7 Features and clustering

7.1 Features

In macine learning, where we want learn from patterns and relations in data, it is obviously very important which data we base our models on. If the features we have available are not very informative about the problems we are addressing, we do not have much chance of solving them. "Garbage in, garbage out" as the proverb says. Thus, it is of great importance that we carefully consider which features to base our models on. If the features we have available are not enough, we should consider gathering more data.

Once we have decided which features to base our analysis on, we need to consider if we need to preprocess the features. Here we can distinguish between the following tasks:

Feature selection If we have a large number of potential features that we can consider, we must choose which to include in our model.

Feature transformation We must decide how to present the features to our learning algorithm. For example, we might consider scaling or normalizing the features, converting between numerical and categorical representations etc.

Feature synthesis We can also consider if it is relevant to create new features by combining some of our existing features. For example, if we have *length* and *width* as features, we might include their product as a new *area* feature.

Feature extraction Some features might be very complex such an image or an audio waveform. For these types of inputs

it could be relevant to extract a small number of features to characterize the raw data more compactly.

It is important to use our background knowledge about the problem to choose features that are likely to be informative. We can also carry out quantitative evaluation of different features or combinations of features to determine which features to include. Some machine learning methods can be somewhat sensitive and perform poorly if we include bad (noisy, uninformative) features, whereas other methods are more robust and learn to ignore the bad features: Thus, how we preprocess the features will also depend on which subseqeuent analysis we plan to conduct. The primary reason to consider various feature transformations is to make the learning problem easier: To learn faster and generalize better.

7.1.1 Feature scaling

A typical way to preprocess numerical features is to scale them so that their range is on a comparable scale.

Min-max normalization A simple way to scale a numerical feature is to map it onto the range [0,1]: If x is our original feature, we can create a new normalized feature x' by subtracting the minimum and dividing by the range,

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

Here, the maximum and minimum is taken over all observed values in the training set. This ensures that all observations in the training set will be in the range [0,1].

Standardization Another common approach is to rescale numerical features to have zero mean and unit standard deviation by subtracting the mean and dividing by the standard deviation.

$$x' = \frac{x - \mu_x}{\sigma_x}.$$

As before, the mean and standard deviation are typically computed on the training set.

Example 7.1 Standardized truck features

Say we have a data set that consists of information about different trucks: The features we have available are the weight of the truck in kilograms and the length of the truck in meters. We have the following data about five trucks:

Weight [kg] (w)	4906	5091	5192	5593	4775
Length [m] (ℓ)	5.0	4.2	7.9	1.6	5.9

We decide to standardize the features to put them on a comparable scale. For the weight feature we compute the mean and standard deviation

$$\mu_w = \frac{1}{5}\sum_{n=1}^{5} w_n \approx 5111$$

$$\sigma_w = \sqrt{\frac{1}{5}\sum_{n=1}^{5}(w_n - \mu_w)^2} \approx 281$$

Based on this we can compute the standardized weights as

$$w' = \frac{w - 5111}{281}.$$

Similarly for the length features, we subtract the mean and divide by the standard deviation. This gives us the following standardized features

Std. weight (w')	-0.732	-0.073	0.287	1.715	-1.198
Std. length (ℓ')	0.039	-0.348	1.441	-1.705	0.474

7.1.2 *Numerical and categorical features*

It can be important to distinguish between *numerical* and *categorical* features.

Numerical features Features that are numerical correspond to counts or measurements. For example, the number of weight or number of pages in a book are numerical features. Numerical data is also often referred to as *quantitative*.

Categorical features Other features represent a characteristic which is absent or present, and cannot be counted or measured. For example, whether a book is a novel, a biography, or a comic book is a categorical feature. Categorical data are also often referred to as *qualitative*.

Often categorical features are represented by whole numbers, for example novel=0, biography=1, comic=2, etc. It is important that we do not treat such features as numerical: The numbers used to represent categorical feature have no meaning whatsoever. When we have categorical features, we must be careful not to use them in a way that is not appropriate, as machine learning methods implicitly assume that the features are numerical. To avoid this situation, it can often be of practical use to transform a categorical feature into several new binary features which correspond to each of the possible categories. This is known as *one-hot-coding* or as creating a set of *dummy features*.

Example 7.2 One-hot-coding the book-type

Consider a feature *book-type* which can take the values 0, 1, or 2, which are interpreted as:

Book-type	Description
0	Novel
1	Biography
2	Comic

and let us say we have the following data set:

Title	Book-type
Gone with the wind	0
Long walk to freedom	1
Astonishing X-Men	2
Amazing Spider-Man	2
The catcher in the rye	0

We can then transform the book-type feature into three new binary dummy features, which indicate the type of book.

Title	Novel	Biography	Comic
Gone with the wind	1	0	0
Long walk to freedom	0	1	0
Astonishing X-Men	0	0	1
Amazing Spider-Man	0	0	1
The catcher in the rye	1	0	0

7.2 Clustering

In cluster analysis, the task is to divide the observations into a number of clusters in such a way that similar observations go in the same group. K-means clustering is a particular algorithm that can give a fast but approximate solution to the cluster analysis problem under certain assumptions which we will go through here.

7.2.1 Centroid-based methods

In the following we will assume that the data we have available is a set of N numerical feature vectors $\mathcal{D}_{\text{u}} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$. We assume that we know the number of clusters K in advance, and that that each cluster can be described by a single representative vector that resides in the same space as the features. These vectors are referred to as cluster centroids and we denote them by $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_K$. The clustering problem consists of assigning each observation to a cluster while at the same time determining the optimal values for the centroids.

Let c_i denote the cluster to which the ith observation is assigned (for example $c_3 = 5$ if observation number 3 is assigned to cluster number 5.) To ensure that each observation is close to its cluster centroid, we can use the squared distance between the observation and its corresponding cluster centroid as a loss function.

$$L = \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{z}_{c_i}\|^2.$$

If the observations and centroids are scalar, this would simply be the squared difference. In the case where observations and centroids are vectors, we use the squared Euclidean distance.

Now, the optimization problem consists of determining the cluster centroids as well as the cluster assignments. This is not a simple optimization problem, because we need to optimize two

things that are strongly dependent: The cluster assignments and the cluster centroids.

7.2.2 *The K-means algorithm*

The K-means algorithm is a fairly simple procedure that aims at finding optimal values in the centroid-based clustering problem. It is based on the idea of alternating between estimating the optimal cluster centroids for a given cluster assignment, and estimating the optimal cluster assignment for a given set of centroids. By going back and forth between estimating the two, after a while the algorithm will converge on a good (albeit not necessarily optimal) solution.

Definition 7.1 K-means clustering

K-means clustering alternates between the following two steps until convergence:

1. Estimate optimal centroids, based on current cluster assignment.
2. Estimate optimal cluster assignment, based on current centroids.

To get the algorithm going, we can start with a random cluster assignment or start with a random set of centroids.

First, we assume that the cluster assignments c_i are known and fixed. Now the problem of estimating the optimal centroids not too difficult: We can simply differentiate the loss with respect to the jth centroid and set it equal to zero to solve for each centroid. In the sum over the N data points in the loss, only some of the terms include the jth centroid $\boldsymbol{z}_j$. It can be useful to define the number of observations assigned to the jth cluster as N_j and define the set of observations in cluster j as $\{i : c_i = j\}$ (the set of all indices i for which the cluster assignment c_i is equal to j). With this notation, we can write sum over the N_j observations in cluster j as a sum over $\{i : c_i = j\}$. Thus, we get the following optimal cluster centroids

$$\frac{\partial L}{\partial \boldsymbol{z}_j} = -2 \sum_{i:c_i=j} (\boldsymbol{x}_i - \boldsymbol{z}_j) = 0 \Rightarrow \boldsymbol{z}_j = \frac{1}{N_j} \sum_{i:c_i=j} \boldsymbol{x}_i. \tag{7.1}$$

This means, that the least squares optimal centroid for a cluster is simply the average of the observations in the cluster.

Example 7.3 Computing centroids

Assume we have $N = 5$ observations

Observation (i)	1	2	3	4	5
Features ($\boldsymbol{x}_i$)	$\begin{bmatrix} 4 \\ 2 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$	$\begin{bmatrix} 2 \\ 4 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 5 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 4 \end{bmatrix}$
Cluster (c_i)	1	2	1	2	2

We can now compute the least squares optimal cluster centroids as

$$\boldsymbol{z}_1 = \frac{1}{2}\left(\begin{bmatrix} 4 \\ 2 \end{bmatrix} + \begin{bmatrix} 2 \\ 4 \end{bmatrix}\right) = \begin{bmatrix} 3 \\ 3 \end{bmatrix},$$
$$\boldsymbol{z}_2 = \frac{1}{3}\left(\begin{bmatrix} 1 \\ 3 \end{bmatrix} + \begin{bmatrix} 1 \\ 5 \end{bmatrix} + \begin{bmatrix} 1 \\ 4 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ 4 \end{bmatrix}.$$

Next, we assume that the cluster centroids $\boldsymbol{z}_i$ are known and fixed. Now, the problem of estimating the optimal cluster assignment is easy: Since the loss function measures the sum of squared distances between the observations and the cluster centroids, the optimal solution is clearly to assign each observation to the closest centroid. Thus, the optimal cluster assignment can be found for each observation by computing the distance to each centroid and finding the minimum distance:

$$c_i = \arg\min_j \|\boldsymbol{x}_i - \boldsymbol{c}_j\|^2.$$

Example 7.4 K-means

Consider again the data set in Example 7.3. Now, we would like to use the K-means algorithm to cluster the five data observations. Say we initialize the algorithm with the following cluster assignment:

i	1	2	3	4	5
c_i	1	1	2	2	2

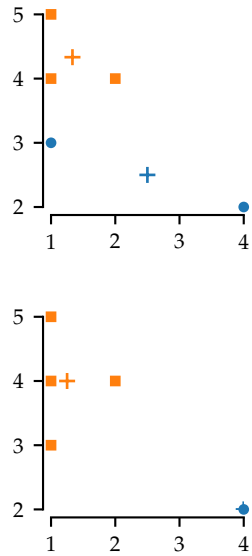

Figure 7.1: Example of K-means clustering of five observations (two-dimensional vectors). Cluster centroids are shown as + symbols and cluster assignment is indicated by color. The top panel shows the initialization, and the bottom panel shows the final result. In this example, the algorithm converges after just one update.

We can then compute the optimal cluster centroids as

$$z_1 = \frac{1}{2}\left(\begin{bmatrix} 4 \\ 2 \end{bmatrix} + \begin{bmatrix} 1 \\ 3 \end{bmatrix}\right) = \begin{bmatrix} 2.5 \\ 2.5 \end{bmatrix},$$
$$z_2 = \frac{1}{3}\left(\begin{bmatrix} 2 \\ 4 \end{bmatrix} + \begin{bmatrix} 1 \\ 5 \end{bmatrix} + \begin{bmatrix} 1 \\ 4 \end{bmatrix}\right) \approx \begin{bmatrix} 1.33 \\ 4.33 \end{bmatrix}.$$

Now, we can compute the distances from each of the five observations to the two centroids

i	1	2	3	4	5
$\|x_i - z_1\|^2$	2.5	2.5	2.5	8.5	4.5
$\|x_i - z_2\|^2$	12.6	1.89	0.56	0.56	0.22

Based on these distances, we see that the new optimal cluster assignment is given by

i	1	2	3	4	5
c_i	1	2	2	2	2

We now update the cluster centroids:

$$z_1 = \begin{bmatrix} 4 \\ 2 \end{bmatrix},$$
$$z_2 = \frac{1}{4}\left(\begin{bmatrix} 1 \\ 3 \end{bmatrix} + \begin{bmatrix} 2 \\ 4 \end{bmatrix} + \begin{bmatrix} 1 \\ 5 \end{bmatrix} + \begin{bmatrix} 1 \\ 4 \end{bmatrix}\right) = \begin{bmatrix} 1.25 \\ 4 \end{bmatrix}.$$

Again we compute the distances from each of the five observations to the two centroids

i	1	2	3	4	5
$\|x_i - z_1\|^2$	0	10	8	18	13
$\|x_i - z_2\|^2$	11.56	1.06	0.56	1.06	0.06

Based on these distances, the optimal cluster assignment does not change and the algorithm has converged. The steps in the algorithm are illustrated in Figure 7.1.

Problems

1. We have a numerical feature x that takes values in the range -17 to 25. Using min-max normalization, how would a value of $x = 0$ be represented?

2. We have 10 observations of a numerical feature x that takes the following values $1, 2, 3, 4, 5, 6, 7, 8, 9, 10$. If we standardize this feature, how would the value $x = 5$ be represented?

3. In Example 7.2 we created three dummy variables *Novel*, *Biography*, and *Comic* to represent the categorical feature *Book-type*. If we decide to throw away the feature *Comic* (so we only have *Novel* and *Biography* left) would we loose any information?

4. What is the value of the loss function in Example 7.3?

Solutions

1. $x' = 0.4048$.
2. $x' \approx -0.174$
3. No, because the feature *Comic* can be inferred from the two other features (if *Novel* and *Biography* are both 0, then *Comic* must be 1). In fact, we can always represent a categorical variable with K categories using $K - 1$ dummy features.
4. $L = 6$