Survey of Machine Learning models, cont'd

Chris Cornwell

April 7, 2025

Outline

Outline

Intro

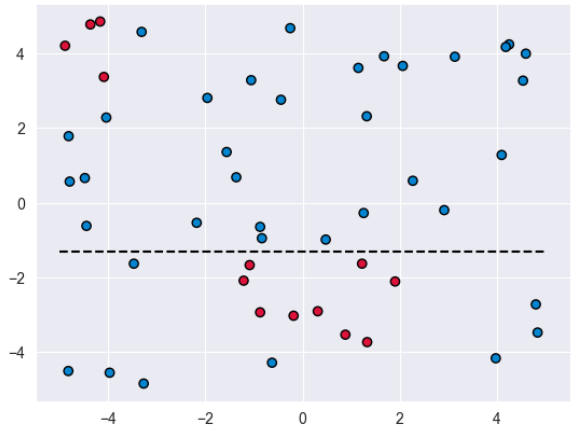
Decision trees are another example of a powerful model function for machine learning.

- Often used for classification (binary or multi-label); however, they can be used for regression tasks as well.
- Similar to neural networks which are capable of approximating any function, if the “size” of the decision tree is unrestrained then it can, in theory, produce an arbitrarily close approximation to any desired classification of points.

Construction of a decision tree

To build a decision tree, begin with a **decision stump** on $\mathbb{R}^d$.

- Let $\omega = (b, \theta, j)$ be a triple consisting of $b \in \mathbb{R}$, $\theta \in \{1, -1\}$, and j an integer with $1 \leq j \leq d$. Then, define $f_\omega(\mathbf{x}) = \text{sign}(\theta(b - \mathbf{x}_j))$, where $\text{sign}(z)$ is 1 if $z \geq 0$ and is -1 otherwise. Such functions are called decision stumps.
- Note that a decision stump is a special type of half-space model that has normal vector with a single non-zero component (i.e., $\mathbf{w} = (0, \ldots, 0, -\theta, 0, \ldots, 0)$).



Construction

- Given sample data, $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^d$, the decision stump which is the best fit may be found simply by minimizing error – that is, find the values of b, θ, and j that achieves the highest accuracy on $\mathcal{S}$.
- If there is no point $\mathbf{x}_i$ between $\{x_{i,j} = b\}$ and $\{x_{i,j} = b'\}$, then the accuracy for (b, θ, j) and (b', θ, j) are the same. So, one can consider just values of b that are an average of two "consecutive" $x_{i,j}$ (and one less than all such j^{th} coordinates, and one larger than all of them). This gives a finite set of decision stumps to check.
- A naive algorithm to determine the most accurate among this finite set of decision stumps takes on the order of dn^2 operations.
  However, there is a more efficient approach, taking only $dn \log(n)$ time.

Construction of a decision tree

A decision stump is a decision tree with "depth" 1. That is, can think of a decision tree as a collection of nested decision stumps, on smaller and smaller subsets of the data.

Begin with a decision stump on all the data; i.e., find a hyperplane $x_j = b$, where $1 \leq j \leq d$ and b is some number and each data point is on one side: for each $\mathbf{x}_i \in \mathcal{S}$, either $x_{i,j} < b$ or $x_{i,j} > b$. This partitions the data into two subsets.

How do you choose where to make a split? You could use the highest accuracy approach described for stumps, but this has disadvantages when the proportion of y_i that are positive, compared to negative, are unbalanced.

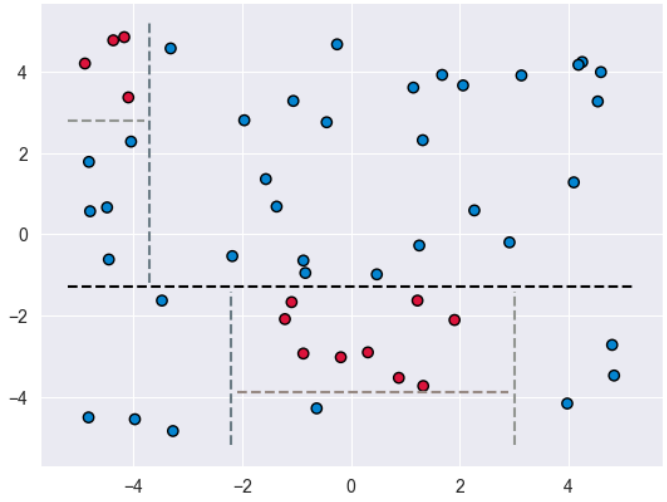
Often, something called **Information Gain** is used, which is defined via an entropy function e. That is, set $e(r) = -r\log(r) - (1-r)\log(1-r)$. Now, before making a split, set r to be the proportion of y_i that are 1 and m the number of points.

Next, define r_+ (resp. r_-) to be the proportion of points on the positive (resp. negative) side of the split that will have label 1, and let m_+ (resp. m_-) be the numbers of points on the positive (resp. negative) side. The information gain of the split is

$$e(r) - \left(\frac{m_+}{m}e(r_+) + \frac{m_-}{m}e(r_-)\right).$$

Multiple branches

The goal of the process is to recursively partition each side. On each step, one chooses the split with maximum Information Gain, at each step restricting to the two subset of data points on one side. The process ends when points in the same part have the same label, or until some predetermined depth is reached. An innermost region (where points have the same label) corresponds to a **leaf** of the tree.



Decision tree model

Start by determining a decision tree on training data, as above. Then, given test data (not yet ”seen” by the model), the decision tree model will check in which partition the test point resides. Then, it labels the test point with the label of the corresponding leaf.
Often (to avoid overfitting), you decide beforehand on the depth of the tree (the maximal number of splits to get to a leaf). Since this will result in having more than one label in some of the partitions (and possibly all), the label given to a test point in each leaf is a function of the training labels in that partition – if the goal is classification, with some categorical labels, the label that has majority; if the goal is regression, with numerical labels, the mean, or the median, could be used.

Outline

What is clustering?

Intuitively, want to group together data points into subsets, i.e., *clusters*, so that *similar* points are in the same cluster and dissimilar points are in different clusters. These data do not have labels.

- Presuming the numerical coordinates of the data have meaning, "similar" points could be those that are sufficiently close.

Goal: Given sample data $\mathcal{S} = \{\mathbf{x}_i\}_{i=1}^n$, want to determine clusters $C_1, C_2, \ldots, C_k$ (for some k), so that $C_1 \cup C_2 \cup \ldots \cup C_K = \mathcal{S}$ and $C_i \cap C_j = \varnothing$ when $i \neq j$.

An inherent difficulty: the clustering problem is ill-defined.

- For example, you might have a sequence of data points $\mathbf{x}_1, \ldots, \mathbf{x}_m$ such that $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ are similar (close enough to be called similar) for each $1 \leq i \leq m-1$, however $\mathbf{x}_1$ and $\mathbf{x}_m$ might be very dissimilar. While $\mathbf{x}_i$ and x_{i+1} should be in the same cluster, this causes the problem of putting $\mathbf{x}_1$ and $\mathbf{x}_m$ ending up in the same cluster.

Applications of clustering

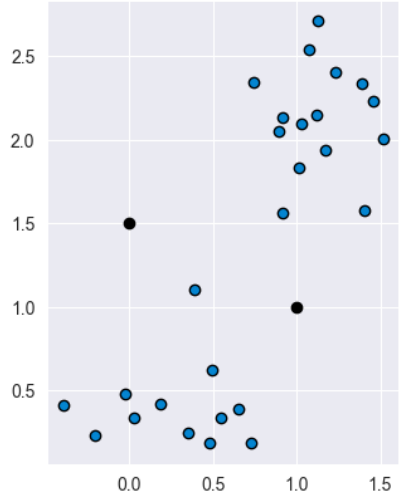
Despite its ill-defined nature, clustering is needed (sometimes in an initial processing step) for many data analysis tasks.
Some examples include:

- Market segmentation: clustering expected customers based on characteristics, either for targeted marketing or to inform product design.
- Gene expression clustering: computational biologists will cluster genes based on how their expression in different experiments.
- Astronomical data analysis: clustering stars or galaxies based on their proximity.
- Social network analysis: for example, identifying “communities” (clusters of users) based on interactions on social media.

k-means

Let $\mathcal{S} = \{\mathbf{x}_i\}_{i=1}^n$, with each $\mathbf{x}_i \in \mathbb{R}^d$ for some dimension d.

- The k-means algorithm is a common approach for clustering $\mathcal{S}$.
- You choose k, and your goal is to find k clusters, $C_1, C_2, \ldots, C_k$ so that $\mathcal{S} = C_1 \cup \ldots C_k$, where we want the sum $\sum_{j=1}^k \sum_{\mathbf{x}_i \in C_j} |\mathbf{x}_i - \mu_j|^2$ to be minimized.
- Here, μ_j is the centroid of C_j, meaning that μ_j is the point in $\mathbb{R}^d$ that minimizes, over $\mu \in \mathbb{R}^d$, the sum $\sum_{\mathbf{x} \in C_j} |\mathbf{x} - \mu|^2$.
  - Note that, in fact, $\mu_j = \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} \mathbf{x}$.



k-means Procedure

The procedure to carry out k-means clustering is the following. First, randomly initialize k centroids. Then:

1. For each $i = 1, 2, \ldots, n$, determine $1 \leq j(i) \leq k$, which is the index such that $\mu_{j(i)}$ is the closest centroid to $\mathbf{x}_i$. A point $\mathbf{x}_i$ is in C_j when $j = j(i)$.
2. Update $\mu_1, \mu_2, \ldots, \mu_k$ so that, for $1 \leq j \leq k$, the vector μ_j is the centroid of the points in C_j.
3. Iterate steps 1 and 2 until there is no $\mathbf{x}_i$ that "changes its cluster", i.e., until the assignment $i \mapsto j(i)$ that is made in 1 is the same as it was in the previous iteration.

At each iteration, if some point has changed the cluster it is in, then the value of $\sum_{j=1}^{k} \sum_{\mathbf{x}_i \in C_j} |\mathbf{x}_i - \mu_j|^2 = \sum_{i=1}^{n} |\mathbf{x}_i - \mu_{j(i)}|^2$ decreases. Hence, the algorithm terminates because, as there are finitely many points in $\mathcal{S}$, there are only a finite number of possibilities for the list $\mu_1, \mu_2, \ldots, \mu_k$.

Visual of k-means in $\mathbb{R}^2$

We show a visualization of the
procedure for k-means on data in $\mathbb{R}^2$, with $k = 2$.

Initially, we choose
two arbitrary centroids. These are drawn in black.
Then, the first step is to
go through all of our data (drawn in) and determine
which of the two centroids is closest to each point.
This assigns each point to one of the two clusters.

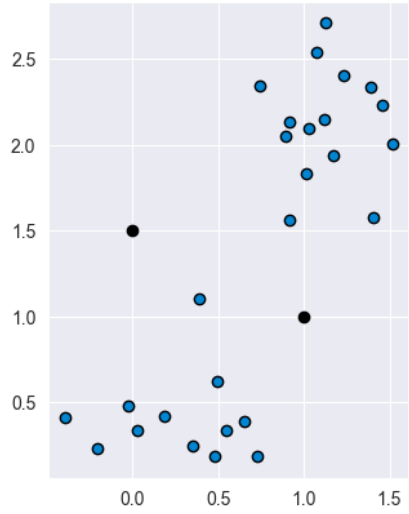


Figure: Initial centroids

Visual of k-means in $\mathbb{R}^2$

We show a visualization of the
procedure for k-means on data in $\mathbb{R}^2$, with $k = 2$.

Here, the two clusters are shown.
The points in one are drawn in light
blue and the points in the other are drawn in red.
The centroid of
each cluster is displayed, outlined in the color of its
cluster. The next step is to recompute the centroids.

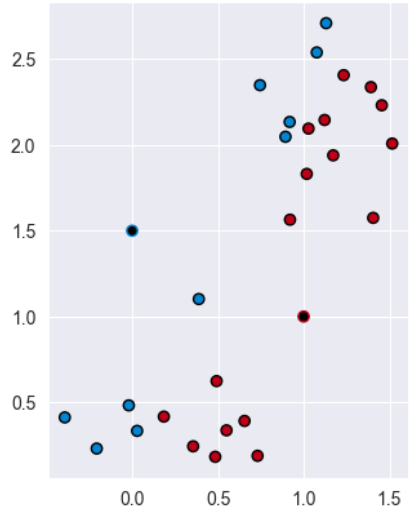


Figure: Assigned clusters

Visual of k-means in $\mathbb{R}^2$

We show a visualization of the
procedure for k-means on data in $\mathbb{R}^2$, with $k = 2$.

The two newly computed centroids are shown.
In addition, we reassign
points based on the (new) centroid that is closest.
We notice that 4 of points have been removed from the blue cluster into the red cluster; also, 3 points have been removed from red cluster into the blue.

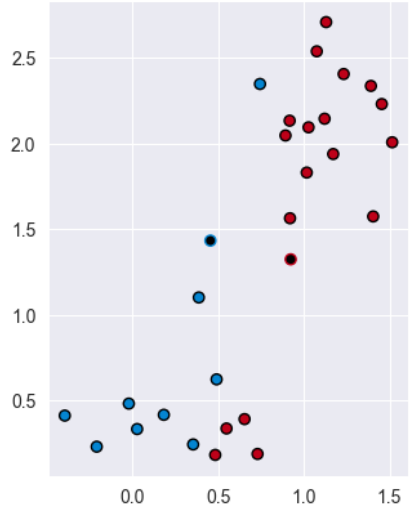


Figure: Updated clusters

Visual of k-means in $\mathbb{R}^2$

We show a visualization of the
procedure for k-means on data in $\mathbb{R}^2$, with $k = 2$.

Again, we compute new centroids and
reassign points based on the centroid that is closest.
In this step,
there is 1 point that was previously in the blue cluster
which changes into red. And 4 points that were in the
red cluster have been moved into the blue cluster.

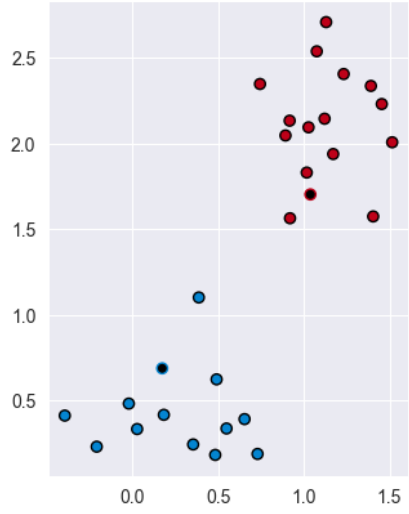


Figure: Updated clusters

Visual of k-means in $\mathbb{R}^2$

We show a visualization of the
procedure for k-means on data in $\mathbb{R}^2$, with $k = 2$.

We compute new centroids again.
This time there
is no change in the way that points are assigned
to clusters when finding the closest centroid.
Since there is no change to the clustering assignment,
the procedure terminates with these centroids.

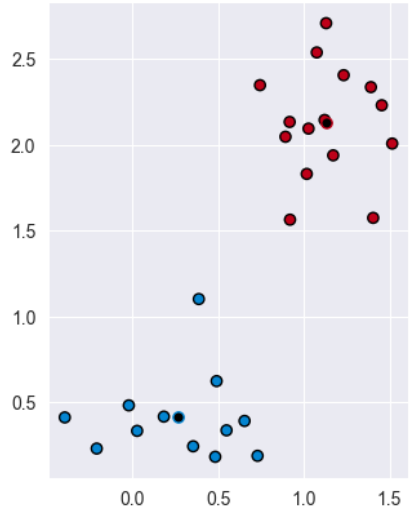


Figure: Updated centroids

k-means depends on Initial Centroids

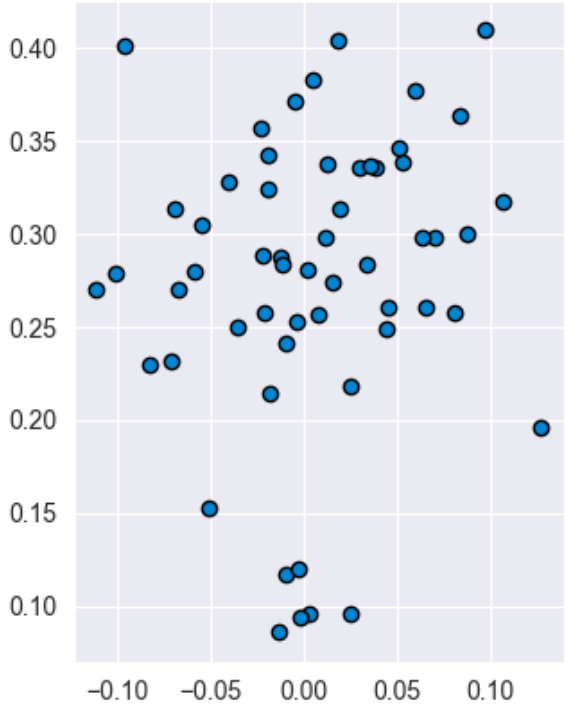


Figure: A data set to cluster

k-means depends on Initial Centroids

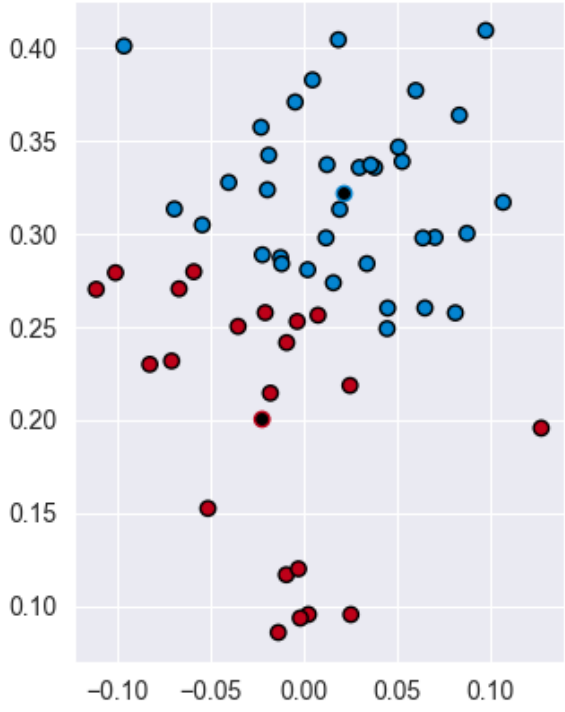


Figure: A data set to cluster

With initial centroids of $(0, 0.5)$ and $(0, 0.25)$.

k-means depends on Initial Centroids

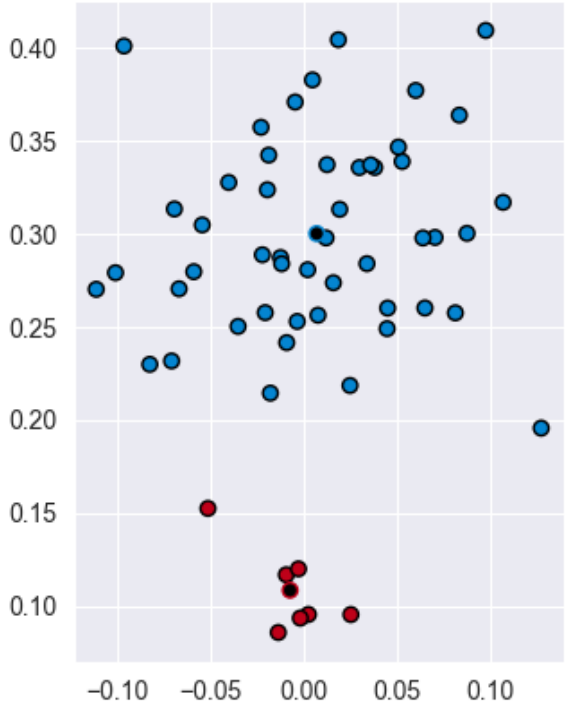


Figure: A data set to cluster

With initial centroids of $(0, 0.26)$ and $(0, 0.1)$.

Density-Based Spatial Clustering for Applications with Noise

DBSCAN clustering does not require choosing a number of clusters at the start.

- Near a given point in $\mathcal{S}$, whether it gets included in a nearby cluster is based on the *density* of points (in $\mathcal{S}$) near that given point.
- The procedure builds a directed graph (network of vertices and edges); vertices are points in $\mathcal{S}$; which edges are included is based on a choice of parameters, and takes into account the nearby density of points.
- Connectedness through edges of the graph determines when points are in the same cluster.

An example of a graph built in the DBSCAN procedure is drawn below. Points are colored according to their cluster.