

Big Data and Automated Content Analysis (12EC)

Week 4: »Statistical Modelling and Machine learning« Wednesday

Damian Trilling
d.c.trilling@uva.nl, @damian0604

March 1, 2023

UvA RM Communication Science

Today

Overview

Supervising Machine Learning: Predicting things

- You have done it before!

- From regression to classification

Unsupervised machine learning

- Finding similar variables

 - An introduction to dimensionality reduction

 - Principal Component Analysis and Singular Value Decomposition

- Finding similar cases

 - k-means clustering

 - Hierarchical clustering

Next steps



Everything clear from last week?

Main points from last week

I assume that by now, everybody knows:

- how to get any data, in any shape, in all common file formats, into a structure usable for future analysis;
- how to create a report in Jupyter Notebook using Markdown cells as well as matplotlib and seaborn (optionally: other visualization libraries).

This week, we will get a general overview of ML and statistical modelling. In Part II of the course, we will go more in-depth and specifically use these techniques in combination with textual data.

Overview

There are bottom-up and top-down approaches. Think: explorative analysis and finding patterns vs. finding something pre-defined and testing a specific hypothesis.

Some terminology

Supervised machine learning

You have a dataset with both predictor and outcome (independent and dependent variables; features and labels) — a *labeled* dataset. Think of regression: You measured x_1 , x_2 , x_3 and you want to predict y , which you also measured

Unsupervised machine learning

You have no labels. (You did not measure y)

You might already know *some* techniques to figure out whether x_1 , x_2, \dots, x_i co-occur

- Principal Component Analysis (PCA) and Singular Value Decomposition (SVD)
- Cluster analysis
- Topic modelling (Non-negative matrix factorization and Latent Dirichlet Allocation)

Some terminology

Supervised machine learning

You have a dataset with both predictor and outcome (independent and dependent variables; features and labels) — a *labeled* dataset. Think of regression: You measured x_1 , x_2 , x_3 and you want to predict y , which you also measured

Unsupervised machine learning

You have no labels. (You did not measure y)

You might already know *some* techniques to figure out whether x_1 , x_2, \dots, x_i co-occur

- Principal Component Analysis (PCA) and Singular Value Decomposition (SVD)
- Cluster analysis
- Topic modelling (Non-negative matrix factorization and Latent Dirichlet Allocation)

Some terminology

Supervised machine learning

You have a dataset with both predictor and outcome (independent and dependent variables; features and labels) — a *labeled* dataset. Think of regression: You measured x_1 , x_2 , x_3 and you want to predict y , which you also measured

Unsupervised machine learning

You have no labels. (You did not measure y)

You might already know *some* techniques to figure out whether x_1 , x_2, \dots, x_i co-occur

- Principal Component Analysis (PCA) and Singular Value Decomposition (SVD)
- Cluster analysis
- Topic modelling (Non-negative matrix factorization and Latent Dirichlet Allocation)

Some terminology

Supervised machine learning

You have a dataset with both predictor and outcome (independent and dependent variables; features and labels) — a *labeled* dataset. Think of regression: You measured x_1 , x_2 , x_3 and you want to predict y , which you also measured

Unsupervised machine learning

You have no labels. (You did not measure y)

You might already know *some* techniques to figure out whether x_1 , x_2, \dots, x_i co-occur

- Principal Component Analysis (PCA) and Singular Value Decomposition (SVD)
- Cluster analysis
- Topic modelling (Non-negative matrix factorization and Latent Dirichlet Allocation)

Some terminology

Supervised machine learning

You have a dataset with both predictor and outcome (independent and dependent variables; features and labels) — a *labeled* dataset. Think of regression: You measured x_1, x_2, x_3 and you want to predict y , which you also measured

Unsupervised machine learning

You have no labels. (You did not measure y)

























You might already know *some* techniques to figure out whether x_1, x_2, \dots, x_i co-occur

- Principal Component Analysis (PCA) and Singular Value Decomposition (SVD)
- Cluster analysis
- Topic modelling (Non-negative matrix factorization and Latent Dirichlet Allocation)

























Let's distinguish four use cases. . .

1. Finding similar variables (dimensionality reduction) – unsupervised
2. Finding similar cases (clustering) – unsupervised
3. Predicting a continuous variable (regression) – supervised
4. Predicting group membership (classification) – supervised

	x1	x2	x3	x4	x5	y
case1	■	■	■	■	■	■
case2	■	■	■	■	■	■
case3	■	■	■	■	■	■
case4	■	■	■	■	■	■

	x1	x2	x3	x4	x5	(y)
case1						
case2						
case3						
case4						

Dimensionality reduction: finding similar variables (features)

	x1	x2	x3	x4	x5	(y)
case1						
case2						
case3						
case4						

Clustering: finding similar cases

	x1	x2	x3	x4	x5	→	y
case1	■	■	■	■	■	→	■
case2	■	■	■	■	■	→	■
case3	■	■	■	■	■	→	■
case4	■	■	■	■	■	→	■
new case	■	■	■	■	■	→	?

Regression and classification: learn how to predict y.

Note, again, that the ■ signs can be *anything*: a person's age, the frequency of a word, the color of a pixel, a number of clicks.

	x1	x2	x3	x4	x5	y
case1	■	■	■	■	■	■
case2	■	■	■	■	■	■
case3	■	■	■	■	■	■
case4	■	■	■	■	■	■

Predicting things

Predicting things

You have done it before!

You have done it before!

Regression

1. Based on your data, you estimate some regression equation
$$y_i = \alpha + \beta_1 x_{i1} + \dots + \beta_p x_{ip} + \varepsilon_i$$
2. Even if you have some *new unseen data*, you can estimate your expected outcome \hat{y} !
3. Example: You estimated a regression equation where y is newspaper reading in days/week:
$$y = -.8 + .4 \times \text{man} + .08 \times \text{age}$$
4. You could now calculate \hat{y} for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset*:
$$\hat{y}_{\text{man}20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$$
$$\hat{y}_{\text{woman}40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$$

You have done it before!

Regression

1. Based on your data, you estimate some regression equation
$$y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$$
2. Even if you have some *new unseen data*, you can estimate your expected outcome \hat{y} !
3. Example: You estimated a regression equation where y is newspaper reading in days/week:
$$y = -.8 + .4 \times man + .08 \times age$$
4. You could now calculate \hat{y} for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset*:
$$\hat{y}_{man20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$$
$$\hat{y}_{woman40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$$

You have done it before!

Regression

1. Based on your data, you estimate some regression equation
$$y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$$
2. Even if you have some *new unseen data*, you can estimate your expected outcome \hat{y} !
3. Example: You estimated a regression equation where y is newspaper reading in days/week:
$$y = -.8 + .4 \times man + .08 \times age$$
4. You could now calculate \hat{y} for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset*:
$$\hat{y}_{man20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$$
$$\hat{y}_{woman40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$$

You have done it before!

Regression

1. Based on your data, you estimate some regression equation
$$y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$$
2. Even if you have some *new unseen data*, you can estimate your expected outcome \hat{y} !
3. Example: You estimated a regression equation where y is newspaper reading in days/week:
$$y = -.8 + .4 \times man + .08 \times age$$
4. You could now calculate \hat{y} for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset*:
$$\hat{y}_{man20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$$
$$\hat{y}_{woman40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$$

You have done it before!

Regression

1. Based on your data, you estimate some regression equation
$$y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$$
2. Even if you have some *new unseen data*, you can estimate your expected outcome \hat{y} !
3. Example: You estimated a regression equation where y is newspaper reading in days/week:
$$y = -.8 + .4 \times \textit{man} + .08 \times \textit{age}$$
4. You could now calculate \hat{y} for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset*:
$$\hat{y}_{\textit{man}20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$$
$$\hat{y}_{\textit{woman}40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$$

You have done it before!

Regression

1. Based on your data, you estimate some regression equation
$$y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$$
2. Even if you have some *new unseen data*, you can estimate your expected outcome \hat{y} !
3. Example: You estimated a regression equation where y is newspaper reading in days/week:
$$y = -.8 + .4 \times \text{man} + .08 \times \text{age}$$
4. You could now calculate \hat{y} for a man of 20 years and a woman of 40 years – *even if no such person exists in your dataset*:
$$\hat{y}_{\text{man}20} = -.8 + .4 \times 1 + .08 \times 20 = 1.2$$
$$\hat{y}_{\text{woman}40} = -.8 + .4 \times 0 + .08 \times 40 = 2.4$$

This is
Supervised Machine Learning!

...but...

- We will only use *half* (or another fraction) of our data to estimate the model, so that we can use the other half to check if our predictions match the manual coding (“labeled data”, “annotated data” in SML-lingo)
 - e.g., 2000 labeled cases, 1000 for training, 1000 for testing — if successful, run on 100,000 unlabeled cases
- We use many more independent variables (“features”)
- For text analysis, IVs can be, e.g., word frequencies

...but...

- We will only use *half* (or another fraction) of our data to estimate the model, so that we can use the other half to check if our predictions match the manual coding (“labeled data”, “annotated data” in SML-lingo)
 - e.g., 2000 labeled cases, 1000 for training, 1000 for testing — if successful, run on 100,000 unlabeled cases
- We use many more independent variables (“features”)
- For text analysis, IVs can be, e.g., word frequencies

...but...

- We will only use *half* (or another fraction) of our data to estimate the model, so that we can use the other half to check if our predictions match the manual coding (“labeled data”, “annotated data” in SML-lingo)
 - e.g., 2000 labeled cases, 1000 for training, 1000 for testing — if successful, run on 100,000 unlabeled cases
- We use many more independent variables (“features”)
- For text analysis, IVs can be, e.g., word frequencies

... but ...

- We will only use *half* (or another fraction) of our data to estimate the model, so that we can use the other half to check if our predictions match the manual coding (“labeled data”, “annotated data” in SML-lingo)
 - e.g., 2000 labeled cases, 1000 for training, 1000 for testing — if successful, run on 100,000 unlabeled cases
- We use many more independent variables (“features”)
- For text analysis, IVs can be, e.g., word frequencies

Predicting things

From regression to classification

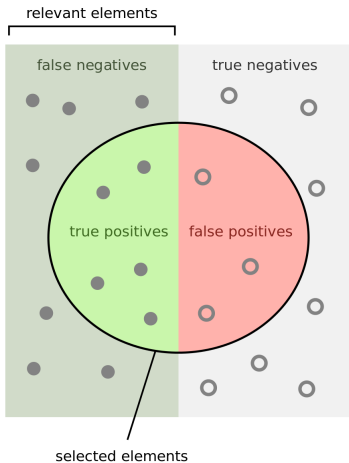
In the machine learning world, predicting some continuous value is referred to as a **regression** task. If we want to predict a binary or categorical variable, we call it a **classification** task.

(quite confusingly, even if we use a logistic regression for the latter)

Classification tasks

For many computational approaches, we are actually not that interested in predicting a continuous value. Typical questions include:

- Is this article about topic A, B, C, D, or E?
- Is this review positive or negative?
- Does this text contain frame F?
- Is this satire?
- Is this misinformation?
- Given past behavior, can I predict the next click?



How many selected
items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant
items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Some measures

- Accuracy
- Recall
- Precision
- $F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$
- AUC (Area under curve)
[0, 1], 0.5 = random
guessing

Different classification algorithms

- It is an empirical question which one works best
- We typically try several ones and select the best
- (remember: we have a test dataset that we did *not* use to train the model, so that we can assess how well it predicts the test labels based on the test features)
- To avoid *p*-hacking-like scenario's (which we call “overfitting”), there are techniques available (cross-validation, later in this course)

To make it easier, we focus on binary classification (“positive”/“negative”), but it doesn't really matter whether there are two or more labels)

Different classification algorithms

- It is an empirical question which one works best
- We typically try several ones and select the best
- (remember: we have a test dataset that we did *not* use to train the model, so that we can assess how well it predicts the test labels based on the test features)
- To avoid *p*-hacking-like scenario's (which we call “overfitting”), there are techniques available (cross-validation, later in this course)

To make it easier, we focus on binary classification (“positive”/“negative”), but it doesn't really matter whether there are two or more labels)

Different classification algorithms

- It is an empirical question which one works best
- We typically try several ones and select the best
- (remember: we have a test dataset that we did *not* use to train the model, so that we can assess how well it predicts the test labels based on the test features)
- To avoid *p*-hacking-like scenario's (which we call “overfitting”), there are techniques available (cross-validation, later in this course)

To make it easier, we focus on binary classification (“positive”/“negative”), but it doesn't really matter whether there are two or more labels)

Different classification algorithms

- It is an empirical question which one works best
- We typically try several ones and select the best
- (remember: we have a test dataset that we did *not* use to train the model, so that we can assess how well it predicts the test labels based on the test features)
- To avoid *p*-hacking-like scenario's (which we call “overfitting”), there are techniques available (cross-validation, later in this course)

To make it easier, we focus on binary classification (“positive”/“negative”), but it doesn't really matter whether there are two or more labels)

Different classification algorithms

- It is an empirical question which one works best
- We typically try several ones and select the best
- (remember: we have a test dataset that we did *not* use to train the model, so that we can assess how well it predicts the test labels based on the test features)
- To avoid *p*-hacking-like scenario's (which we call “overfitting”), there are techniques available (cross-validation, later in this course)

To make it easier, we focus on binary classification (“positive”/“negative”), but it doesn't really matter whether there are two or more labels)

Different classification algorithms

- It is an empirical question which one works best
- We typically try several ones and select the best
- (remember: we have a test dataset that we did *not* use to train the model, so that we can assess how well it predicts the test labels based on the test features)
- To avoid *p*-hacking-like scenario's (which we call “overfitting”), there are techniques available (cross-validation, later in this course)

To make it easier, we focus on binary classification (“positive”/“negative”), but it doesn't really matter whether there are two or more labels)

Naïve Bayes

Bayes' theorem

$$P(A | B) = \frac{P(B | A) \times P(A)}{P(B)}$$

A = Text is about sports

B = Text contains 'very', 'close', 'game'. Furthermore, we simply multiply the propabilities for the features:

$$P(B) = P(\text{very close game}) = P(\text{very}) \times P(\text{close}) \times P(\text{game})$$

We can fill in all values by counting how many articles are about sports, and how often the words occur in these texts. (Fully elaborated

example on <https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/>)

Naïve Bayes

Bayes' theorem

$$P(A | B) = \frac{P(B | A) \times P(A)}{P(B)}$$

A = Text is about sports

B = Text contains 'very', 'close', 'game'. Furthermore, we simply multiply the probabilities for the features:

$$P(B) = P(\text{very close game}) = P(\text{very}) \times P(\text{close}) \times P(\text{game})$$

We can fill in all values by counting how many articles are about sports, and how often the words occur in these texts. (Fully elaborated

example on <https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/>)

Naïve Bayes

Bayes' theorem

$$P(A | B) = \frac{P(B | A) \times P(A)}{P(B)}$$

A = Text is about sports

B = Text contains 'very', 'close', 'game'. Furthermore, we simply multiply the probabilities for the features:

$$P(B) = P(\text{very close game}) = P(\text{very}) \times P(\text{close}) \times P(\text{game})$$

We can fill in all values by counting how many articles are about sports, and how often the words occur in these texts. (Fully elaborated

example on <https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/>)

Naïve Bayes

- It's “naïve” because the features are treated as completely independent (\neq “controlling” in regression analysis)
- It's fast and easy
- It's a good *baseline* for binary classification problems

Naïve Bayes

- It's “naïve” because the features are treated as completely independent (\neq “controlling” in regression analysis)
- It's fast and easy
- It's a good *baseline* for binary classification problems

Naïve Bayes

- It's “naïve” because the features are treated as completely independent (\neq “controlling” in regression analysis)
- It's fast and easy
- It's a good *baseline* for binary classification problems

Naïve Bayes

$$P(\text{label} \mid \text{features}) = \frac{P(x_1 \mid \text{label}) \cdot P(x_2 \mid \text{label}) \cdot P(x_3 \mid \text{label}) \cdot P(\text{label})}{P(x_1) \cdot P(x_2) \cdot P(x_3)}$$

- Formulas always look intimidating, but we only need to fill in how many documents containing feature x_n have the label, how often the label occurs, and how often each feature occurs
- Also for computers, this is *really easy and fast*
- Weird assumption: features are independent
- Often used as a baseline

Logistic Regression

Probability of a binary outcome in a regression model

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n)}}$$

Just like in OLS regression, we have an intercept and regression coefficients. We use a threshold (default: 0.5) and above, we assign the positive label ('good movie'), below, the negative label ('bad movie').

Logistic Regression

- The features are *not* independent.
- Computationally more expensive than Naïve Bayes
- We can get probabilities instead of just a label
- That allows us to say how sure we are for a specific case
- ...or to change the threshold to change our precision/recall-tradeoff

Logistic Regression

- The features are *not* independent.
- Computationally more expensive than Naïve Bayes
- We can get probabilities instead of just a label
- That allows us to say how sure we are for a specific case
- ...or to change the threshold to change our precision/recall-tradeoff

Logistic Regression

- The features are *not* independent.
- Computationally more expensive than Naïve Bayes
- We can get probabilities instead of just a label
- That allows us to say how sure we are for a specific case
- ...or to change the threshold to change our precision/recall-tradeoff

Logistic Regression

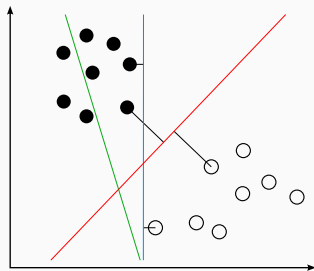
- The features are *not* independent.
- Computationally more expensive than Naïve Bayes
- We can get probabilities instead of just a label
- That allows us to say how sure we are for a specific case
- ...or to change the threshold to change our precision/recall-tradeoff

Logistic Regression

- The features are *not* independent.
- Computationally more expensive than Naïve Bayes
- We can get probabilities instead of just a label
- That allows us to say how sure we are for a specific case
- ...or to change the threshold to change our precision/recall-tradeoff

Support Vector Machines

- Idea: Find a hyperplane that best separates your cases
- Can be linear, but does not have to be (depends on the so-called kernel you choose)
- Very popular



https://upload.wikimedia.org/wikipedia/commons/b/b5/Svm_separating_hyperplanes_%28SVG%29.svg

(Further reading:

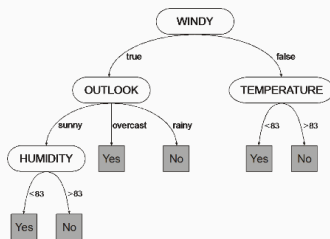
<https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/>)

SVM vs logistic regression

- for *linearly separable* classes not much difference
- with the right hyperparameters, SVM is less sensitive to outliers
- biggest advantage: with the *kernel trick*, data can be transformed that they *become* linearly separable

Decision Trees and Random Forests

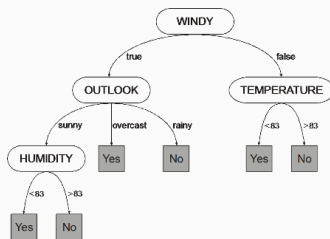
- Model problem as a series of decisions (e.g., if cloudy then ...if temperature > 30 degrees then ...)
- Order and cutoff-points are determined by an algorithm
- Big advantage: Model non-linear relationships
- And: They are easy to interpret (!) (“white box”)



https://upload.wikimedia.org/wikipedia/en/4/4f/GEP_decision_tree_with_numeric_and_nominal_attributes.png

Decision Trees and Random Forests

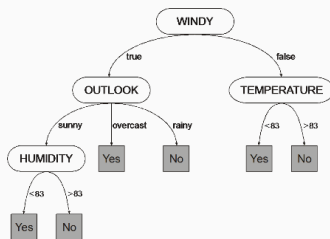
- Model problem as a series of decisions (e.g., if cloudy then ...if temperature > 30 degrees then ...)
- Order and cutoff-points are determined by an algorithm
- Big advantage: Model non-linear relationships
- And: They are easy to interpret (!) (“white box”)



https://upload.wikimedia.org/wikipedia/en/4/4f/GEP_decision_tree_with_numeric_and_nominal_attributes.png

Decision Trees and Random Forests

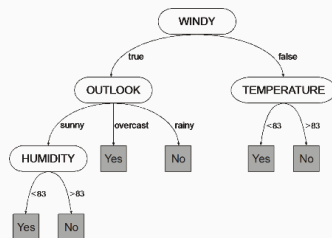
- Model problem as a series of decisions (e.g., if cloudy then ...if temperature > 30 degrees then ...)
- Order and cutoff-points are determined by an algorithm
- Big advantage: Model non-linear relationships
- And: They are easy to interpret (!) (“white box”)



https://upload.wikimedia.org/wikipedia/en/4/4f/GEP_decision_tree_with_numeric_and_nominal_attributes.png

Decision Trees and Random Forests

- Model problem as a series of decisions (e.g., if cloudy then ... if temperature > 30 degrees then ...)
- Order and cutoff-points are determined by an algorithm
- Big advantage: Model non-linear relationships
- And: They are easy to interpret (!) (“white box”)



https://upload.wikimedia.org/wikipedia/en/4/4f/GEP_decision_tree_with_numeric_and_nominal_attributes.png

Decision Trees and Random Forests

Disadvantages of decision trees

- comparatively inaccurate
- once you are in the wrong branch, you cannot go 'back up'
- prone to overfitting (e.g., outlier in training data may lead to completely different outcome)

Therefore, nowadays people use *random forests*: Random forests *combine* the predictions of *multiple* trees \Rightarrow might be a good choice for your non-linear classification problem

Decision Trees and Random Forests

Disadvantages of decision trees

- comparatively inaccurate
- once you are in the wrong branch, you cannot go 'back up'
- prone to overfitting (e.g., outlier in training data may lead to completely different outcome)

Therefore, nowadays people use *random forests*: Random forests *combine* the predictions of *multiple* trees \Rightarrow might be a good choice for your non-linear classification problem

Unsupervised ML

A lot of applications and use cases, . . .

. . . but we'll distinguish two today:

1. Finding similar variables (dimension reduction)
2. Finding similar cases (clustering)

Are we more interested in which features “belong together” or which cases “belong together”?

There are many other techniques than those presented today, and vice versa, those presented today can also be used for other purposes

A lot of applications and use cases, . . .

. . . but we'll distinguish two today:

1. Finding similar variables (dimension reduction)
2. Finding similar cases (clustering)

Are we more interested in which features “belong together” or which cases “belong together”?

There are many other techniques than those presented today, and vice versa, those presented today can also be used for other purposes

Unsupervised ML

Finding similar variables

Dimensionality reduction

dimensionality = the number of features we have

(1) Explorative data analysis and visualization

- No good way to visualize 10,000 dimensions (or even 4)

(2) The curse of dimensionality

More features means more data (good!), but:

- Too many features can lead to unfeasible computation times
- We need more training cases to increase the likelihood that the possible combinations actually occur

Dimensionality reduction

dimensionality = the number of features we have

(1) Explorative data analysis and visualization

- No good way to visualize 10,000 dimensions (or even 4)

(2) The curse of dimensionality

More features means more data (good!), but:

- Too many features can lead to unfeasible computation times
- We need more training cases to increase the likelihood that the possible combinations actually occur

Dimensionality reduction

Feature extraction

- Create a smaller set of features
- E.g.: 1,000 features → PCA to reduce to 50 components → SML with these 50 component scores as features

Dimensionality reduction

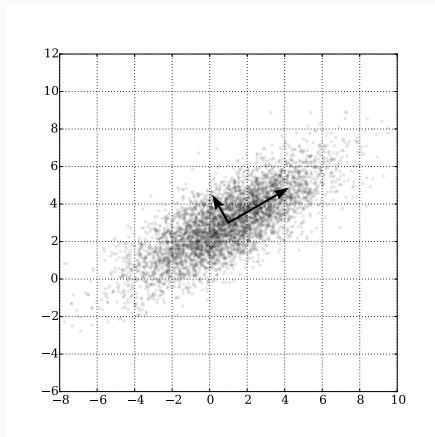
So, we can use unsupervised ML as a dimension reduction step in a supervised ML pipeline.

But it can also be a goal in itself, to understand the data better or to visualize them.

PCA

- related to and often confused with Factor Analysis (same menu item in SPSS – many people who believe they run FA actually run PCA!)
- Components are ordered (first explains most variance)
- Components do *not* necessarily carry a meaningful interpretation

PCA



<https://upload.wikimedia.org/wikipedia/commons/f/f5/GaussianScatterPCA.svg>

Preparation: Import modules and get some sample data

```
1 import pandas as pd
2 from sklearn.decomposition import PCA
3 from sklearn.preprocessing import StandardScaler
4
5 df = pd.read_csv("https://cssbook.net/d/eurobarom_nov_2017.csv")\
6     .dropna()\
7     .groupby(["country"])[[
8         "support_refugees_n",
9         "support_migrants_n",
10        "age",
11        "educational_n"]].mean()
```

	country	support_refugees_n	support_migrants_n	age	educational_n
2	Austria	2.847674	2.576744	49.445349	18.930233
3	Belgium	2.772619	2.252381	53.002381	19.338095
4	Bulgaria	2.075159	1.769427	51.517197	19.652229
5	Croatia	2.717105	2.002193	47.001096	18.899123
6	Cyprus	3.018141	2.104308	53.734694	18.598639

Running PCA

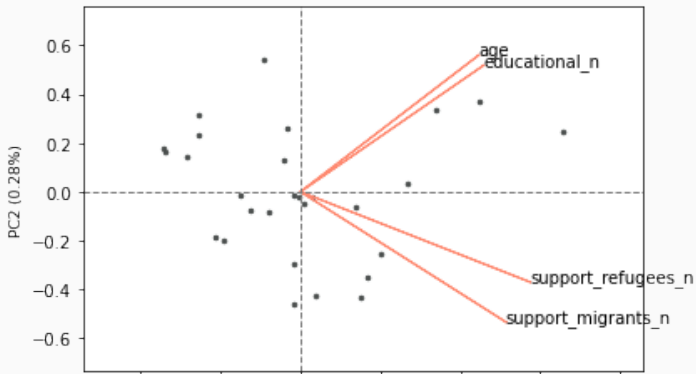
```
1  # we first standardize our data to z-scores
2  scaler = StandardScaler()
3  df_s = scaler.fit_transform(df)
4
5  mypca = PCA()
6  componentscores = mypca.fit_transform(df_s)
7  scores_df = pd.DataFrame(componentscores, index=df.index)
8
9  components_df = pd.DataFrame(data=mypca.components_.T)
10 components_df.index = df.columns
11 print("Variable loadings on the 4 components:")
12 print(components_df)
13
14 # This is the transformed dataset
15 # As we will see in a minute, we retain 86\% of the
16 # variance even if we keep only the first 2 components
17 print("The component scores for each case:")
18 print(scores_df)
```

Running PCA

```
1 Variable loadings on the 4 components:
2           0           1           2           3
3 support_refugees_n 0.573292 -0.369010 0.139859 0.718058
4 support_migrants_n 0.513586 -0.533140 -0.094283 -0.665659
5 age                0.445117 0.558601 0.670994 -0.199005
6 educational_n      0.457642 0.517261 -0.722023 0.041073
7
8
9 The component scores for each country:
10           0           1           2           3
11 country
12 Austria      -0.103285 -1.220018 -0.535673 0.066888
13 Belgium      -0.029355 -0.084707 0.051515 0.227609
14 Bulgaria     -1.660518 0.949533 -0.480337 -0.151837
15 Croatia      -1.267502 -0.819093 -0.920657 0.843682
16 Cyprus       0.060590 -0.195928 0.573670 0.812519
17 Czech republic -2.219795 0.675655 -0.763679 -0.086147
18 Denmark      2.925631 1.516636 -0.748940 0.495489
19 Estonia      -0.602217 2.206418 0.785869 -0.152262
20 Finland      2.224310 1.384983 -0.169499 -0.468742
21 France       -0.102062 -0.046415 0.228587 -0.122965
22 Germany      0.917864 -0.259560 0.940792 0.319158
23 Greece       -0.832343 -0.313480 0.329525 0.683748
24 Hungary      -2.234701 0.716823 0.391788 -0.315312
25 Ireland      0.993243 -1.767147 -0.592957 -0.168892
```

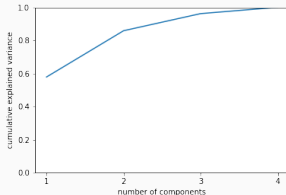
Plotting the result

```
1 import bioinfokit.visuz
2
3 bioinfokit.visuz.cluster.biplot(cscore=componentscores,
4     loadings=mypca.components_,
5     labels=df.columns,
6     var1=round(mypca.explained_variance_ratio_[0],2),
7     var2=round(mypca.explained_variance_ratio_[1],2),
8     show=True)
```



Plotting the result

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 fig, ax = plt.subplots()
5 ax.plot(np.cumsum(mypca.explained_variance_ratio_))
6 ax.set_xticks(range(mypca.n_components_))
7 ax.set_xticklabels(range(1, mypca.n_components_+1))
8 ax.set_xlabel("number of components")
9 ax.set_ylabel("cumulative explained variance")
10 ax.set_ylim(0,1)
11 fig.savefig("pca-explained-variance.png")    # optional of course
```



Singular value decomposition

PCA uses a dense matrix!

In “real life” with larger datasets (essentially, everything (!) involving text), we never use PCA but SVD. It works exactly the same (check out scikit-learn.org), but does not require to hold a dense matrix in memory. Instead, it operates on a sparse matrix, in which only non-zero values are stored. (this will make a lot of sense once we talk about textual data)

* It's mathematically different, but SVD is even used “under the hood” by several PCA modules to solve PCA problems. More info and background: <https://towardsdatascience.com/pca-and-svd-explained-with-numpy-5d13b0d2a4d8>

Unsupervised ML

Finding similar cases

Grouping features vs grouping cases

- In the previous example, we established that *age* and *education* measure almost the same thing, and so do *support_refugees* and *support_migrants*.
- We could now, for instance, check out which countries receive similar scores to group them.
- But if grouping countries (instead of variables) is our goal in the first place, we can directly do so.

Grouping features vs grouping cases

- In the previous example, we established that *age* and *education* measure almost the same thing, and so do *support_refugees* and *support_migrants*.
- We could now, for instance, check out which countries receive similar scores to group them.
- But if grouping countries (instead of variables) is our goal in the first place, we can directly do so.

Grouping features vs grouping cases

- In the previous example, we established that *age* and *education* measure almost the same thing, and so do *support_refugees* and *support_migrants*.
- We could now, for instance, check out which countries receive similar scores to group them.
- But if grouping countries (instead of variables) is our goal in the first place, we can directly do so.

k-means clustering

- Goal: group cases into k clusters
- k is set in advance
- Algorithm to determine k centroids (points in the middle of the cases that belong to it) such that the distances between the cases and their centroids are minimized
- non-deterministic: starts with a randomly chosen centroids (there are other versions)
- Cheap to compute: works even with large number of cases
- We can run PCA first to reduce the number of features if we want/need to

k-means clustering

- Goal: group cases into k clusters
- k is set in advance
- Algorithm to determine k centroids (points in the middle of the cases that belong to it) such that the distances between the cases and their centroids are minimized
- non-deterministic: starts with a randomly chosen centroids (there are other versions)
- Cheap to compute: works even with large number of cases
- We can run PCA first to reduce the number of features if we want/need to

k-means clustering

- Goal: group cases into k clusters
- k is set in advance
- Algorithm to determine k centroids (points in the middle of the cases that belong to it) such that the distances between the cases and their centroids are minimized
- non-deterministic: starts with a randomly chosen centroids (there are other versions)
- Cheap to compute: works even with large number of cases
- We can run PCA first to reduce the number of features if we want/need to

k-means clustering

- Goal: group cases into k clusters
- k is set in advance
- Algorithm to determine k centroids (points in the middle of the cases that belong to it) such that the distances between the cases and their centroids are minimized
- non-deterministic: starts with a randomly chosen centroids (there are other versions)
- Cheap to compute: works even with large number of cases
- We can run PCA first to reduce the number of features if we want/need to

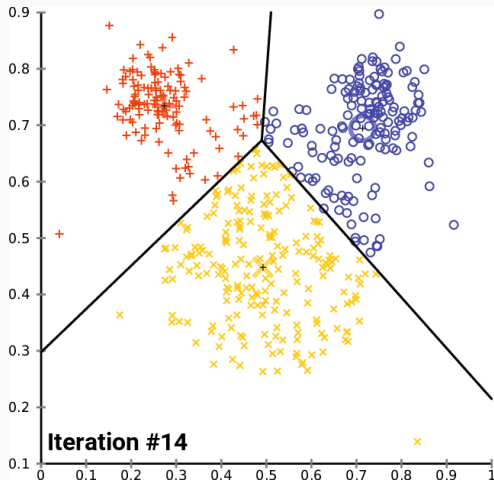
k-means clustering

- Goal: group cases into k clusters
- k is set in advance
- Algorithm to determine k centroids (points in the middle of the cases that belong to it) such that the distances between the cases and their centroids are minimized
- non-deterministic: starts with a randomly chosen centroids (there are other versions)
- Cheap to compute: works even with large number of cases
- We can run PCA first to reduce the number of features if we want/need to

k-means clustering

- Goal: group cases into k clusters
- k is set in advance
- Algorithm to determine k centroids (points in the middle of the cases that belong to it) such that the distances between the cases and their centroids are minimized
- non-deterministic: starts with a randomly chosen centroids (there are other versions)
- Cheap to compute: works even with large number of cases
- We can run PCA first to reduce the number of features if we want/need to

k-means clustering



https://upload.wikimedia.org/wikipedia/commons/e/ea/K-means_convergence.gif

Notice the big symbols indicating the centroids.

```
1 from sklearn.cluster import KMeans
2
3 # everything needs to be measured on the same scale,
4 # therefore we take the scaled dataset
5
6 mykm = KMeans(n_clusters=3).fit(df_s)
7 pd.DataFrame({"country":df.index, "cluster":mykm.labels_})
```

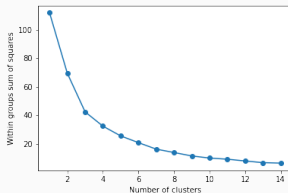
```
1      country cluster
2 0      Austria      1
3 1      Belgium      1
4 2    Bulgaria      0
5 3    Croatia      0
6 4      Cyprus      1
7 5 Czech republic      0
```

Finding the optimal k

- The only way to find k is to estimate multiple models with different k s
- No single best solution; finding a balance between error within clusters (distances from centroid) and low number of clusters.
- An elbow plot can be helpful

Finding the optimal k

```
1 wss = []
2 for i in range(1, 15):
3     km = KMeans(n_clusters=i)
4     km.fit(df_s)
5     wss.append(km.inertia_)
6
7 fig, ax = plt.subplots()
8 ax.plot(range(1, 15), wss, marker="o")
9 ax.set_xlabel("Number of clusters")
10 ax.set_ylabel("Within groups sum of squares")
```



Downsides of k-means clustering

k-means is fast, but has problems:

- k can only be determined by fitting multiple models and comparing them
- bad results if the wrong k is chosen
- bad results if the (real) clusters are non-spherical
- bad results if the (real) clusters are not evenly sized

Hierarchical clustering

General idea

- To start, each case has its own cluster
- Merge the two clusters that are most similar
- Repeat until desired number of clusters is reached

Different options

- Stopping criterion: based on numerical statistic (e.g., Duda-Hart) or dendrogram
- Linkage: how to determine which two clusters should be merged?

Hierarchical clustering

General idea

- To start, each case has its own cluster
- Merge the two clusters that are most similar
- Repeat until desired number of clusters is reached

Different options

- Stopping criterion: based on numerical statistic (e.g., Duda-Hart) or dendrogram
- Linkage: how to determine which two clusters should be merged?

Let's look into some options

`https://scikit-learn.org/stable/modules/clustering.html#hierarchical-clustering`

⇒ Ward's linkage is a good default all-rounder choice, especially if you encounter the problem that other linkages lead to almost all cases ending up in one cluster.

Hierarchical clustering takeaway

- The main reason *not* to use hierarchical methods (but k -means) is their computational cost: when clustering survey data of media users, never use k -means!
- But for NLP/ML, costs may be too high (if not used carefully)
- Very much worth considering, though, if you are really into grouping cases!

Important notes (for *all* techniques today)

Consider the scales of measurement

Clustering is based on distances – if your features are not measured on the same scale, or if it is not meaningful to calculate a numerical distance, it won't produce meaningful results!

Consider standardizing/whitening your features!

Pay attention outliers/extreme cases

Extreme cases or outliers can have a strong influence.

Do proper pre-processing

To reduce the number of features, but also to have *meaningful* features (dimensions on which you expect high distances between the clusters).

Important notes (for *all* techniques today)

Consider the scales of measurement

Clustering is based on distances – if your features are not measured on the same scale, or if it is not meaningful to calculate a numerical distance, it won't produce meaningful results!

Consider standardizing/whitening your features!

Pay attention outliers/extreme cases

Extreme cases or outliers can have a strong influence.

Do proper pre-processing

To reduce the number of features, but also to have *meaningful* features (dimensions on which you expect high distances between the clusters).

Important notes (for *all* techniques today)

Consider the scales of measurement

Clustering is based on distances – if your features are not measured on the same scale, or if it is not meaningful to calculate a numerical distance, it won't produce meaningful results!

Consider standardizing/whitening your features!

Pay attention outliers/extreme cases

Extreme cases or outliers can have a strong influence.

Do proper pre-processing

To reduce the number of features, but also to have *meaningful* features (dimensions on which you expect high distances between the clusters).



Any questions?

Next steps

Make sure you understood all of today's
concepts.

Re-read the chapters.

I prepared exercises to work on *during* the
Friday meeting (alone or in teams):

[https://github.com/uvacw/teaching-bdaca/blob/
main/12ec-course/week04/exercises/](https://github.com/uvacw/teaching-bdaca/blob/main/12ec-course/week04/exercises/)

Overview
○○○○○○○○○

Predicting things
○○○○○○○○○○○○○○○○○○○

Unsupervised ML
○○○○○○○○○○○○○○○○○○○○○○○○○○○○○

Next steps
○○●○

Take-home exam next week Friday!

