

Unsupervised learning

Magnus Nielsen, SODAS, UCPH

Agenda

- Survey
- Unsupervised learning
 - Dimensionality reduction
 - Clustering
- Text as data
 - Dictionary based methods
 - Bag-of-words

Survey

Heterogeneous student body

- Both across and within Aarhus and Copenhagen
- Primary job responsibilities are of course main priority

You want more lecturing, less assistance for coding

Solution...?

- Less focus on Python in lectures
- More code given in exercises

Unsupervised learning

We do not have/use a given target

- Different models 'create' their own target and structure

Structure not always necessary

- Depends more on domain rather than supervised or not
- Utilize data sources such as text and images in novel ways

Dimensionality reduction

Question

Have you worked with dimensionality reduction before?

What methods did you use?

Why?

Reducing dimensionality of the data is done primarily for three reasons:

1. It reduces computation time for later models
2. It can increase performance (the curse of dimensionality)
3. It makes visualizations easier

PCA

The most common method for dimensionality reduction is (probably) Principal Component Analysis

Main idea is to create new variables:

- Which are orthogonal
- Each capture as much variance as possible

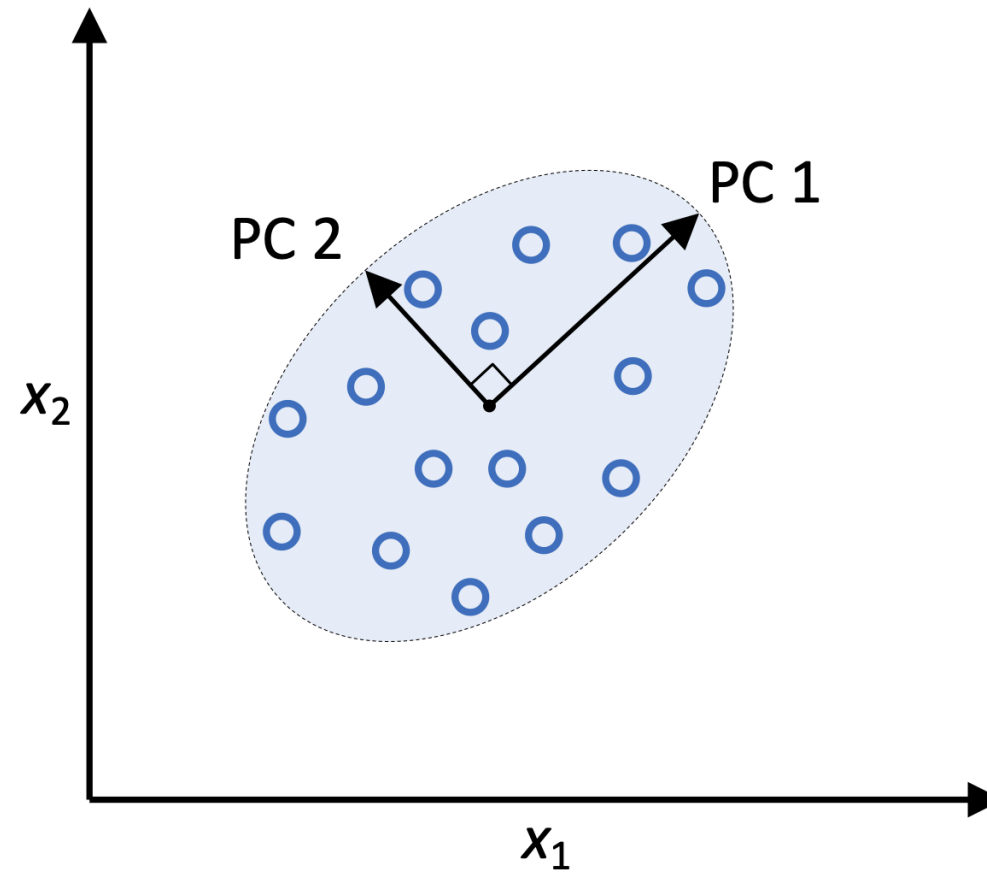
Creates linear 'latent variables'

The method 'intuitively'

- Find the direction of max variance in data
 - This is the first component
- Subtract the first component
- Find the direction of max variance in data
 - This is the second component
- Subtract, and create additional components until no more variance in data

Earlier components will explain more variance!

An illustration



Principal components in 2D

Source: Raschka & Mirjalili, 2022, ch. 6

The method ‘mathematically’

1. Standardize the features
2. Calculate the correlation matrix
3. Perform an eigenvector decomposition of the correlation matrix
4. Order eigenvectors by the size of the eigenvalues
5. Transform original data into K -dimensional space using K first eigenvectors

K is a hyperparameter that we choose - More on how to choose later

The method 'sklearnically'

Supports fit and transform in PCA from `sklearn.decomposition`

- Can be used in pipelines just like the other methods
- Centers, but does not scale
 - Generally preceded by `StandardScaler`

`n_components` is the parameter of interest, and can be set in two ways:

- A positive integer, corresponding to K
 - i.e. 'I would like $K = 5$ components'
- A float between 0 and 1, corresponding to the amount of variance that is retained
 - i.e. 'I would like enough components to keep $K = 95$ of the variance'

How to evaluate

- Downstream performance
 - Treat K as any other hyperparameter
- Computational limits
 - Use as many dimensions as feasible
- Explained variance plots
 - Stop when adding more components doesn't add much information

Explained variance as a guide

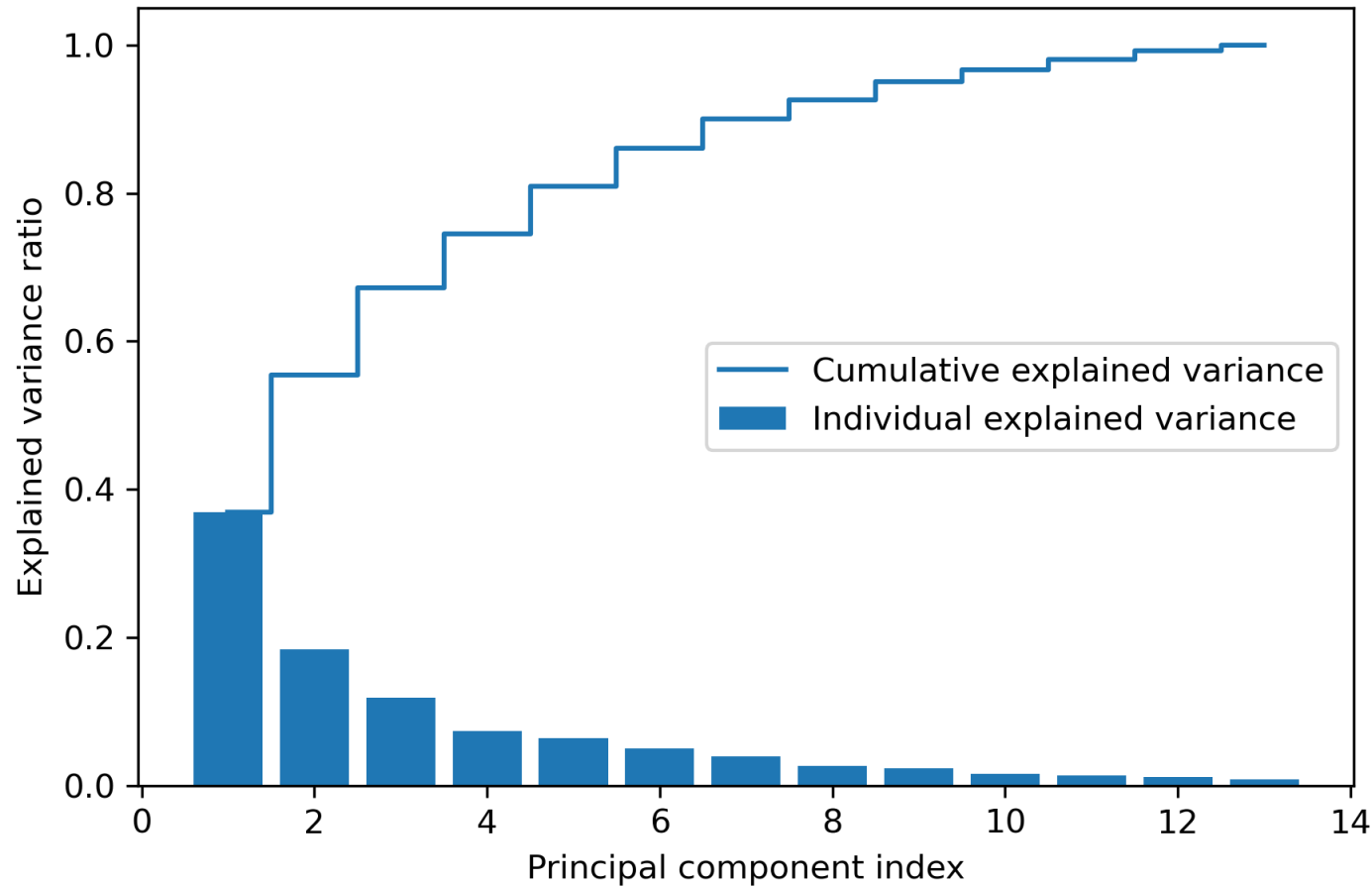
Each principal components explains part of the variance in the original data

If some variables covary, a single principal component can explain much of the variance in the data

Most often, a relatively low amount of principal components explain a lot of the variance

- We can look for an ‘elbow’ in a plot of the explained variance to guide us

Introducing the scree plot



Scree plot

Source: Raschka & Mirjalili, 2022, ch. 5

Interpretation

As PCA is a linear transformation, we can look at how the variables are transformed

- Available in `pca.components_`

A way of interpreting the new variables

- Remember that variables are standardized

Not done that often

- Everything loads on everything due to no restrictions

Non-linearity

If we want to incorporate non-linearity, one of two things are commonly done:

1. Do a polynomial transformation of the input before doing PCA
2. Use a kernel ([kernelPCA](#))

Clustering

Question

Have you worked with clustering before?

What methods did you use?

Clustering

Find unknown groups within populations

- Observations within clusters are alike
- Observations between clusters are different

What constitutes different and alike?

A broad introduction

Different ways of formulating this, with three broad categories:

- Hierarchical clustering
- Prototype-based clustering
- Density-based clustering

Some methods fall into more than one category and there are other clustering methods (e.g. graph-based)

Hierarchical clustering

Clusters are based sequential merging or division of clusters

- Agglomerative, start with n clusters and sequentially join clusters
- Divisive, start with one cluster and sequentially divide into n clusters
- Main hyperparameters: Distance measure (no. of clusters)
- Creates neat dendrograms
- No notion of noise, and all points will belong to a cluster
- An example: [AgglomerativeClustering](#)

Prototype-based clustering

A cluster is represented by a prototype (centroid)

- Main hyperparameter: Number of clusters
- Clusters will generally be convex (circular or ellipsoidal)
- No notion of noise, and all points will belong to a cluster
- An example: [Kmeans](#)

Density-based clustering

A cluster is represented by a dense area of points

- Main hyperparameters: What constitutes ‘dense’?
- Clusters can be any shape
- Distant points are labeled as outliers
- An example: **DBSCAN**

K-means

K-means aims to minimize the within-cluster sum-of-squares:

$$\sum_{i=0}^n \min_{\mu_j \in C} (\|x_i - \mu_j\|^2)$$

Where μ_j are the centroids

This corresponds to an Euclidian distance, and thus we (often) standardscale

Iterative updating

This is done by:

- Initialize clusters
- Assigning points to nearest cluster
- Move centroids to mean values of the cluster samples

Repeating procedure until convergence

Illustrations are nice

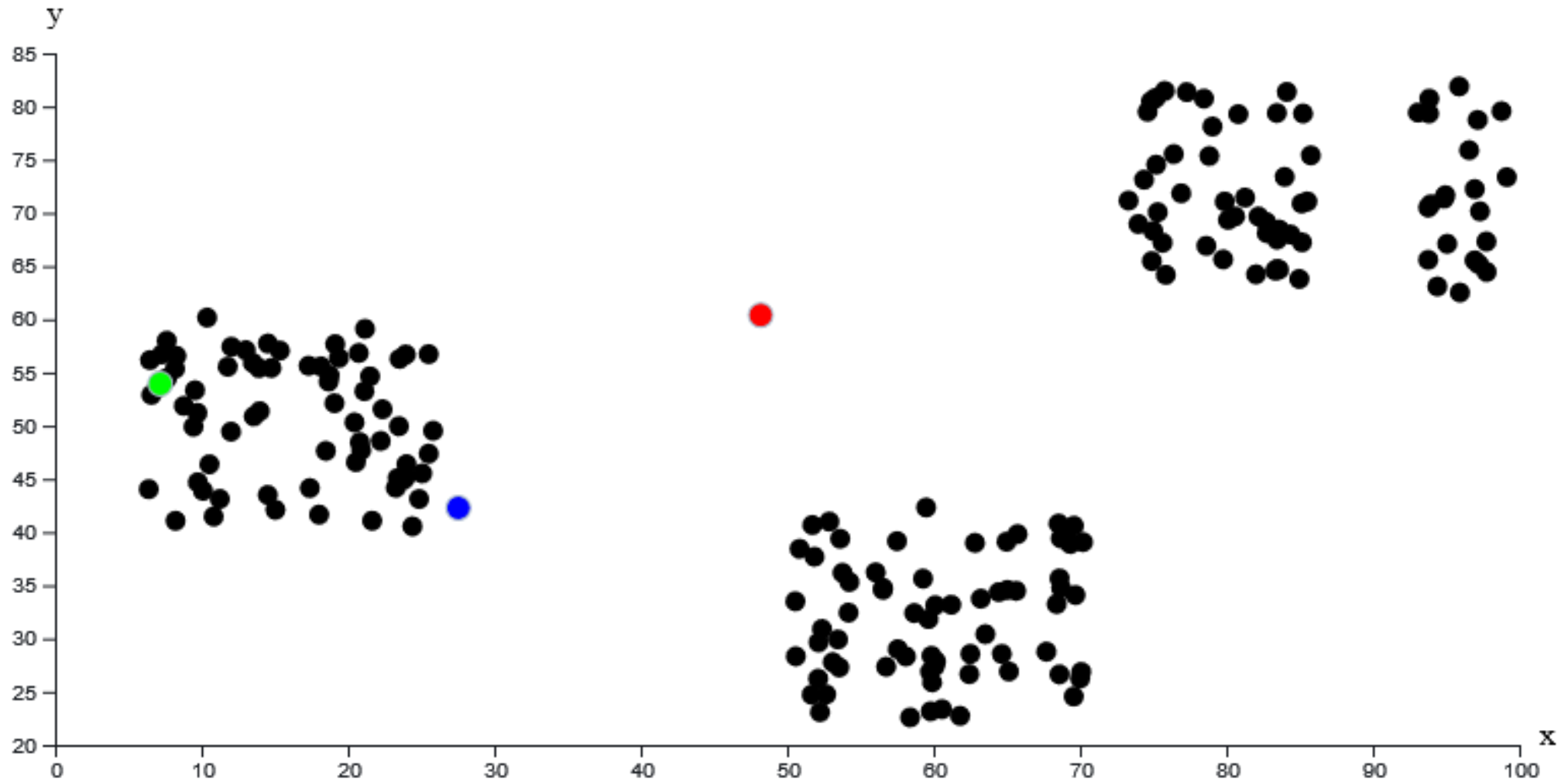
The easiest way to get to know K-means is to look at the process

The following figures are from a cool tool made by [brooklybrand](#)

Note that this is the best case scenario:

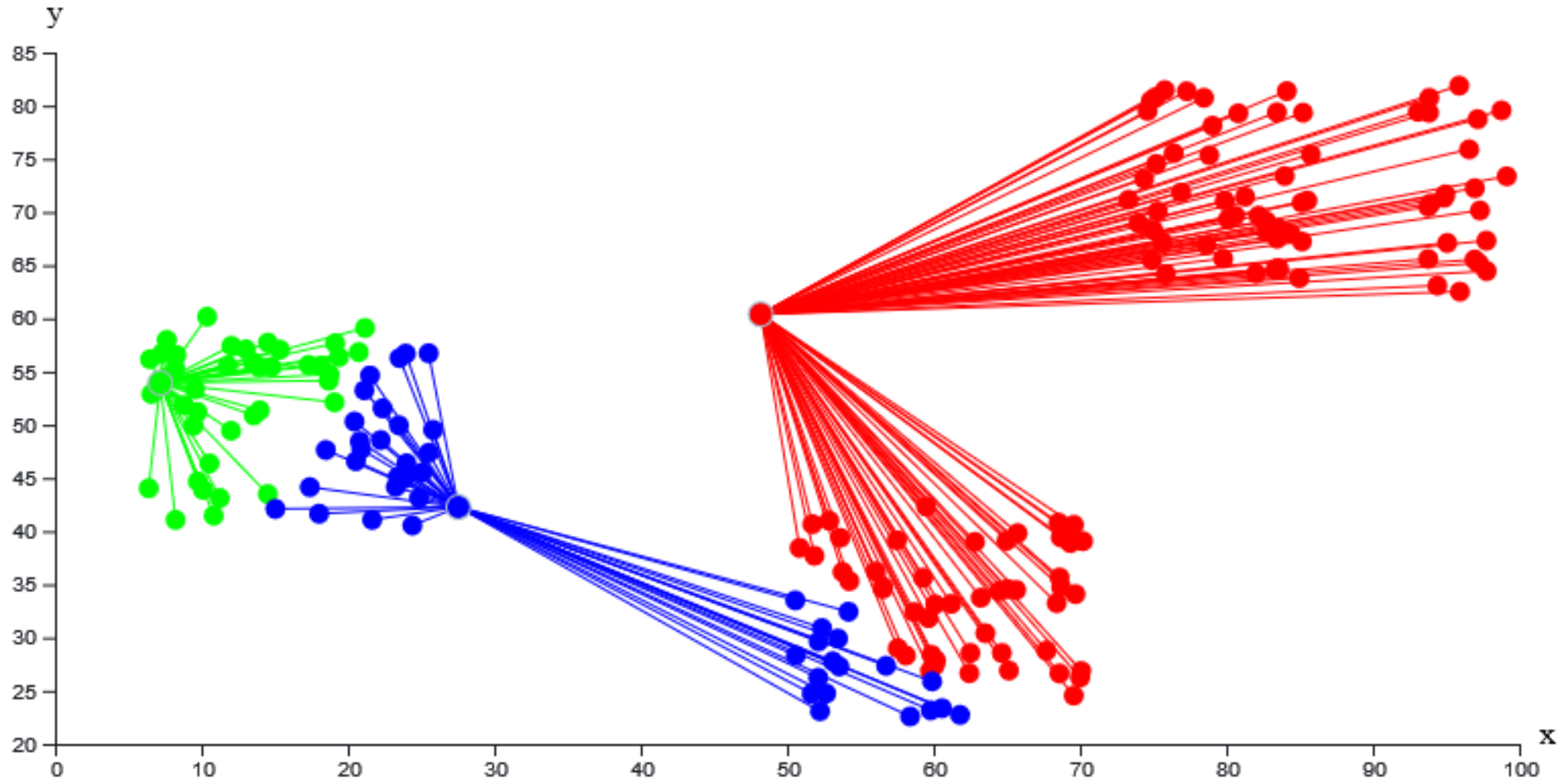
- Three clusters of data and three centroids
- Convex clusters

Initialize centroids



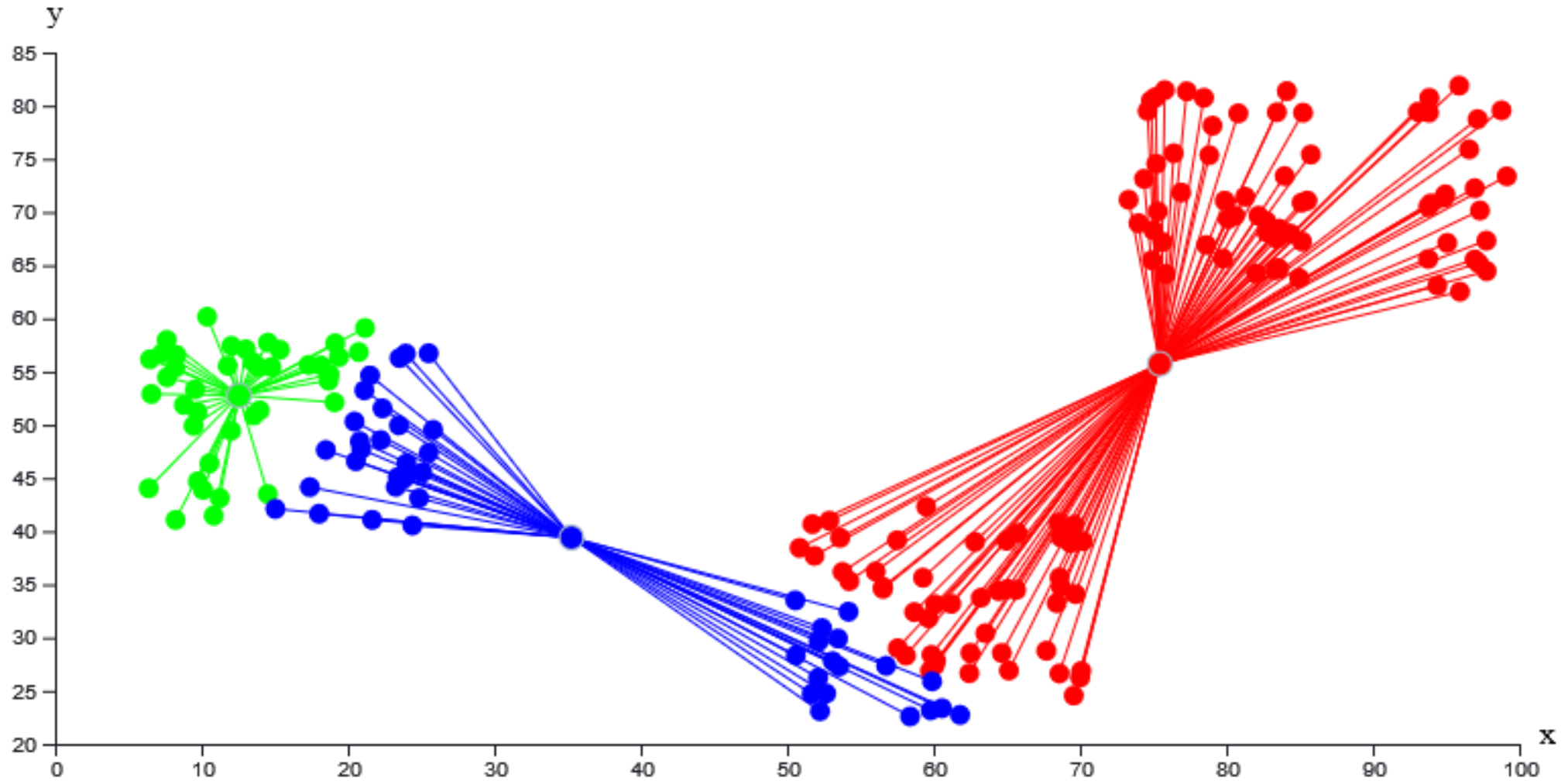
Centroids are randomly placed

Assign points



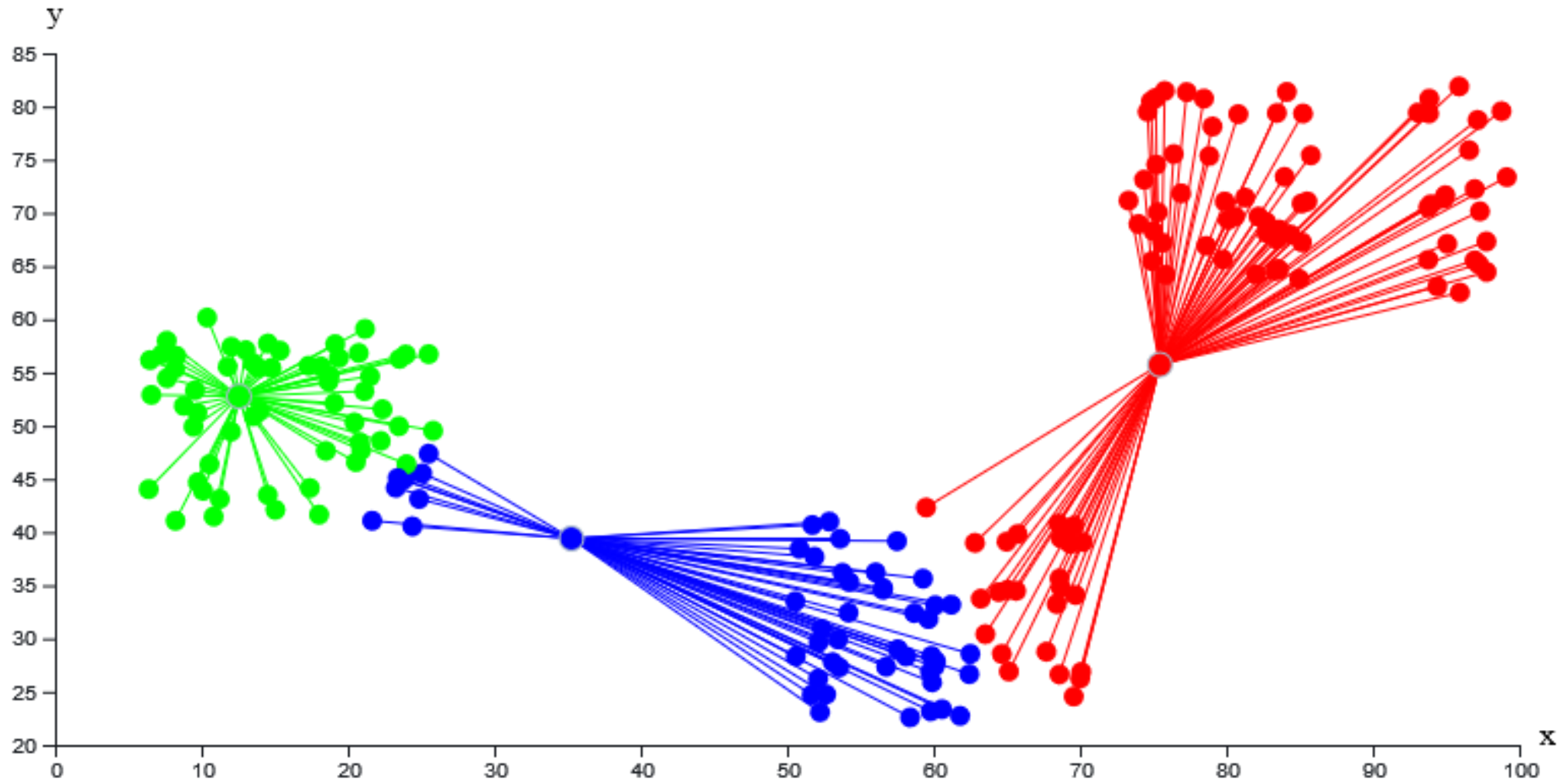
All points are assigned to the nearest centroid

Update centroids



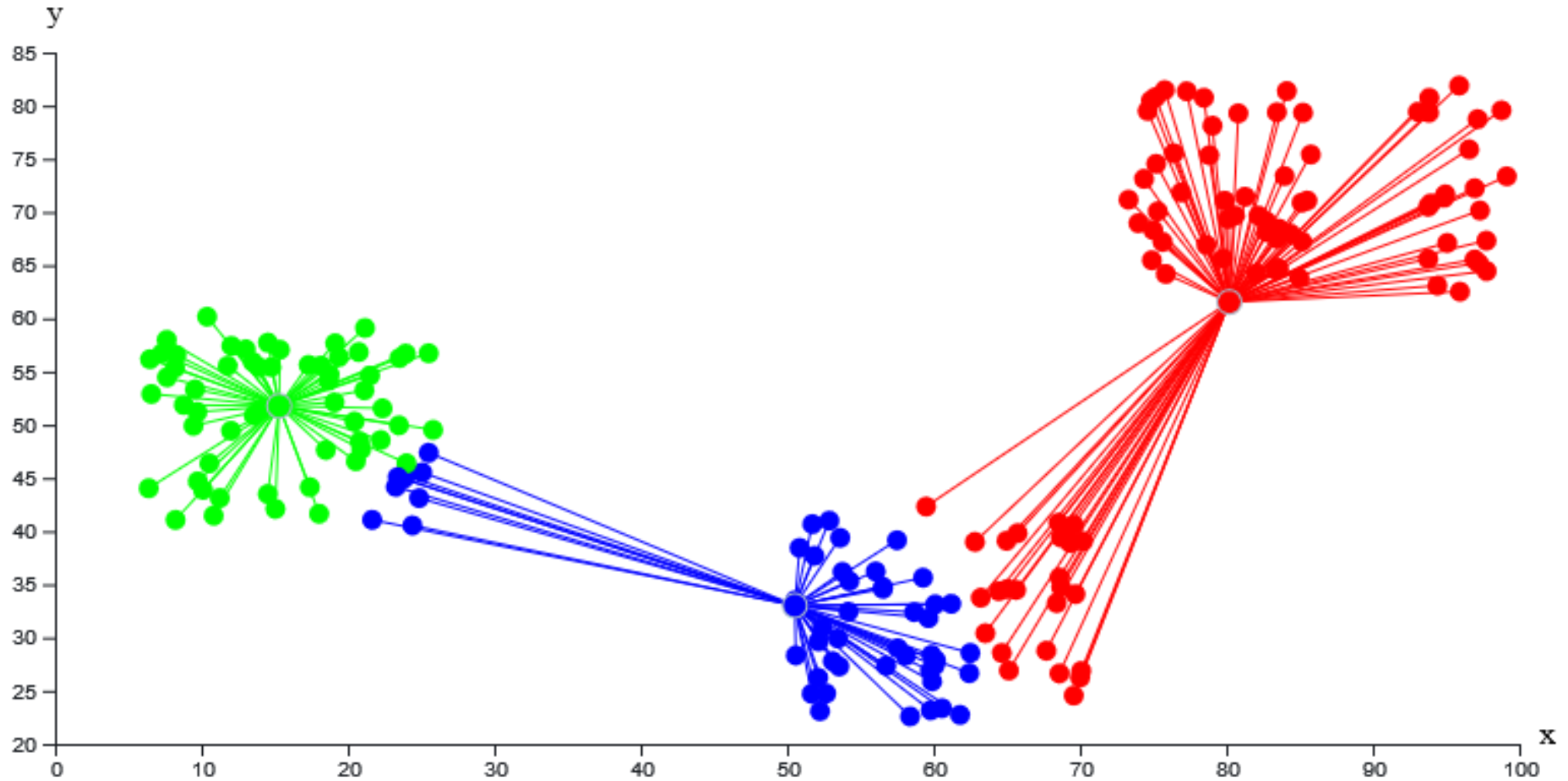
Centroids are updated to the mean of assigned points

Assign points



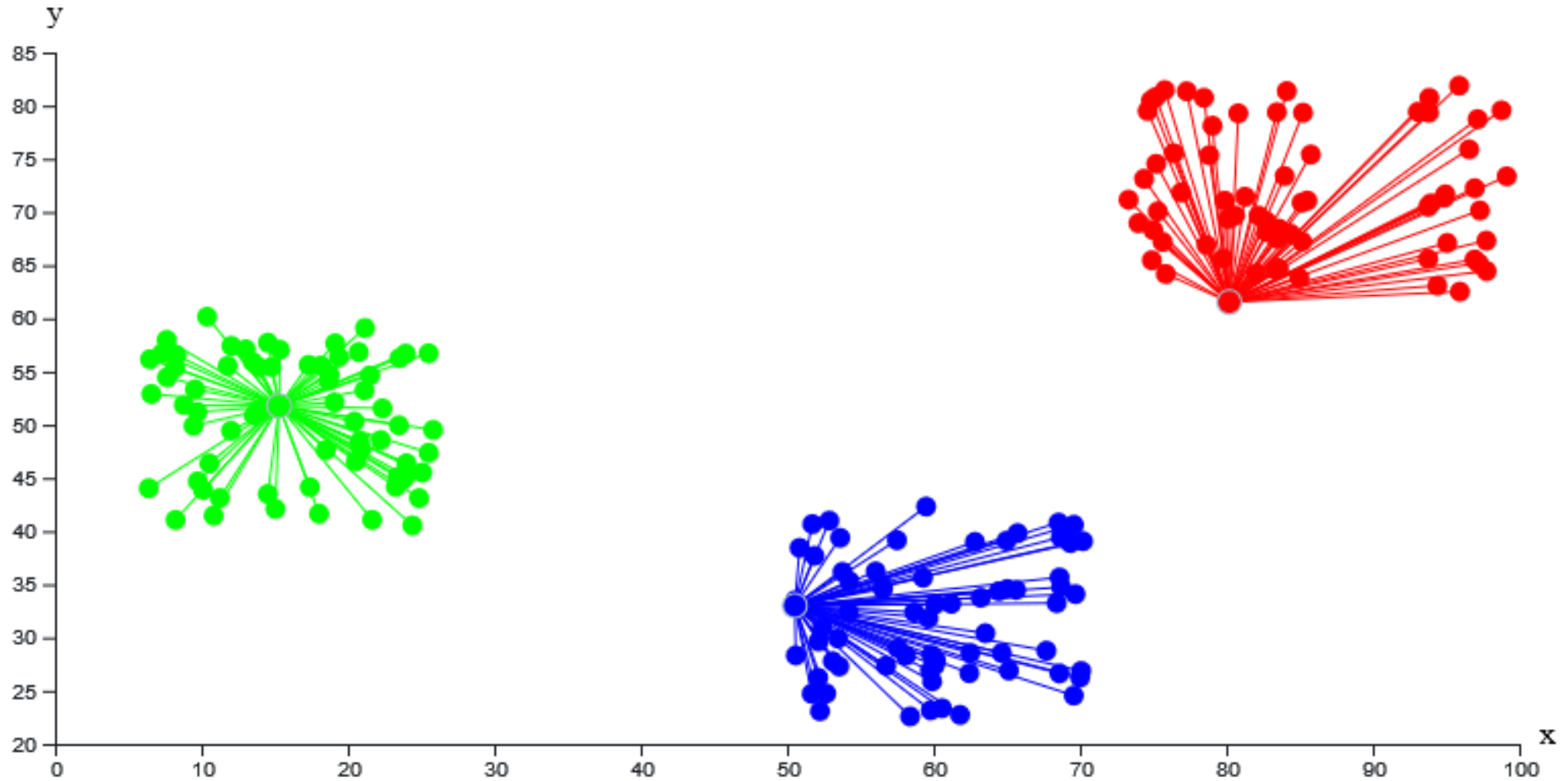
All points are assigned to the nearest centroid after update

Update centroids



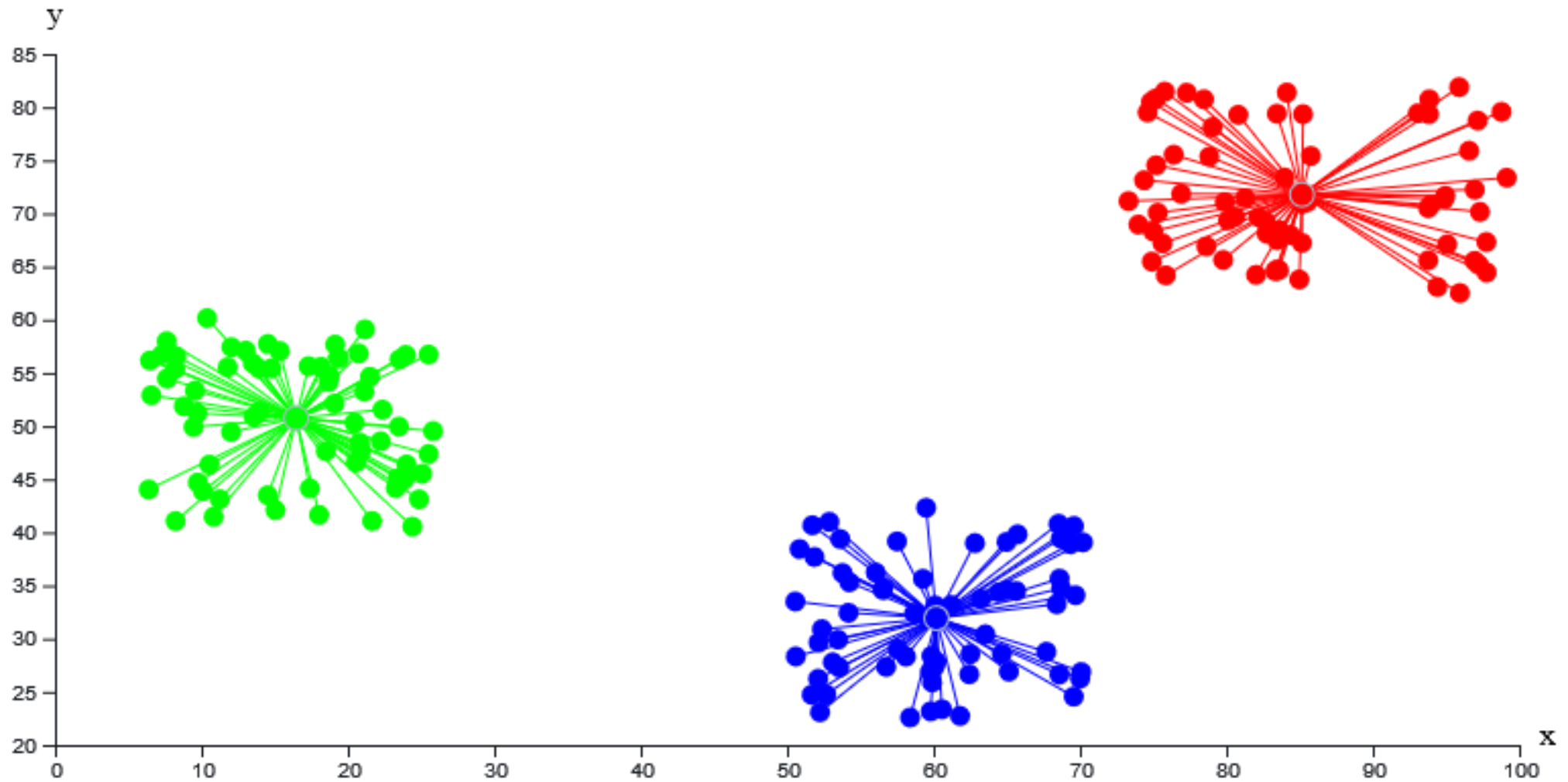
Centroids are updated to the mean of assigned points

Assign points



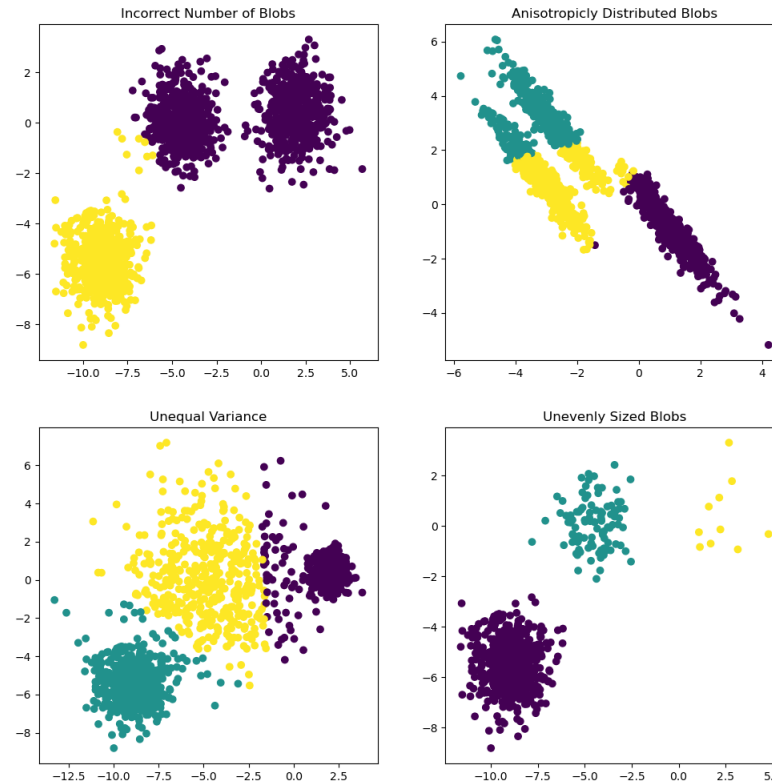
All points are assigned to the nearest centroid after update

Update centroids



At this point, we've converged

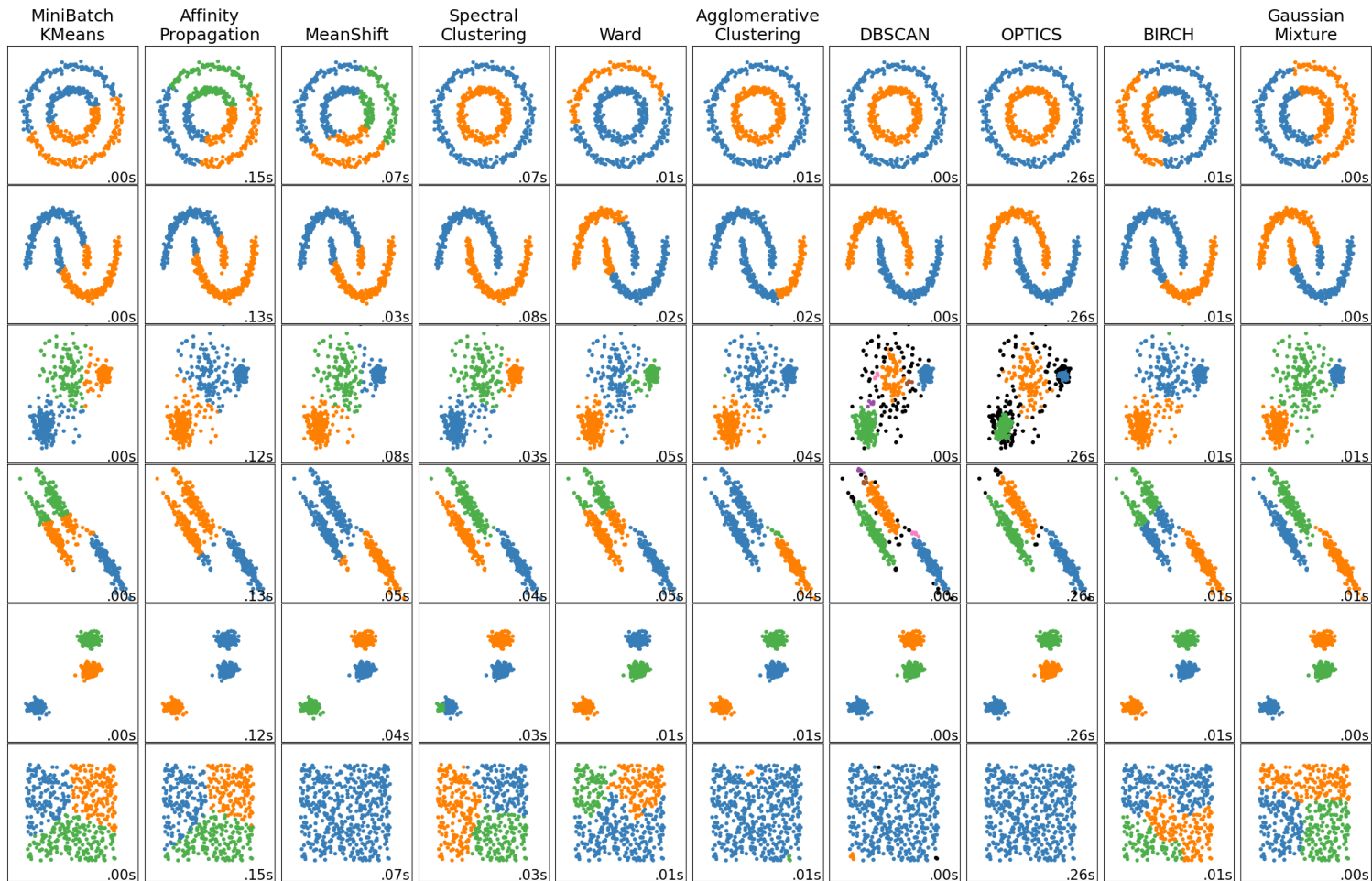
Assumptions



Violated assumptions

Source: [sklearn examples](#), 'Demonstration of k-means assumptions'

There are other methods



Methods implemented in sklearn

Source: [sklearn User Guide, 'Overview of clustering methods'](#)

Evaluation

It is generally not easy to evaluate clustering methods, but methods exist

These can be used to evaluate hyperparameter choice or method choice

- Visualize it
- Domain knowledge
- Elbow plots
- Silhouette coefficient

If ground truth is known, there are [more methods](#)

Choosing the right amount of cluster

Models optimize some sort of metric

- For K-means: Sum of squared distances

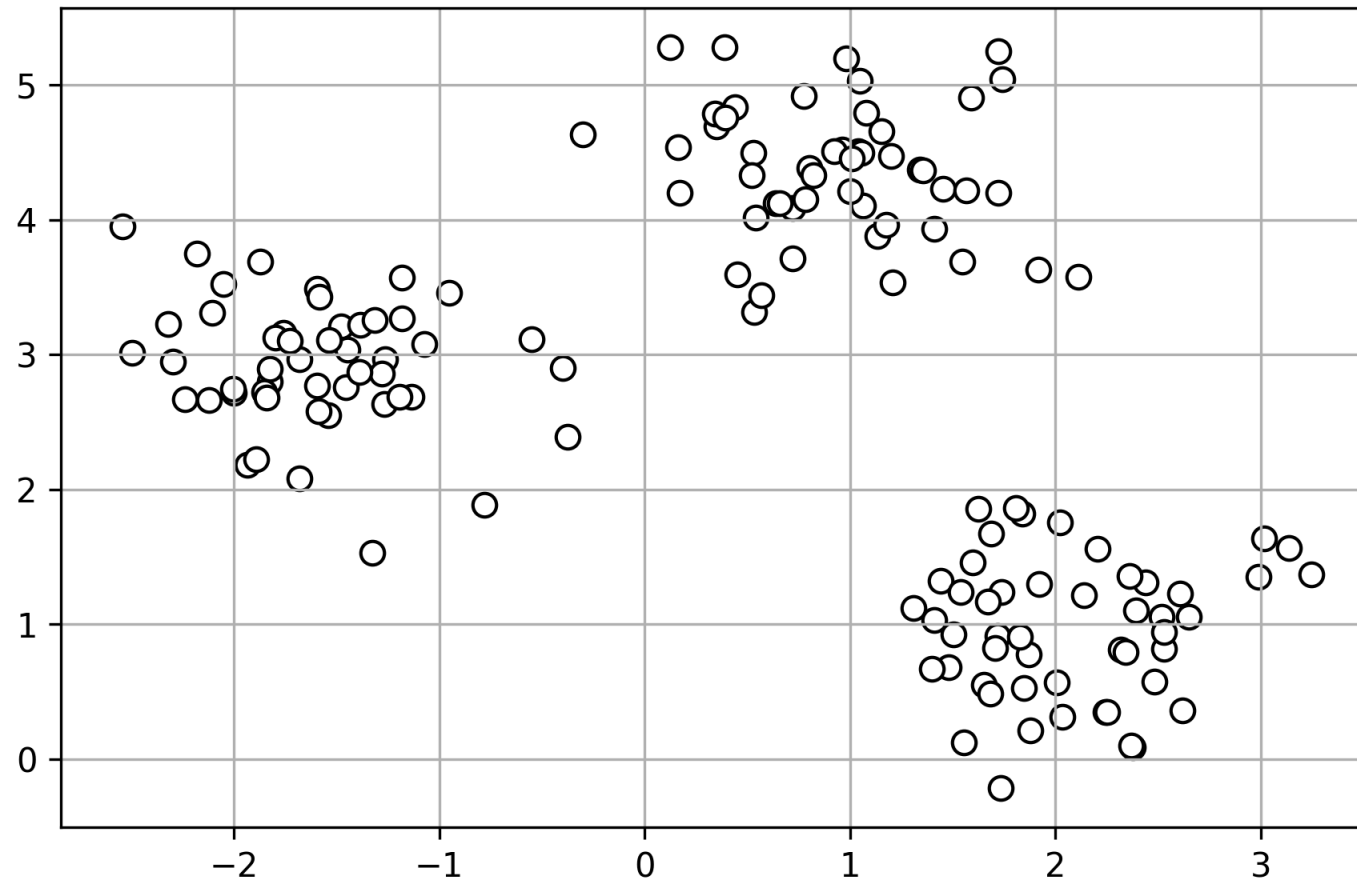
Generally, more clusters cause a better fit

When this additional increase in fit becomes small, we have 'enough' clusters

- Back to elbow plots!

If different models optimize different metrics, this cannot be used across models

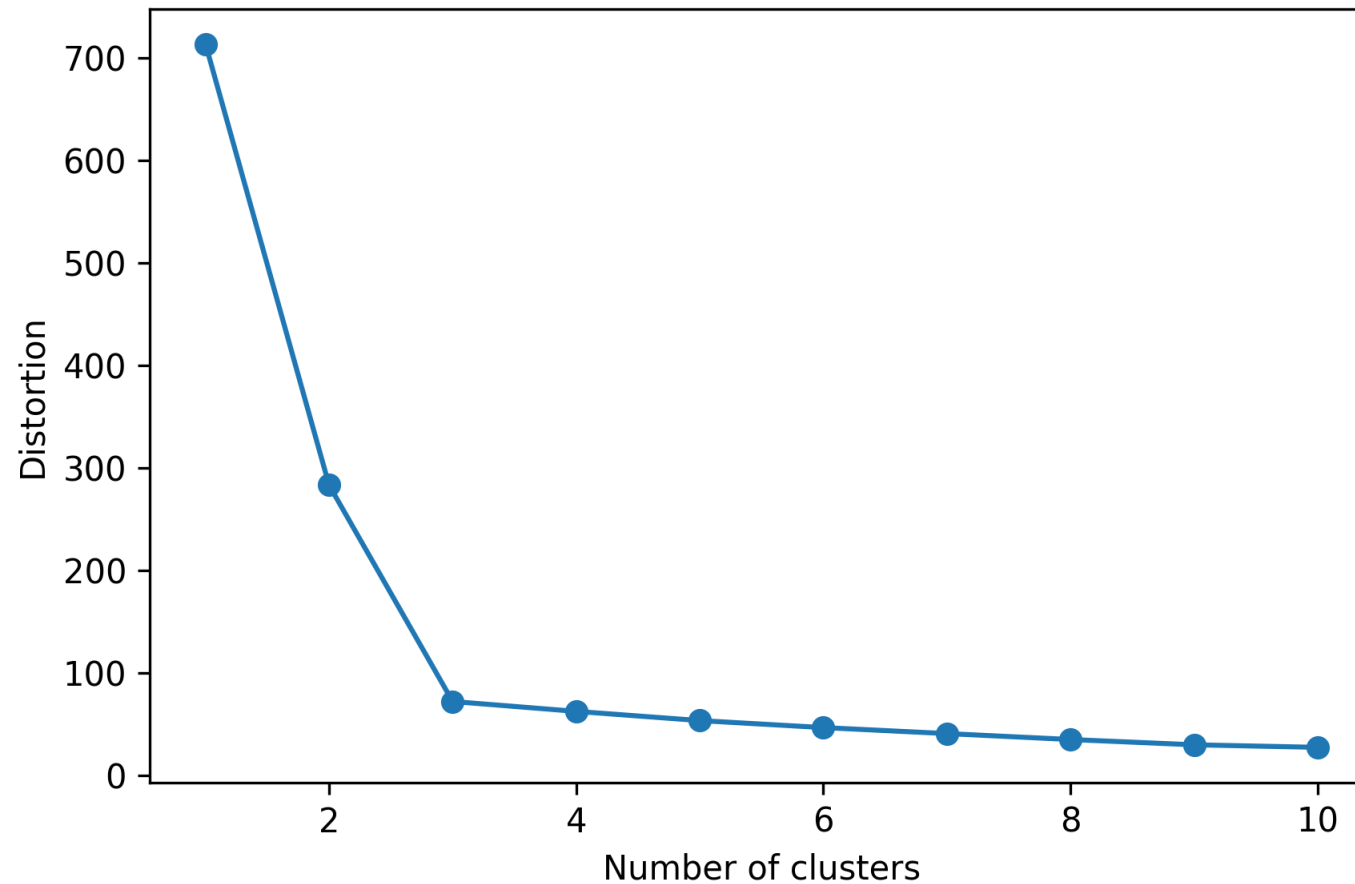
Example data



Clustered data

Source: Raschka & Mirjalili, 2022, ch. 11

K-means elbow plot



Sum of squared distances to nearest centroid as a function of cluster amount

Source: Raschka & Mirjalili, 2022, ch. 11

Silhouette scores

Based on the mean intra-cluster distance, a_i (cohesion), and the mean nearest-cluster distance, b_i (separation), for an observation i :

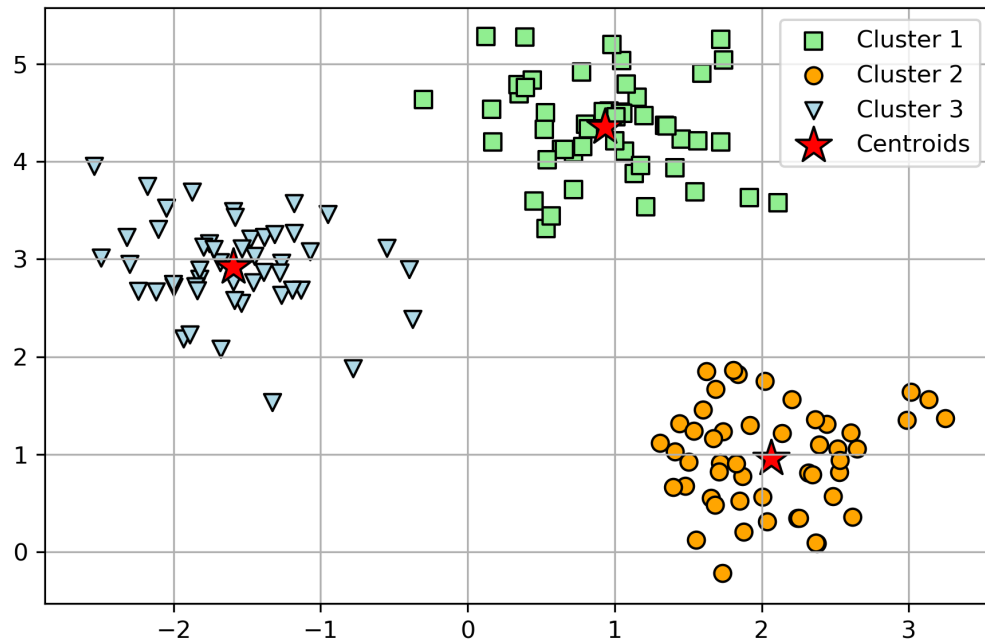
$$s_i = \frac{(b_i - a_i)}{\max(a_i, b_i)}$$

Bounded between -1 and 1,

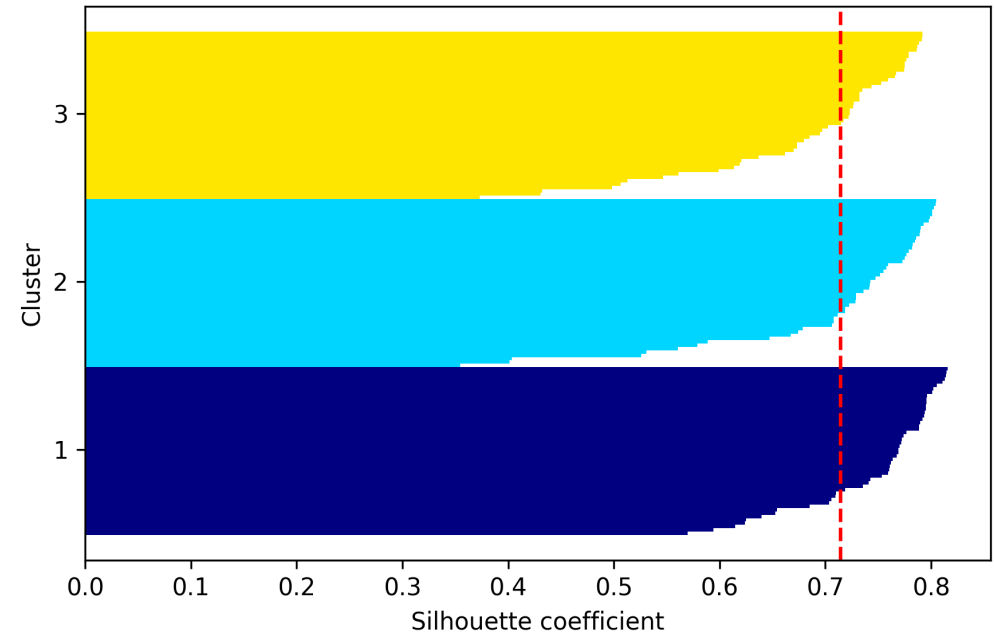
- Higher values are preferred
- Values of 0 indicate overlapping clusters
 - Cohesion equals separation

These can be plotted!

A good number



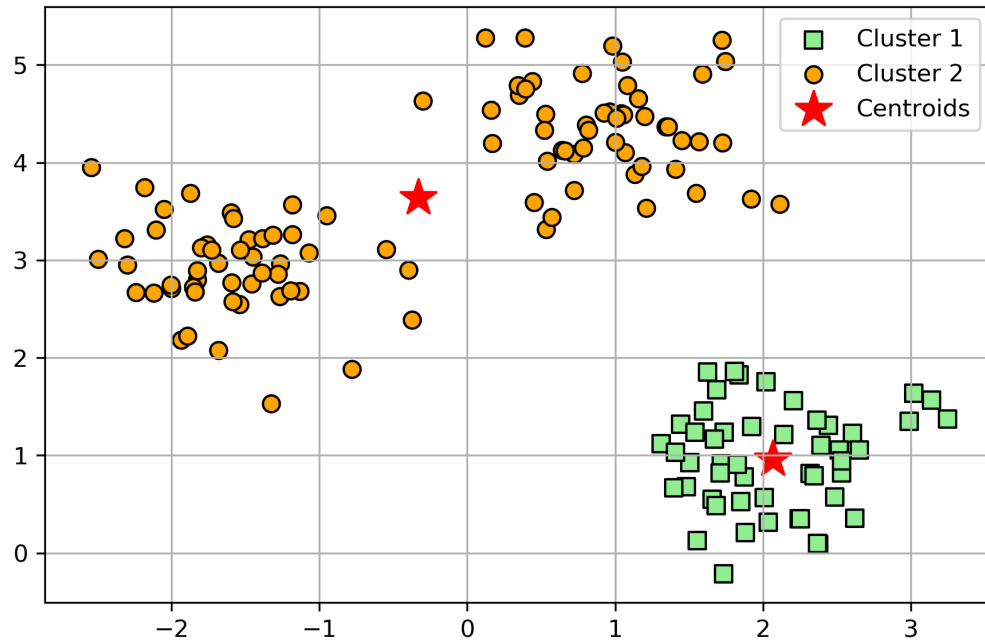
Three clusters



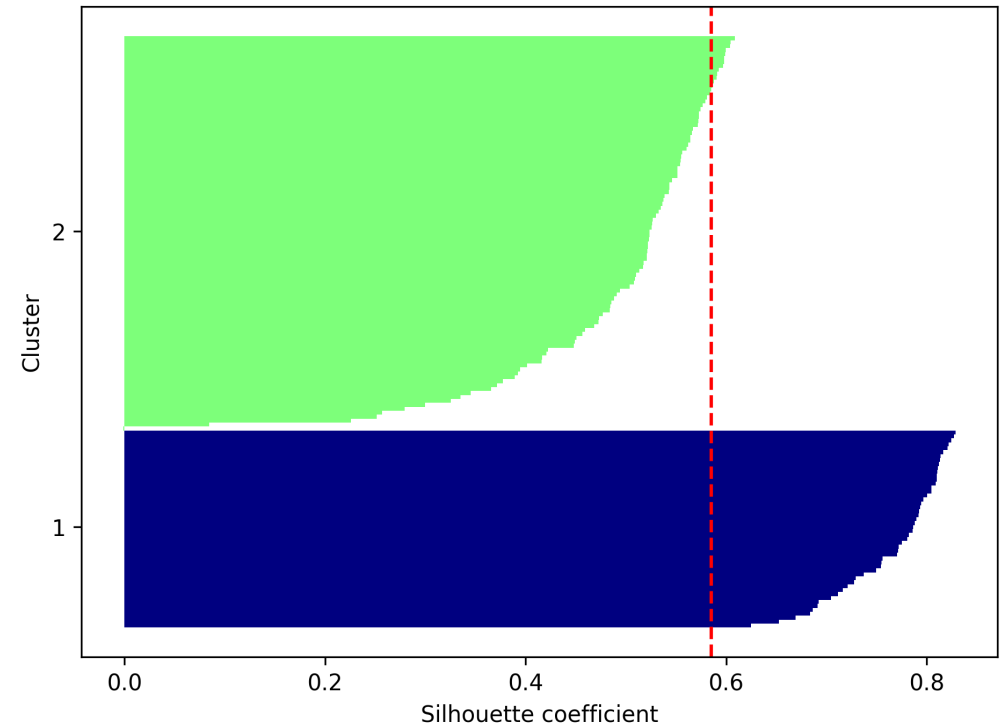
Silhouette plot for three clusters

Source: Raschka & Mirjalili, 2022, ch. 6

A not so good number



Two clusters



Silhouette plot for two clusters

Source: Raschka & Mirjalili, 2022, ch. 6

Convexity

Silhouette scores based on the idea that we want samples to be

- Near their cluster companions
- Far away from the nearest neighbor clusters

Works across models, but favors models that create convex clusters

Text as data

Question

Have any of you used text as data in an analysis?

Was it done using qualitative methods or quantitative?

Aim today

Natural Language Processing (NLP) is a huge field

Aim today is to get you thinking about text as data

- How do I start working with text
- What do people do

Some terminology

A word is the basic unit of discrete data

- E.g. a word, but can also be parts of words

A document is a collection of words

- E.g. a sentence, paragraph or full documents

A corpus is a collection of documents

- I.e. a dataset

How to use text as data?

There are many different ways of using text as data

- There is no ‘correct’ way
- The way in which you use text should be guided by what you’re interested in

The most basic method way to use text is to read the text

- Does not scale very well, to say the least...

Today we will focus on dictionary methods and bag-of-word models

Dictionary based methods

Words to values

The key in this method is a dictionary which includes:

- Words of interest
- Scores associated with these words

Using these dictionaries, we can

- Go through sentences and give each word a score
- Summarize these scores

How to get these dictionaries?

Dictionaries are generally either:

- General
 - Often a scale of an emotion (sentiment analysis)
 - More thoroughly tested and verified
- Handcrafted for the purpose
 - Subject specific and often very niche
 - No to little testing

Sentiment analysis

One well known dictionary is VADER (Hutto & Gilbert, 2014), Valence Aware Dictionary and sEntiment Reasoner

- Encodes sentiment as both *polarity* and *intensity*




Verified through experiments and benchmarks

- A benefit over handcrafted dictionaries

Scores every sentence according to how positive or negative they are (-1 to 1), and the proportion of negative, neutral and positive words

Examples

Text	Pos	Neu	Neg	Comp
VADER is smart, handsome, and funny.	0.746	0.254	0.0	0.8316
VADER is smart, handsome, and funny!	0.752	0.248	0.0	0.8439
VADER is very smart , handsome, and funny.	0.701	0.299	0.0	0.8545
VADER is VERY SMART , handsome, and FUNNY .	0.754	0.246	0.0	0.9227

Text	Pos	Neu	Neg	Comp
VADER is VERY SMART, handsome, and FUNNY!!!	0.767	0.233	0.0	0.9342
VADER is not smart, handsome, nor funny.	0.0	0.354	0.646	-0.7424
Make sure you :) or :D today!	0.706	0.294	0.0	0.8633
Catch utf-8 emoji such as  and  and 	0.279	0.721	0.0	0.7003

One small problem

It's not Danish

Here we can use **AFINN** instead

- Less nuanced
- Multiple languages
- Positive and negative words on a scale from -5 to 5
- Mean calculated at the sentence level

The difference

‘VADER is **not** smart, handsome, nor funny.’ classified as positive

- Picks up on smart, handsome and funny
- Does not capture the negation

You can play around with it at [this website](#), both in Danish and English

Use case: Racial animus in politics

Stephens-Davidowitz (2014) examines whether prejudice against blacks remains a potent factor within American politics (at the time)

People sometimes lie on surveys

- Could be a problem when asked about racial animus

A handcrafted dictionary

To examine this, they use Google searches up to the 2008 election in the US split by media market and examine the ‘racially charged search rate’, which is a fraction of the type

$$\frac{\text{Google searches including some word(s)}}{\text{All Google searches}}$$

Essentially counting

Question

What word(s) would you include in this dictionary?

Simplicity at its best

Stephens-Davidowitz (2014) uses just a single derogatory term

- Can be found in the appendix for the curious

Using this measure, they find that it is a robust negative predictor of Obama's performance

- Racial animus causes about a 4%-point loss in the elections in 2008 and 2012

Downsides of dictionary based methods

Requires a dictionary

- Getting an appropriate one can be hard

Sometimes too simple

- Hard time with subtleties, i.e. difference between *I hate this* and *I do not hate this*

Bag-of-words

Question

How could one represent sentences numerically?

Counting words systematically

We can treat words as unique tokens

- Count how often each word appears in a sentence
- Much like categorical variables

High-dimensional and sparse

- Good that we know both how to regularize and reduce dimensionality!

An example

Sentence	dogs	like	i	bananas
Dogs like dogs	2	1	0	0
I like dogs	1	1	1	0
I like bananas	0	1	1	1

This allows us to go from sentences to a tabular format

A variant: tf-idf

A common variant is a term frequency times inverse document (tf-idf) matrix

- Here we normalize by how often the word appears in documents
- Intuition is that words which appear in only a few documents are more important for those documents than those who appear in many documents

A variant: n-grams

N-gram models are also another common variant

- Extend the simple unigram bag-of-words to consecutive words
 - e.g. Bigram ($n = 2$) models would also include all pairs of words in order
 - 'Dogs like dogs' would become {'dogs':2, 'like':1, 'dogs like':1, 'like dogs':1}

Common preprocessing

You should consider whether you want to:

- Lowercase everything
- Remove stopwords
 - See e.g. `nltk.corpus.stopwords.words('danish')` or `spacy.lang.da.stop_words.STOP_WORDS`
- Remove punctuation
- Either stem or lemmatize words
 - Stemming removes endings based on some fixed rules, see e.g. `nltk.stem.snowball.DanishStemmer`
 - Lemmatization aims to obtain the grammatically correct form, see e.g. [lemmy](#)
 - Reduces dimensionality
- Remove rare and/or common words

You're the experts

In the end, it's text

- Read it, get a feel for it!
- Why are you interested in this text?

Room for subject-specific preprocessing

- What to do with links? Hashtags? Emojis?
- Specific terms such as names
- `regex` is a nice tool for working with text
 - Can be daunting

Many further avenues

- Deep learning, e.g. transformers
 - Can do translation, prediction, generation and more
 - [HuggingFace](#) is a repository of (relatively) easily accessible models
- Topic modelling, e.g. Latent Dirichlet Allocation
 - Find topics in corpus and assign words and documents to topics
 - Focus on interpretability

References

Hutto, C., & Gilbert, E. (2014, May). Vader: A parsimonious rule-based model for sentiment analysis of social media text. In Proceedings of the international AAAI conference on web and social media (Vol. 8, No. 1, pp. 216-225).

Raschka, S., Liu, Y. H., Mirjalili, V., & Dzhulgakov, D. (2022). Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python. Packt Publishing Ltd.

Stephens-Davidowitz, S. (2014). The cost of racial animus on a black candidate: Evidence using Google search data. *Journal of Public Economics*, 118, 26-40.

To the exercises!

