TUTORIAL 0

k-means clustering

Course: Math 535 (http://www.math.wisc.edu/~roch/mmids/) - Mathematical Methods in Data Science (MMiDS)
Author: Sebastien Roch (http://www.math.wisc.edu/~roch/), Department of Mathematics, University of Wisconsin-Madison
Updated: Sep 1, 2020

Recap from the lectures

The setup We are given n vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in $\mathbb{R}^d$. Our goal is to find a good clustering (https://en.wikipedia.org/wiki/Cluster_analysis): loosely speaking, we want to partition these data points into k disjoint subsets -- or clusters -- with small pairwise distances within clusters and large pairwise distances across clusters. To make this rather imprecise problem more precise, we consider a specific objective function known as the k-means objective.

Fix a number of clusters k. A partition of $[n] = \{1, \ldots, n\}$ of size k is a collection of disjoint non-empty subsets $C_1, \ldots, C_k \subseteq [n]$ that covers all of $[n]$, that is, such that $\cup_{i=1}^k C_i = [n]$. The cost of $C_1, \ldots, C_k$ is then defined as

$$\mathcal{G}(C_1, \ldots, C_k) = \min_{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k} \sum_{i=1}^k \sum_{j \in C_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2.$$

We think of $\boldsymbol{\mu}_i$ as the representative -- or center -- of cluster C_i. Note that $\boldsymbol{\mu}_i$ need not be one of the $\mathbf{x}_j$'s. Our goal is to find a partition that minimizes $\mathcal{G}(C_1, \ldots, C_k)$.

The k-means algorithm is a popular heuristic. It is based on the idea that the following two sub-problems are easy to solve:

1. finding the optimal representatives for a fixed partition
2. finding the optimal partition for a fixed set of representatives.

One then alternates between the two.

The theory We proved the following.

Lemma (Optimal Representatives): Fix a partition $C_1, \ldots, C_k$. The optimal representatives are the centroids

$$\boldsymbol{\mu}_i^* = \frac{1}{|C_i|} \sum_{j \in C_i} \mathbf{x}_j.$$

Lemma (Optimal Clustering): Fix the representatives $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$. An optimal partition is obtained as follows. For each j, find the $\boldsymbol{\mu}_i$ that minimizes $\|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2$ (picking one arbitrarily in the case of ties) and assign $\mathbf{x}_j$ to C_i.

Theorem (Convergence of k-means): The sequence of objective function values produced by the k-means algorithm is non-increasing.

The algorithm

In [1]:
```
# Julia version: 1.5.1
using CSV, DataFrames, Plots, LinearAlgebra
```

In [2]:
```
function opt_clust(X, k, reps)
    n, d = size(X) # n=number of rows, d=number of columns
    dist = zeros(Float64, n) # distance to rep
    assign = zeros(Int64, n) # cluster assignments
    for i = 1:n
        dist[i], assign[i] = findmin([norm(X[i,:] .- reps[j,:]) for j=1:
k])
    end
    @show G = sum(dist.^2)
    return assign
end
```

Out[2]:
```
opt_clust (generic function with 1 method)
```

In [3]:
```
function opt_reps(X, k, assign)
    n, d = size(X)
    reps = zeros(Float64, k, d) # rows are representatives
    for j = 1:k
        in_j = [i for i=1:n if assign[i] == j]
        reps[j,:] = sum(X[in_j,:],dims=1) ./ length(in_j)
    end
    return reps
end
```

Out[3]:
```
opt_reps (generic function with 1 method)
```

In [4]:
```
function mmids_kmeans(X, k; maxiter=10)
    n, d = size(X)
    assign = [rand(1:k) for i=1:n] # start with random assignments
    reps = zeros(Int64, k, d) # initialization of reps
    for iter = 1:maxiter
        # Step 1: Optimal representatives for fixed clusters
        reps = opt_reps(X, k, assign)
        # Step 2: Optimal clusters for fixed representatives
        assign = opt_clust(X, k, reps)
    end
    return assign
end
```

Out[4]: mmids_kmeans (generic function with 1 method)

1 Species delimitation

We will look again at the classical iris dataset (https://en.wikipedia.org/wiki/Iris_flower_data_set) first analyzed by Fisher. We will upload the data in the form of a `DataFrame` -- similar to a spreadsheet -- where the columns are different measurements (or features) and the rows are different samples.

In [5]:
```
df = CSV.read("iris-measurements.csv")
first(df, 5)
```

Out[5]: 5 rows × 5 columns

	Id	PetalLengthCm	PetalWidthCm	SepalLengthCm	SepalWidthCm
	Int64	Float64	Float64	Float64	Float64
1	1	1.4	0.2	5.1	3.5
2	2	1.4	0.2	4.9	3.0
3	3	1.3	0.2	4.7	3.2
4	4	1.5	0.2	4.6	3.1
5	5	1.4	0.2	5.0	3.6

In [6]:
```
describe(df)
```

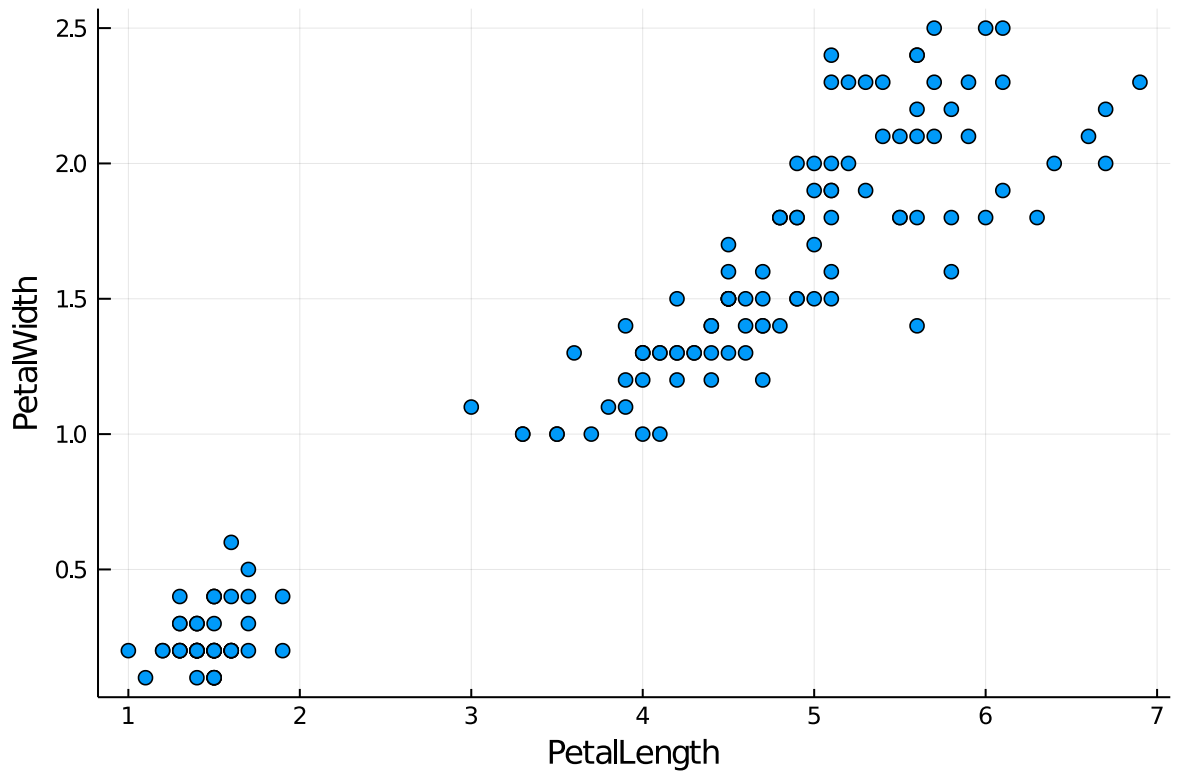
Out[6]: 5 rows × 8 columns

	variable	mean	min	median	max	nunique	nmissing	eltype
	Symbol	Float64	Real	Float64	Real	Nothing	Nothing	DataType
1	Id	75.5	1	75.5	150			Int64
2	PetalLengthCm	3.75867	1.0	4.35	6.9			Float64
3	PetalWidthCm	1.19867	0.1	1.3	2.5			Float64
4	SepalLengthCm	5.84333	4.3	5.8	7.9			Float64
5	SepalWidthCm	3.054	2.0	3.0	4.4			Float64

Let's first extract the columns and combine them into a data matrix $\texttt{X}$, and visualize the 2-dimensional petal data.

In [7]:
```
X = reduce(hcat, [df[:,:PetalLengthCm], df[:,:PetalWidthCm],
        df[:,:SepalLengthCm], df[:,:SepalWidthCm]]);
scatter(X[:,1], X[:,2],
    legend=false, xlabel="PetalLength", ylabel="PetalWidth")
```

Out[7]:



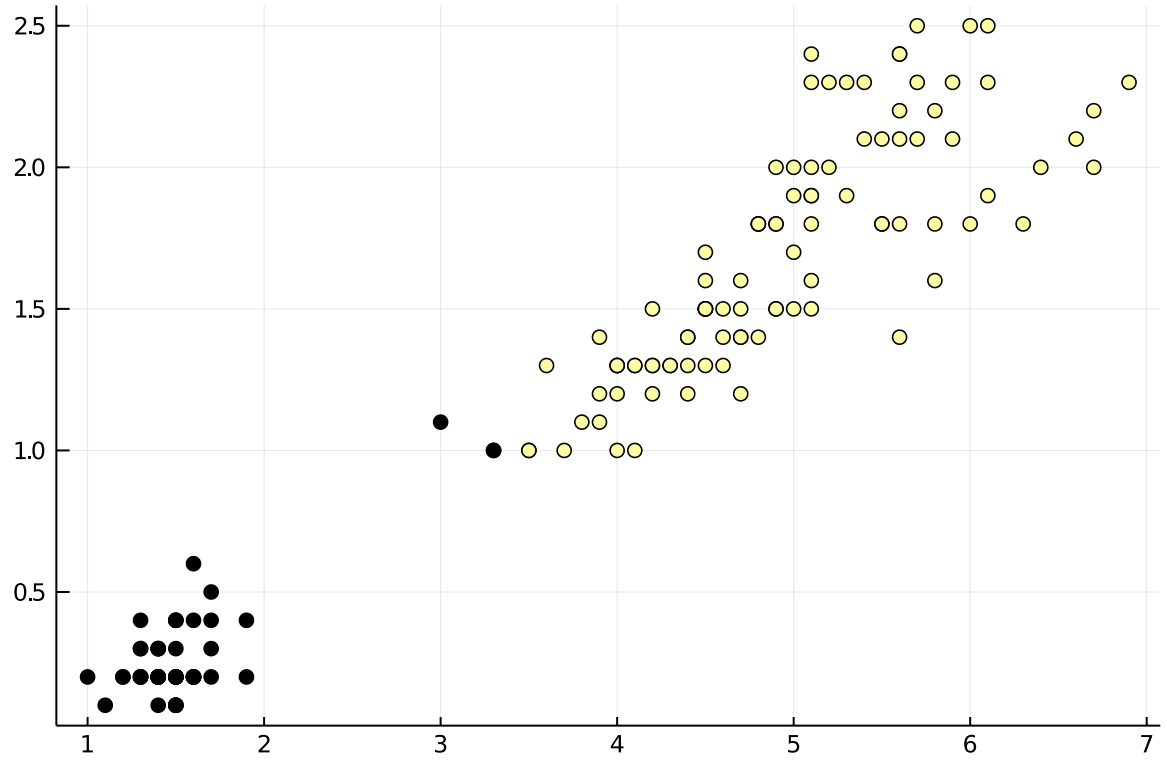
We run the k-means algorithm with $k = 2$ clusters.

In [8]:
```
assign = mmids_kmeans(X, 2);
```

```
G = sum(dist .^ 2) = 622.7644623999738
G = sum(dist .^ 2) = 156.29984835441667
G = sum(dist .^ 2) = 152.53336910043723
G = sum(dist .^ 2) = 152.36870647733903
G = sum(dist .^ 2) = 152.36870647733903
G = sum(dist .^ 2) = 152.36870647733903
G = sum(dist .^ 2) = 152.36870647733903
G = sum(dist .^ 2) = 152.36870647733903
G = sum(dist .^ 2) = 152.36870647733903
G = sum(dist .^ 2) = 152.36870647733903
```

In [9]:
```
scatter(X[:,1], X[:,2], marker_z=assign, legend=false)
```

Out[9]:



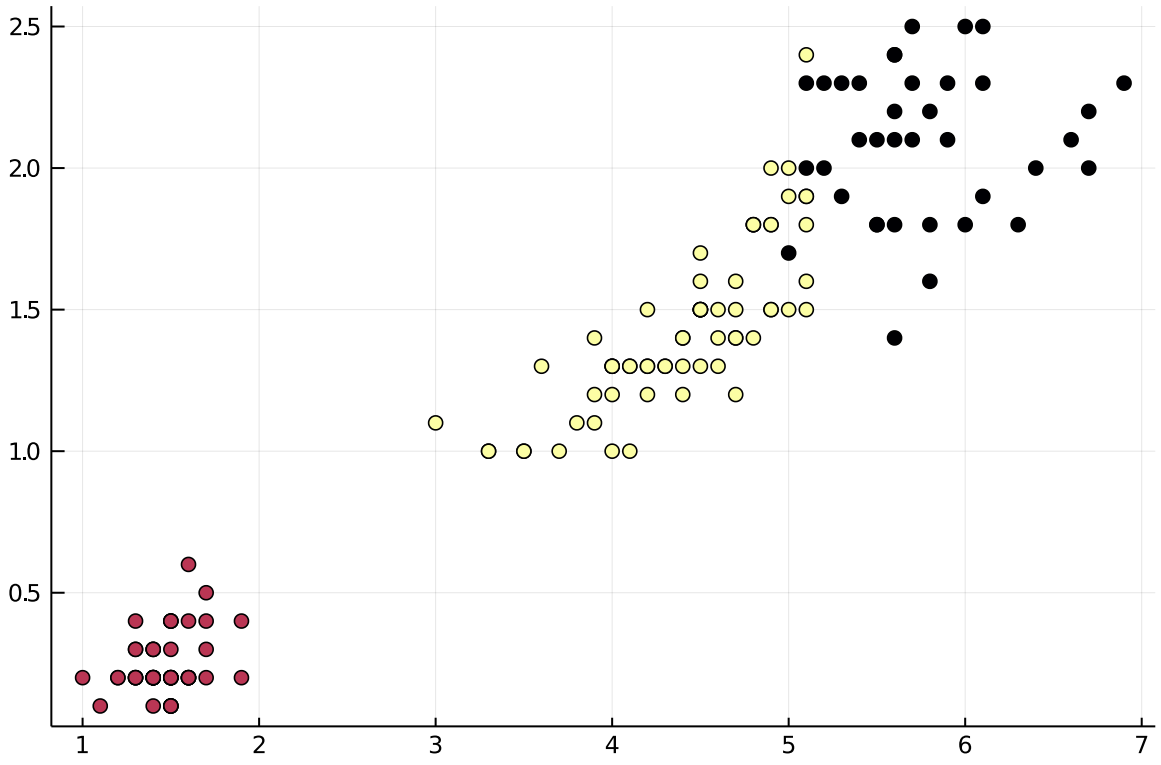
This clustering may seem less than perfect. But recall that (1) in this plot we are looking at only two of the four variables while k-means uses all of them, (2) we are not guaranteed to find the best solution, (3) our objective function is somewhat arbitrary, and (4) it is not clear what the right choice of k is. In fact, the original dataset provided the correct answer as determined by a botanist (https://en.wikipedia.org/wiki/Edgar_Anderson). Despite what the figure above may lead us to believe, there are in reality three separate species. So let's try with $k = 3$ clusters.

In [10]:
```
assign = mmids_kmeans(X, 3; maxiter=20);
```

```
G = sum(dist .^ 2) = 543.1623859092733
G = sum(dist .^ 2) = 101.63572149600577
G = sum(dist .^ 2) = 88.35799644444447
G = sum(dist .^ 2) = 85.04157943238867
G = sum(dist .^ 2) = 84.10217888865148
G = sum(dist .^ 2) = 83.13638186876973
G = sum(dist .^ 2) = 81.8390020677262
G = sum(dist .^ 2) = 80.895776
G = sum(dist .^ 2) = 79.96297983461302
G = sum(dist .^ 2) = 79.43376414532675
G = sum(dist .^ 2) = 79.01070972222222
G = sum(dist .^ 2) = 78.94506582597728
G = sum(dist .^ 2) = 78.94506582597728
G = sum(dist .^ 2) = 78.94506582597728
G = sum(dist .^ 2) = 78.94506582597728
G = sum(dist .^ 2) = 78.94506582597728
G = sum(dist .^ 2) = 78.94506582597728
G = sum(dist .^ 2) = 78.94506582597728
G = sum(dist .^ 2) = 78.94506582597728
G = sum(dist .^ 2) = 78.94506582597728
```

In [12]:
```
p3clust = scatter(X[:,1], X[:,2],
    marker_z=assign, legend=false)
```

Out[12]:



Let's load the truth and compare.

In [13]:
```
df_truth = CSV.read("iris-species.csv")
first(df_truth, 5)
```

Out[13]: 5 rows × 2 columns

	Id	Species
	Int64	String
1	1	Iris-setosa
2	2	Iris-setosa
3	3	Iris-setosa
4	4	Iris-setosa
5	5	Iris-setosa

The species are:

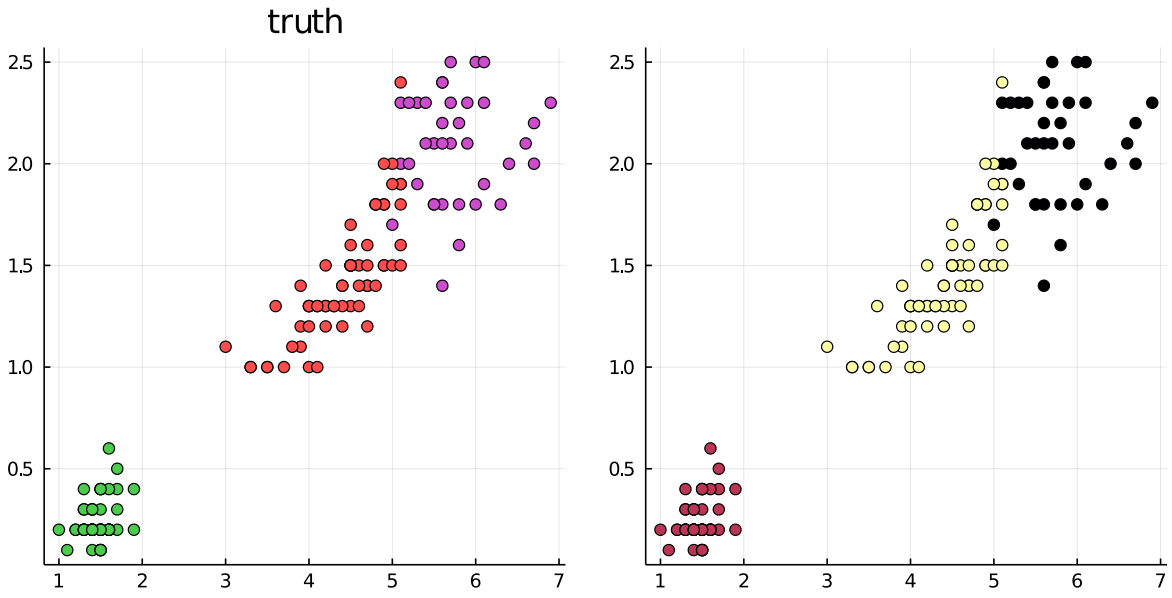
In [14]:
```
species = df_truth[:,:Species]
unique(species)
```

Out[14]:
```
3-element Array{String,1}:
 "Iris-setosa"
 "Iris-versicolor"
 "Iris-virginica"
```

To plot the outcome, we rename the species arbitrarily as 1, 2, and 3 using a [dictionary (https://docs.julialang.org/en/v1/base/collections/#Dictionaries-1)](https://docs.julialang.org/en/v1/base/collections/#Dictionaries-1).

In [15]:
```
species2number = Dict("Iris-setosa"=>1, "Iris-versicolor"=>2, "Iris-virginica"=>3)
truth = map(i -> get(species2number,i,0), species)
ptruth = scatter(X[:,1], X[:,2],
    marker_z=assign, legend=false, seriescolor=:lightrainbow, title="truth")
plot(ptruth, p3clust, layout=(1,2), size=(750,375))
```

Out[15]:



This time we get a perfect outcome...

Determining the appropriate number of clusters is not a straighforward problem. To quote Wikipedia (https://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set):

> The correct choice of k is often ambiguous, with interpretations depending on the shape and scale of the distribution of points in a data set and the desired clustering resolution of the user. In addition, increasing k without penalty will always reduce the amount of error in the resulting clustering, to the extreme case of zero error if each data point is considered its own cluster (i.e., when k equals the number of data points, n). Intuitively then, the optimal choice of k will strike a balance between maximum compression of the data using a single cluster, and maximum accuracy by assigning each data point to its own cluster. If an appropriate value of k is not apparent from prior knowledge of the properties of the data set, it must be chosen somehow. There are several categories of methods for making this decision.

In practice, several heuristic approaches (https://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set) are in use. Other approaches to clustering, e.g. DBSCAN (https://en.wikipedia.org/wiki/DBSCAN) and hierarchical clustering (https://en.wikipedia.org/wiki/Hierarchical_clustering), do not require a number of clusters as input.