MATH 462 NOTES
NEAREST NEIGHBORS AND K-MEANS CLUSTERING

ADAM M. OBERMAN

TODO: add the nearest neighbor, classify to e_i distance. Then relate to similar pairs.

$$s(x, x') \text{ if } d(x, x') < d_0$$

similarity classification.

Can also do cosine similarity, since for unit vectors $x \cdot x'$ is related by

$$(x - x')^2 = x^2 - 2xx' + x'^2 = 2(1 - xx')$$

so

$$x \cdot z = 1 - d(x, z)^2/2$$

1. Nearest Neighbors

Given a labelled dataset,

We are given a dataset (S_m) consisting of m pairs of (x_i, y_i), $i = 1, \dots, m$, of vector data, $x_i \in \mathcal{R}^d$ and labels, $y_i \in \mathcal{Y}$,

$$S_m = \{(x_1, y_1), \dots, (x_m, y_m)\} \tag{S_m}$$

Given a new example x, define $h(x)$ by

(1) find the nearest neighbor, x_j closest to x by $\min_i \|x_i - x\|^2$.
(2) Define $y(x) = y_j$ to have the same label.

1.1. **Nearest neighbor function class.** Given k vectors $w_1, \dots w_k$, written as the single array of vectors $W = (w_1, \dots, w_k)$ define the nearest neighbor (index) function by

$$h(x, W) = j^* = \operatorname*{arg\,min}_{j \in 1, \dots, k} \|x - w_j\|^2 \tag{1}$$

which returns *the index of the vector closest w_i to x.*[1] Then we can define the nearest neighbor classifier by

$$y_{nn}(x, W) = y_j, \quad j = h(x, W)$$

The function $h_W(x)$ is *piecewise constant*. The pieces are determined by the sets

$$V_j = \{x \in \mathbb{R}^d \mid h(x, W) = j\}$$

which are the Voronoi cells corresponding to the points `https://en.wikipedia.org/wiki/Voronoi_diagram`. See Figure 1.

Date: October 5, 2023.

[1]We leave the function undefined at the points where there is more than one minimizer

Given a larger dataset $S = \{x_1, \dots, x_m\}$ when we clusder the points according to the nearest neighbor, each cluster

$$C_j = \{x \in S \mid h(x, W) = j\}$$

partitionts the dataset.

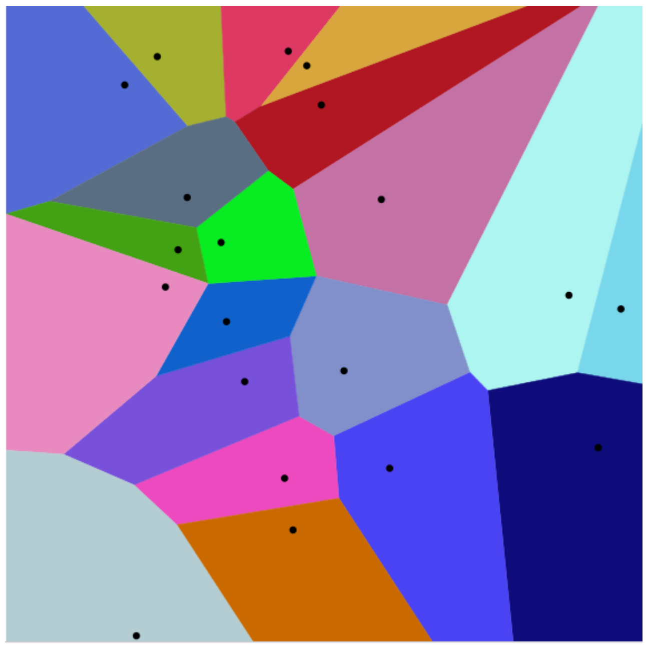

FIGURE 1. Voronoi diagram illustrating the function

Example 1.1. Consider in $d = 2$ the points $W = \{(\pm 1, \pm 1)\}$ (vertices of a square). Find the Voroinoi cells. Given the dataset $S = \{(\pm 1, \pm 2), (\pm 2, \pm 1), (\pm 2, \pm 2)\}$, (points on a larger square), partition is into clusters according to the nearest neighbors.

2. k MEANS CLUSTERING

KNN used the idea of nearest neighbors to give a label. Now use the same idea to cluster points when we have not labels.

References

- Clustering [SSBD14, Chapter 22]
- Vector Calculus [DFO20, Chapter 5]

2.1. **Introduction and problem setup.** In k-means clustering, we want to partition the data into k sets, where each partition contains similar data. In our case we consider vector data and use distance as measure of similarity.

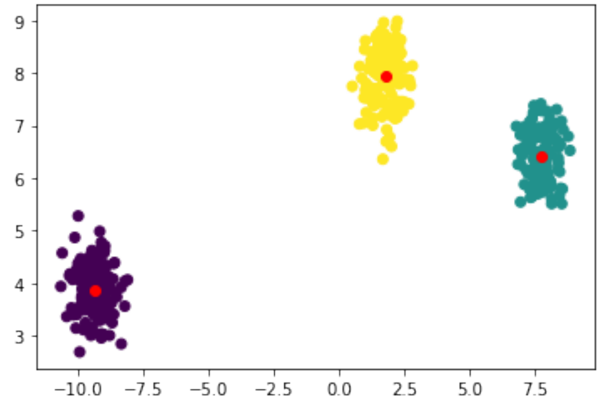

FIGURE 2. Example of a $k = 3$ cluster, with means in red.

Givens.

- a dataset, S, consisting of m vectors in d-dimensions, $\mathbb{R}^d$ (no labels),

$$S = \{x_1 \ldots, x_m\}$$

- k, the number of partitions required. (Note, if we can't see the data, we may not know the best k but can try the different values of k).

Goal: We want to partition the data into k clusters (disjoint sets),

$$S = C_1 \cup C_2 \cup \cdots \cup C_k$$

in such a way that 'similar' points belong to the same partition. Each partition C_j is represented by a vector, w_j, which is called a 'mean'.

Model: Similarity is a semantic[2] relation. It is replaced by the a mathematical relation of distance. The distance function we use is the usual Euclidean distance, $d(x, x') = \|x - x'\|$, where

$$\|x - y\|^2 = (x_1 - y_1)^2 + \cdots + (x_d - y_d)^2$$

Formally our model substitutes semantic similarity for *geometric* similarity via

$$d(x, x') \text{ small means } x \text{ and } x' \text{ are similar}$$

Method: The k-means algorithm.

Randomly choose initial means $W = (w_1, \ldots, w_k)$ which is a list of k different vectors. Can be chosen randomly (without replacement) from the vectors themselves.

- Assign each point x in dataset S^m to the cluster C_i corresponding to the closest mean w_i.
- Update the means by setting w_i to be the mean of the vectors in the cluster C_i

Repeat until convergence (meaning the w don't change).

Example 2.1. Do a one dimensional example. Let $S = \{-3, -2, -1, 2, 34\}$ Perform k means starting from $(-1, 4)$.

[2]semantic: relating to meaning

2.2. **Discussion.** Clustering is visually simple and the algorithm is also simple to implement and understand.

In what follows, we will *deliberately make things complicated*. Why? We are using this example of k-means clustering to introduce some concepts which will appear later in a more complicated context.

Analysis:

- We will analyze the problem, using simple examples to show what can happen.
- We will give a variational interpretation of the algorithm, and prove that each step of the algorithm improves the cluster, until the algorithm terminates at a fixed point.

3. Python code

The main Numpy code for k-means is given here.

```
# find the squared distance to each of the means
for j in range(k):
    # subtract the j-th mean and square each component
    Xtemp = (X - means[j,:])**2
    # sum the squares of each vector
    dist[j,:] = np.sum(Xtemp,axis=1)
# Find the cluster for each data point
labels = np.argmin(dist,axis=0)  # returns the index
# Update means
for j in range(k):
    # extract the vectors in the jth cluster
    Xjj = X[labels==j,:]
    # compute their mean
    means[j,:] = np.mean(Xjj,axis=0)
```

4. Analysis via examples

[Pictures]

5. Analysis via loss

5.1. Loss functional.

Definition 5.1 (Empirical Loss functional)**.** Given an unlabeled dataset, S, and a function $h : \mathbb{R}^d \to \mathbb{R}^d$, define the empirical loss functional to be the average squared distance from a point to its image under the transformation $h(x)$,

$$L(h, S) = \frac{1}{m} \sum_{i=1}^{m} \|h(x_i) - x_i\|^2$$

Remark 5.2. It has the form

$$L(h) = \frac{1}{|S|} \sum_{x \in S} \ell(h(x), x)$$

in the case of the loss $\ell(x_1, x_2) = \|x_1 - x_2\|^2$, The k-means loss functional comes from minimizing this over functions of the form

$$h(x, W) = w_j^* = \underset{j \in 1, \ldots, k}{\arg\min} \|x - w_j\|^2 \tag{2}$$

which returns the nearest neighbor (vector) to x in W, where $W = (w_1, \ldots, w_k)$ which leads to

$$\min_W L(h(x, W), S)$$

Now these functions partition the loss as follows.

Lemma 5.3. *Given a function h_W of the form* (2)*, we can write*

$$L(h_W) = \frac{1}{m} \sum_{j=1}^{k} \sum_{x \in C_j} \|x - w_j\|^2$$

Proof. Rewrite the loss as

$$\begin{aligned} L(h_W) &= \frac{1}{m} \sum_{j=1}^{k} \sum_{x \in C_j} \|x - h_W(x)\|^2 && \text{since } C_1, \ldots C_k \text{ is partition of } S^m \\ &= \frac{1}{m} \sum_{j=1}^{k} \sum_{x \in C_j} \|x - w_j\|^2 && \text{by definition} \end{aligned}$$

□

5.2. **Algorithm.** Here we rewrite the simple k-means algorithm described above in terms of the hypothesis.

Given an initial (e.g. random) choice of W^0, for any t, given W^t, define

$$w_j^{t+1} = \underset{w \in \mathbb{R}^d}{\arg\min} \sum_{x \in C_j} \|x - w\|^2, \qquad j = 1, \ldots, k \tag{3}$$

thus *in each cluster, the w_j^t is updates to one which improves the sum of the distances over the cluster*

Remark 5.4. In other parts of the course, we will consider algorithms which update the loss using a gradient with respect to the weights. However, in this case, gradient based algorithm are not appropriate because h_W is piecewise constant, so not really differentiable in W.

Lemma 5.5. *Suppose we update h_W according to* (3)*. Then we have*

$$L(h_W^{t+1}, S) \leq L(h_W^t, S)$$

with a strict inequality, unless $W^{t+1} = W^t$

But may not reach a global mimimum. See example with rectangle four corners.

Example 5.6. $S = \{(\pm 2, \pm 1)\}$ Can find two fixed points (by averaging corners horizontal or vertical).

But, doing a few random initializations is usually good enough to find a better one.
+ heuristic for how many k to choose.
(Homework).

References

[DFO20] Marc Peter Deisenroth, A Aldo Faisal, and Cheng Soon Ong. *Mathematics for machine learning*. Cambridge University Press, 2020.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.