Contents

Chapter 1

Working with Unlabeled Data – Clustering Analysis

1.1 Grouping objects by similarity using k-means

1.1.1 k-means clustering using scikit-learn

Algorithm 1: The k-means algorithm

begin
  Randomly pick k centroids from the examples as initial cluster centers;
  repeat
    Assign each example to the nearest centroid, $\mu^{(i)}, j \in \{1, \ldots, k\}$;
    Move the centroids to the center of the examples that were assigned to it;
  until *the cluster assignments do not change or a user-defined tolerance or maximum number of iterations is reached*;

A problem with k-means is that one or more clusters can be empty.

Feature scaling

When we are applying k-means to real-world data using a Euclidean distance metric, we want to make sure that the features are measured on the same scale and apply z-score standardization or min-max scaling if necessary.

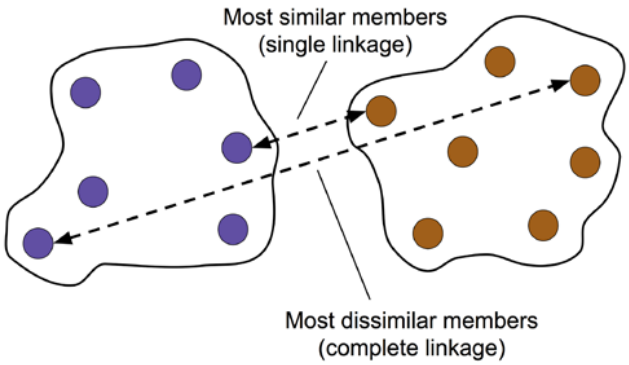

Figure 1.1: The complete linkage approach

1.2 Organizing clusters as a hierarchical tree

1.2.1 Grouping clusters in a bottom-up fashion

The two standard algorithms for agglomerative hierarchical clustering are single linkage and complete linkage. Using single linkage, we compute the distances between the most similar members for each pair of clusters and merge the two clusters for which the distance between the most similar members is the smallest. The complete linkage approach is similar to single linkage but, instead of comparing the most similar members in each pair of clusters, we compare the most dissimilar members to perform the merge. This is shown in Figure 1.1:

Alternative types of linkages

Other commonly used algorithms for agglomerative hierarchical clustering include average linkage and Ward's linkage. In average linkage, we merge the cluster pairs based on the minimum average distances between all group members in the two clusters. In Ward's linkage, the two clusters that lead to the minimum increase of the total within-cluster SSE are merged.

1.3 Locating regions of high density via DBSCAN

According to the DBSCAN algorithm, a special label is assigned to each example (data point) using the following criteria:

- A point is considered a **core point** if at least a specified number (MinPts) of neighboring points fall within the specified radius, ϵ

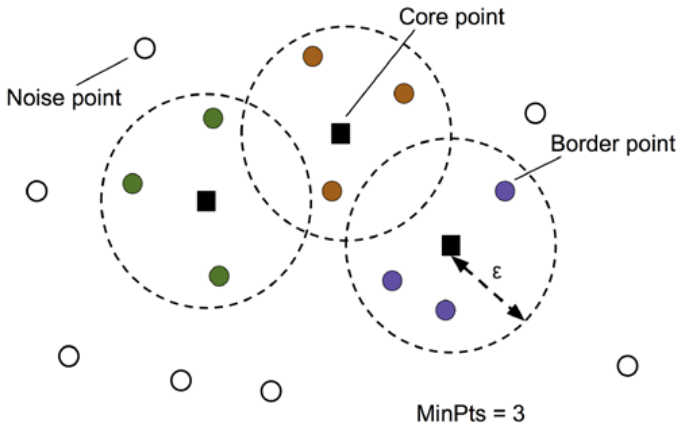

Figure 1.2: Core, noise, and border points for DBSCAN

- A **border point** is a point that has fewer neighbors than MinPts within ε , but lies within the ε radius of a core point
- All other points that are neither core nor border points are considered **noise points**

Chapter 2

Implementing a Multilayer Artificial Neural Network from Scratch

2.1 Modeling complex functions with artificial neural networks

2.1.1 Introducing the multilayer neural network architecture

Adding additional hidden layers

We can add any number of hidden layers to the MLP to create deeper network architectures. Practically, we can think of the number of layers and units in an NN as additional hyperparameters that we want to optimize for a given problem task using the cross-validation technique,

2.1.2 Activating a neural network via forward propagation

Furthermore, we can generalize this computation to all n examples in the training dataset:

$$\begin{aligned} \boldsymbol{Z}^{(h)} &= \boldsymbol{X}^{(in)}\boldsymbol{W}^{(h)T} + \boldsymbol{b}^{(h)} \\ \boldsymbol{A}^{(h)} &= \sigma(\boldsymbol{Z}^{(h)}) \end{aligned} \tag{2.1}$$

Here, $\boldsymbol{X}^{(in)}$ is now an $n \times m$ matrix, and the matrix multiplication will result in an $n \times d$ dimensional net input matrix, $\boldsymbol{Z}^{(h)}$. Finally, we apply the activation function $\sigma(\cdot)$ to each value in the net input matrix to get the $n \times d$ activation matrix in the next layer.

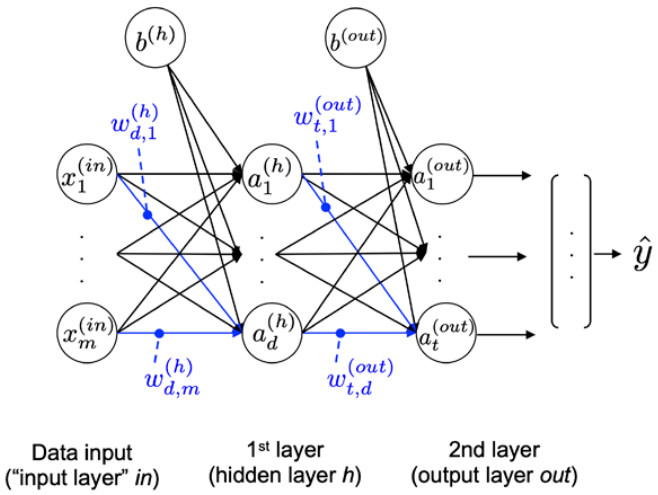

Figure 2.1: A two-layer MLP. We will use the *in* superscript for the input features, the *h* superscript for the hidden layer, and the *out* superscript for the output layer. For instance, $x_i^{(in)}$ refers to the ith input feature value, $a_i^{(h)}$ refers to the ith unit in the hidden layer, and $a_i^{(out)}$ refers to the ith unit in the output layer. Note that the b's denote the bias units. In fact, $b^{(h)}$ and $b^{(out)}$ are vectors with the number of elements being equal to the number of nodes in the layer they correspond to. For example, $b^{(h)}$ stores d bias units, where d is the number of nodes in the hidden layer. The connection between the kth unit in layer l to the jth unit in layer $l+1$ will be written as $w^{(l)}_{j,k}$.