Contents

Chapter 1

Linear Regression

1.1 Fitting a robust regression model using RANSAC

Algorithm 1: RANdom SAmple Consensus (RANSAC) algorithm

```
begin
    repeat
        Select a random number of examples to be inliers and fit the model;
        Test all other data points against the fitted model and add those points
          that fall within a user-given tolerance to the inliers;
        Refit the model using all inliers;
        Estimate the error of the fitted model versus the inliers;
    until the performance meets a certain user-defined threshold or if a fixed
      number of iterations was reached;
```

Chapter 2

Clustering

The goal of clustering is to find a natural grouping in data so that items in the same cluster are more similar to each other than to those from different clusters.

2.1 Prototype-based clustering

Prototype-based clustering means that each cluster is represented by a prototype, which is usually either the **centroid** (average) of similar points with continuous features, or the **medoid** (the most representative or the point that minimizes the distance to all other points that belong to a particular cluster) in the case of categorical features.

2.1.1 k-means clustering

K-means algorithms partition cases in a dataset into *k* clusters, where *k* is an integer defined by us. The clusters returned by k-means algorithms tend to be n-dimensionally spherical (where n is the number of dimensions of the feature space). This means the clusters tend to form a circle in two dimensions, a sphere in three dimensions, and a hypersphere in more than three dimensions. K-means clusters also tend to have a similar diameter. These are traits that may not be true of the underlying structure in the data.

There are a number of k-means algorithms, but some commonly used ones are as follows:

- Lloyd algorithm (also called Lloyd-Forgy algorithm)
- MacQueen algorithm
- Hartigan-Wong algorithm

Algorithm 2: Lloyd's algorithm

```
1 begin
2     Select k;
3     Randomly initialize k centers in the feature space;
4     repeat
5         foreach case(sample) do
6             Calculate the distance between the case and each center;
7             Assign the case to the cluster of the nearest centroid;
8             Place each center at the mean of the cases assigned to its cluster;
9     until no cases change clusters or a maximum number of iterations is
        reached;
```

Lloyd's algorithm

MacQueen's algorithm

MacQueen's algorithm is extremely similar to Lloyd's algorithm, varying just subtly in when the centroids get updated. Lloyd's algorithm is called a *batch* or *offline* algorithm, meaning it updates the centroids together at the end of an iteration. MacQueen's algorithm, on the other hand, updates the centroids each time a case changes clusters and once the algorithm has passed through all the cases in the data.

Algorithm 3: MacQueen's algorithm

```
1  begin
2      Select k;
3      Randomly initialize k centers in the feature space;
4      Assign each case to the cluster of its nearest center;
5      Place each center at the mean of the cases assigned to its cluster;
6      repeat
7          foreach case(sample) do
8              Calculate the distance between the case and each centroid;
9              Assign the case to the cluster of the nearest centroid;
10             if the case changed clusters then
11                 update the position of the new and old centroids.
12     until no cases change clusters;
```

Hartigan-Wong algorithm

The third k-means algorithm is a little different from the Lloyd and MacQueen algorithms. The Hartigan-Wong algorithm starts by initializing k random centers and assigning each case to the cluster of its nearest center, just as we saw in the other two algorithms. Here's the different bit: for each case in the dataset, the algorithm calculates the sum of squared error of that case's current cluster if that case was removed, and the sum of squared error of each of the other clusters if that case was included in those clusters.

Algorithm 4: Hartigan-Wong algorithm

```
1  begin
2      Select $k$;
3      Randomly initialize $k$ centers in the feature space;
4      Assign each case to the cluster of its nearest center;
5      Place each center at the mean of the cases assigned to its cluster;
6      repeat
7          foreach case(sample) do
8              Calculate the sum of squared error for its cluster, omitting the case
                 under consideration;
9              Calculate the sum of squared error for the other clusters, as if that
                 case were included.;
10             Assign the case to the cluster with the smallest sum of squared
                 error;
11             if the case changed clusters then
12                 update the position of the new and old centroids.
13     until no cases change clusters;
```

The Hartigan-Wong algorithm tends to find a better clustering structure than either the Lloyd or MacQueen algorithms, although we are always subject to the "no free lunch" theorem. Hartigan-Wong is also more computationally expensive than the other two algorithms, so it will be considerably slower for large datasets.

2.1.2 k-means++

2.1.3 Hard versus soft clustering

Hard clustering describes a family of algorithms where each example in a dataset is assigned to exactly one cluster, as in the **??** and algorithm 5. In contrast, algorithms for **soft clustering** (sometimes also called **fuzzy clustering**) assign an example to one or more clusters. A popular example of soft clustering is the **fuzzy C-means (FCM)** algorithm (also called **soft k-means** or **fuzzy k-means**).

2.1.4 Fuzzy C-means

The objective function of FCM—we abbreviate it as J_m:

$$J_m = \sum_{i=1}^{n} \sum_{j=1}^{k} w^{(i,j)^m} ||\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(j)}||_2^2 \tag{2.1}$$

Algorithm 5: The k-means++ algorithm

```
1 begin
2     Initialize an empty set, $M$, to store the $k$ centroids being selected;
3     Randomly choose the first centroid from the input examples and $M \leftarrow \mu^{(j)}$;
4     repeat
5         For each example, $\boldsymbol{x}^{(i)}$, that is not in $M$, find the minimum squared
            distance, $d(x^{(i)}, M)^2$, to any of the centroids in $M$;
6         To randomly select the next centroid, $\mu^{(p)}$, from a weighted
            probability distribution equal to $\frac{d(\mu^{(p)},\mathbf{M})^2}{\sum_i d(x^{(i)},\mathbf{M})^2}$;
7     until k centroids are chosen;
8     Proceed with the classic k-means algorithm;
```

Algorithm 6: The FCM algorithm

```
1 begin
2     Specify the number of k centroids and randomly assign the cluster
        memberships for each point;
3     repeat
4         Compute the cluster centroids, $\boldsymbol{\mu}^{(i)}, j \in \{1, \ldots, k\}$;
5         Update the cluster memberships for each point;
6     until the membership coefficients do not change or a user-defined
        tolerance or maximum number of iterations is reached;
```

We added an additional exponent to $w^{(i,j)}$; the exponent m, any number greater than or equal to one (typically $m = 2$), is the so-called **fuzziness coefficient** (or simply **fuzzifier**), which controls the degree of fuzziness.

The larger the value of m, the smaller the cluster membership, $w^{(i,j)}$, becomes, which leads to fuzzier clusters. The cluster membership probability itself is calculated as follows:

$$w^{(i,j)} = \left[\sum_{c=1}^{k} \left(\frac{||\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(j)}||_2}{||\boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(c)}||_2} \right)^{\frac{2}{m-1}} \right]^{-1} \tag{2.2}$$

The center, $\boldsymbol{\mu}^{(j)}$, of a cluster itself is calculated as the mean of all examples weighted by the degree to which each example belongs to that cluster ($w^{(i,j)^m}$):

$$\boldsymbol{\mu}^{(j)} = \frac{\sum_{i=1}^{n} w^{(i,j)^m} \boldsymbol{x}^{(i)}}{\sum_{i=1}^{n} w^{(i,j)^m}} \tag{2.3}$$

2.2 Agglomerative hierarchical clustering

2.3 Density-based clustering

2.3.1 Density-based spatial clustering of applications with noise

2.4 Graph-based clustering

Chapter 3

Others

3.1 Distance

3.1.1 Euclidean distance