Machine Intelligence

10. Unsupervised Learning: Clustering

Learning on your own!

Álvaro Torralba

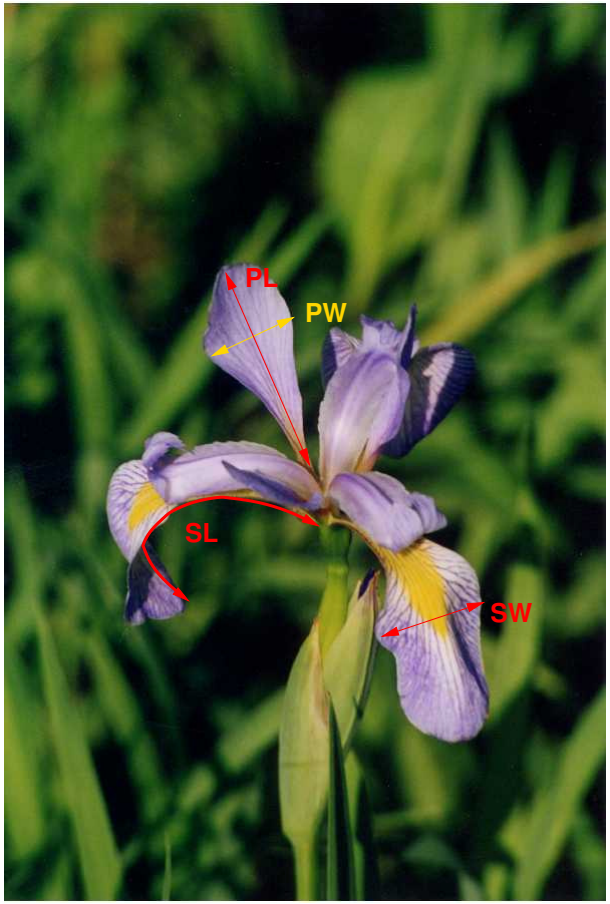
AALBORG UNIVERSITET

Fall 2022

Thanks to Thomas D. Nielsen and Jörg Hoffmann for slide sources

So far... Supervised Learning

Supervised Learning: Learn from labelled data



Example: Iris Dataset

Measurement of petal width/length and sepal width/length for 150 flowers of 3 different species of Iris.

first reported in:
Fisher,R.A. "The use of multiple measurements in taxonomic problems" Annual Eugenics, 7 (1936).

Attributes				**Class variable**
SL	SW	PL	PW	Species
5.1	3.5	1.4	0.2	Setosa
4.9	3.0	1.4	0.2	Setosa
6.3	2.9	6.0	2.1	Virginica
6.3	2.5	4.9	1.5	Versicolor
...	...	...	...	...

→Given the value of the input attributes, predict the value of the target variable

Unsupervised Learning

- Obtaining labelled data is expensive (typically done manually) or even impossible
- Sometimes we have a lot of data but we do not know what kind of question to ask

Suppose that we have a dataset without any target variable:

	Attributes		
SL	SW	PL	PW
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
6.3	2.9	6.0	2.1
6.3	2.5	4.9	1.5
...	...	...	...

Can we learn something, even if we do not know the answers (or the questions for that matter)?

Yes! We can try to learn **patterns or structural properties of the data**

$\rightarrow$ In the case of the Iris, instead of predicting the species as defined by humans, our learning algorithm can "invent" or "discover" its own species classification!

Our Agenda for this Chapter

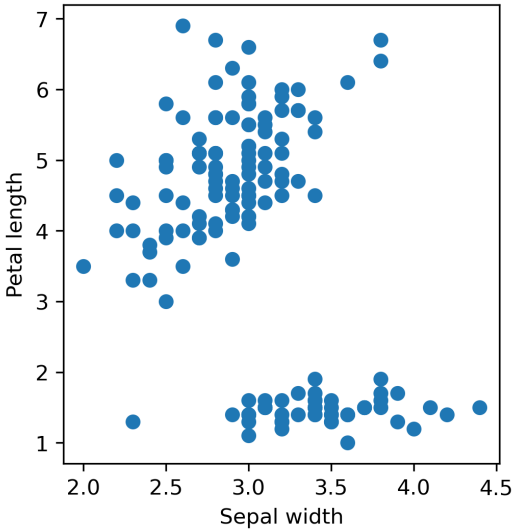
- **Clustering**. A problem for unsupervised learning
  → **What does clustering mean and what is it useful for?**
- **k-means**: A clustering method
  → **How can we find clusters in the dataset?**
- **Evaluation of Clustering Models**:
  → **How to know if the model is correct?**
- **Preprocessing Optimizations**: Excluding Outliers and Normalization
  → **Is it possible to do it better?**
- **Soft Clustering: The EM Algorithm**
  → **Using probabilities once again**
- **Autoencoders**: Unsupervised learning using neural networks
  → **What the hell is that?**
- **Semi-supervised learning**: Combining supervised and unsupervised learning
  → **Taking advantage of the information we already have**

Clustering Definition

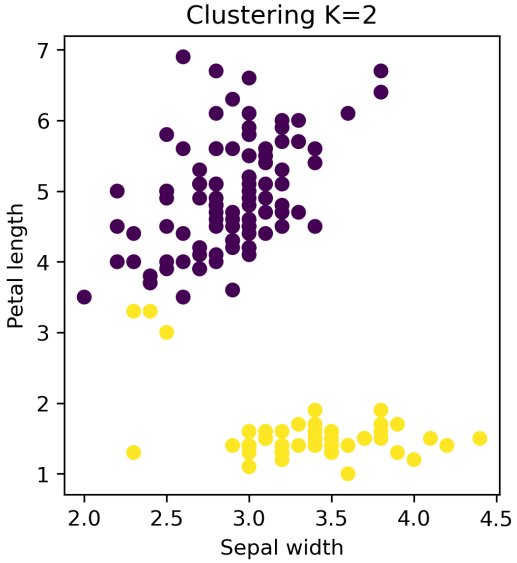
Definition (Clustering). A clustering of the data examples $E = \mathbf{e}_1, \ldots, \mathbf{e}_N$ consists of a set $C = \{c_1, \ldots, c_k\}$ of *cluster labels*, and a *cluster assignment* $ca : E \to C$.

In the unlabelled Iris-Dataset, can get find a pattern in the data?

	Attributes		
SL	SW	PL	PW
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
6.3	2.9	6.0	2.1
6.3	2.5	4.9	1.5
...	...	...	...



Unlabeled Iris

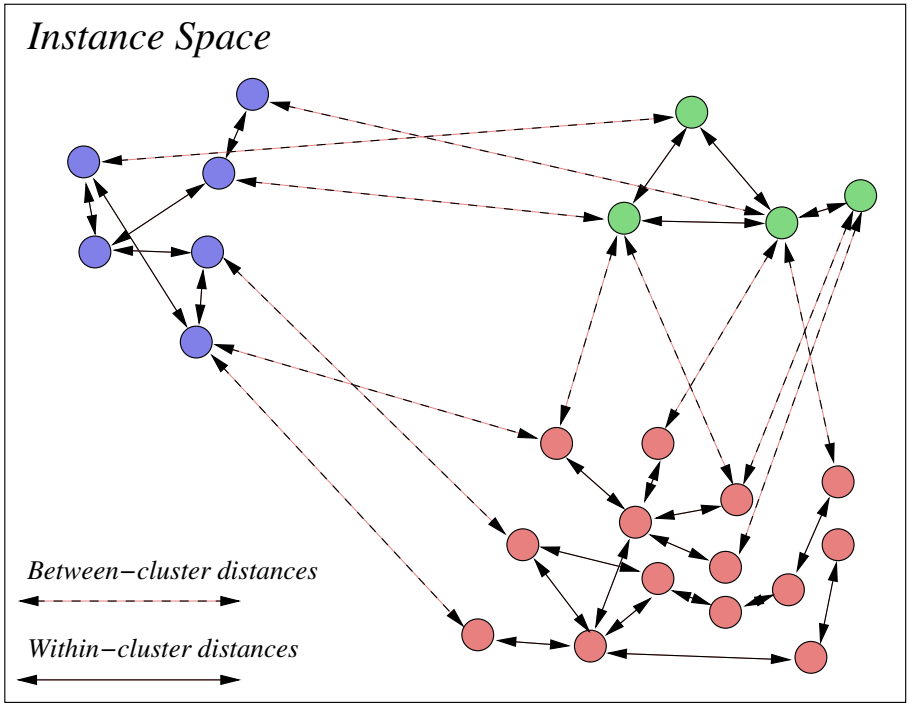


Clustering Iris with $C = \{class1, class2\}$:

Yes, we can divide our data into two distinct clusters!

Note: Here we are only plotting 2 of the 4 dimensions (so there may be further patterns hidden in the data)

What is a Good Clustering?



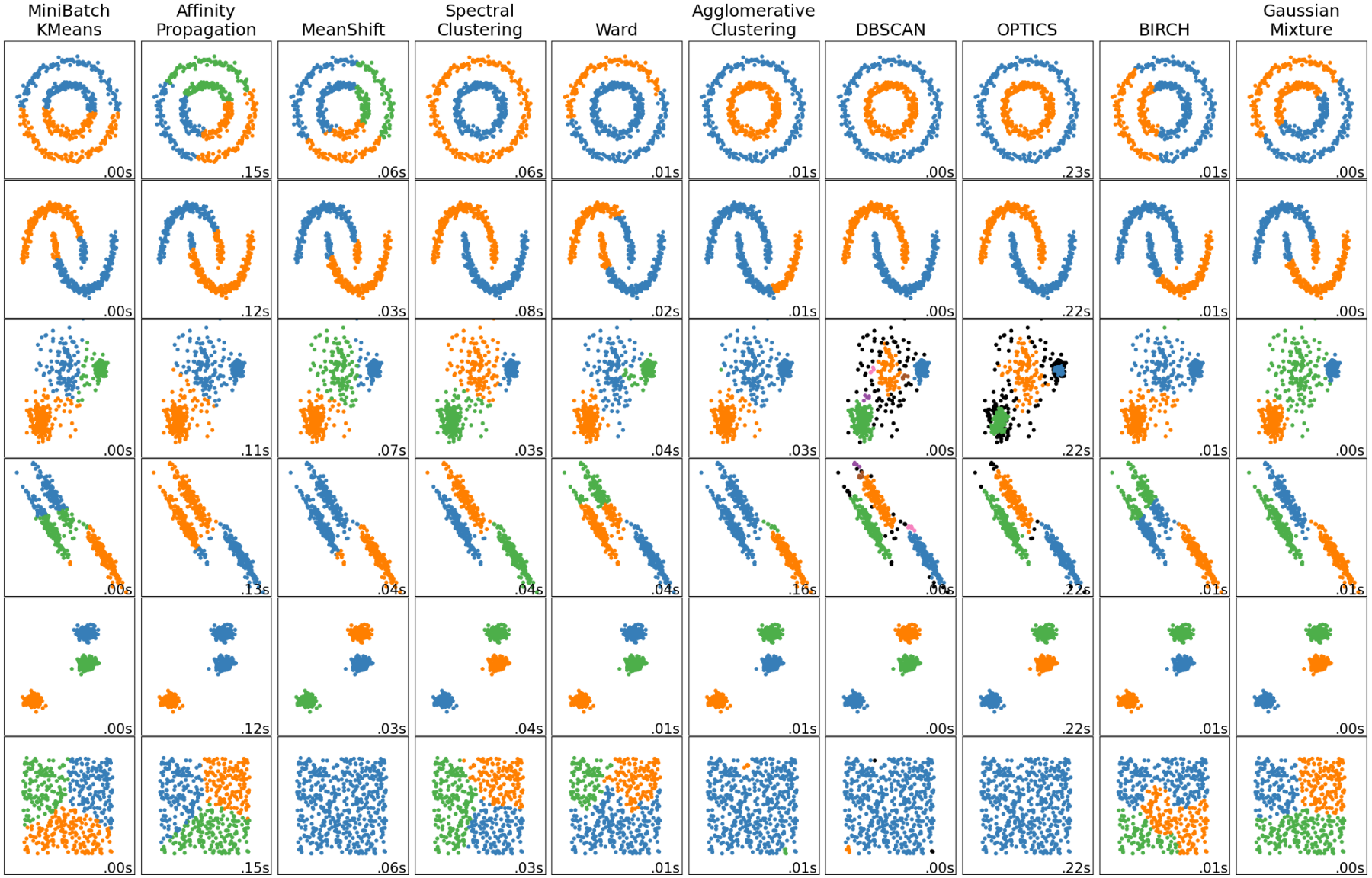
A candidate clustering (indicated by colors) of data cases in instance space. Arrows indicate between- and within-cluster distances (selected).

In this lecture, we will focus on finding clusterings such that:

- large between-cluster variation (sum of between-cluster distances)
- small within-cluster variation (sum of within-cluster distances)

(where distance function can be e.g. the Euclidean distance).

What is a Good Clustering?



Note: There are many clustering algorithms and distance metrics are not the only option to extract useful patterns

Clustering: Applications

- Based on customer data, find groups of customers with similar profiles.
- Based on image data, find groups of images with similar motif.
- Based on article data, find groups of articles with the same topics.
- ...

Also, unsupervised learning can be used as a preparation to perform supervised learning (see end of this chapter)

The k-means algorithm

We consider the scenario, where

- the number k of clusters is known.
- we have a distance measure $d(\mathbf{x}_i, \mathbf{x}_j)$ between pairs of data points (feature vectors).
- we can calculate a centroid for a collection of data points $S = \{\mathbf{x}_1, \ldots \mathbf{x}_n\}$.

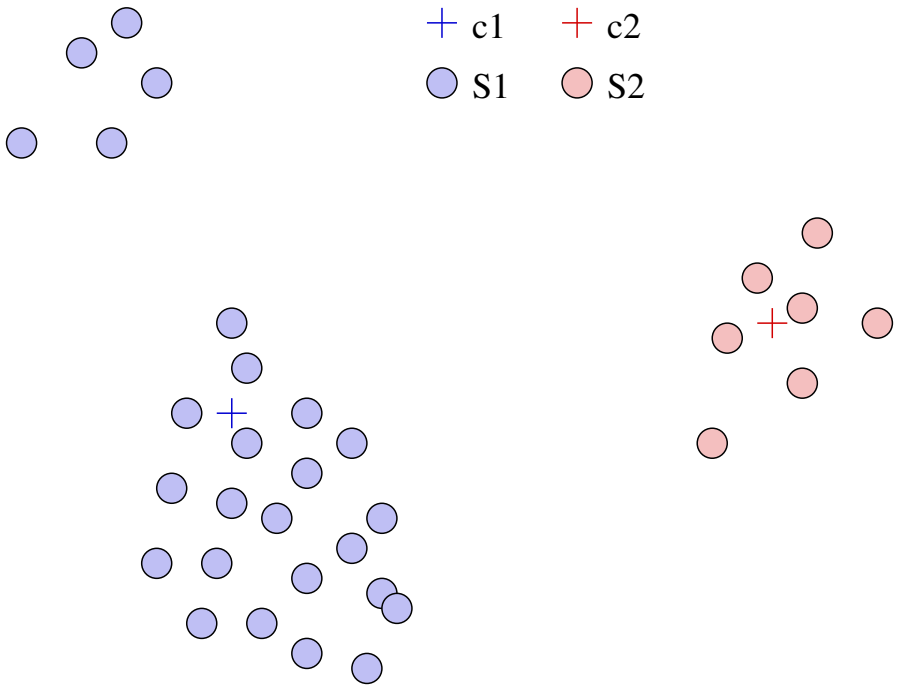
Initialize: *randomly pick k data points as initial cluster centers* $\mathbf{c} = c_1, \ldots, c_k$ from S
`repeat`
    Form k clusters by assigning each point in S to its closest centroid.
    Recompute the centroid for each cluster.
`until` Centroids do not change

The k-means algorithm: Example

$k = 3$:

Different k

Result for clustering the same data with $k = 2$:



Result can depend on choice of initial cluster centers!

The k-means algorithm: Properties

Convergence

The k-means algorithm is guaranteed to converge

- Each step reduces the sum of squared errors
- There is only a finite number of cluster assignments

Optimality Analysis

k-means is not guaranteed to reach the global optimum:

- Performance depends on the random initialization
- Improve by running with multiple random restarts

The k-means algorithm: Background

k-means as an optimization problem

Assume that we use the Euclidean distance d as proximity measure and that the quality of the clustering is measured by the sum of squared errors:

$$SSE = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} d(\mathbf{c}_i, \mathbf{x})^2,$$

where:

- $\mathbf{c}_i$ is the i'th centroid
- $C_i \subseteq S$ is the points closets to $\mathbf{c}_i$ according to d.

In principle ...

We can minimize the SSE by looking at all possible partitionings $\rightsquigarrow$ not feasible!

Instead, k-means

The centroid that minimizes the SSE is the *mean* of the data-points in that cluster:

$$\mathbf{c}_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$$

Cluster evaluation

A clustering algorithm applied to a dataset will return a clustering - even if there is no meaningful structure in the data!

Question: Do the clusters actually correspond to meaningful groups of data instances?
Question: Are all the clusters relevant, or are there some real and some meaningless clusters?

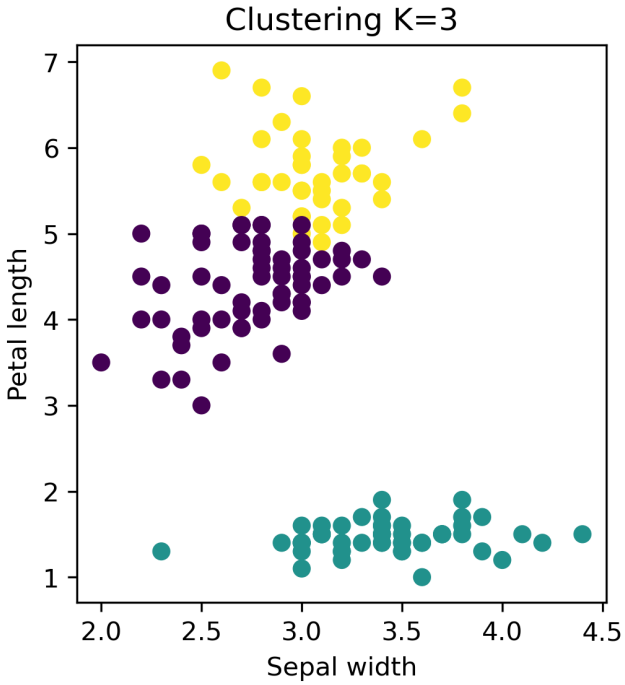
Two types of evaluation:

- Supervised
- Unsupervised

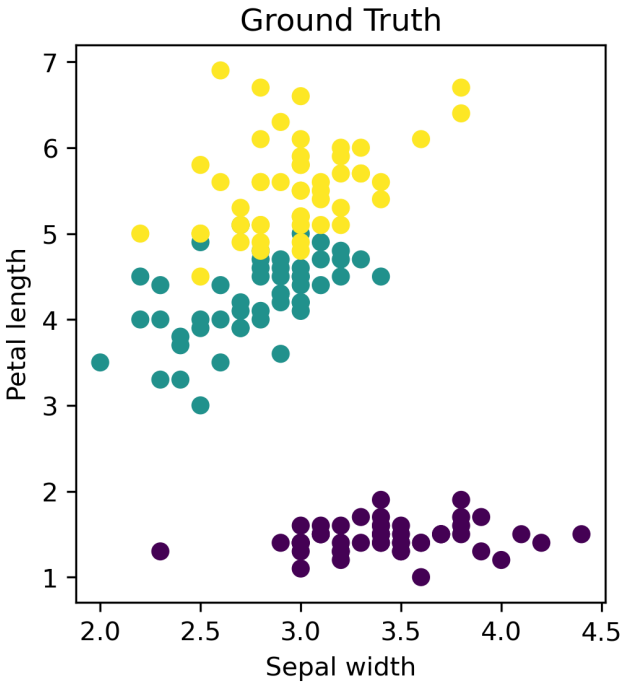
Supervised Evaluation of Clustering Approaches

Uses external information, e.g. a true class label as the "gold standard" for actual groups in the data

Example: Clustering Iris with 3-means



Clustering found by k-means

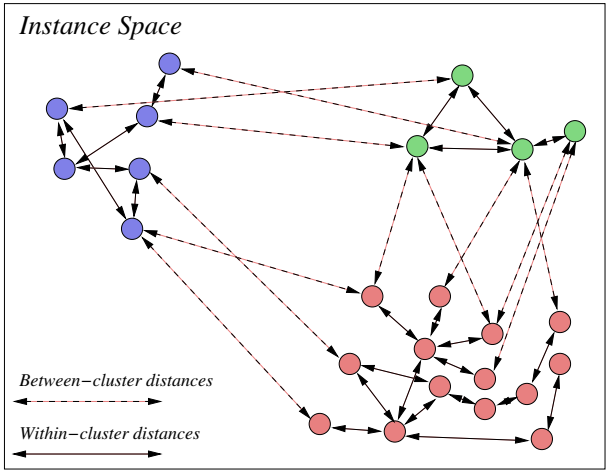


Iris true classes

- Not representative for actual clustering applications
- Can be useful to evaluate *clustering algorithms*
- Caveat: no guarantee that the class labels actually describe the most natural or relevant groups in the data

Unsupervised Evaluation of Clustering Approaches

- Uses only the data as given to the clustering algorithm, and the resulting clustering
- The realistic scenario: If you have labelled data, why not doing supervised learning instead?
- Example: distance between clusters:



$\rightarrow$ Other metrics are out of the scope of this course

Some Practical Issues

Some Practical Issues:

1. Outliers
2. Different Measuring Scales

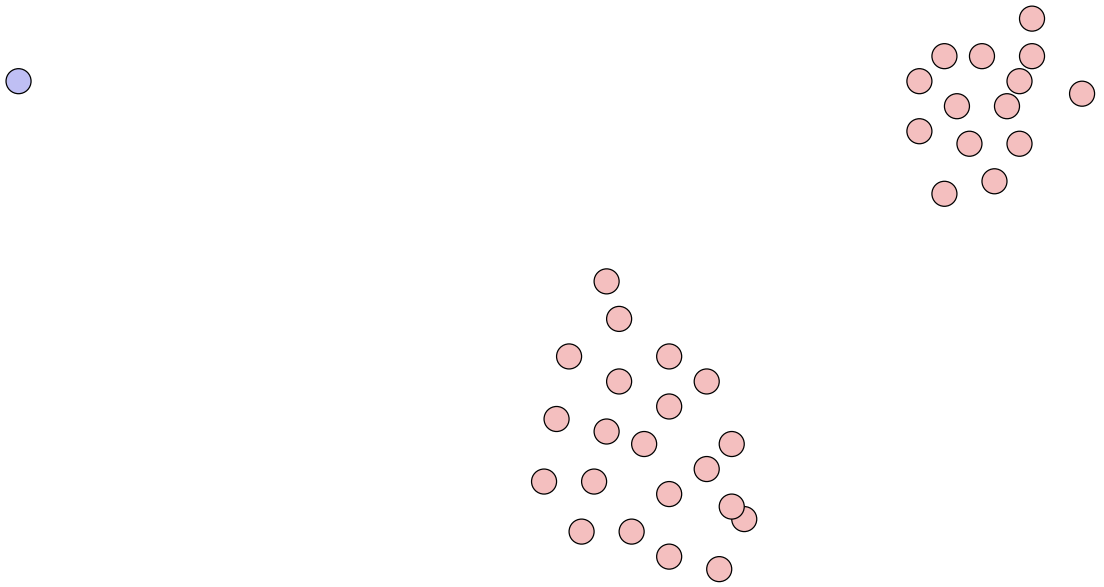
We can address them by some "massaging" of the data

Outliers

Issue: Outliers

The result of partitional clustering can be skewed by outliers

Example with $k = 2$:



Possible solution:

$\rightsquigarrow$ useful preprocessing: outlier detection and elimination.

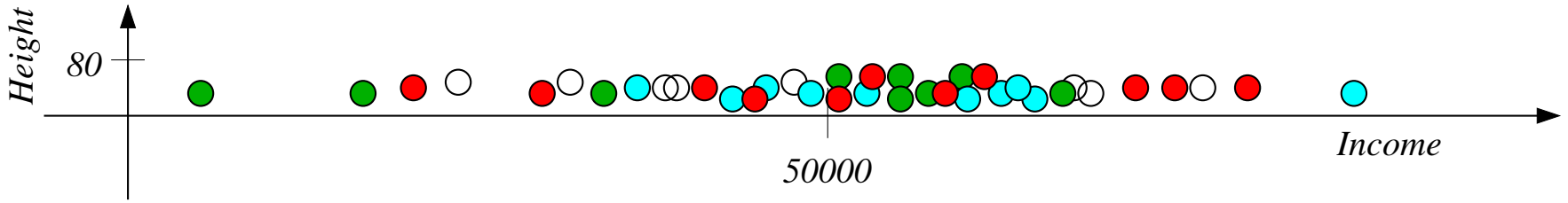
Different Measuring Scales

Issue: Different Measuring Scales

If attributes are measured in different scales, the Euclidean distance is heavily biased and completely ignores those in smaller scales!

Example: Dataset with two attributes

$$A_1 = \textit{height in inches} \text{ and } A_2 = \textit{annual income in \$}:$$



- Height is irrelevant because distances are dominated by the difference in income
- Problem is that we are comparing attributes in different units/magnitudes
- $\rightsquigarrow$ may need to *rescale* or *normalize* continuous attributes. We see two ways to do so:
  1. Min-Max Normalization
  2. Z-score Normalization

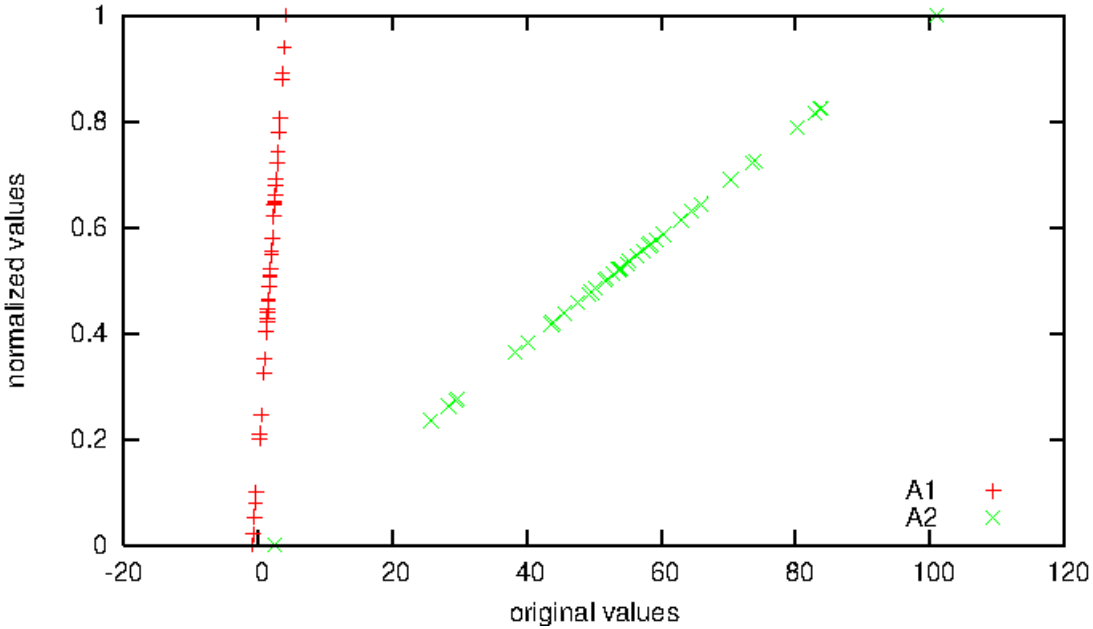
Min-Max Normalization

Min-Max Normalization

replace A_i with

$$\frac{A_i - \min(A_i)}{\max(A_i) - \min(A_i)}$$

($\min(A_i), \max(A_i)$ are min/max values of A_i appearing in the data)



Properties

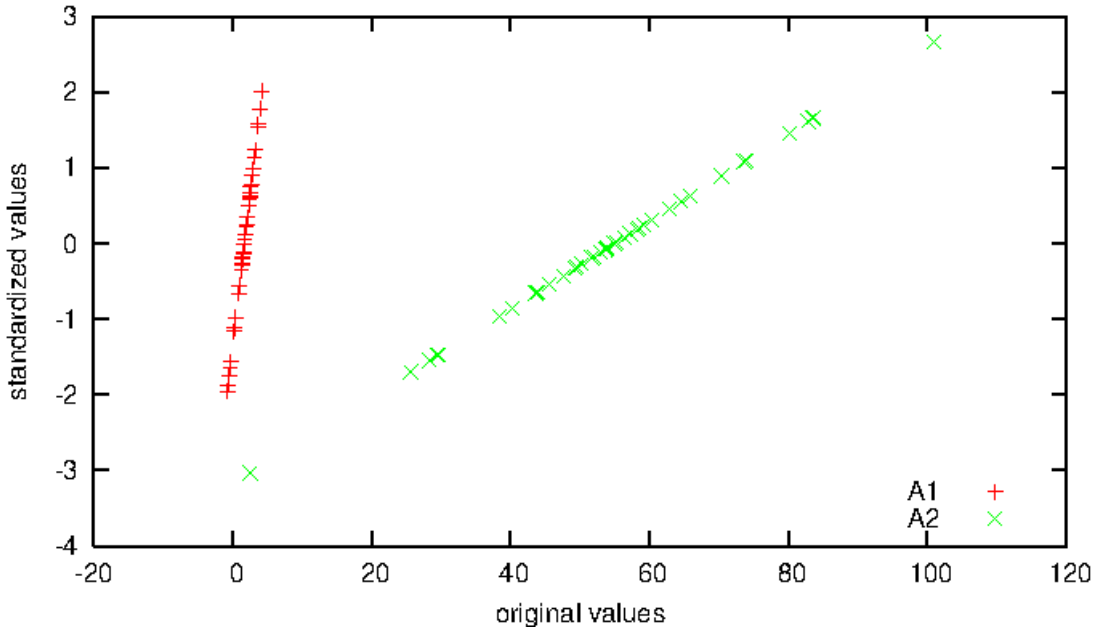
- Ensures that all attributes have only values between 0 and 1
- Very sensitive with respect to outliers

Z-Score Standardization

Z-score Standardization

replace A_i with

$$\frac{A_i - mean(A_i)}{standarddeviation(A_i)}$$



where

$$mean(A_i) = \frac{1}{n}\sum_{j=1}^{n} a_{j,i}$$

$$standarddeviation(A_i) = \sqrt{\frac{1}{n-1}\sum_{j=1}^{n}(a_{j,i} - mean(A_i))^2}$$

Properties

- Intuitively, it measures how many standard deviations is each example far from the mean
- Less sensitive to outliers
- Not all attributes have the same range (but they are still relatively close)

Soft Clustering

The k-means algorithm generates a *hard* clustering: each example is assigned to a single cluster.

Original dataset

F_1	F_2	F_3
t	t	t
t	f	t
t	f	f
f	f	t

Hard clustering (K=2)

F_1	F_2	F_3	C
t	t	t	$class1$
t	f	t	$class2$
t	f	f	$class1$
f	f	t	$class1$

Soft clustering (K=2)

F_1	F_2	F_3	$class1$	$class2$
t	t	t	0.9	0.1
t	f	t	0.2	0.8
t	f	f	0.7	0.3
f	f	t	1	0

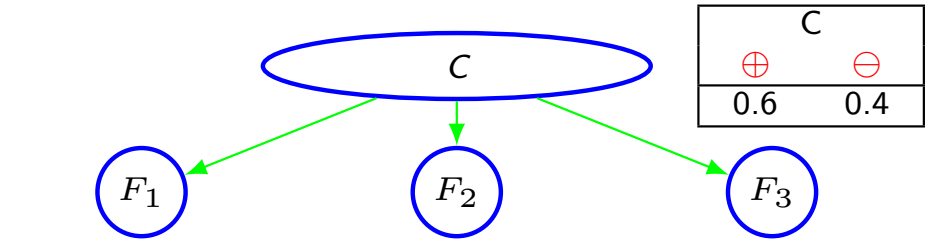
Soft Clustering: Each example is assigned to a cluster with a certain probability
We need:

1. A way of representing the mapping from examples to probabilities. Any ideas? (Note that in this section, we assume for simplicity that our features are discrete)
2. An algorithm to decide how to cluster the data (an alternative to k-means that works with probabilities) $\rightarrow$ **Expectation-Maximization**

Soft Clustering Using Naive Bayes

We use a Naive Bayes Network to represent the soft clustering

We can compute $P(C|F_1, F_2, F_3)$ to retrieve the probability of belonging to each class! (c.f. **Chapter 6**)



C	⊕	⊖
	0.6	0.4

C	F_1 t	f
⊕	0.6	0.4
⊖	0.4	0.6

C	F_2 t	f
⊕	0.6	0.4
⊖	0.4	0.6

C	F_3 t	f
⊕	0.6	0.4
⊖	0.4	0.6

Question!

Considering the network above, compute the probability that each example belongs to ⊕. Which of them have probability ≥ 0.5?

(A): ttt **(B):** tft

(C): tff **(D):** fft

How to Fill In Probability Tables: Expectation Maximization

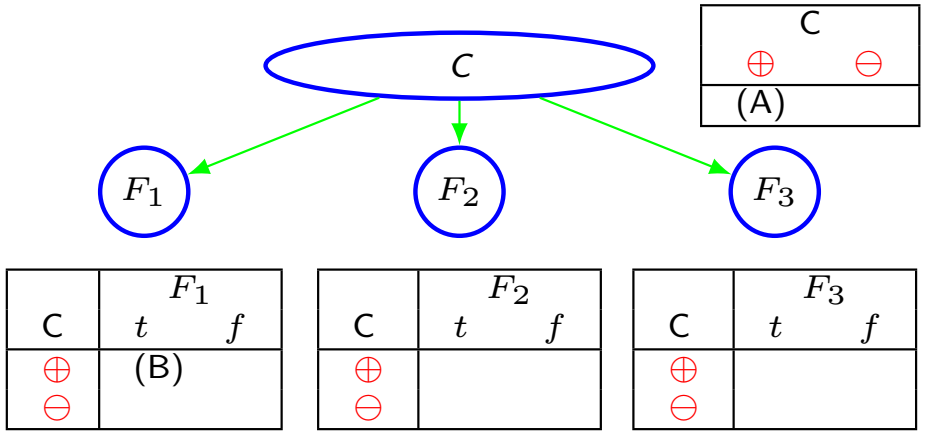
We need to estimate $P(C)$, $P(C|F_1)$, $P(C|F_2)$, $P(C|F_3)$
How to do that?

The Expectation Maximization (EM) Algorithm

- Create a model with the right number of clusters (K)
- Initialize probabilities of the model ($P(C)$, $P(F_1|C)$, $P(F_2|C)$, and $P(F_3|C)$) at random
- Until convergence:
  1. Expectation: for each example in the training set, compute estimated class probabilities based on the current table probabilities
  2. Maximization: Re-compute the current table probabilities based on the estimated class probabilities

Maximization Step

Re-compute the current table probabilities based on the estimated class probabilities

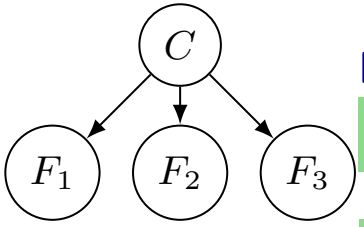


Question!

Compute the following:

(A): $P(C = \oplus)$ **(B):** $P(C = \oplus | F_1 = t)$

EM for soft clustering: an example



Data:

F_1	F_2	F_3
t	t	t
t	f	t
t	f	f
f	f	t

Expectation

Probability tables:

$P_0(C) = (0.6, 0.4)$ $P_1(C) = (0.63, 0.37)$ $P_2(C) = (0.62, 0.38)$

$P_0(F_1\|C)$	$C=1$	$C=2$
$F_1 = t$	0.6	0.4
$F_1 = f$	0.4	0.6

Count table $A(F_1, F_2, F_3, C)$**:**

F_1	F_2	F_3	$P(C\|F_1, F_2, F_3)$
t	t	t	$(0.84, 0.16)(0.88, 0.12)$
t	f	t	$(0.69, 0.31)(0.66, 0.34)$
t	f	f	$(0.5, 0.5)(0.48, 0.52)$
f	f	t	$(0.5, 0.5)(0.47, 0.53)$

$\ldots\|C)$	
$= 1$	C
81	0
19	0

Also $P_{01}(F_2|C)$ and $P_{01}(F_3|C)$

Maximization

Expectation

- Fractional counts are being calculated by probability updating.

Maximization

$$P_1(C) = \frac{1}{4} \sum_{F_1, F_2, F_3} A(F_1, F_2, F_3, C) = \frac{1}{4}(0.84 + 0.69 + 0.5 + 0.5, 0.16 + 0.31 + 0.5 + 0.5)$$

$$= (0.63, 0.37)$$

Maximization

The EM algorithm: Properties

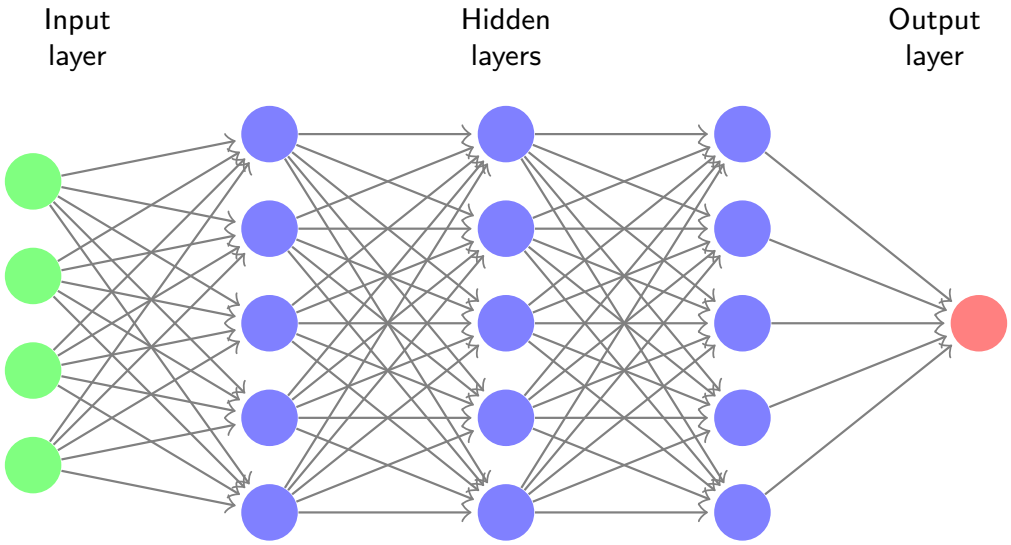
- **Convergence** The sequence of probability estimates generated by the EM algorithm converges to a local maximum (in rare cases: a saddle point) of the marginal likelihood given the data.
- **Non-optimality**: May converge to a local maximum (non-necessarily the global maximum)
- To find better local maxima: run EM several times with different starting points.

Notes

- Any permutation of the cluster labels of a local maximum will also be a local maximum.
- Clustering an existing or new instance $\mathbf{x}$ amounts to calculating $P(C|\mathbf{x})$.

Neural Networks for Unsupervised Learning

Can we use Neural Networks for Unsupervised Learning?

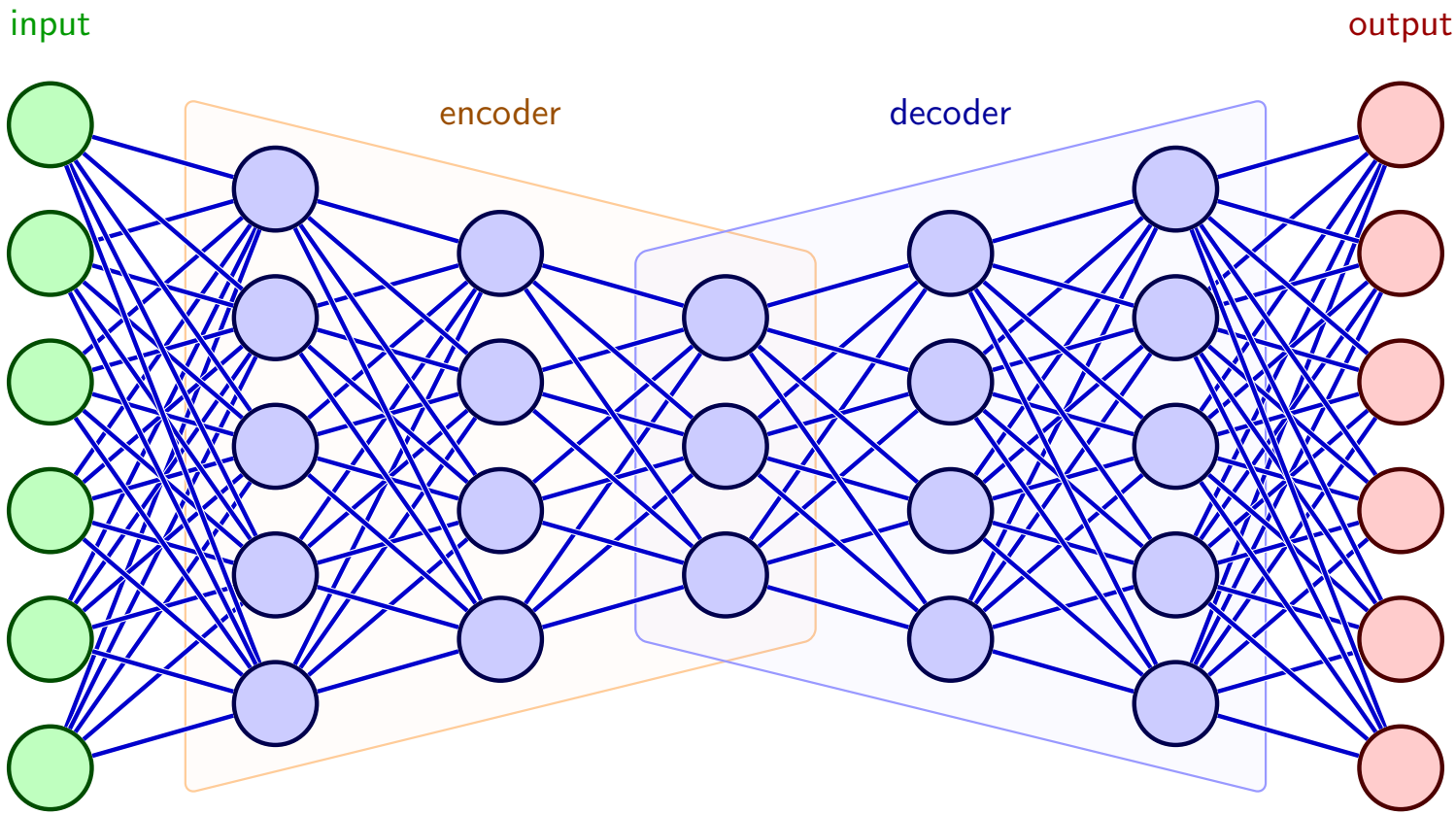


Problem:

The whole concept of training is to minimize the error with gradient descent. How do we compute the error if we do not have any target label?

Autoencoder

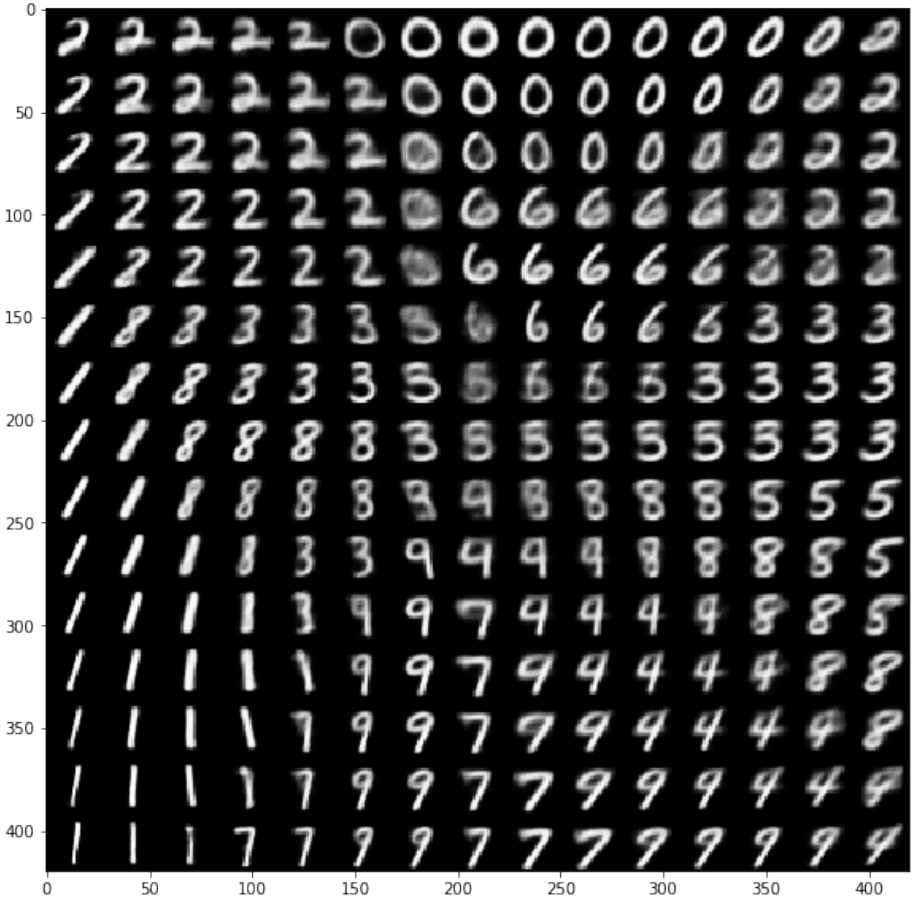
- Output = Input: Train the network to predict the input from the input
- Hidden layers have less neurons so the data needs to be compressed and then uncompressed
- After learning all weights, we can separate the encoder and the decoder



Embeddings

Aim: extract good representation from data without the need for any manual labels

Embeddings: compressed representation of the original data



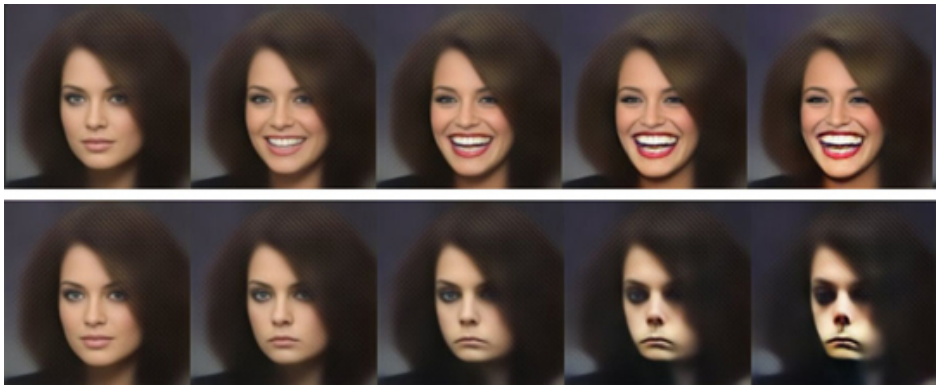
(example where images are reduced to two values (x, y), taken from `https://github.com/lyeoni/keras-mnist-VAE/`)

Why is this possible?

Using Decoders

We can use decoders to obtain new examples that are not in our original dataset:

1. Train Autoencoder on a large dataset (e.g. of face images)
2. Identify which node in the latent representation correspond to smile (possibly forcing this by altering the training)
3. Get a new image, and obtain its latent representation by using the encoder
4. Modify the value of the value corresponding to "smile"
5. Use the decoder to obtain a new image



(image taken from
`https://gaussian37.github.io/deep-learning-chollet-8-4/`)

Semi-supervised Learning

We have seen the two extremes:

- Supervised Learning: All our dataset is labelled
- Unsupervised Learning: We do not have any labels

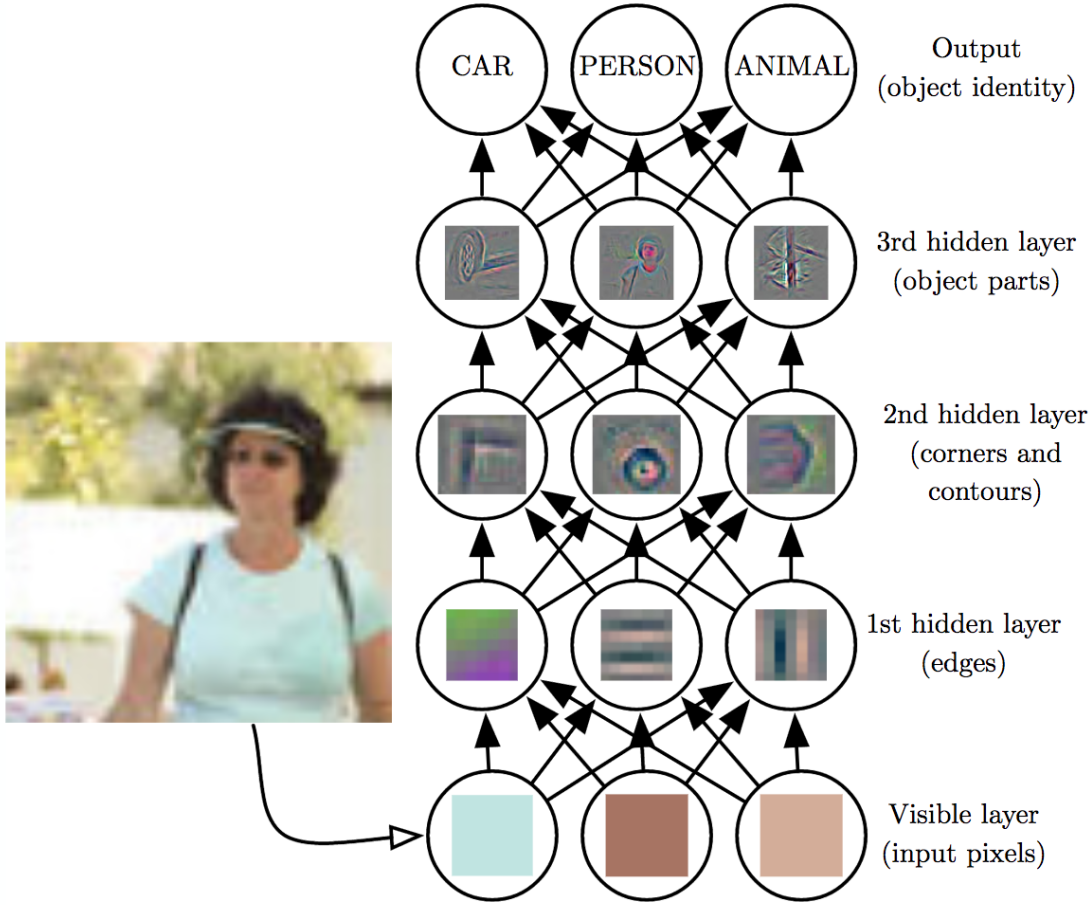
Semi-supervised Learning: We have only some examples labelled.

- Very realistic scenario: labelling data is expensive so we want to do it only for few examples
- Example: We have all images on the internet as our dataset. We label some of them (e.g. "like" and "dislike", or "dogs" and "cats"). In supervised learning, unlabelled images cannot be used. In semi-supervised learning we learn also from unlabelled data

Autoencoders for Semi-supervised Learning

1. Train Autoencoder on a large dataset (e.g. of face images)
2. Get the (possibly few) labelled images
3. Train a new "decoder"

$\rightarrow$ The "compressed" representation obtained by the encoder makes much easier to learn the classifier



Summary

- **Unsupervised Learning** techniques learn from unlabelled examples.
- **Clustering** techniques divide the data into multiple clusters.
- **K-means** is a clustering technique, where each cluster is specified by its centroid. It alternates between setting the centroid and re-classifying the data.
- **Soft clustering** assigns each example a probability of belonging to each of the clusters.
- **Expectation Maximization (EM)** is a soft clustering algorithm, which can be used in combination with a Naive Bayes model.
- **Autoencoders** can be used to obtain a latent representation of the data.

Reading

- *Chapter 10.2 Unsupervised Learning* from the book "Artificial Intelligence:Foundations of Computational Agents (2nd edition)
- Extra Reading: To go much further, you can read the Lecture Notes of the Stanford Course in Machine Learning. In this lecture we cover