Selecting the Machine Learning Model

Chris Cornwell

April 14, 2025

Outline

Overfitting

Having only concern be to minimize empirical loss

Clustering

Outline

Overfitting

Having only concern be to minimize empirical loss

Clustering

Intro

Given sample data $\mathcal{S}$, consisting of greyscale images of handwritten digits $0, 1, \ldots, 9$, say we want a decision tree model to predict the correct digit, given an image.
Let the images be p pixels by p pixels. For an image, we might take the p^2 greyscale values (0–255) and lay them end to end to get a vector in $\mathbb{R}^{p^2}$. Then, could train a decision tree with however many splits are needed so that each leaf has points in $\mathcal{S}$ with only one label.

- Gives 100% accuracy on $\mathcal{S}$, the data used to determine the parameters.
- But, the model won't perform nearly as well on data not from $\mathcal{S}$.

Let's look at doing exactly this, using images of digits available from the Python package scikit-learn.

Decision tree model for handwritten digits

The images of digits from scikit-learn are 8 pixels by 8 pixels. With the submodule `datasets` imported from `sklearn`, the data can be loaded as follows.

```
data = datasets.load_digits()
x = data.images
y = data.target
```

To convert each 8×8 array to a vector in $\mathbb{R}^{64}$, can use the `reshape` method; after assigning `x` and `y` as above, code below would output `(1797, 64)`.

```
# make each 8x8 array into 1d array with 64 entries
x = x.reshape(len(x), -1)
x.shape
```

Randomly select 20% of the data to test model with – it won't be used to determine the decision tree.

```
n_test = int(0.2*len(x))
test_indices = np.random.choice(len(x), size=n_test, replace=False)
train_indices = np.delete(np.arange(len(x)), test_indices)
x_test, y_test = x[test_indices], y[test_indices]
x_train, y_train = x[train_indices], y[train_indices]
```

**Alternatively, you can use the function* train_test_split *from the scikit-learn submodule* model_selection.

Decision tree model for handwritten digits

With the setup from the last slide, use the training data to determine the decision tree.
There is a submodule tree within scikit-learn that we can use.

```
model = tree.DecisionTreeClassifier()
model.fit(x_train, y_train)
```

The default behavior of the `.fit()` method is that the tree has as many splits (decision stumps) as needed so that each leaf consists of points with a single label. The accuracy on x_train is, therefore, 100%.
The accuracy of the model on x_test will depend some on which points were put into x_test; however, it tends to be around only 85%.
What happened is that the *hypothesis class*, that set of functions that were possible outcomes to be the trained model, was too large (allowed too many possibilities for the resulting model).

Keeping the hypothesis class small

We discuss a (somewhat absurd) example to demonstrate the point that

small empirical loss $\not\Rightarrow$ small population loss

Allowing for *any* function as a possible model, for any training data $\mathcal{S} = \{(\mathbf{x}_i, y_i)_{i=1}^n$, with labels in $\{1, -1\}$, we set

$$f_S(\mathbf{x}) = \begin{cases} y_i, & \text{if } \mathbf{x} = \mathbf{x}_i \\ -1, & \text{otherwise.} \end{cases}$$

Regardless of $\mathcal{S}$, or the population distribution that it is a sample from, the empirical loss $\mathcal{L}_S(f_S) = \frac{\#\{i:\, y_i \neq f_S(\mathbf{x}_i)\}}{n}$ is zero.

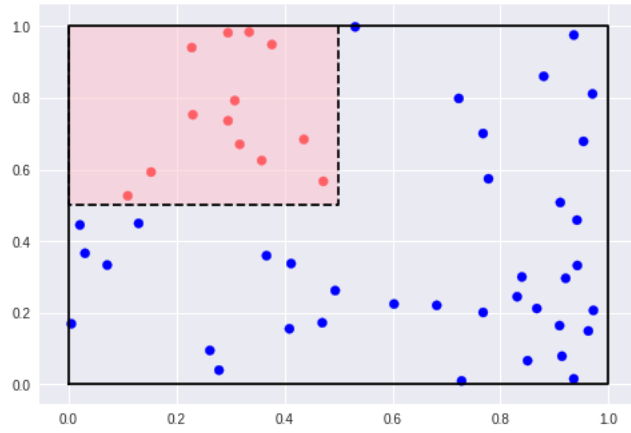


Figure: Upper-left square labeled -1

Keeping the hypothesis class small

We discuss a (somewhat absurd) example to demonstrate the point that

small empirical loss $\not\Rightarrow$ small population loss

Suppose that the population data consists of points in $[0, 1]^2$ (the square of points with both coordinates between 0 and 1), and that a point (x_1, x_2) has label -1 if and only if $0 \leq x_1 < 0.5$ and $0.5 < x_2 \leq 1$ (See Figure).

Since a sample $\mathcal{S}$ is finite, a randomly chosen point has probability 0 of being in $\mathcal{S}$. Thus, given a random $\mathbf{x} \in [0, 1]^2$, with probability $3/4$ the predicted label $f_S(\mathbf{x})$ is wrong.
The issue
with both this model, and the decision tree that had no restriction on depth, is that the **variance** *of the hypothesis class* is too large.

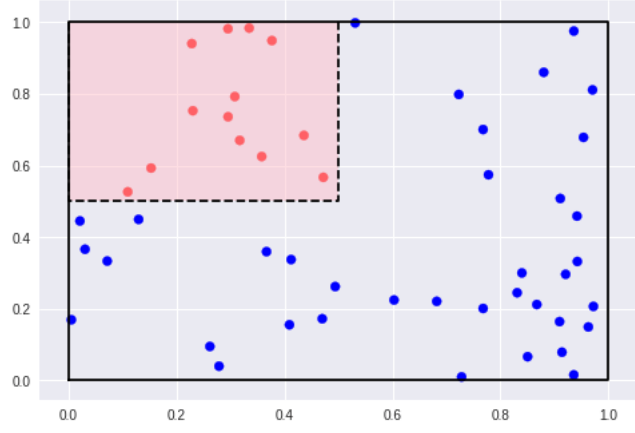


Figure: Upper-left square labeled -1

Bias-Variance Trade-off

Suppose that we have chosen a hypothesis class (a class of parameterized functions, say), and a procedure that, given a training sample $\mathcal{S}$ from the population, determines a prediction function f_S.
In order to understand the *expected* population loss, we want to consider the variance of $f_S(\mathbf{x})$, not only over the input space but over possible training sets $\mathcal{S}$. That is, over all $(\mathbf{x}, y)$ and S, the expected value $\mathbb{E}[(f_S(\mathbf{x}) - y)^2]$.
Let $\bar{h}$ be the expected prediction function, over training sets.[1] In practice, you could approximate $\bar{h}$ in the following way: for samples $S_1, \ldots, S_N$, each with the same number of points, drawn i.i.d. from the population, then for sufficiently large N,

$$\bar{h}(\mathbf{x}) \approx \frac{1}{N} \sum_{i=1}^{N} h_{S_i}(\mathbf{x}).$$

[1] This takes some advanced mathematics to carefully describe. It means there is a "measure" on sets of functions, which allows you to integrate over a set of functions to get a new function.

Outline

Overfitting

Having only concern be to minimize empirical loss

Clustering

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

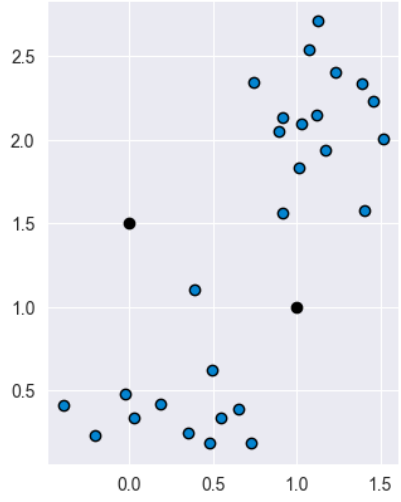
Despite its ill-defined nature, clustering is needed (sometimes in an initial processing step) for many data analysis tasks.
Some examples include:

- 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.

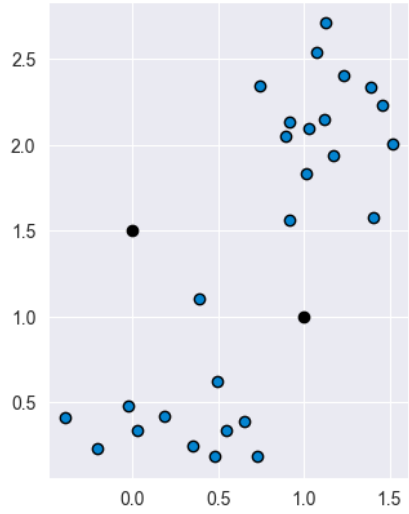


Figure: Initial centroids

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.

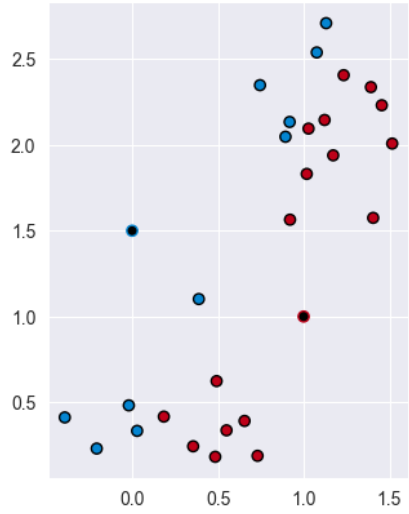


Figure: Assigned clusters

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.

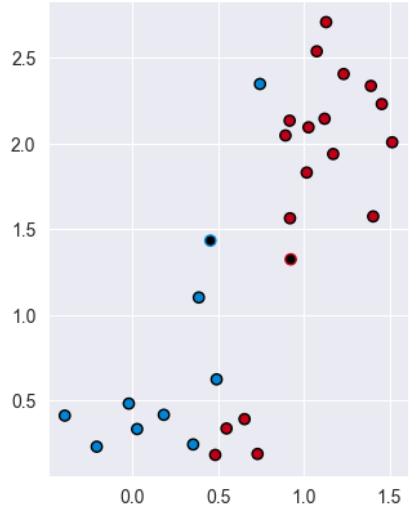


Figure: Updated clusters

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.

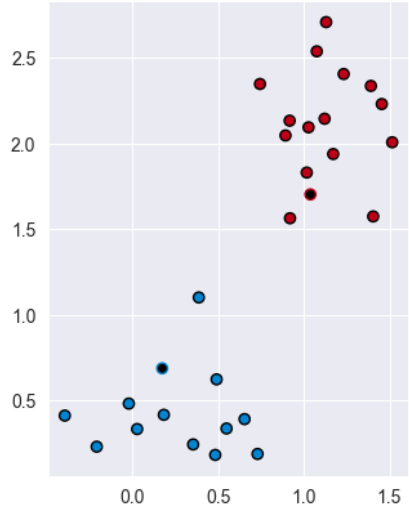


Figure: Updated clusters

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.

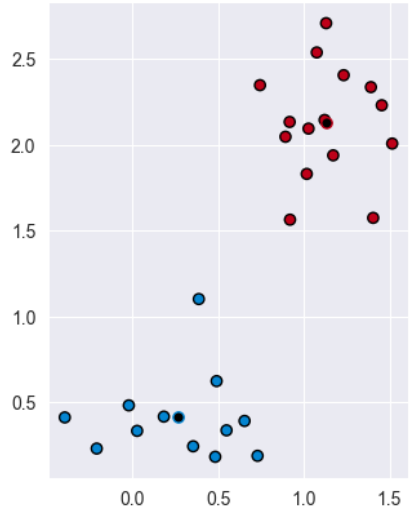


Figure: Updated centroids

k-means depends on Initial Centroids

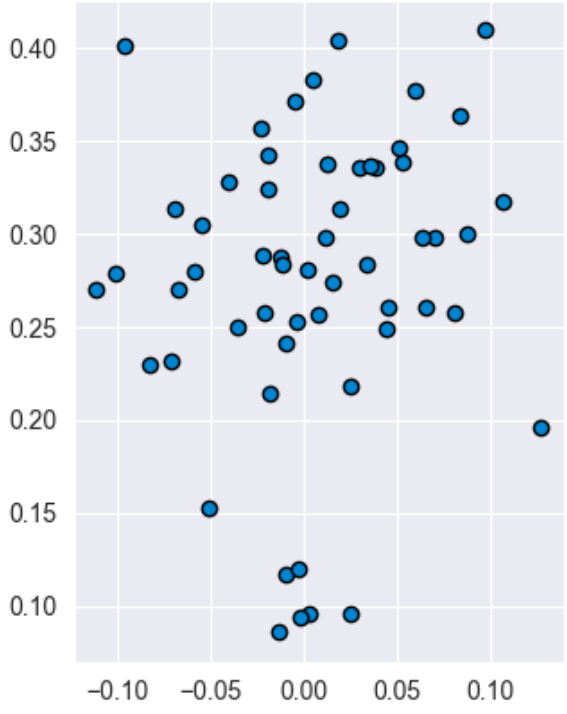


Figure: A data set to cluster

k-means depends on Initial Centroids

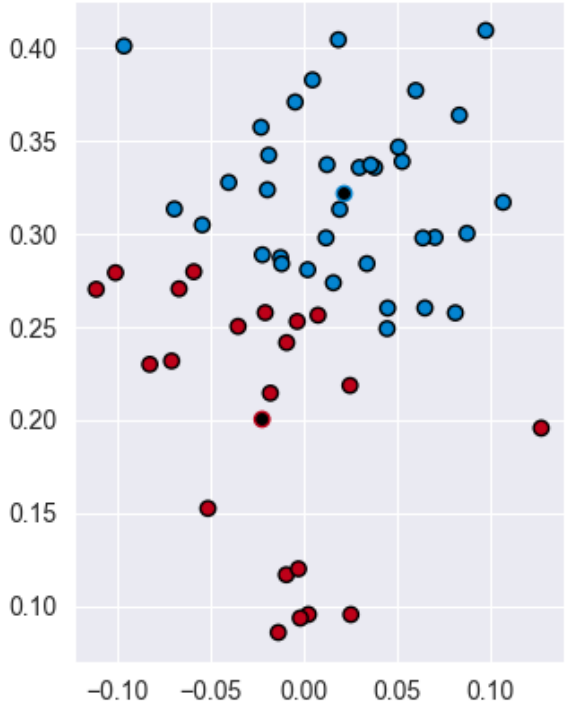


Figure: A data set to cluster

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

k-means depends on Initial Centroids

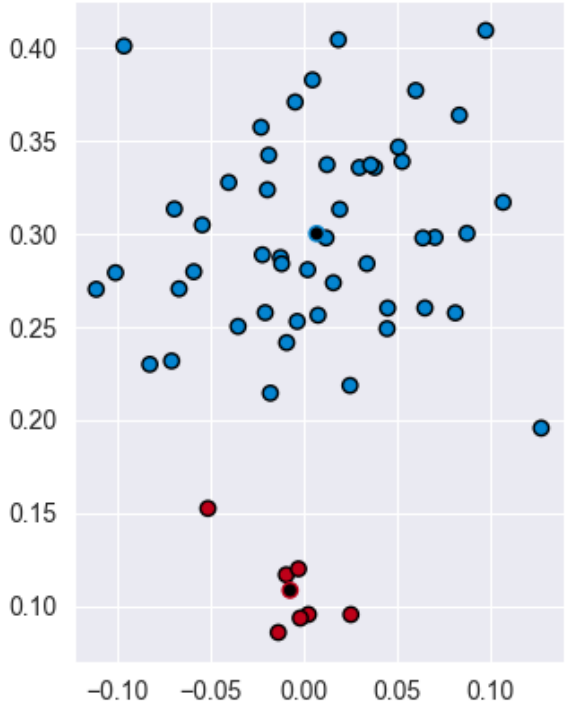


Figure: A data set to cluster

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.