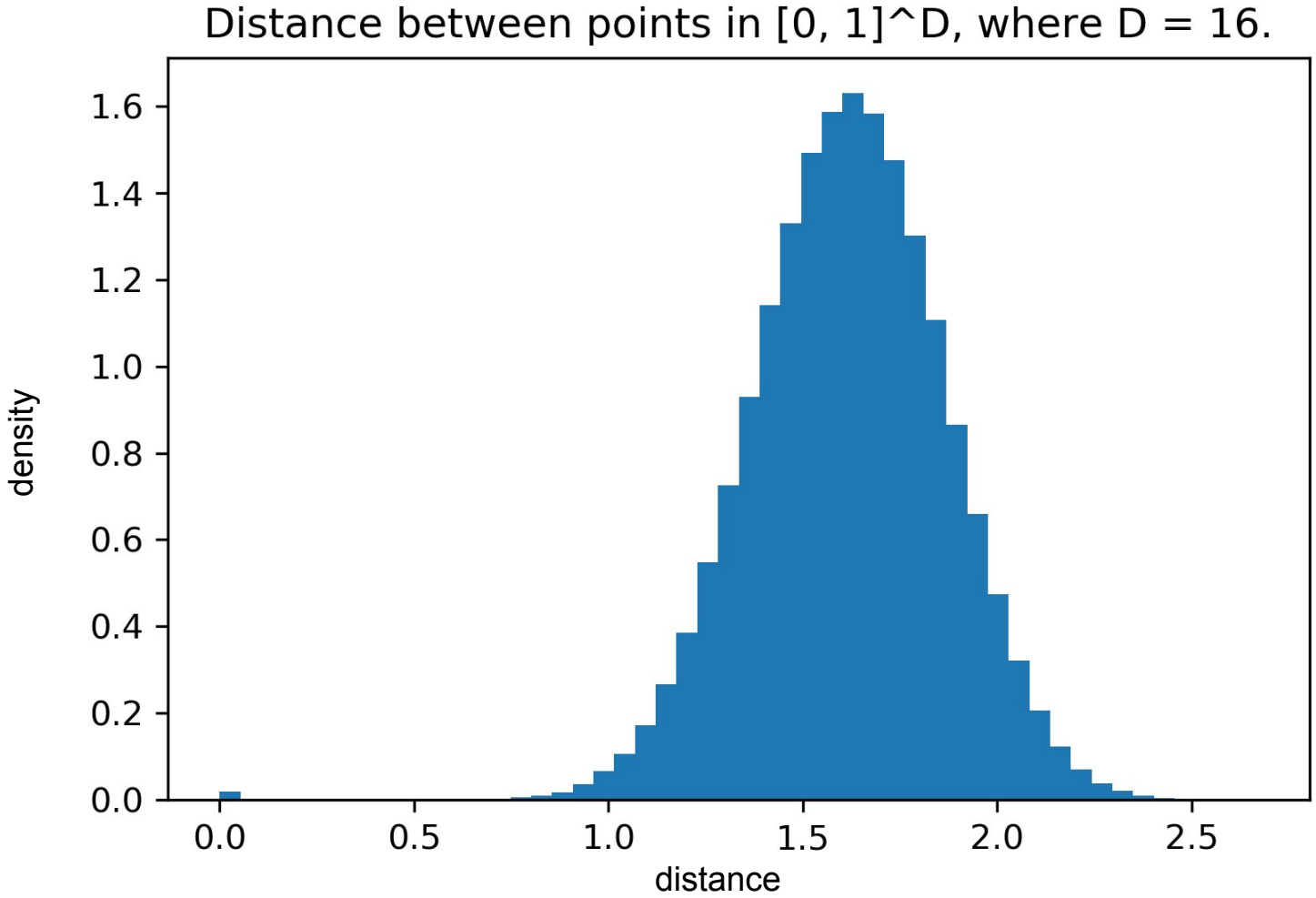
Curse of Dimensionality: strange things happen in very high-dimensional spaces

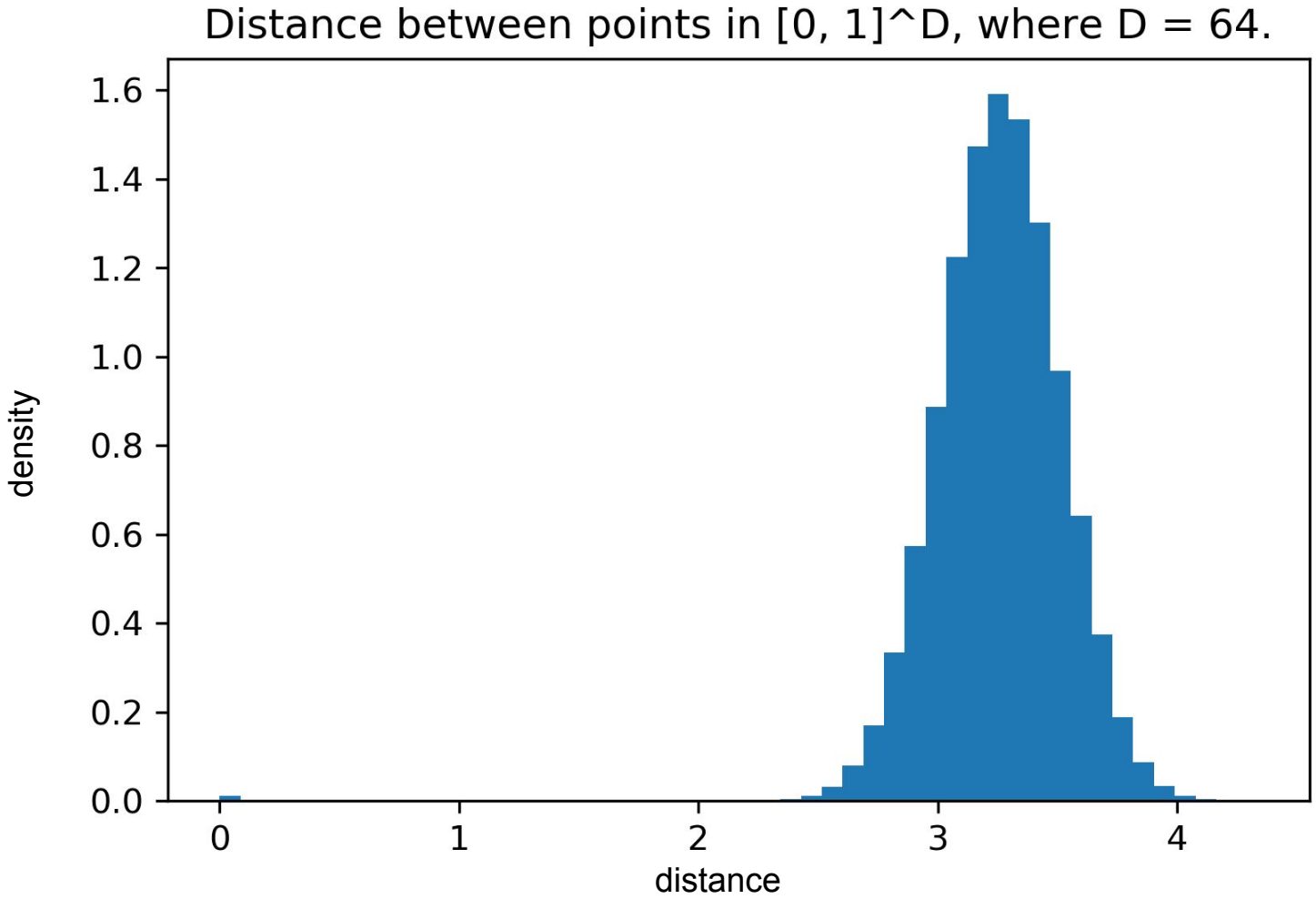
- A fixed number of datapoints occupy a fraction of the space that decreases exponentially with the number of dimensions.
- The portion of a volume near the surface increases in the number of dimensions.
- Distances between points become increasingly indistinguishable

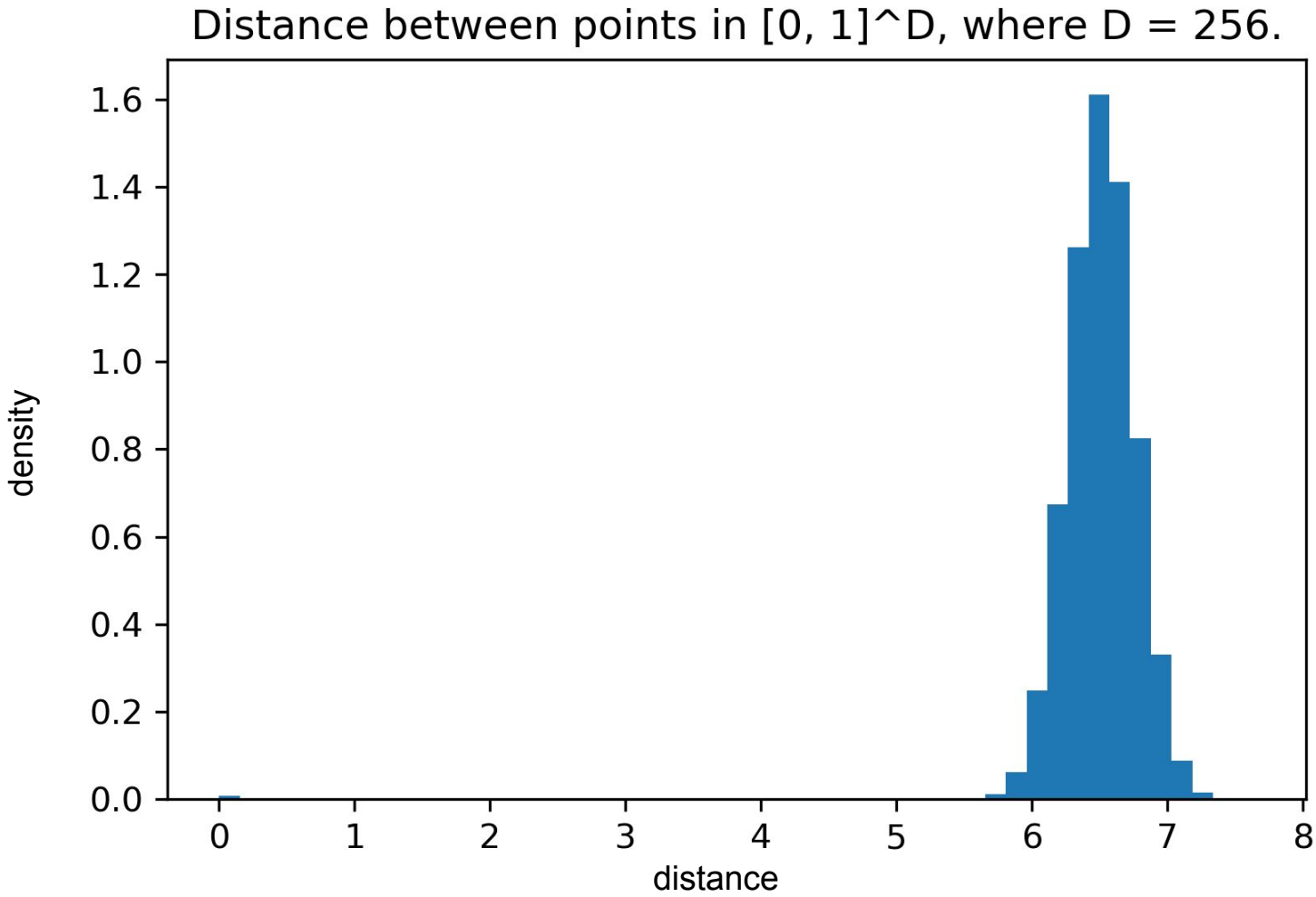

Distance between points in [0, 1]^D, where D = 1.
density
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.2
0.4
0.6
0.8
1.0
distance

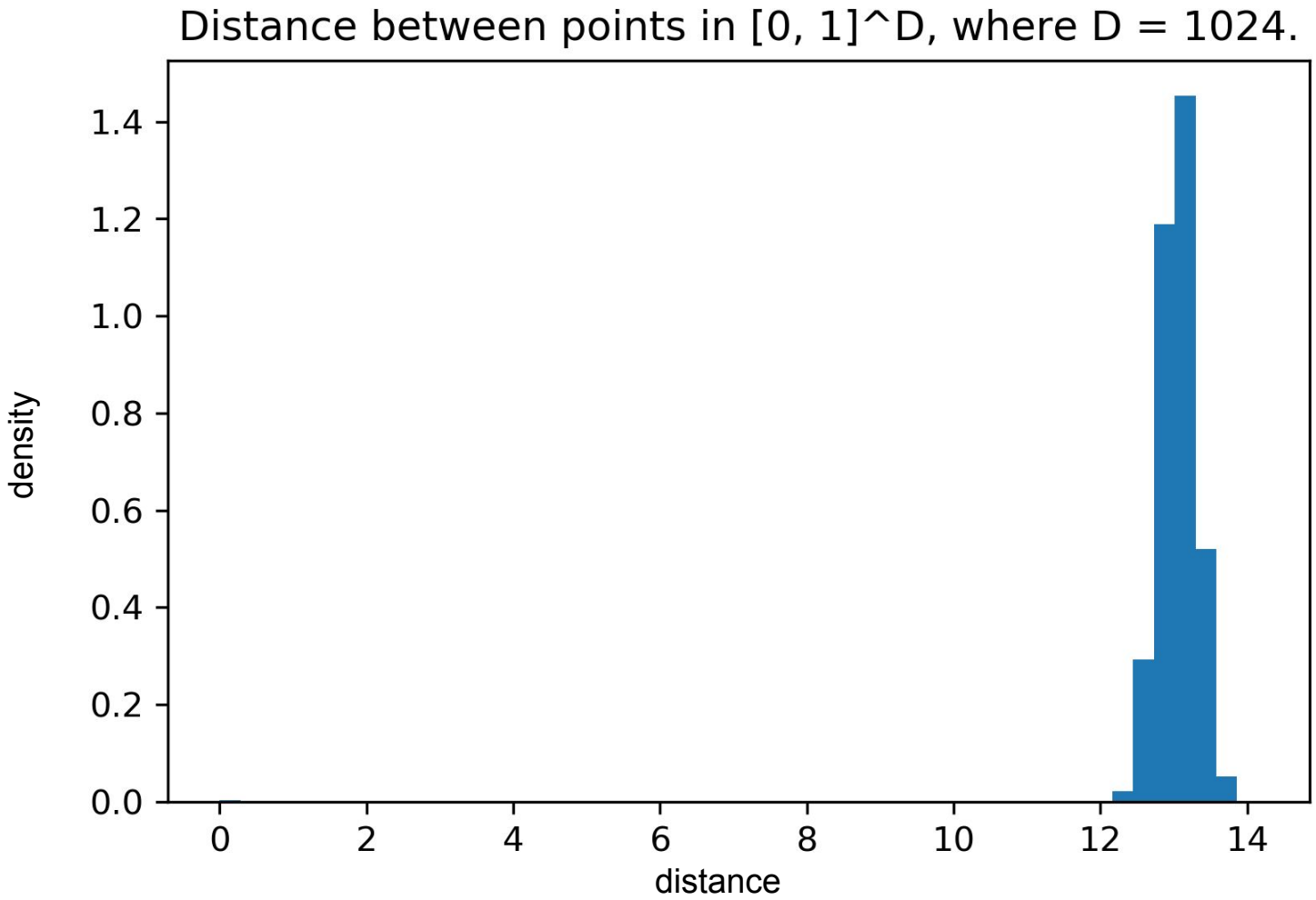

Distance between points in [0, 1]^D, where D = 2.
density
distance
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

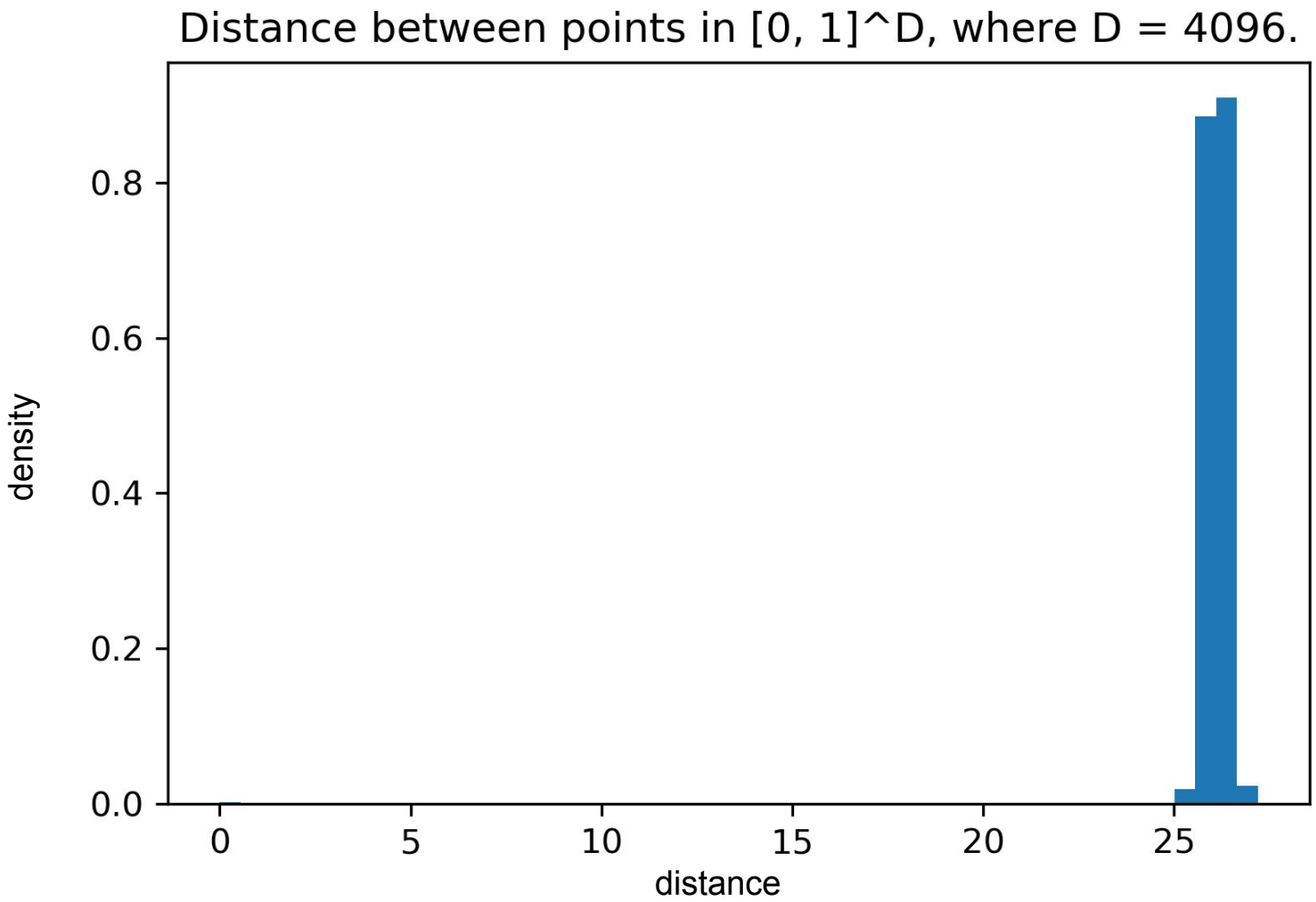

Distance between points in [0, 1]^D, where D = 4.
density
distance
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75

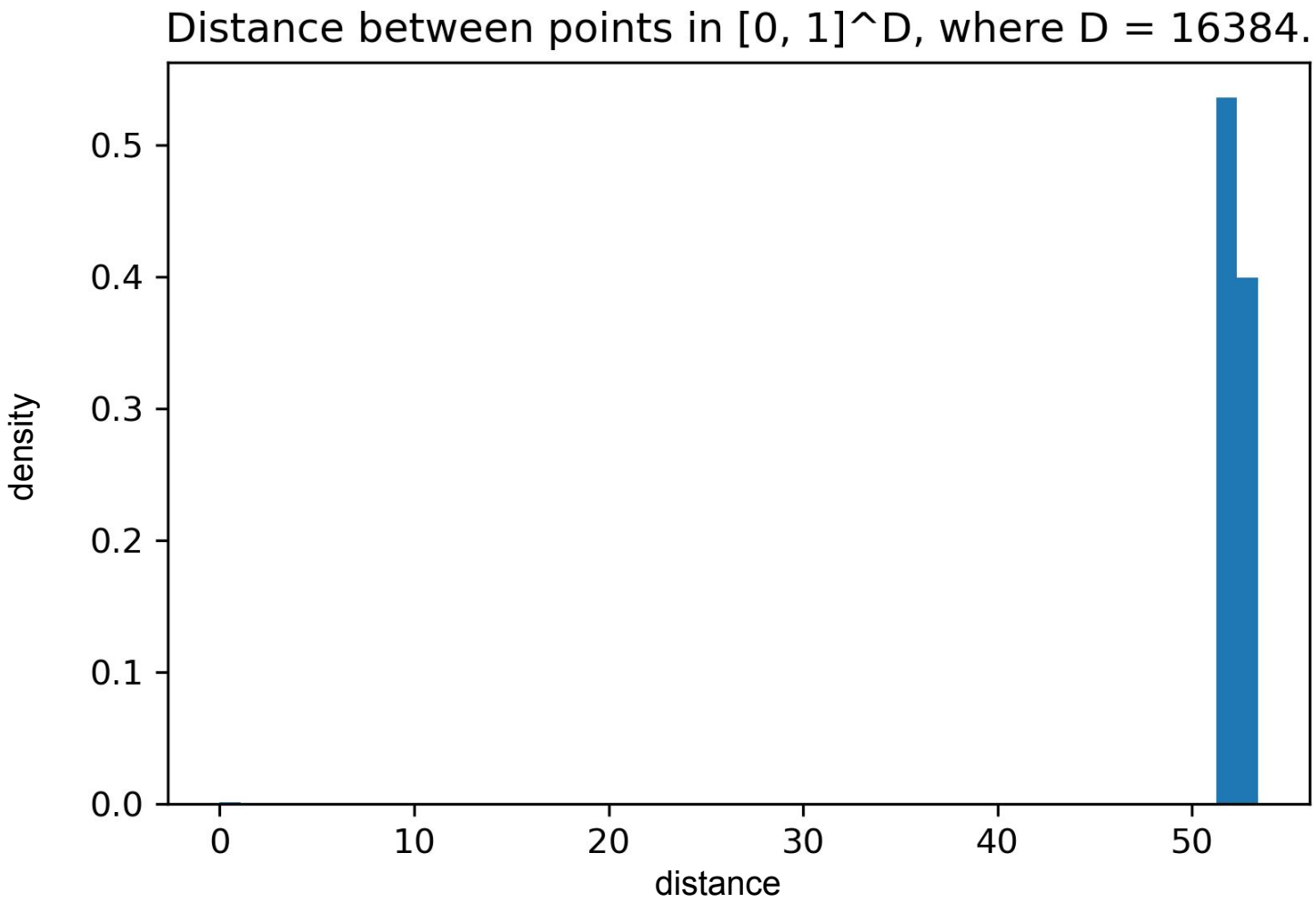

Distance between points in [0, 1]^D, where D = 16.
density
distance
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
0.0
0.5
1.0
1.5
2.0
2.5

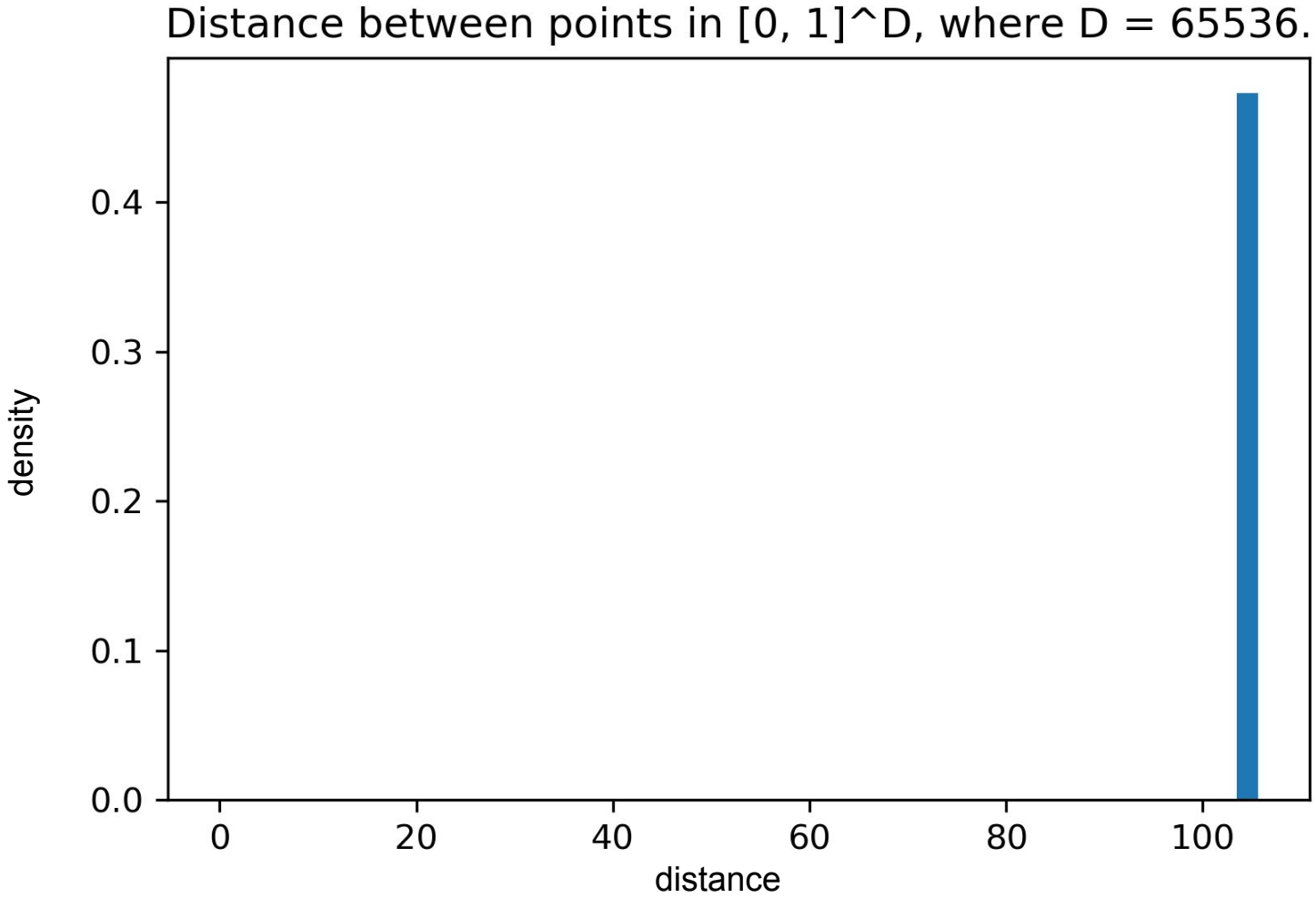

Distance between points in [0, 1]^D, where D = 64.
density
distance
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
0
1
2
3
4


Distance between points in [0, 1]^D, where D = 256.
density
distance
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
0
1
2
3
4
5
6
7
8

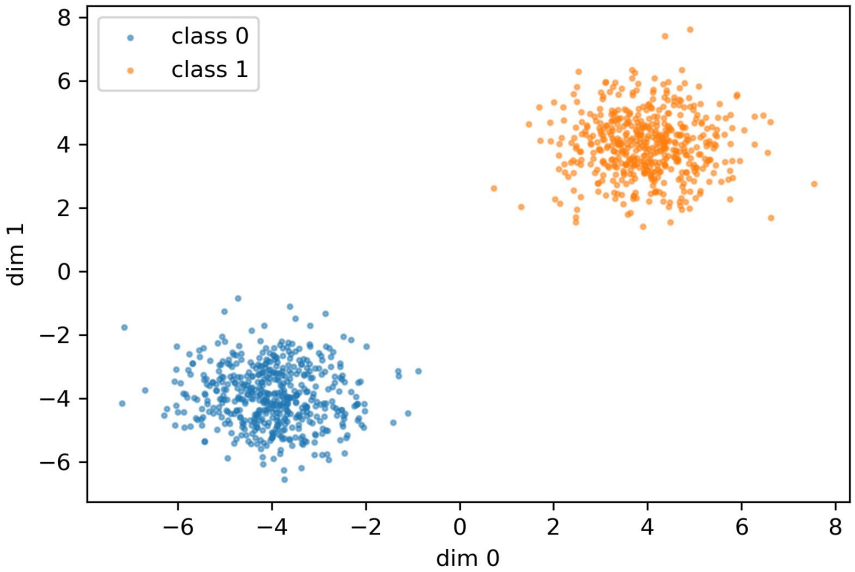

Distance between points in [0, 1]^D, where D = 1024.
density
distance
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0
2
4
6
8
10
12
14


Distance between points in [0, 1]^D, where D = 4096.
density
0.0
0.2
0.4
0.6
0.8
0
5
10
15
20
25
distance

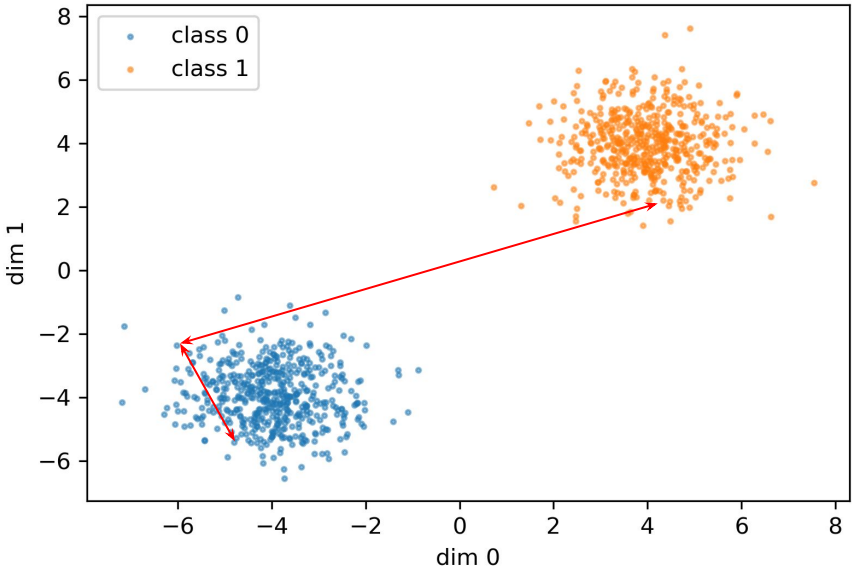

Distance between points in [0, 1]^D, where D = 16384.
density
0.0
0.1
0.2
0.3
0.4
0.5
0
10
20
30
40
50
distance

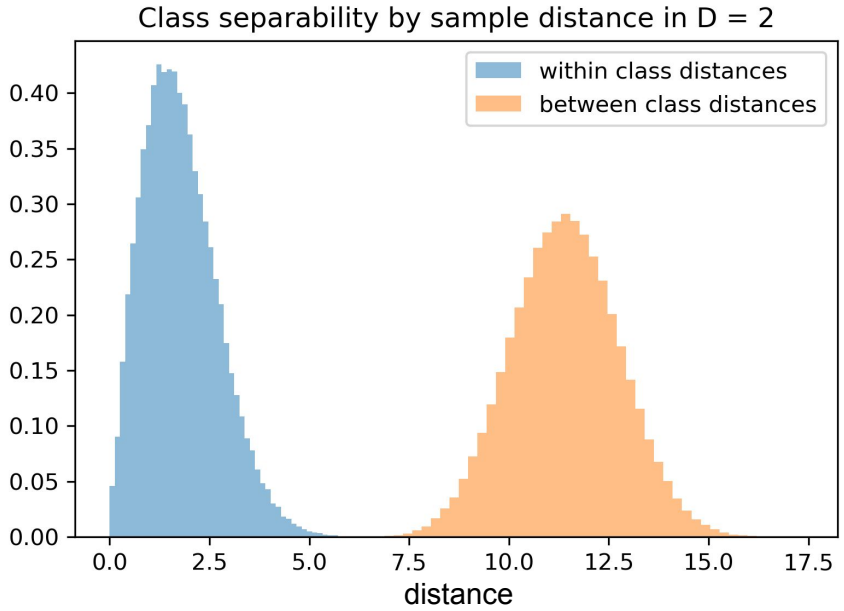

Distance between points in [0, 1]^D, where D = 65536.
density
0.0
0.1
0.2
0.3
0.4
0
20
40
60
80
100
distance

Distractors

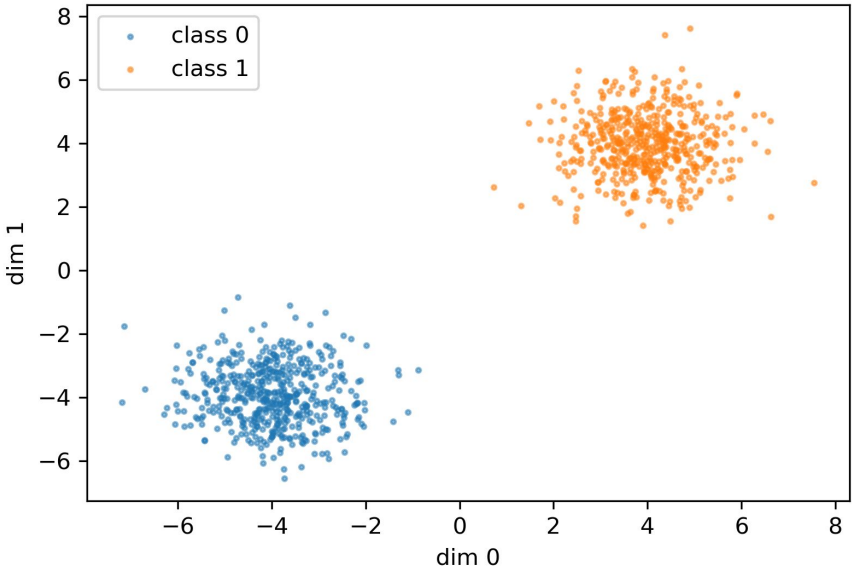


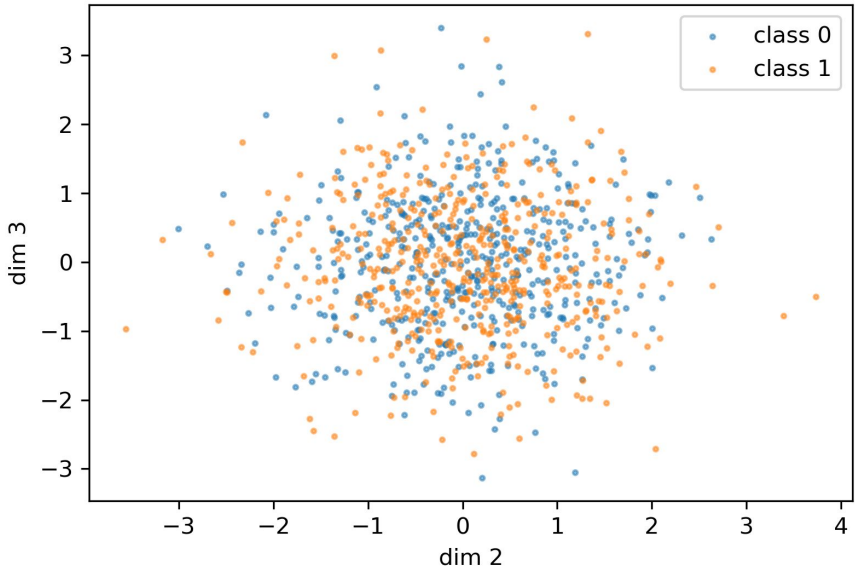
Distractors

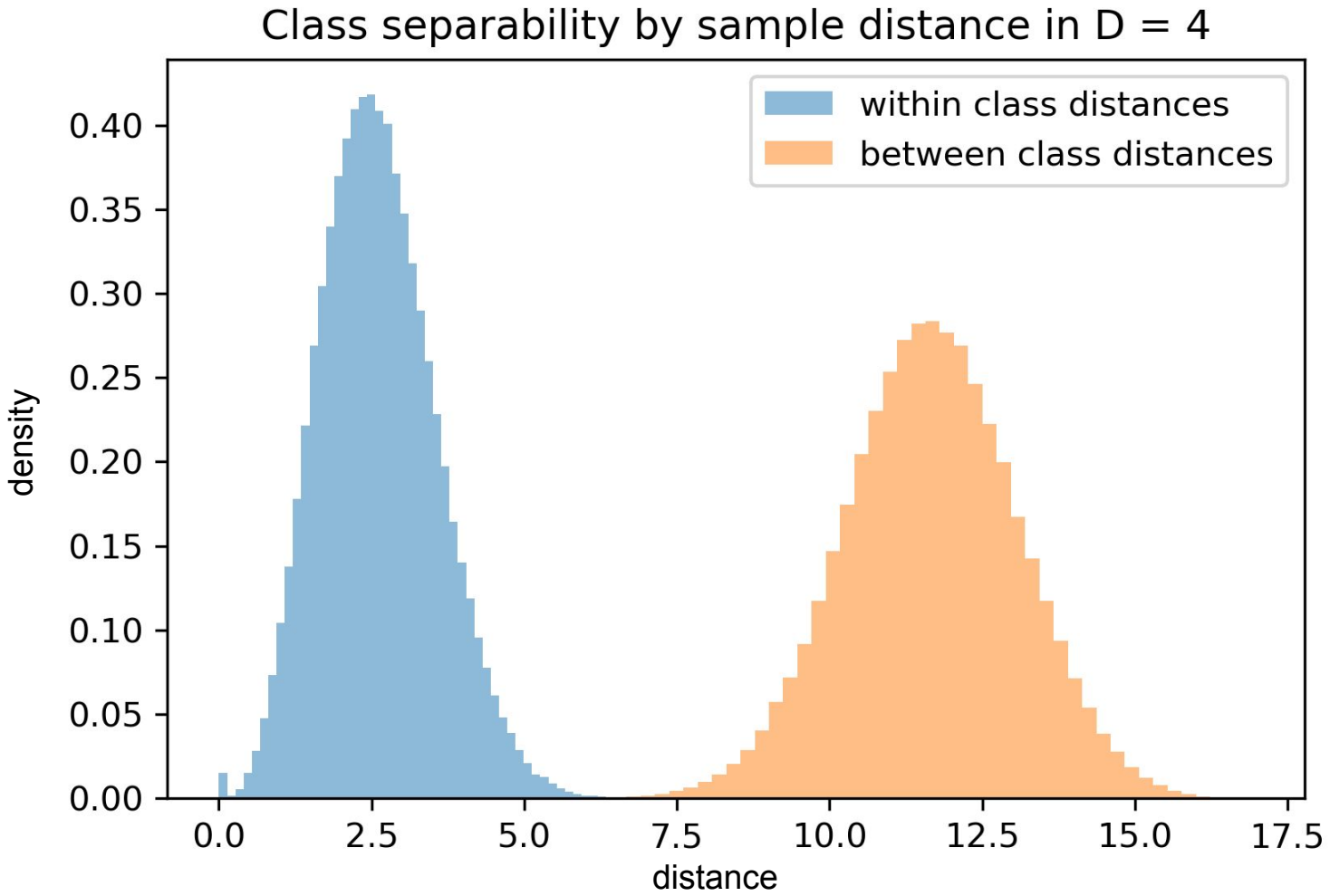


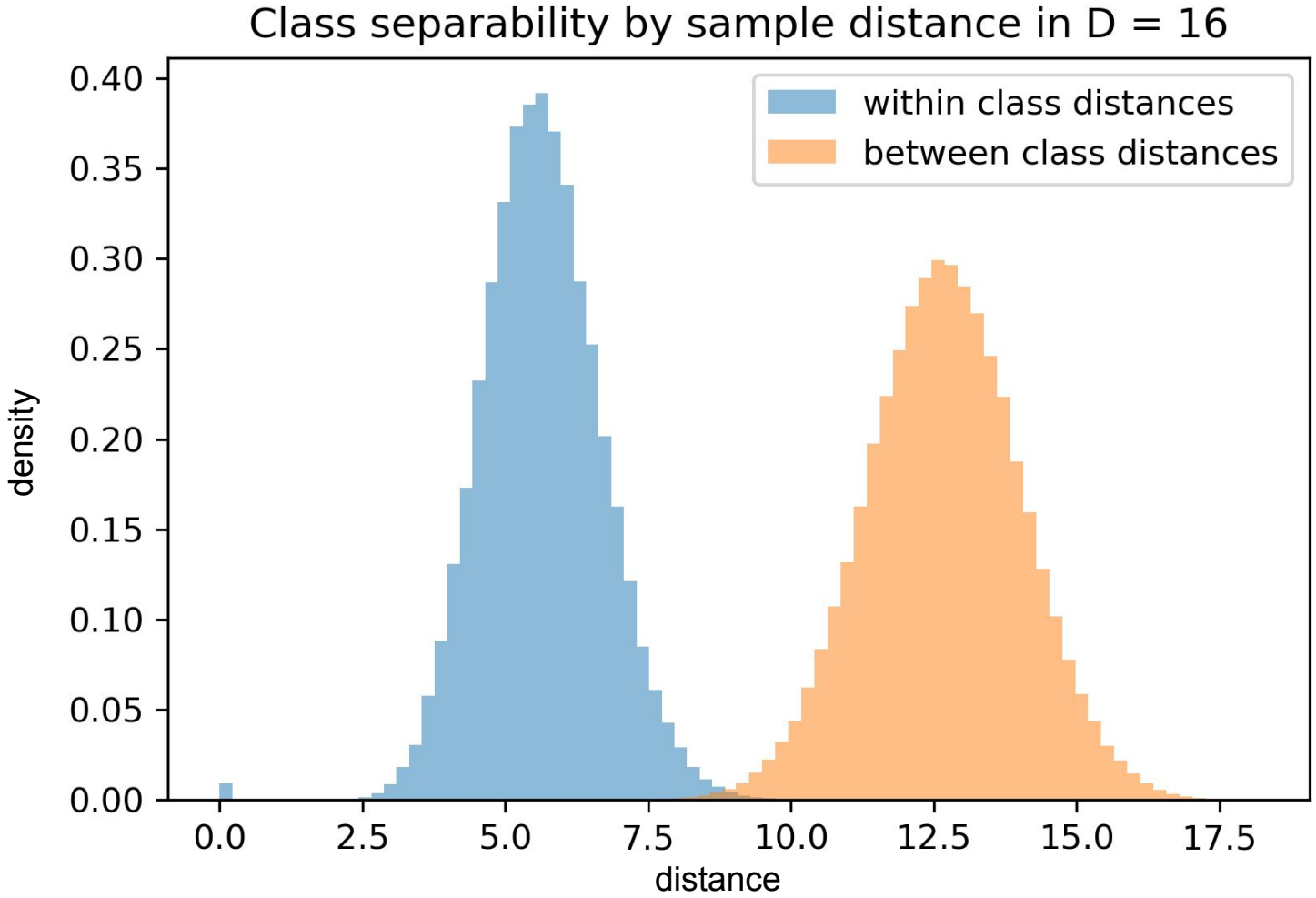


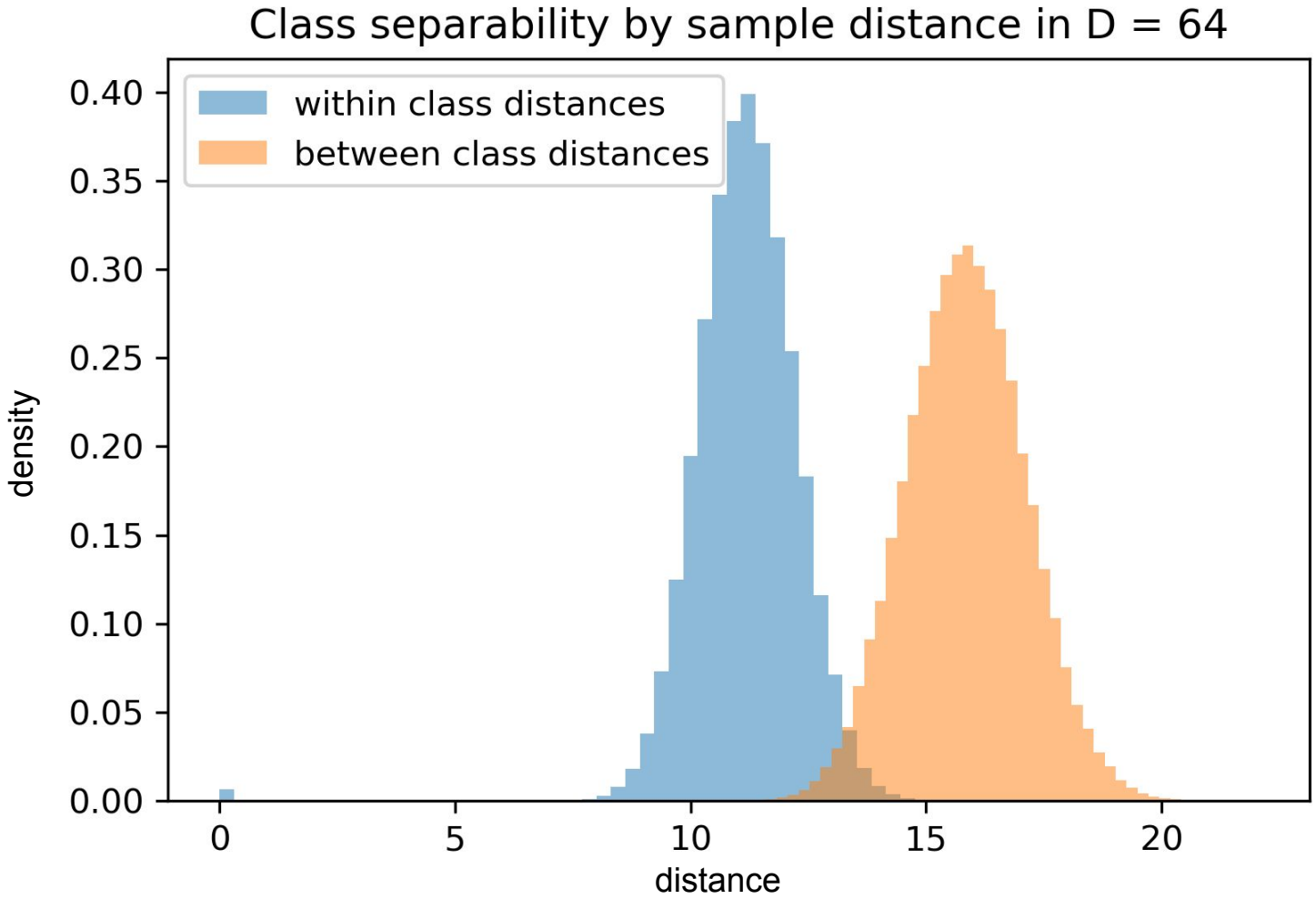
Distractors

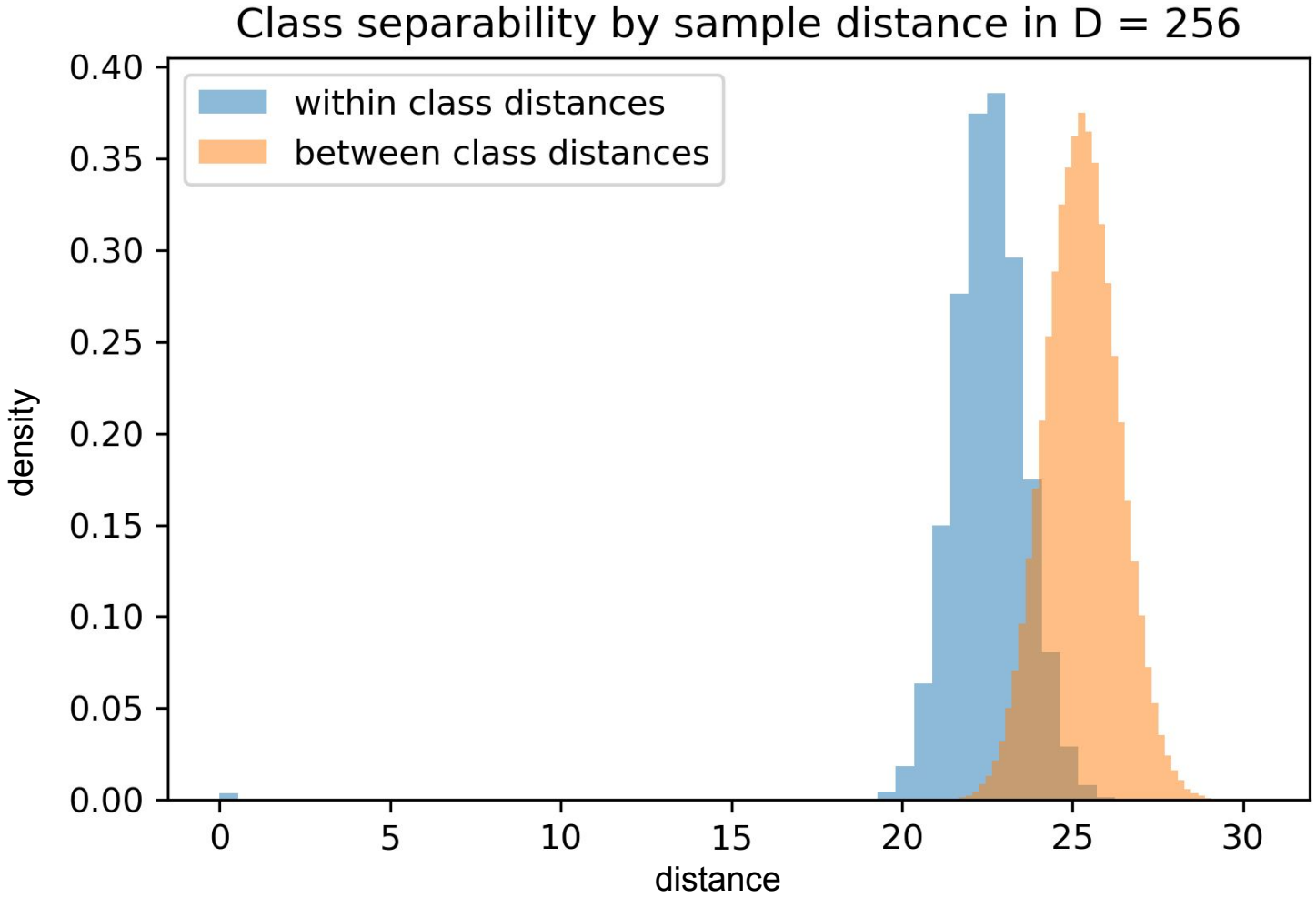


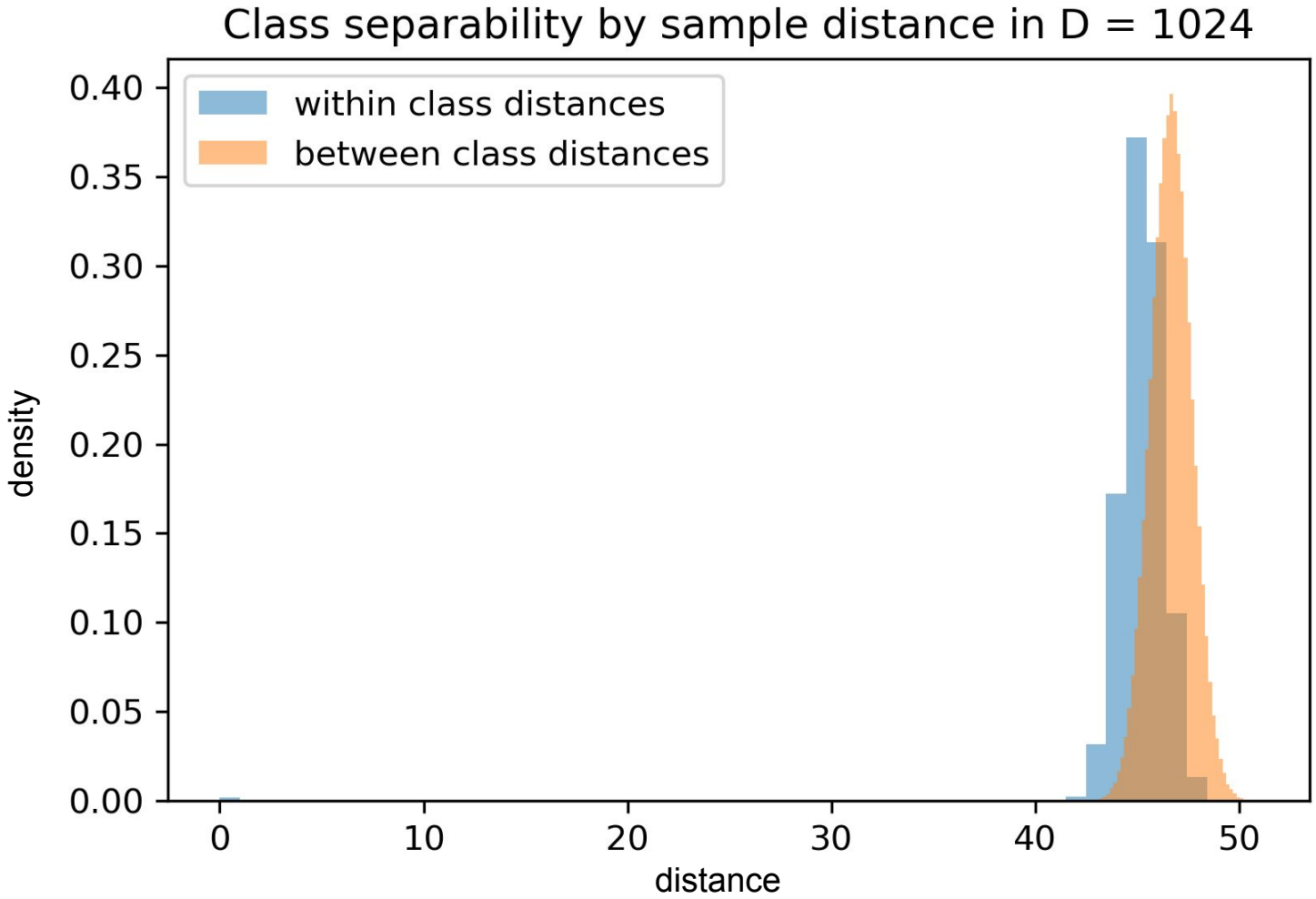


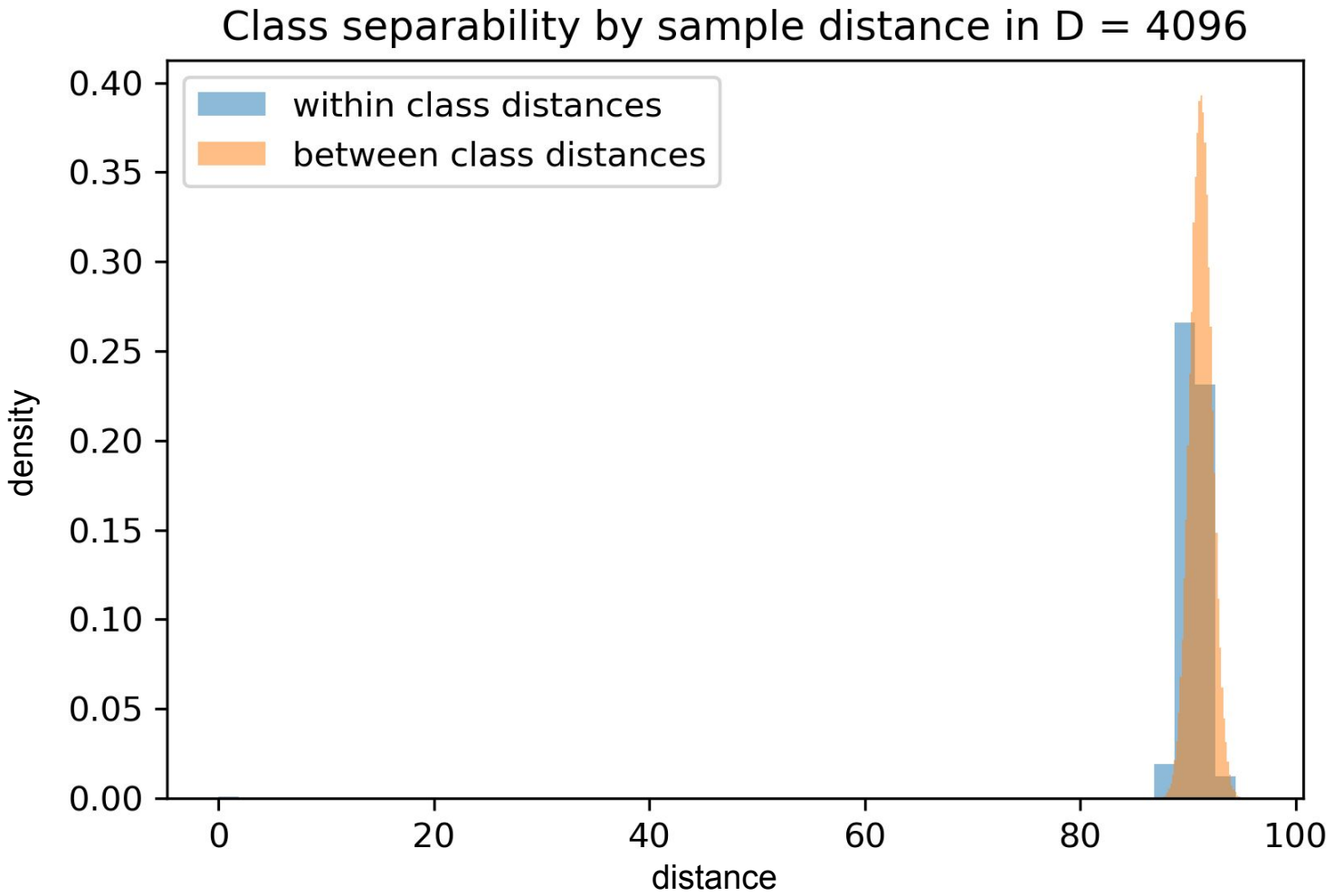

Class separability by sample distance in D = 4
within class distances
between class distances
density
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
distance

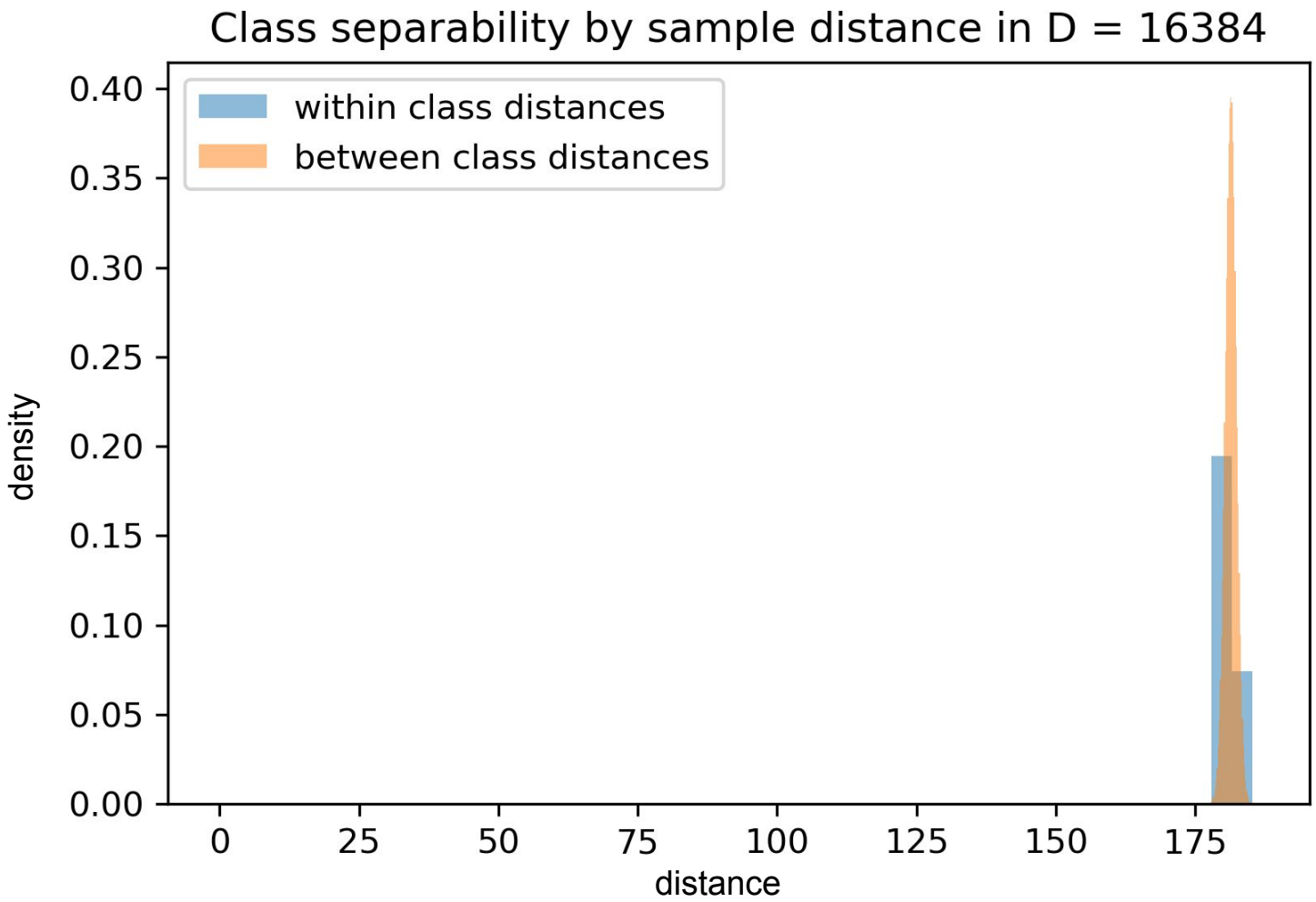

Class separability by sample distance in D = 16
within class distances
between class distances
density
distance
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

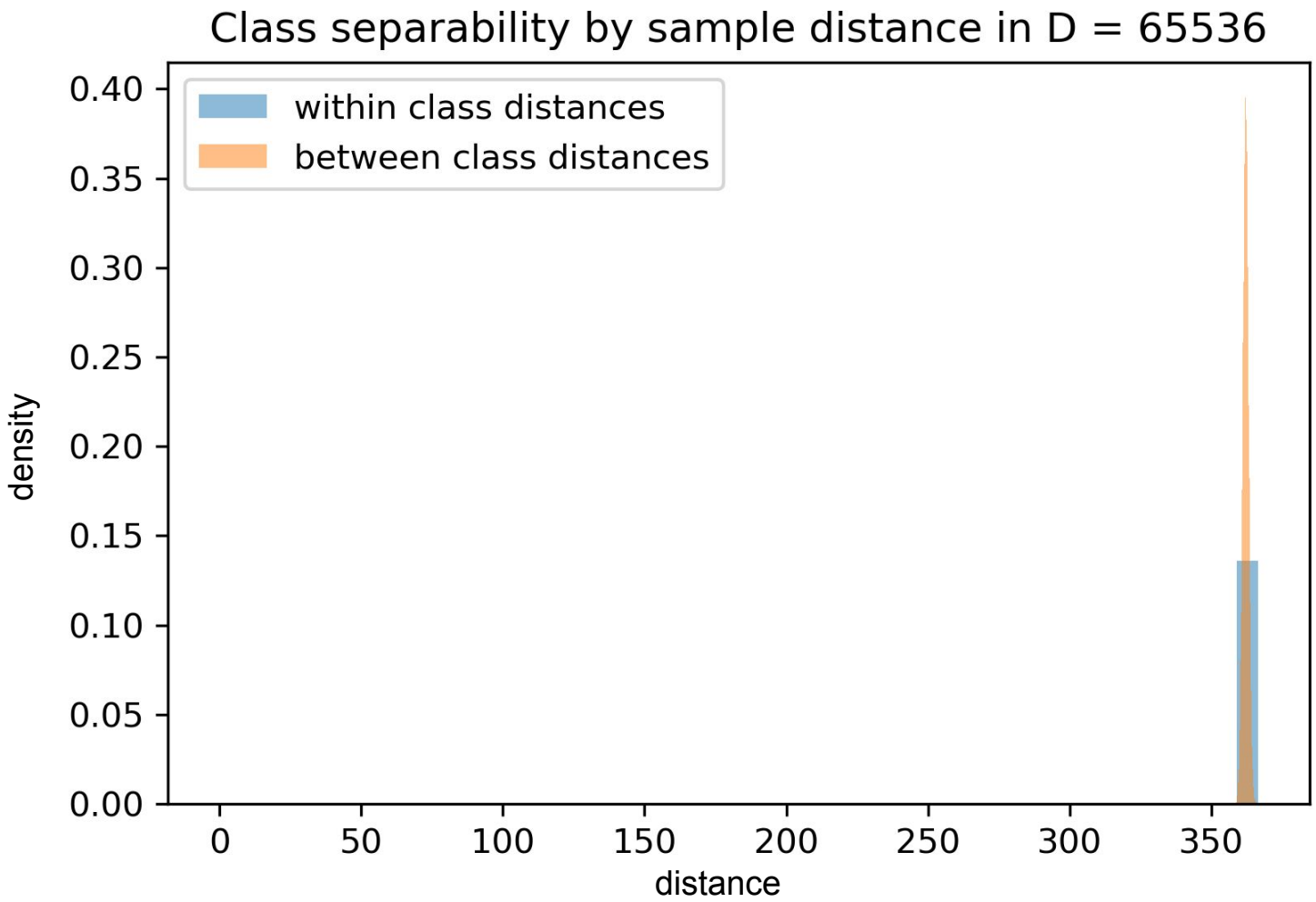

Class separability by sample distance in D = 64
within class distances
between class distances
density
distance
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0
5
10
15
20


Class separability by sample distance in D = 256
within class distances
between class distances
density
distance
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0
5
10
15
20
25
30


Class separability by sample distance in D = 1024
within class distances
between class distances
density
distance
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0
10
20
30
40
50


Class separability by sample distance in D = 4096
within class distances
between class distances
density
distance
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0
20
40
60
80
100

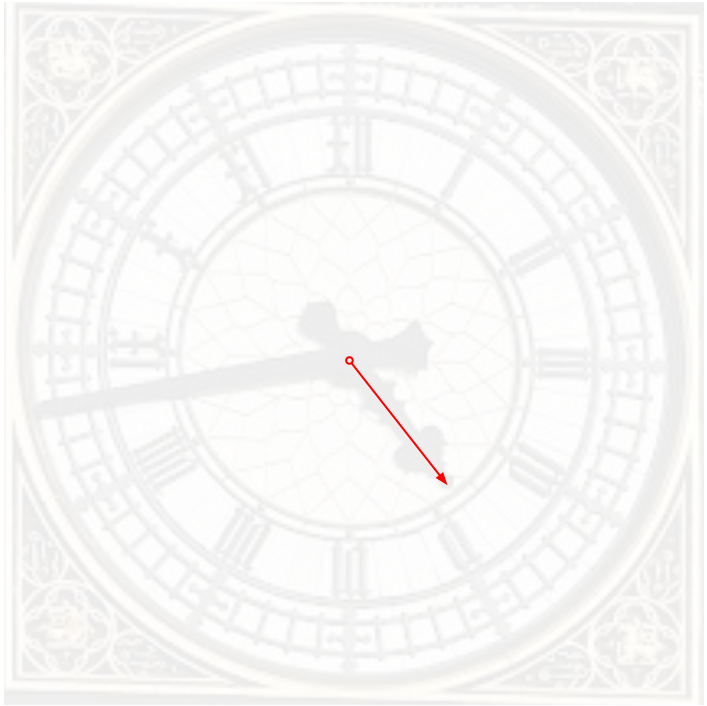

Class separability by sample distance in D = 16384
within class distances
between class distances
density
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
25
50
75
100
125
150
175
distance


Class separability by sample distance in D = 65536
within class distances
between class distances
density
distance
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0
50
100
150
200
250
300
350

Which features are useful?

what time is it?

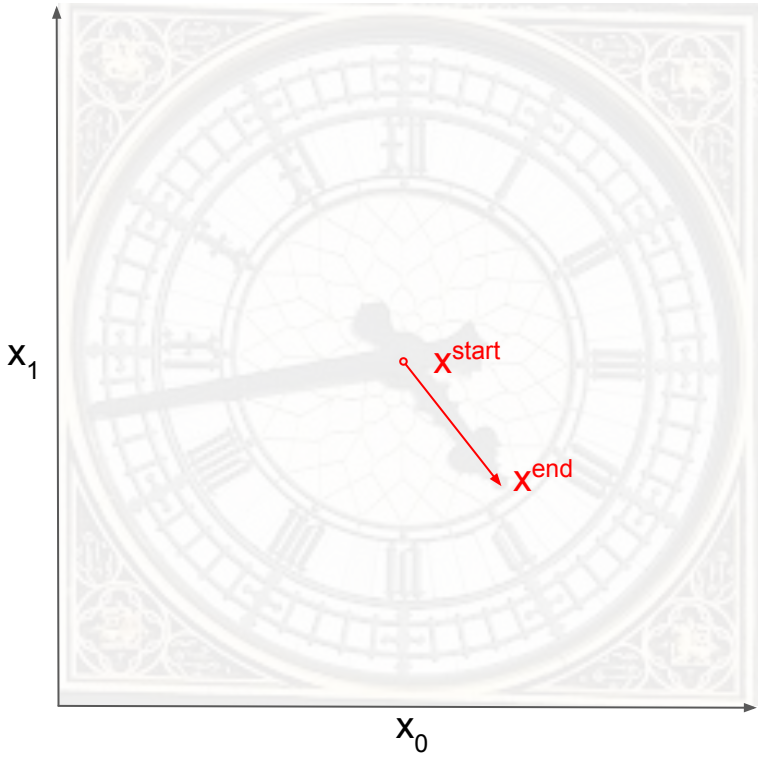
Which features are useful?



What time is it?

- Hour hand is sufficient to tell time

Which features are useful?



What time is it?

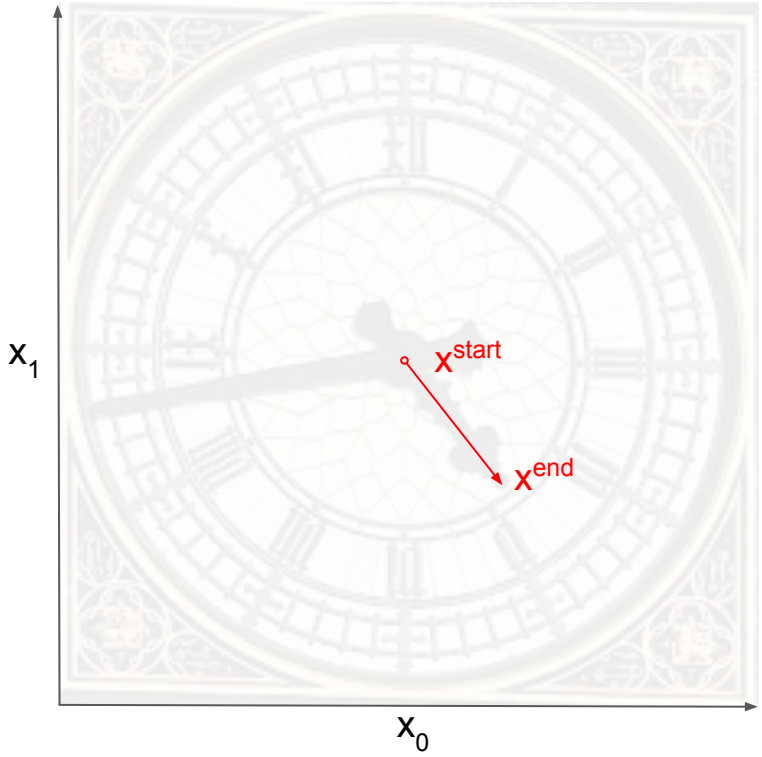
- Hour hand is sufficient to tell time
- Assume we had an hour hand detector

$$[x_0^{start}, x_1^{start}, x_0^{end}, x_1^{end}]$$

What qualities does a good representation have?

- Geometry of the problem
- Translation invariance
- Scale invariance
- Orientation invariance

Which features are useful?



What time is it?

- Hour hand is sufficient to tell time
- Assume we had an hour hand detector

$$[x_0^{start}, x_1^{start}, x_0^{end}, x_1^{end}]$$

Polar coordinate representation

$$[d_0 : (x_0^{end} - x_0^{start}), d_1 : (x_1^{end} - x_1^{start})]$$

$$\left[r : \sqrt{d_0^2 + d_1^2}, \varphi : atan2(d_1, d_0)\right]$$

Scale invariance

$$[\varphi]$$

Orientation Invariance, assuming some reference angle

$$[\varphi - \theta]$$

Are the same features useful for different tasks?

Is this a clock or pressure gauge?

Which features are useful?

What year was the clock built?

Which features are useful?

How will this image look 5 minutes later?

Which features are useful?

How does the surrounding context look?

Overview

1. Case Study 3
2. What is Feature Engineering?
3. Feature Selection: Finding Relevant Subsets of Features
    - 3.1. Filtering Methods
    - 3.2. Wrapper Methods
    - 3.3. Embedding Methods
4. Feature Extraction

Feature Selection

Context

Dataset with D features per datapoint.

Goal

Find the subspace of M << D features with minimal error on hold-out set.

- Brute-force subset evaluation is exponential in the number of features (NP-hard problem)
- Heuristics
  - Filtering: use surrogate measure and test individual features' relevance
  - Wrappers: greedily build feature set and evaluate using predictive model
  - Embedding: predictive model selects features during learning process (e.g., LASSO)

Filtering Methods

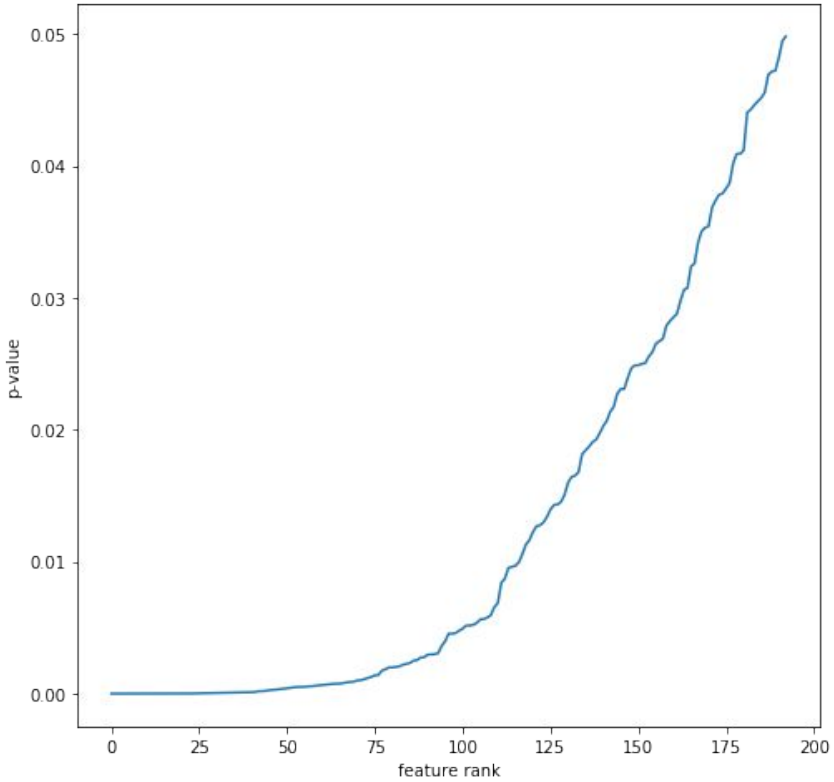
Idea:

1. Measure *relevance* of individual feature for dependent variable.
2. Rank features by relevance
3. Keep top K relevant features; k could be chosen via cross-validation

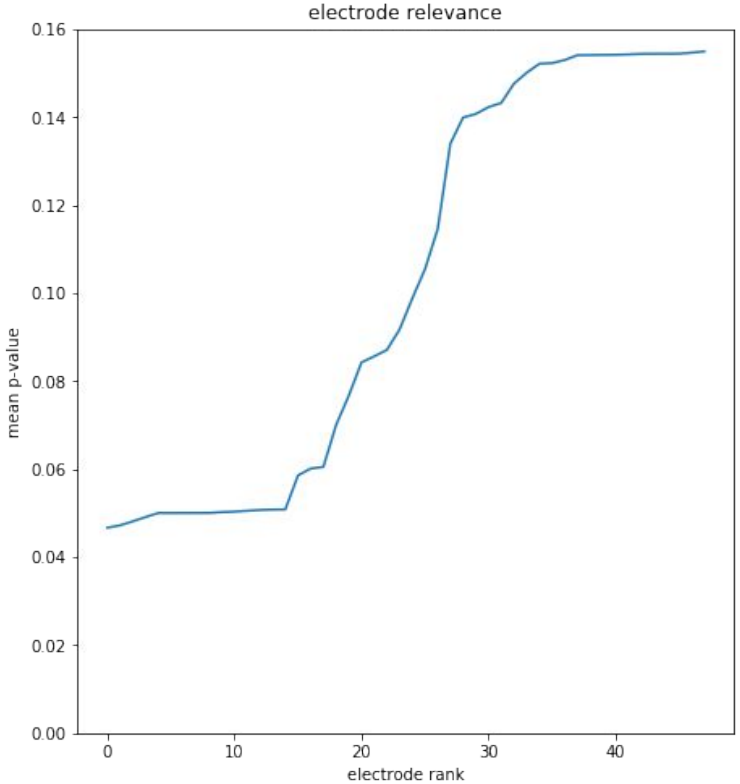
Statistical tests used as relevance scores:

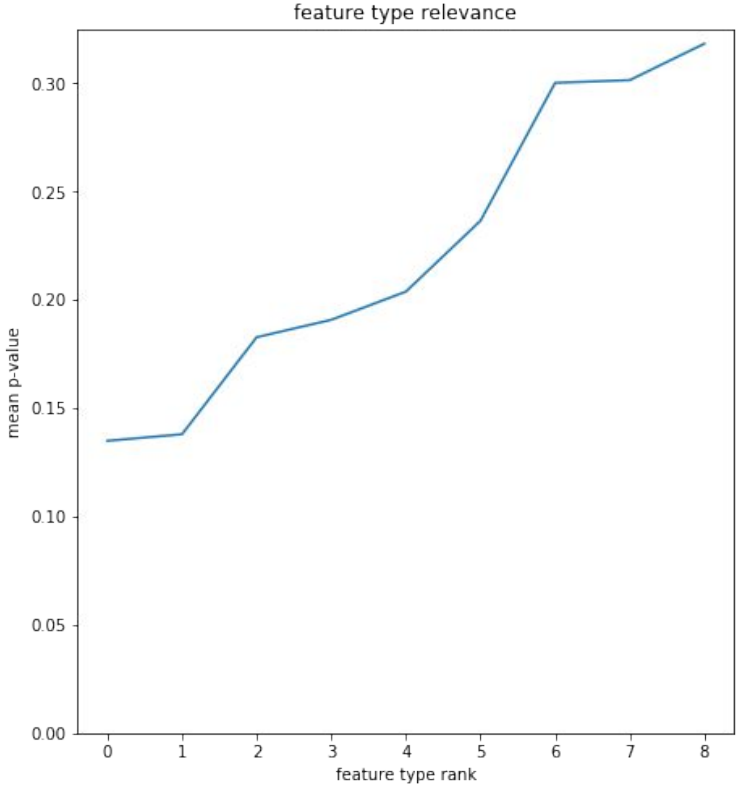
- Pearson Correlation
- Chi-Square
- Mutual Information
- t-test / ANOVA

Filtering: Case Study 3

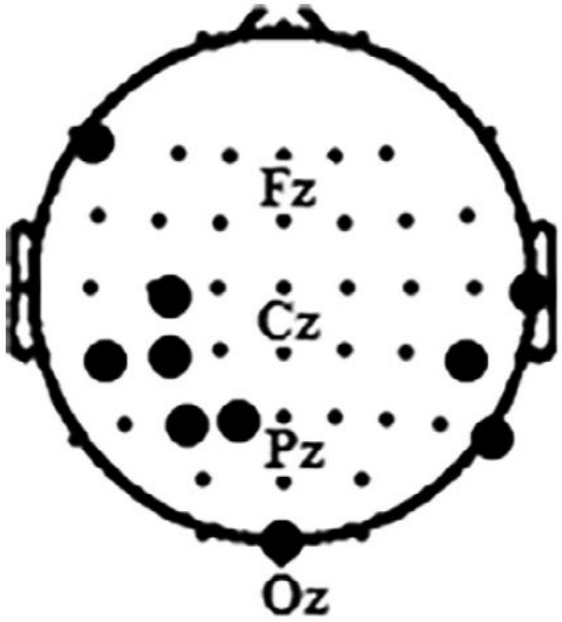


Filtering: Case Study 3





Filtering: Case Study 3



PDP vs PNP

Wrapper Methods

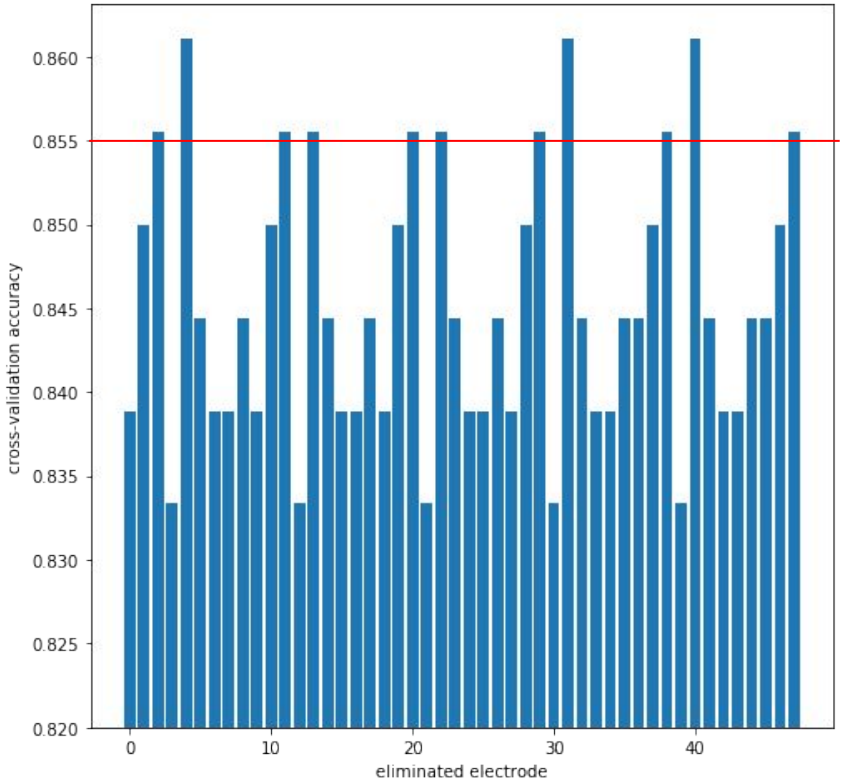
Backward feature elimination

1. Initialize the feature set F to include all features F_0: $\{f_0, f_1, \ldots, , f_{M-1}\}$ and evaluate performance
2. Evaluate performance with feature sets $F \backslash f_i$, removing a single feature f_i from F.
3. Update F to exclude f_i for the one feature that maximally maintained or increased performance.
4. Repeat steps 2 and 3 until
    a. Performance degrades
    b. Target number of features is reached

Backward Electrode Elimination: Case Study 3

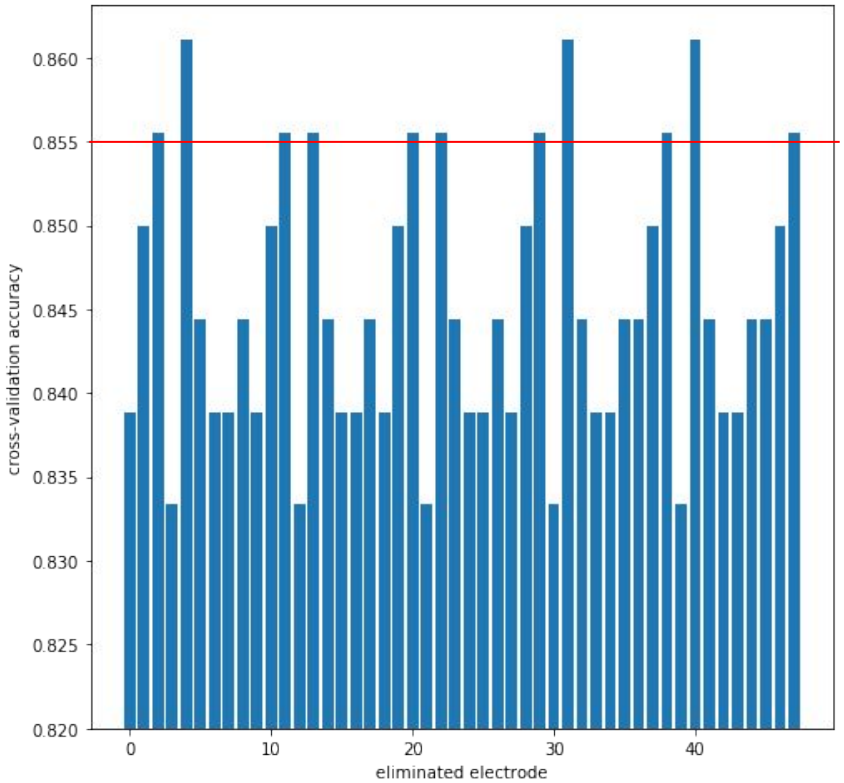
Iteration 1

baseline performance



Iteration 1

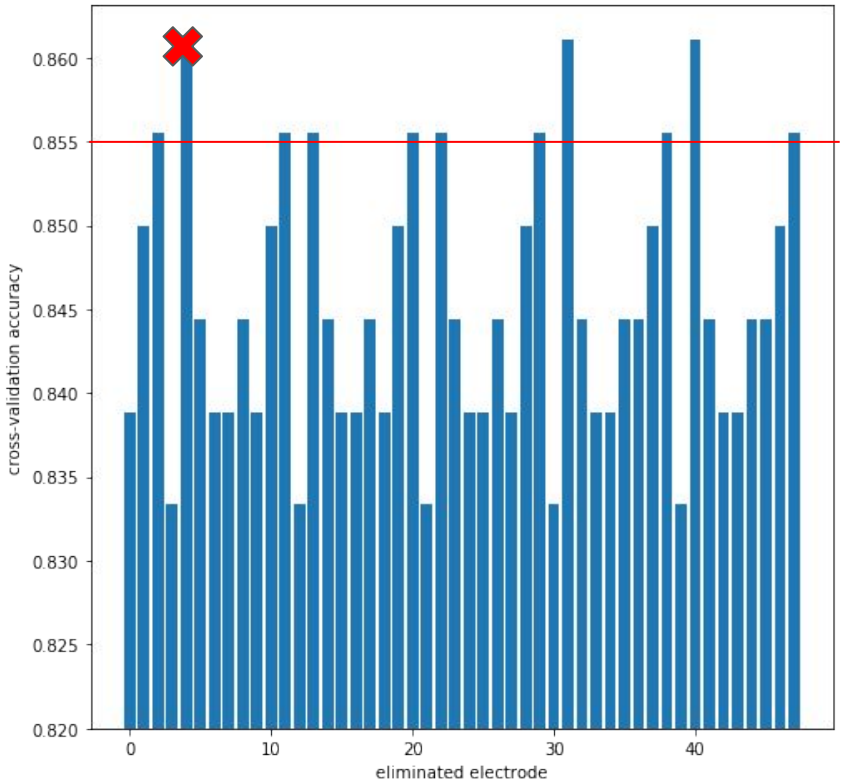
baseline performance



← Removing any of these electrodes improves performance

Iteration 1

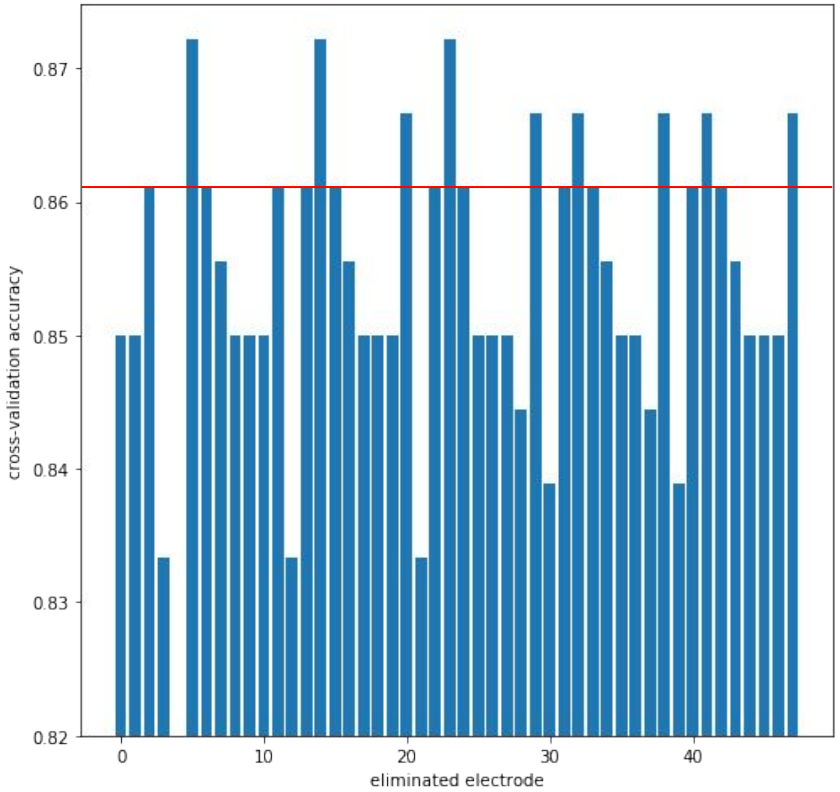
baseline performance



← Removing electrode 4

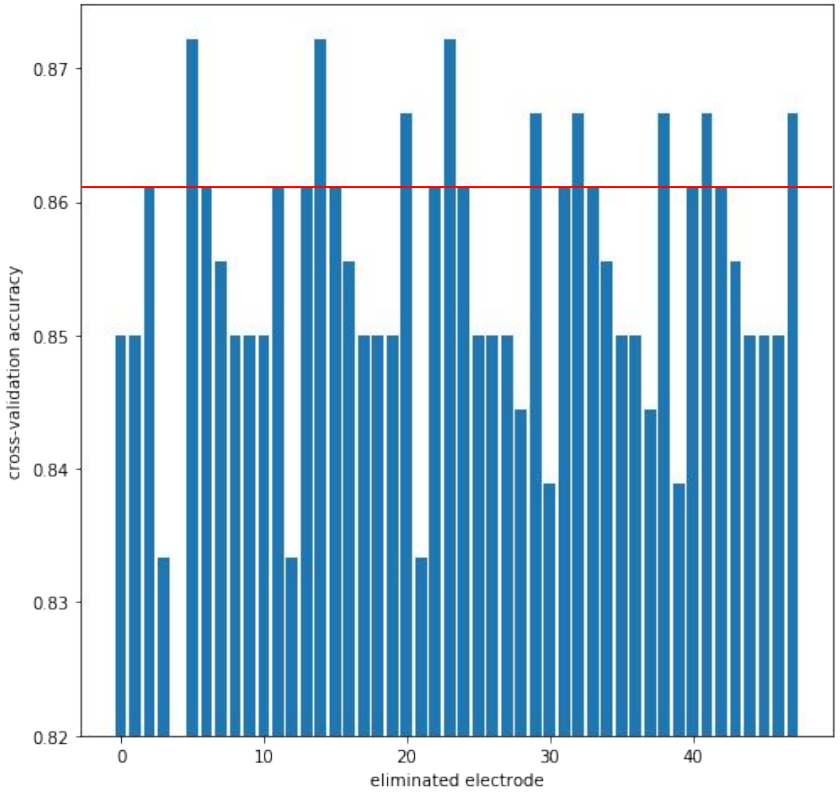
Iteration 2

new baseline performance



Iteration 2

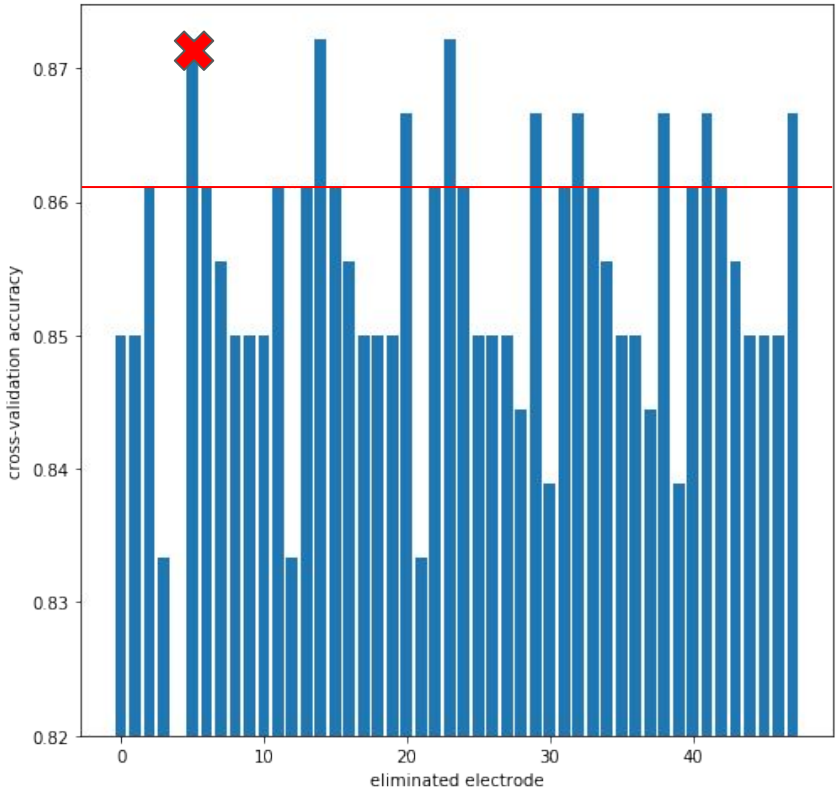
new baseline performance



← Removing any of these electrodes improves performance

Iteration 2

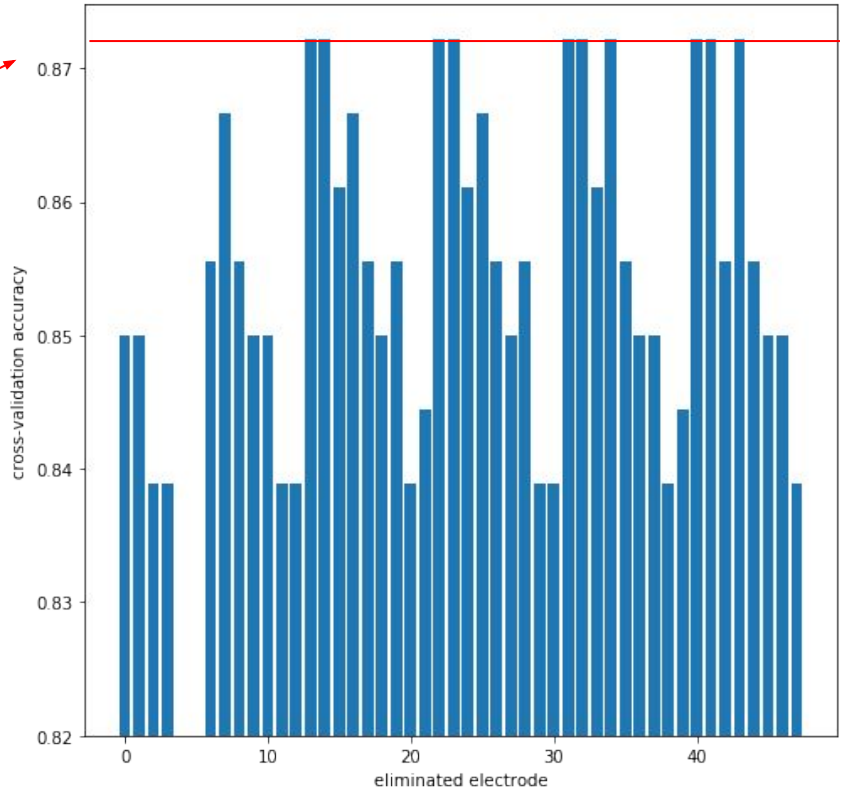
new baseline performance



← Removing electrode 5

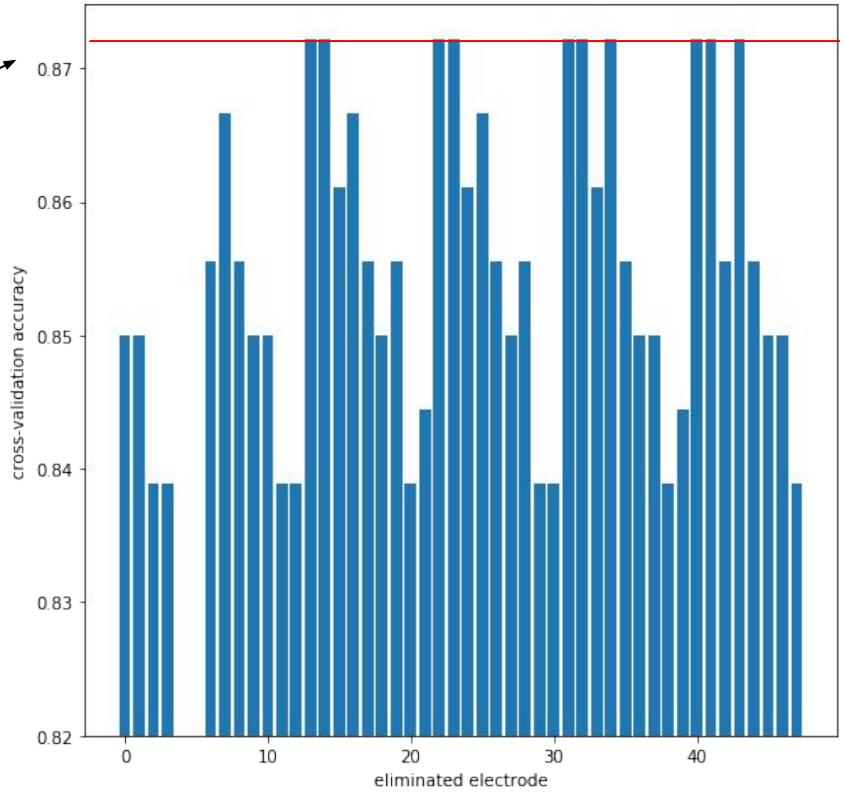
Iteration 3

new baseline performance



Iteration 3

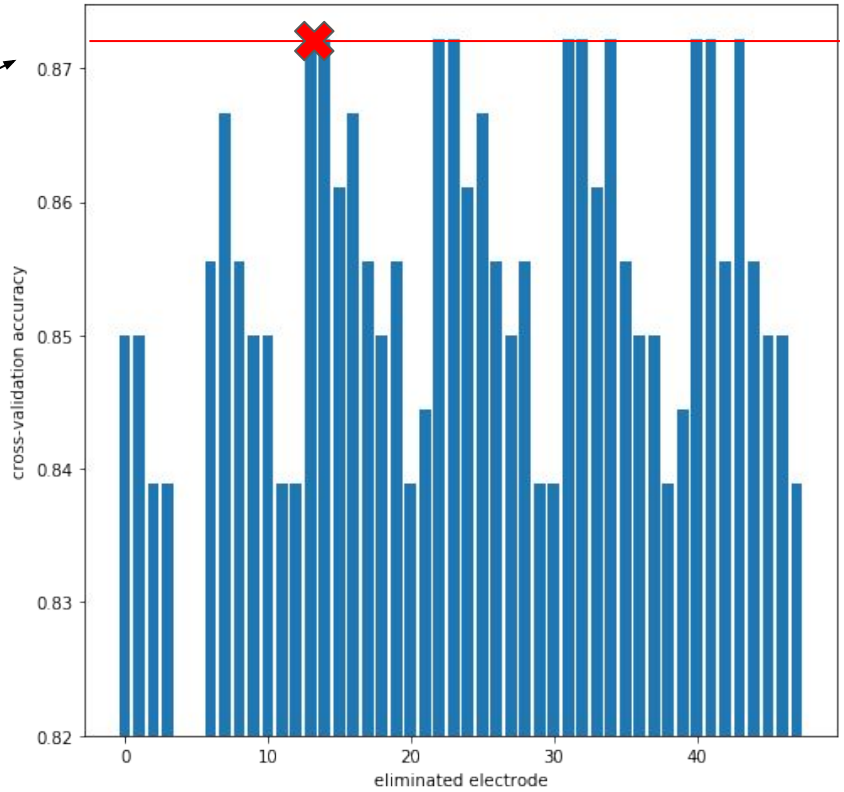
new baseline performance



Removing any of these electrodes does not degrade performance.

Iteration 3

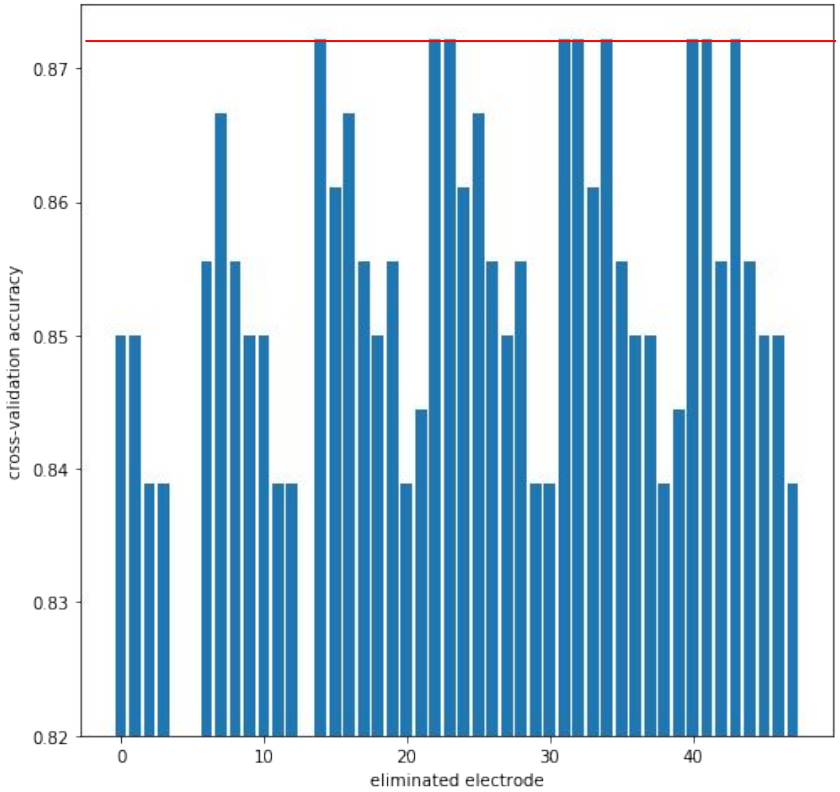
new baseline performance



Removing electrode 13

Iteration 4

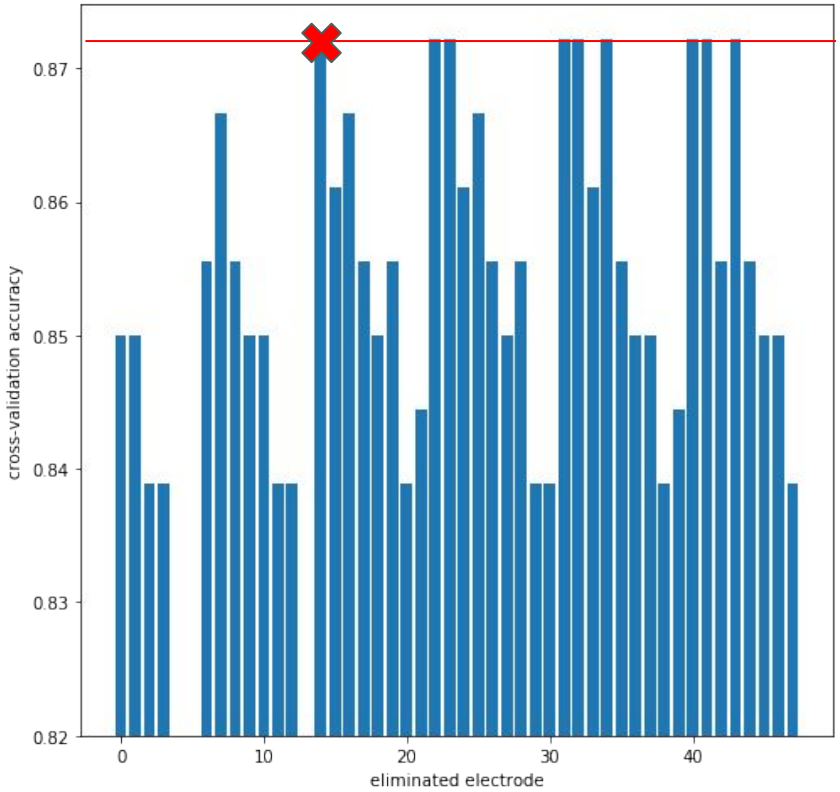
unchanged baseline performance



Removing any of these electrodes does not degrade performance.

Iteration 4

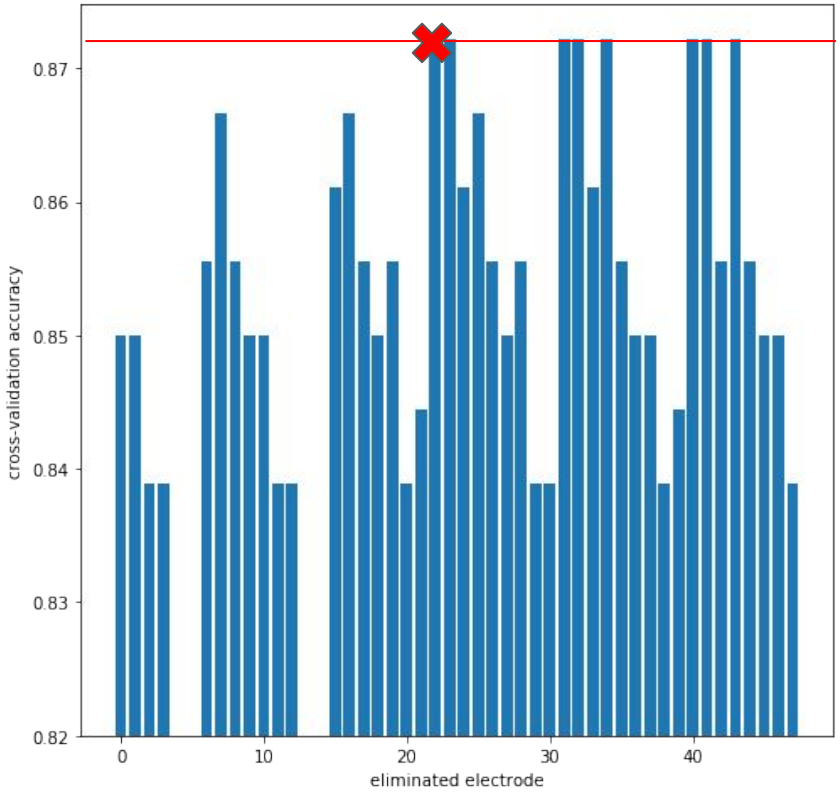
unchanged baseline performance



Removing electrode 14

Iteration 5

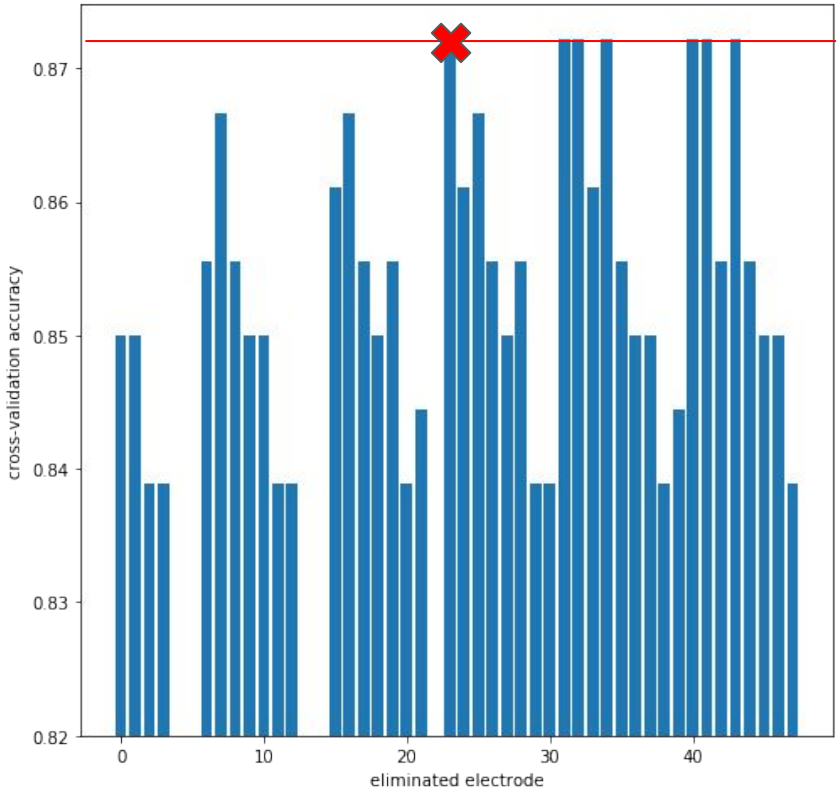
unchanged baseline performance



Removing electrode 22

Iteration 6

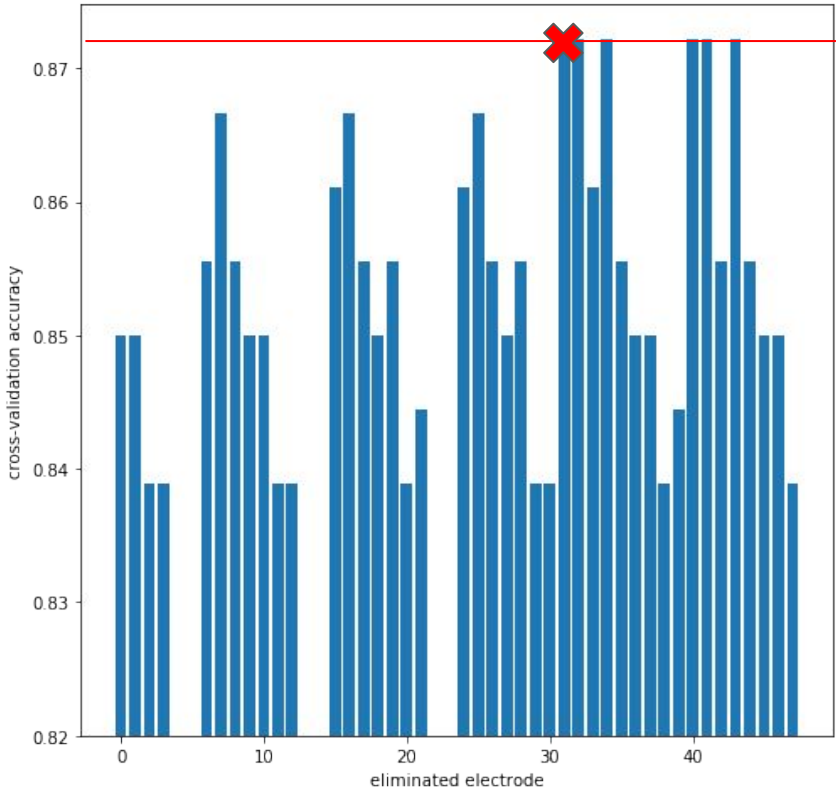
unchanged baseline performance



Removing electrode 23

Iteration 7

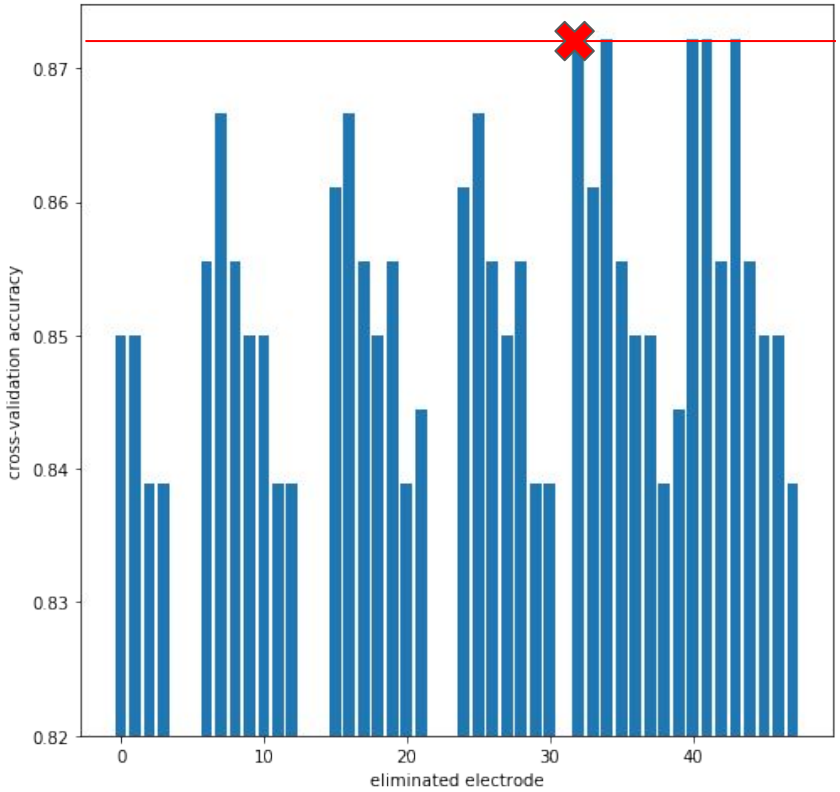
unchanged baseline performance



Removing electrode 31

Iteration 8

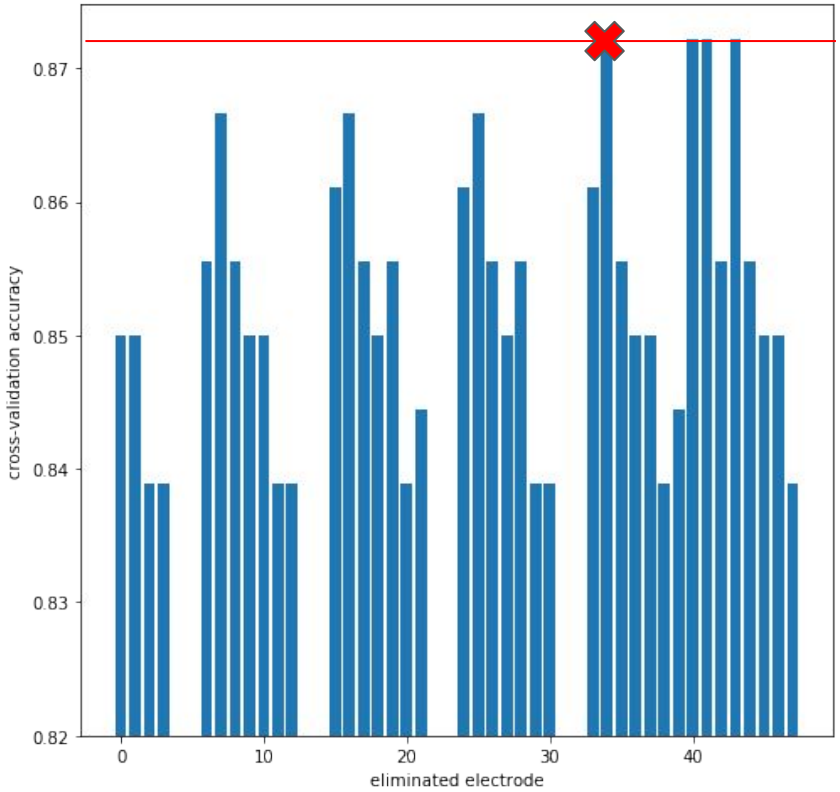
unchanged baseline performance



Removing electrode 32

Iteration 9

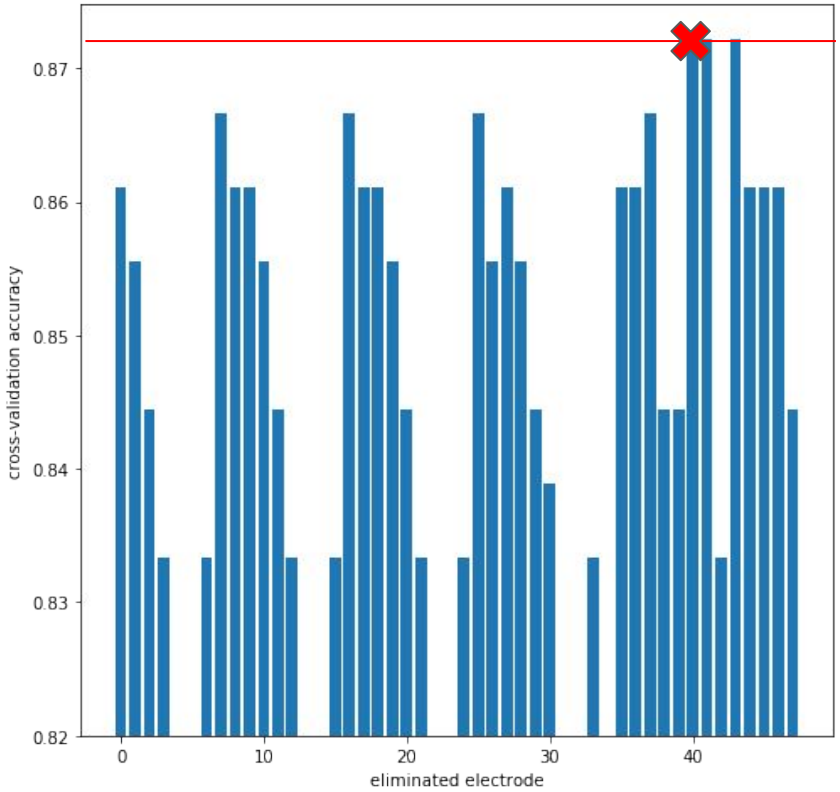
unchanged baseline performance



Removing electrode 34

Iteration 10

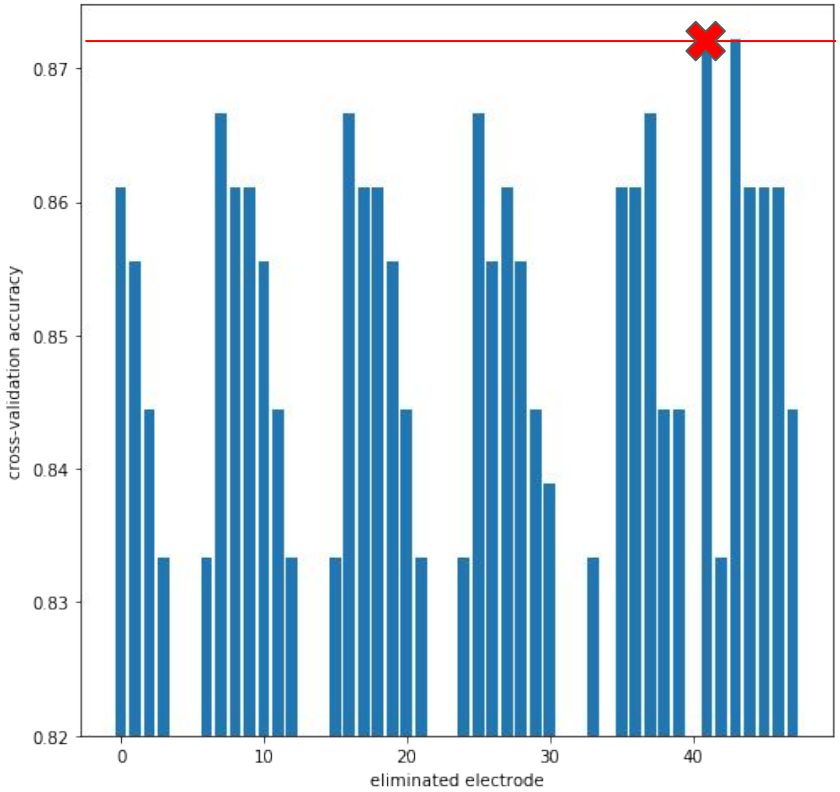
unchanged baseline performance



Removing electrode 40

Iteration 11

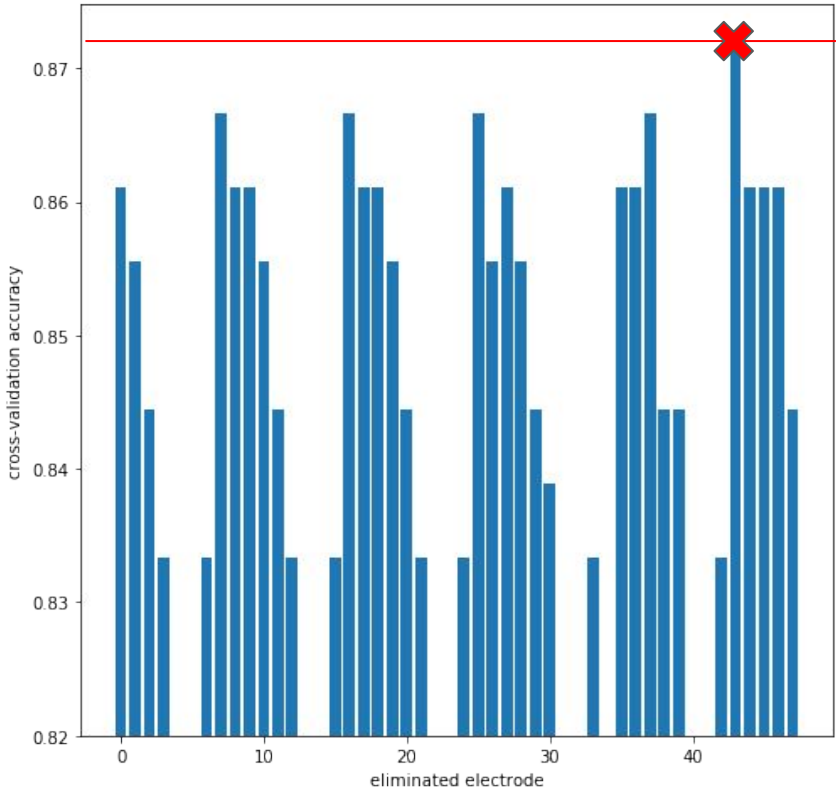
unchanged baseline performance



Removing electrode 41

Iteration 12

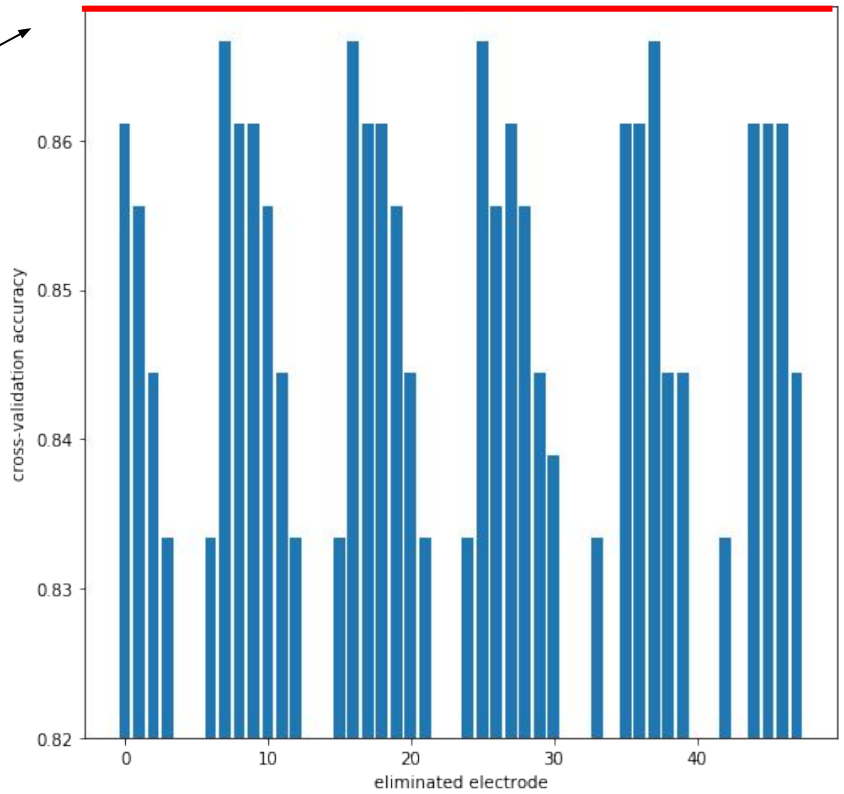
unchanged baseline performance



Removing electrode 43

Iteration 13

unchanged baseline performance



STOPPING HERE removing any additional electrode would degrade performance.

Wrapper Methods - Variants

Backward feature elimination

1. Initialize the feature set F to include all features F_0: $\{f_0, f_1, \ldots, , f_{M-1}\}$ and evaluate performance
2. Evaluate performance with feature sets $F \backslash f_i$, removing a single feature f_i from F.
3. Update F to exclude f_i for the one feature that maximally maintained or increased performance.
4. Repeat steps 2 and 3 until
    a. Performance degrades
    b. Target number of features is reached

Forward feature selection

Same in reverse: evaluate classifier with each feature individually, add feature and repeat.

Prioritized selection/ elimination

Add/ remove features in order determined by some feature scoring method.

Filtering vs. Wrapper

The main differences between the filter and wrapper methods for feature selection are:

- Filter methods are much faster compared to wrapper methods as they do not involve training the models.
- Filter methods measure the relevance of features by their correlation/ mutual information with dependent variable.
- Wrapper methods measure the usefulness of a subset of features by actually training a model on it.
- Filter methods use statistical methods for evaluation of features
- Wrapper methods use cross validation.
- Feature selection using wrapper methods risk making the model more prone to overfitting.

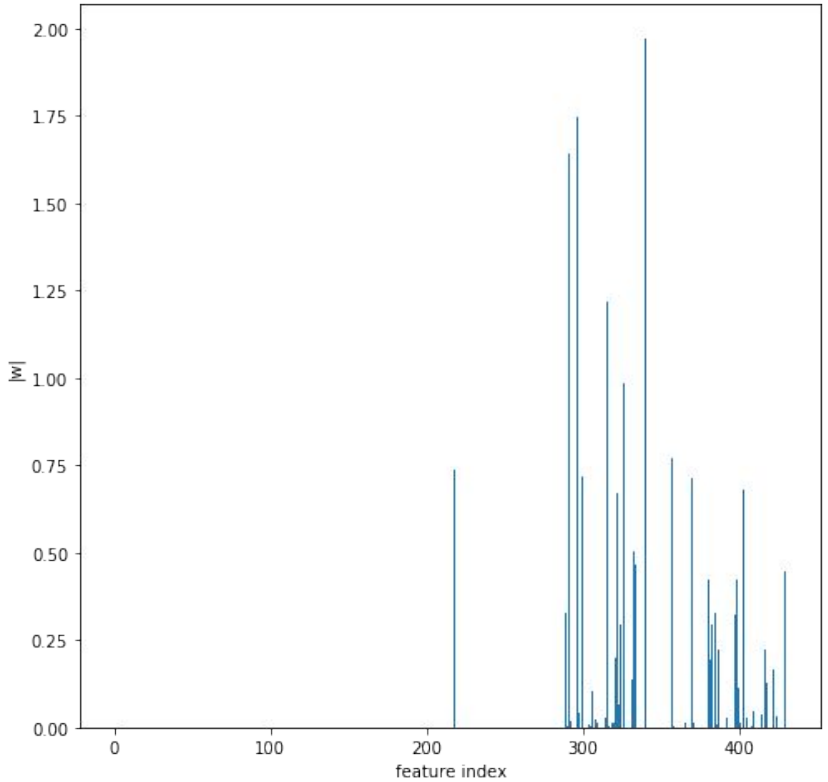
Embedding

Objectives with weight regularization favour solutions with specific weight characteristics.

- L_1 regularization induces sparsity
- L_2 regularization keeps weights near zero

Absolute value of the weights indicates classifier sensitivity to feature values.

Embedding using L_1: Case Study 3



Overview

1. Case Study 1
2. What is Feature Engineering?
3. Feature Selection
    - 3.1. Filtering Methods
    - 3.2. Wrapper Methods
    - 3.3. Embedding Methods
4. Feature Extraction: Constructing Features Manually

Tabular Data: Categorical Data

Categorical data cannot be interpreted by numerical models directly

- Mapping onto numerical ‘index’ variables is problematic
  - Numbers are arbitrary, predictive function almost always non-linear in the mapping
  - Distance-based methods assume triangle inequality is valid

$$d(A, B) \leq d(A, C) + d(C, B)$$

$$d(car, book) \leq_? d(car, cat) + d(cat, book)$$

- Transform categorical data into one-hot encoding

$$\phi('cat') = [0, 0, 1]$$

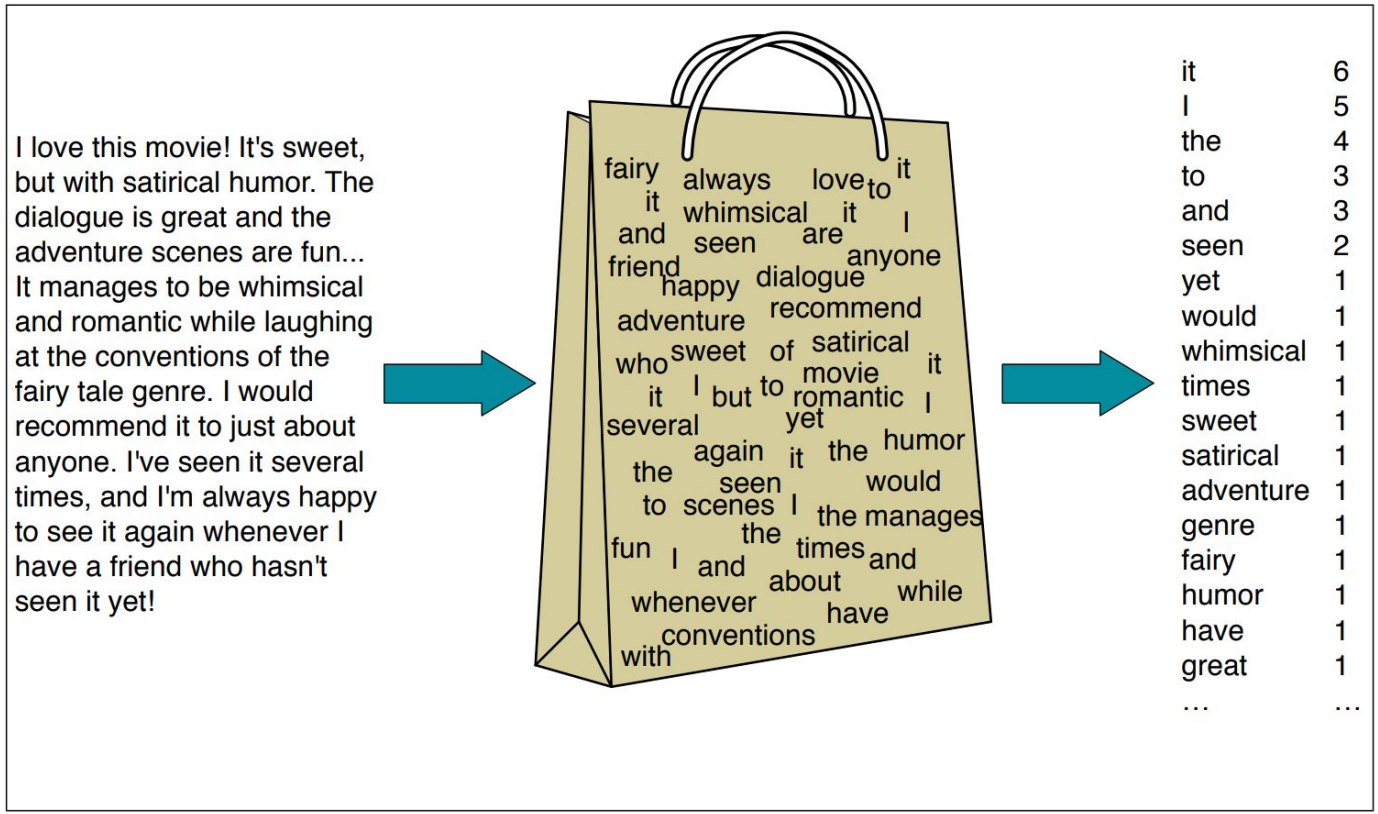
is_car is_book is_cat

Tabular Data: Timestamps

- Timestamps represented as seconds since 01.01.1970 UTC (UNIX Epoch)
- Predictive functions are usually non-linear in raw timestamps (e.g., bicycle rental demand).
- Transforming the data into something more useful is trivial
- Only retain time information that's pertinent to your modelling problem.
  - Time of day (routines), day of week (restaurants), month (seasonal adjustment)
  - Rush hour yes/ no (traffic)
  - Time since last event (customer retention)
  - Time as an aggregator (spend per month; customer value)

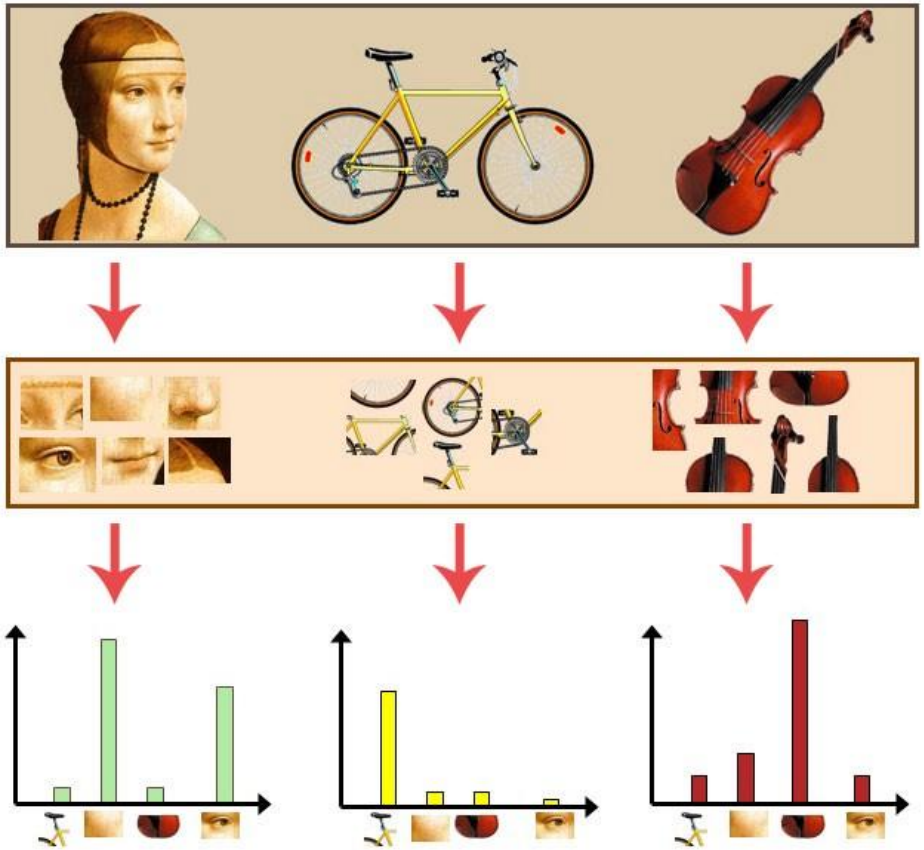
$$\phi(t = 1571745600) = [day\_of\_week = Tuesday, is\_rush\_hour = False]$$

Aggregation: Bag of Words



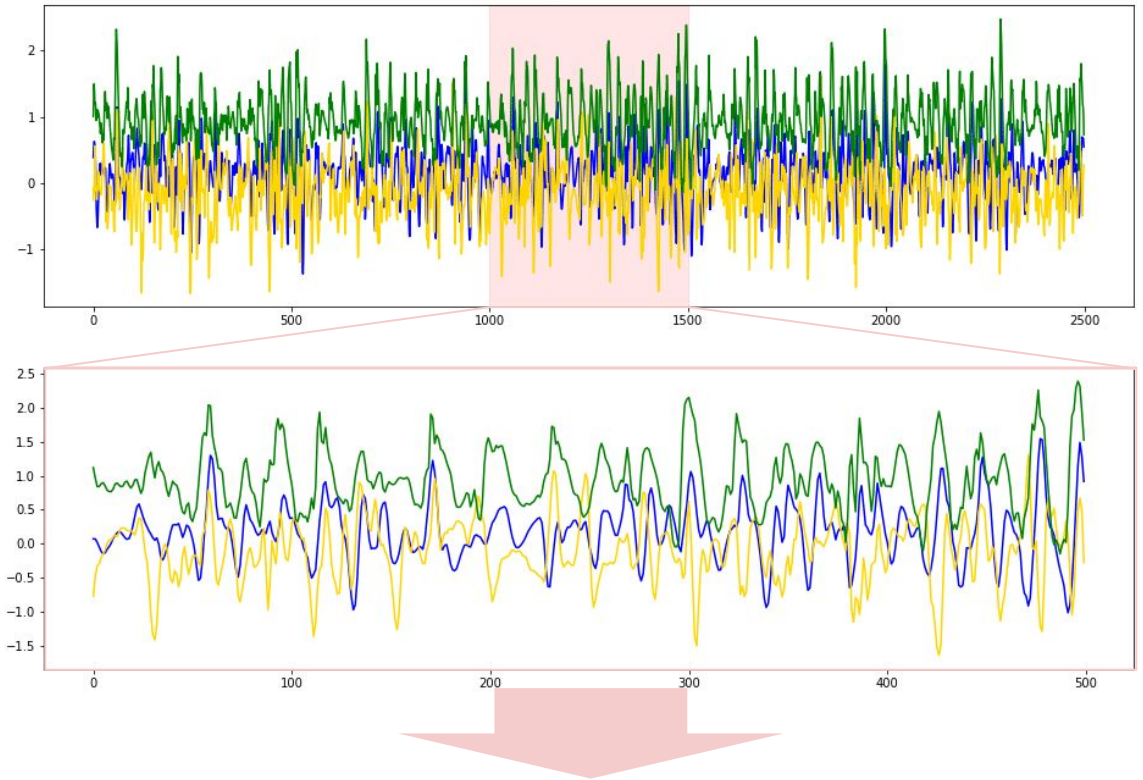
Jurafsky & Martin, Speech and Language Processing (3rd ed. draft) http://web.stanford.edu/~jurafsky/slp3/4.pdf

Aggregation: Bag of Visual Words



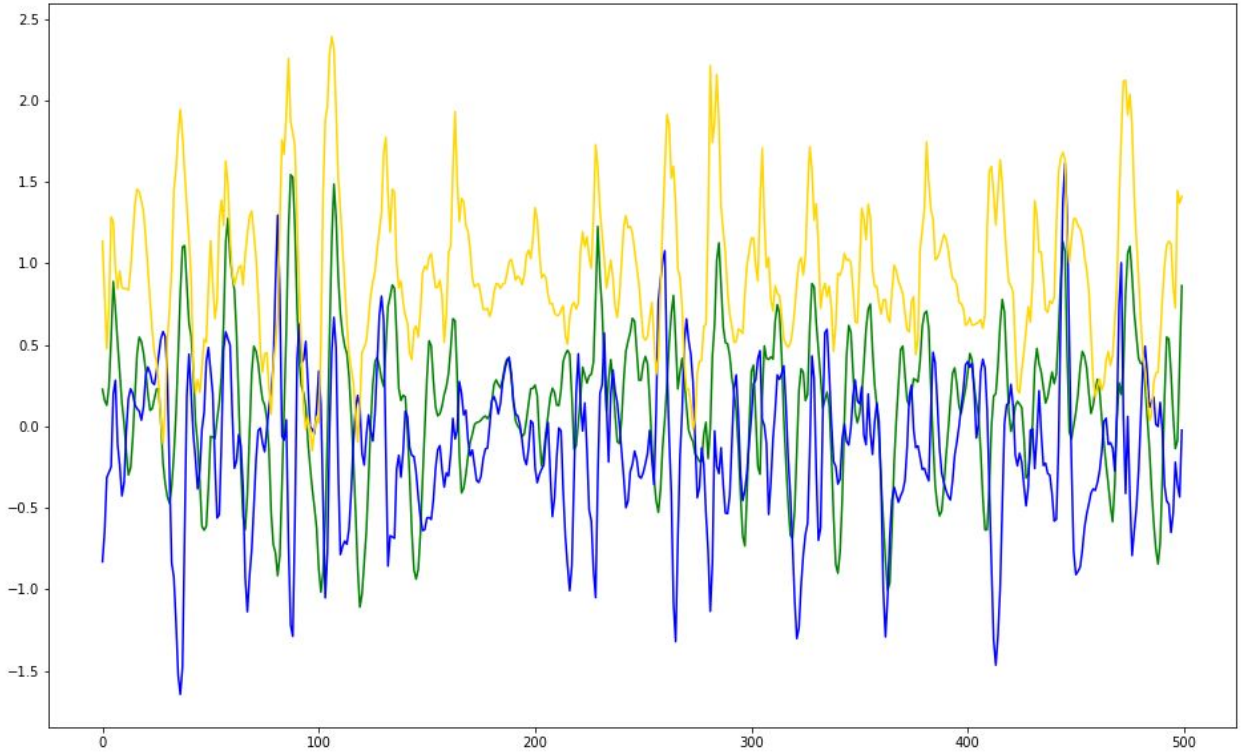
Dictionary of K “visual words” often learned using K-means clustering

Cropping: Sliding Window

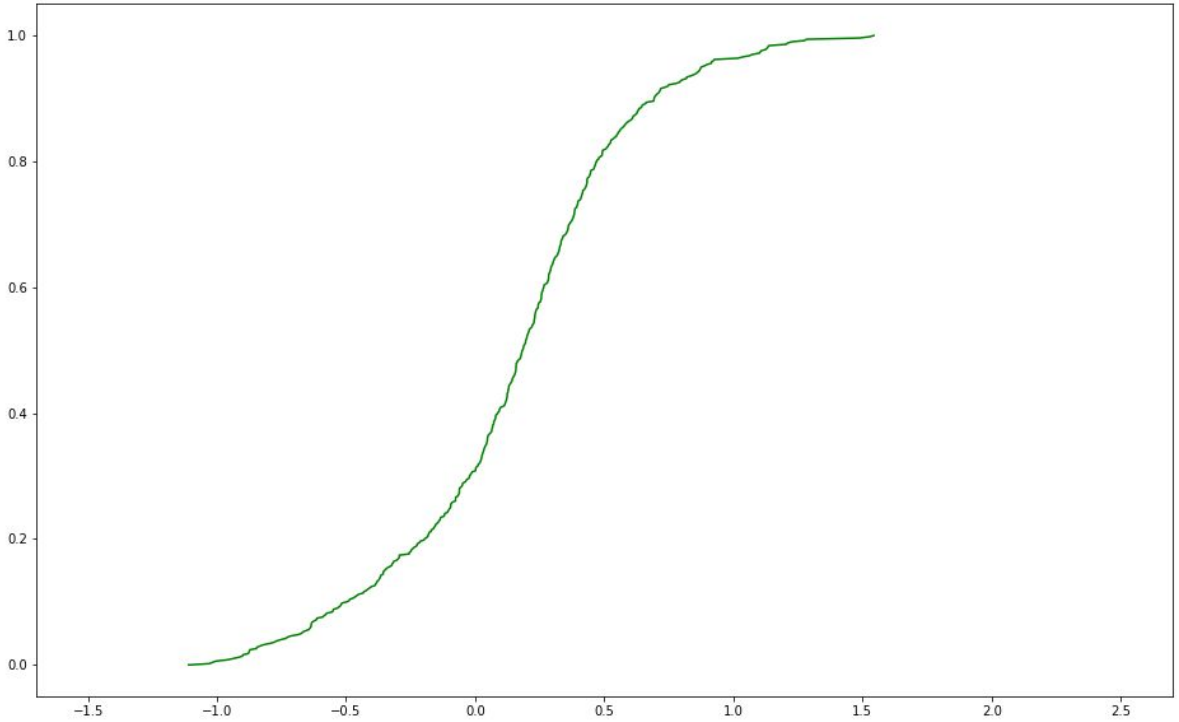


Running/ Walking/ Cycling?

Aggregation

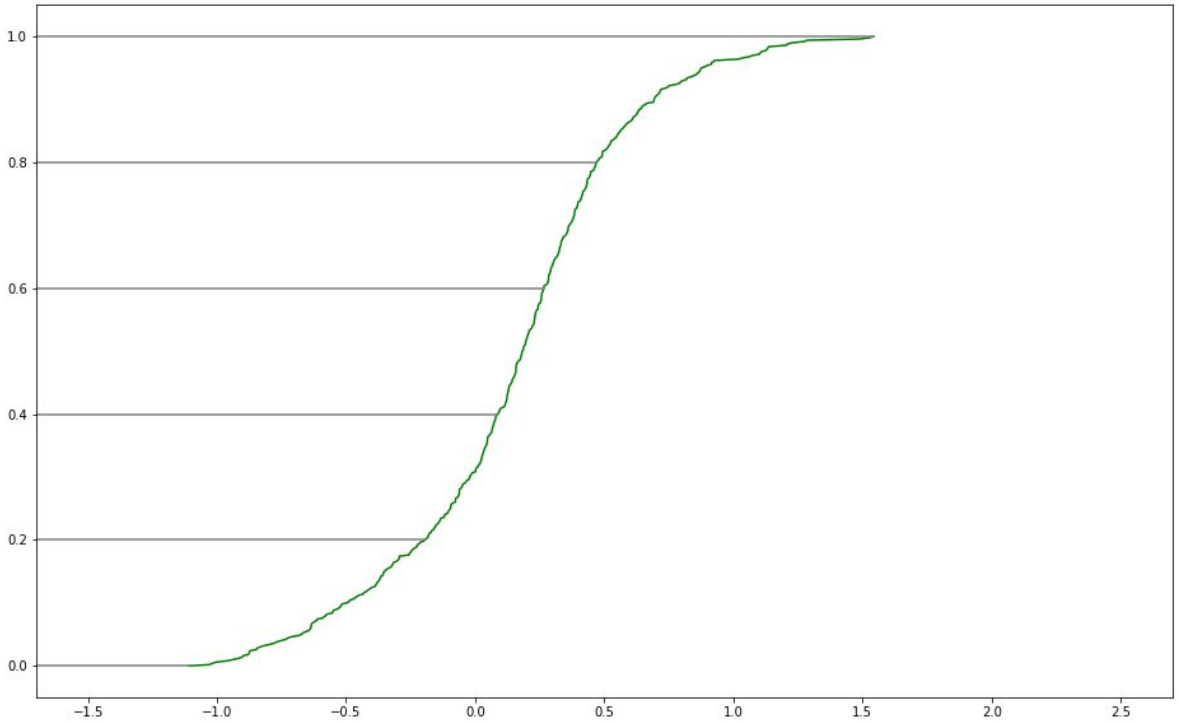


Aggregation: Empirical Cumulative Density Function



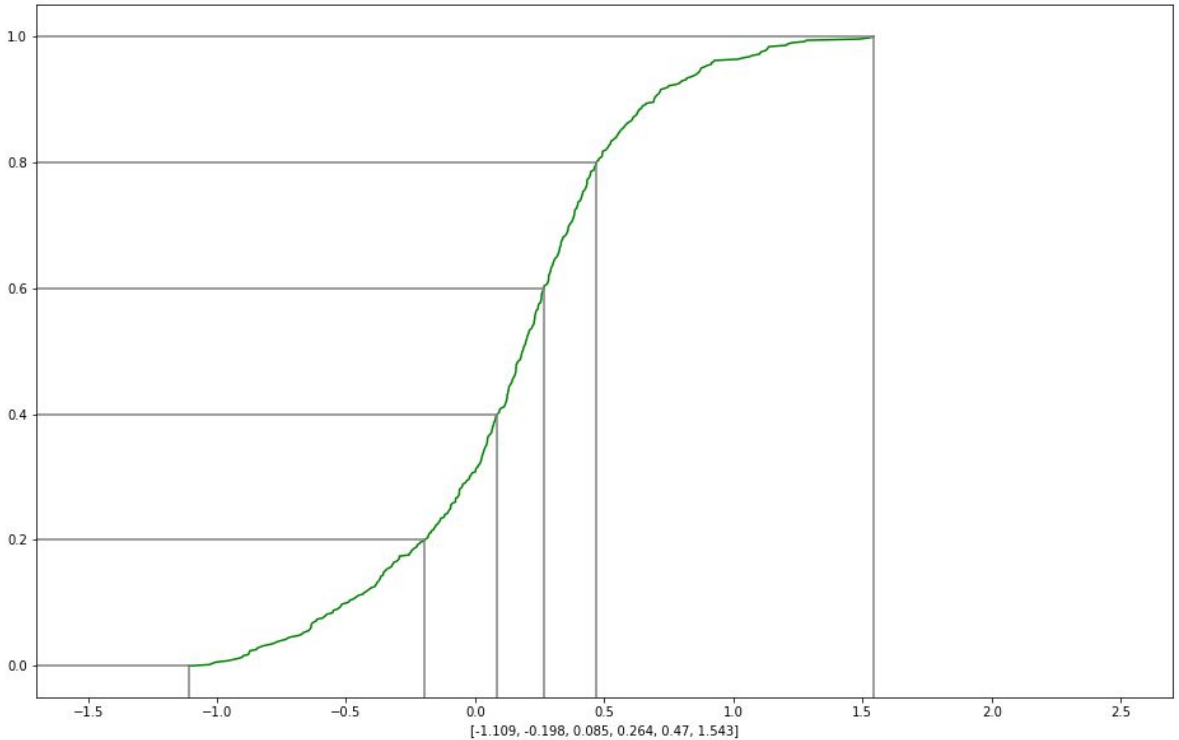
$$F_X(x) := Pr(X \leq x) = p$$

Aggregation: Empirical Cumulative Density Function



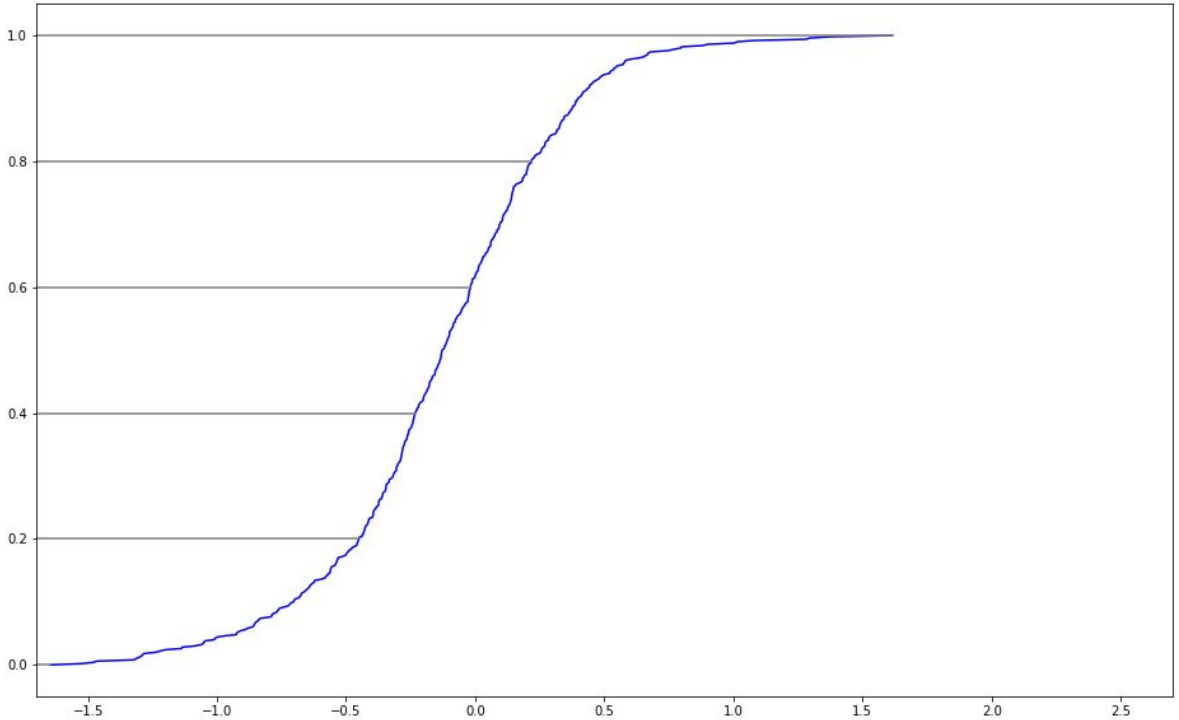
$$Q(p) = \inf\{x \in \mathbb{R} : p \leq F(x)\}$$

Aggregation: Empirical Cumulative Density Function

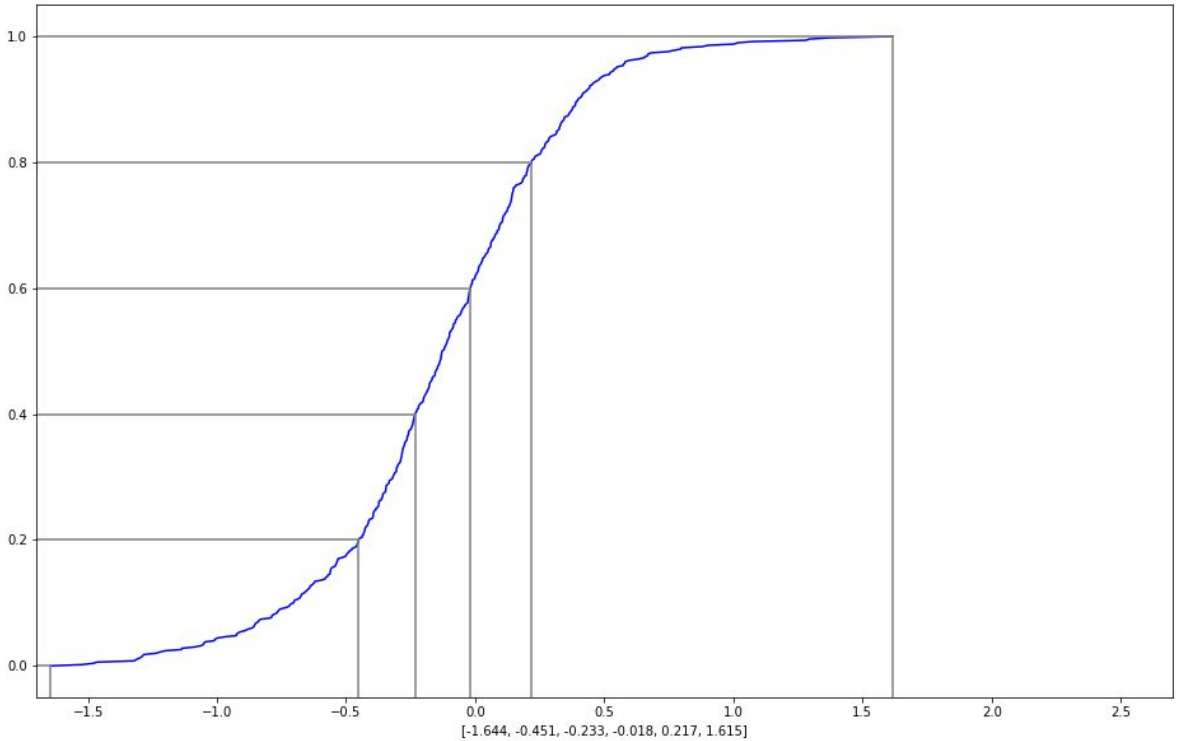


$$Q(p) = \inf\{x \in \mathbb{R} : p \leq F(x)\}$$

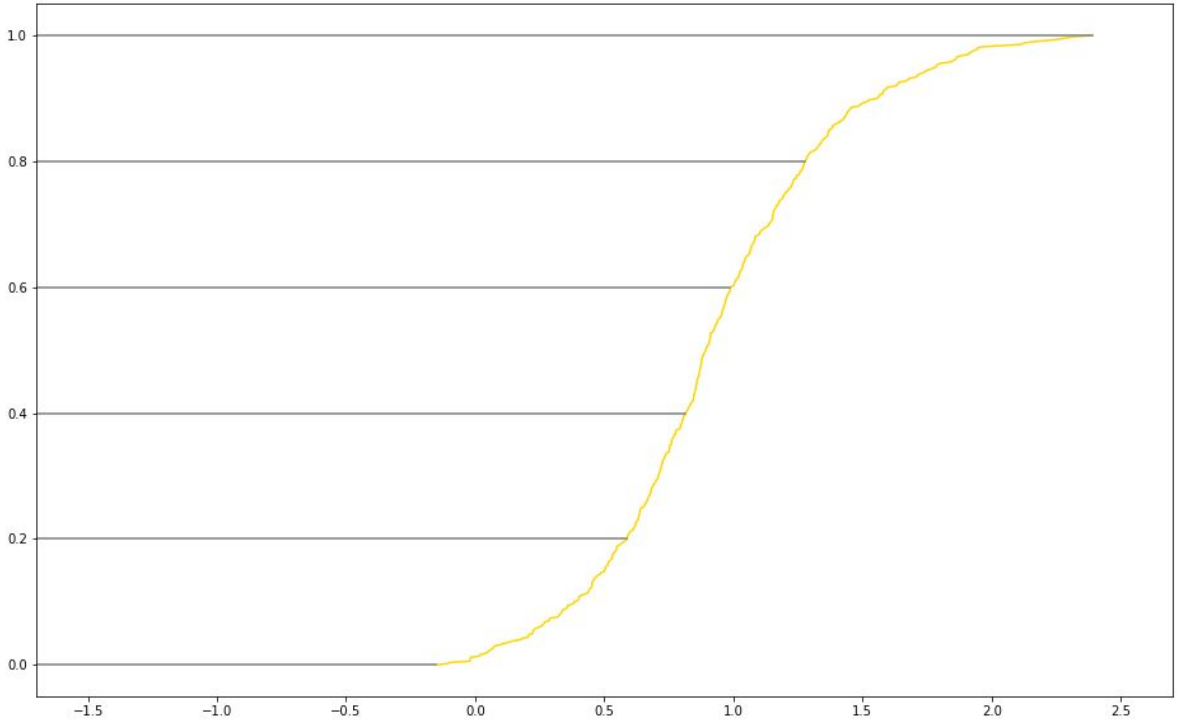
Aggregation: Empirical Cumulative Density Function



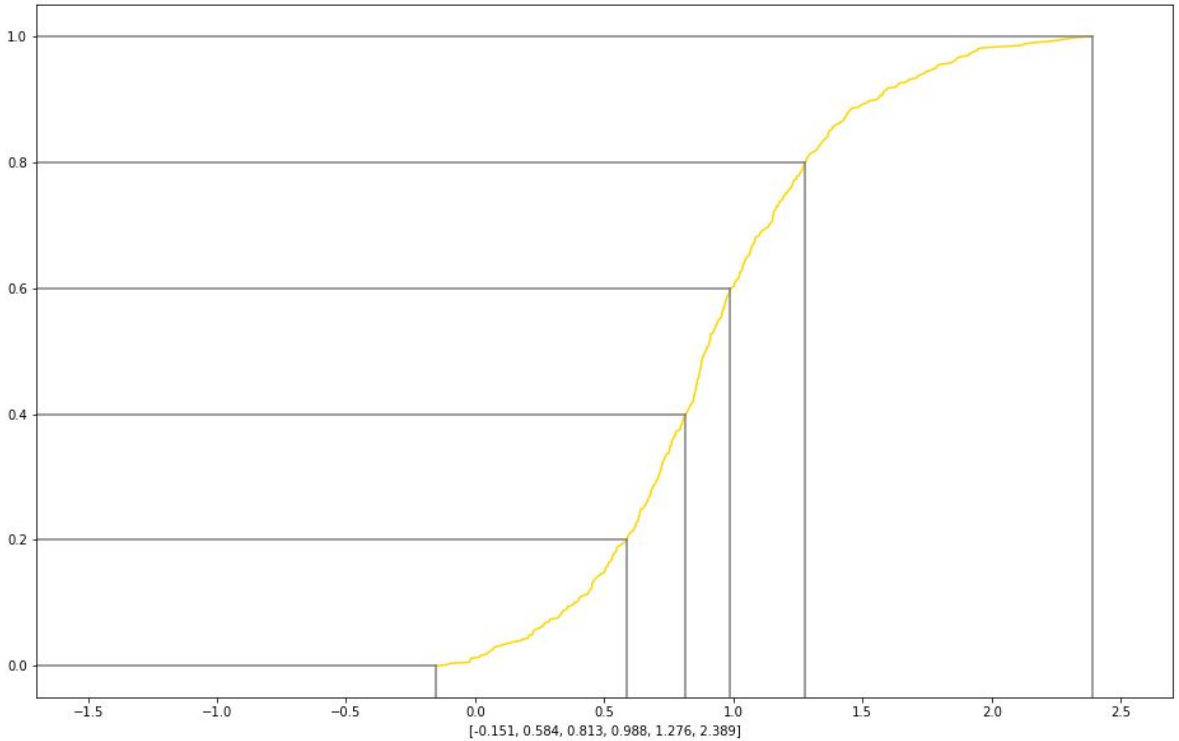
Aggregation: Empirical Cumulative Density Function



Aggregation: Empirical Cumulative Density Function



Aggregation: Empirical Cumulative Density Function



Aggregation: Empirical Cumulative Density Function

```
def encode_ecdf(x, num_features):
    x.sort()
    qs = np.linspace(0, len(x) - 1e-9, num_features)
    ecdf = [ x[int(q*len(x))] for q in qs]
    return ecdf
```

Normalization

- Raw data value ranges can vary wildly.
- Distances and objectives become governed by features with large values

Standardisation

$$x'_i = \frac{x_i - \mu_i}{\sigma_i} \qquad \mu_i = \frac{1}{N}\sum x_i \quad \sigma_i = \frac{1}{N-1}\sum (x_i - \mu_i)^2$$

Min-Max normalization

$$x'_i = \frac{x_i - min(x_i)}{max(x_i) - min(x_i)}$$

Scaling to unit length

$$\mathbf{x}' = \frac{\mathbf{x}}{\|\mathbf{x}\|}$$

Projection

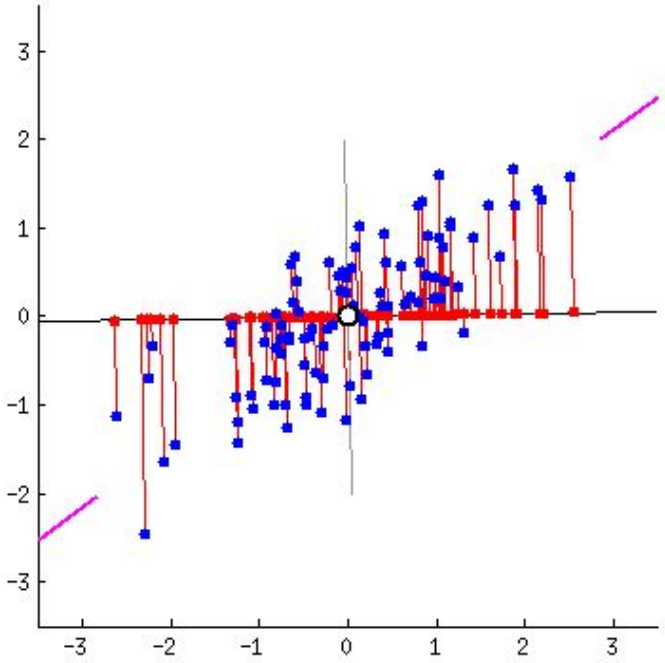
How much of the input space does the data occupy?

- Set of face images among all possible images of 100 x 100 pixels.
- Set of English language utterances among all possible 16kHz audio.
- Set of human hand gestures among all possible 2 second accelerometer data.

Projection Methods

- PCA
- SVD
- Randomized variants
- See also non-linear projection methods (e.g., autoencoders)

Projection: Principal Component Analysis (PCA)



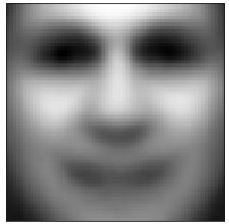
https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues

Eigenfaces

Mean Face

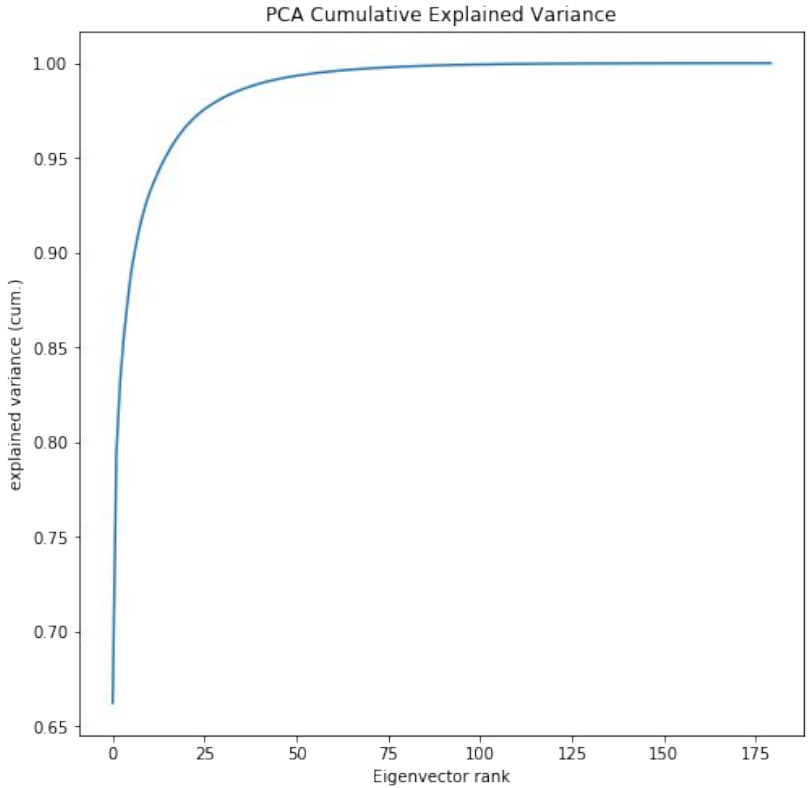
Top Eigenvectors (Eigenfaces)

Eigenfaces

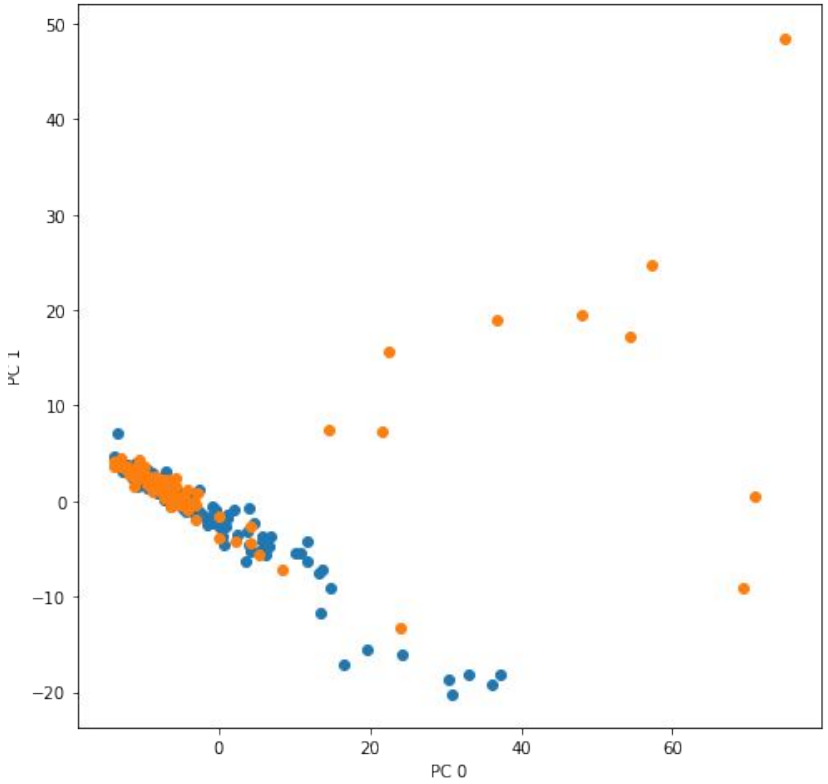
Encoding  =  + 0.8  + -0.2  + …

- Faceness of an image (reconstruction error)
- Recognition (nearest neighbour)
- Classification (via attribute labels)

PCA: Case Study



PCA: Case Study



Summary

1. Case Study 3: Prediction Central Neuropathic Pain

2. What is Feature Engineering?

   Transforming data to facilitate downstream machine learning tasks

3. Feature Selection: Finding Relevant Subsets of Features

   3.1. Filtering Methods (rank features, keep top k)
   3.2. Wrapper Methods (greedy forward/ backward selection)
   3.3. Embedding Methods (LASSO)

4. Feature Extraction

   4.1. Tabular Data Transformations (binarization, one-hot encoding)
   4.2. Cropping (temporal sliding windows)
   4.3. Aggregation (Empirical Cumulative Density, Bag-of-Words)
   4.4. Normalization (e.g., standardisation, scaling)
   4.5. Projection (PCA)