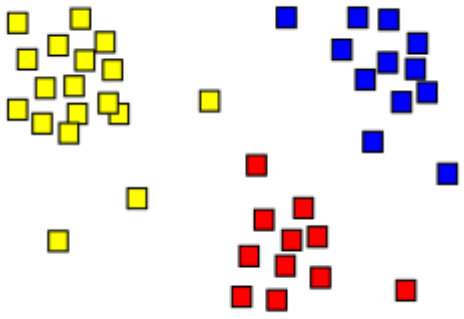
TOPIC 0

Introduction

2 Clustering: an objective and an algorithm

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 8, 2020

Consider the following fundamental problem in data science. 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.



(Source) (https://commons.wikimedia.org/wiki/File:Cluster-2.svg)

Fix a number of clusters k. Formally, we think of a clustering as a partition.

Definition (Partition): A partition of $[n] = \{1, \dots, n\}$ of size k is a collection of non-empty subsets $C_1, \dots, C_k \subseteq [n]$ that:

- are pairwise disjoint, i.e., $C_i \cap C_j = \emptyset, \forall i \neq j$
- cover all of $[n]$, i.e., $\cup_{i=1}^{k} C_i = [n]$. $\lhd$

2.1 The k-means objective

Given n vectors $\mathbf{x}_1, \dots, \mathbf{x}_n$ in $\mathbb{R}^d$ and a partition $C_1, \dots, C_k \subseteq [n]$, it will be useful to have notation for the corresponding cluster assignment: we define $c(i) = j$ if $i \in C_j$.

Under the k-means objective, the cost of $C_1, \dots, C_k$ is then defined as

$$\mathcal{G}(C_1, \dots, C_k) = \min_{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k \in \mathbb{R}^d} G(C_1, \dots, C_k; \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k)$$

where

$$G(C_1, \dots, C_k; \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k) = \sum_{j=1}^{k} \sum_{i \in C_j} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 = \sum_{i=1}^{n} \|\mathbf{x}_i - \boldsymbol{\mu}_{c(i)}\|^2.$$

Here $\boldsymbol{\mu}_i \in \mathbb{R}^d$ is the representative -- or center -- of cluster C_i. Note that $\boldsymbol{\mu}_i$ need not be one of the $\mathbf{x}_j$'s.

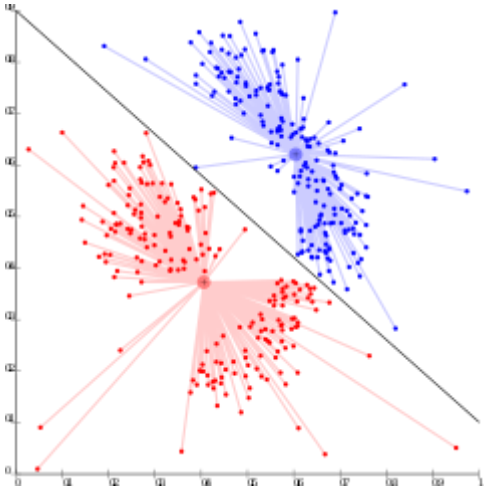
Our goal is to find a partition that minimizes $\mathcal{G}(C_1, \dots, C_k)$, i.e., to solve the problem:

$$\min_{C_1, \dots, C_k} \mathcal{G}(C_1, \dots, C_k)$$

over all partitions of $[n]$ of size k.

To quote Wikipedia (https://en.wikipedia.org/wiki/Cluster_analysis#Centroid-based_clustering):

> In centroid-based clustering, clusters are represented by a central vector, which may not necessarily be a member of the data set. When the number of clusters is fixed to k, k-means clustering gives a formal definition as an optimization problem: find the k cluster centers and assign the objects to the nearest cluster center, such that the squared distances from the cluster are minimized.



(Source) (https://commons.wikimedia.org/wiki/File:KMeans-density-data.svg)

In general, the problem is NP-hard (https://en.wikipedia.org/wiki/NP-hardness), that is, no fast algorithm is expected to exist to solve it. 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 (perhaps until progress falls below a tolerance). To elaborate on the first step, we review an elementary fact about quadratic functions.

2.1.1 Preliminary result: minimizing a quadratic function

A key step of the algorithm involves minimizing a quadratic function (https://en.wikipedia.org/wiki/Quadratic_function). Consider the function

$$q(x) = ax^2 + bx + c.$$

When $a > 0$, q has a unique minimum.

Lemma (Minimum of Quadratic Function): Let $q(x) = ax^2 + bx + c$ where $a > 0$ and $x \in \mathbb{R}$. The unique minimum of q is achieved at $x^* = -b/2a$.

Proof: By the *First-Order Necessary Condition*, a global minimizer of q (which is necessarily a local minimizer) satisfies the condition

$$\frac{\mathrm{d}}{\mathrm{d}x} q(x) = 2ax + b = 0,$$

whose unique solution is

$$x^* = -\frac{b}{2a}.$$

To see that x^* is indeed a global minimizer, we re-write q as

$$\begin{aligned} q(x) &= a\left(x^2 + 2\left[\frac{b}{2a}\right]x\right) + c \\ &= a\left(x^2 + 2\left[\frac{b}{2a}\right]x + \left[\frac{b}{2a}\right]^2\right) - a\left[\frac{b}{2a}\right]^2 + c \\ &= a(x - x^*)^2 + \left[c - \frac{b^2}{4a}\right]. \end{aligned}$$

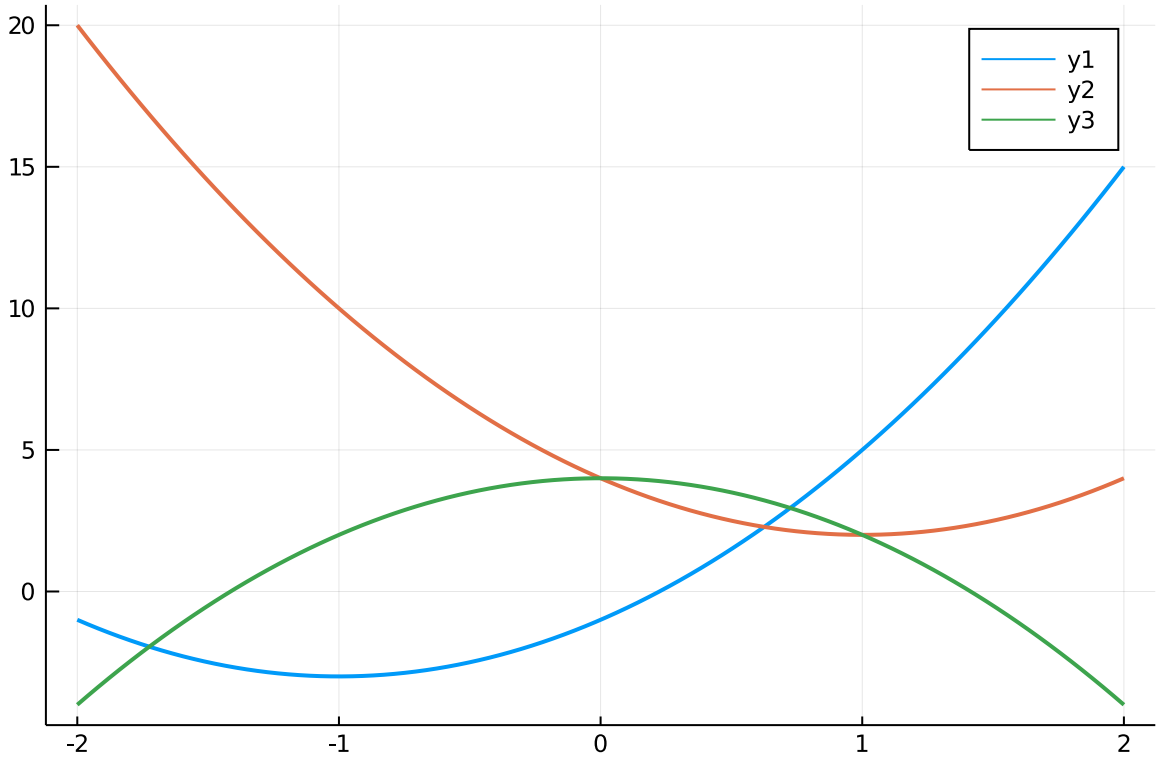
Clearly, any other x gives a higher value for q. □

NUMERICAL CORNER Here's a numerical example.

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

In [2]:
```
q(x) = a*x^2 + b*x + c;

x = LinRange(-2, 2, 100); # 100 equally spaced points between -2 and 2

a, b, c = 2, 4, -1;
y₁ = q.(x); # the `.` applies the function `q` element-wise to the 1d ar
ray `x`
plot(x, y₁, label="y1", lw=2)

a, b, c = 2, -4, 4;
y₂ = q.(x);
plot!(x, y₂, label="y2", lw=2) # the `!` indicates that it modifies the
 existing plot

a, b, c = -2, 0, 4;
y₃ = q.(x);
plot!(x, y₃, label="y3", lw=2)
```

Out[2]:



2.1.2 Finding the optimal representatives

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

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

Proof idea: The objective G can be written as a sum, where each term is a quadratic function in one component of one of the $\boldsymbol{\mu}_i$'s. Each of these terms is minimized by the average of the corresponding components of the $\mathbf{x}_j$'s belonging C_i.

Proof: Note that we can expand the k-means objective as

$$\begin{aligned}\sum_{i=1}^{k} \sum_{j \in C_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2 &= \sum_{i=1}^{k} \sum_{j \in C_i} \sum_{m=1}^{d} (x_{j,m} - \mu_{i,m})^2 \\ &= \sum_{m=1}^{d} \sum_{i=1}^{k} \left[\sum_{j \in C_i} (x_{j,m} - \mu_{i,m})^2 \right].\end{aligned}$$

The expression in square brackets is a quadratic function in $\mu_{i,m}$:

$$q_{i,m}(\mu_{i,m}) = \left\{ \sum_{j \in C_i} x_{j,m}^2 \right\} + \left\{ -2 \sum_{j \in C_i} x_{j,m} \right\} \mu_{i,m} + \{|C_i|\} \, \mu_{i,m}^2,$$

and therefore, by the formula for the minimum of a quadratic function, is minimized at

$$\mu_{i,m}^* = -\frac{-2 \sum_{j \in C_i} x_{j,m}}{2|C_i|} = \frac{1}{|C_i|} \sum_{j \in C_i} x_{j,m}.$$

Since each term in the sum making up G is strictly minimized at $\boldsymbol{\mu}_1^*, \ldots, \boldsymbol{\mu}_k^*$, so is G. □

2.1.3 Finding the optimal partition

Lemma (Optimal Clustering): Fix the representatives $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$. An optimal partition under G 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.

Proof: By definition, each term in

$$G(C_1, \ldots, C_k; \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k) = \sum_{i=1}^{n} \|\mathbf{x}_i - \boldsymbol{\mu}_{c(i)}\|^2$$

is, when the $\boldsymbol{\mu}_j$'s are fixed, minimized by the assignment in the statement. Hence so is G. □

2.2 The k-means algorithm

We are now ready to describe the k-means algorithm (https://en.wikipedia.org/wiki/K-means_clustering). We start from a random assignment of clusters. We then alternate between the optimal choices in the lemmas. In lieu of pseudo-code, we write out the algorithm in Julia.

The input `X` is assumed to be a collection of n vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ stacked into a matrix. The other input, `k`, is the desired number of clusters. There is an optional input `maxiter` for the maximum number of iterations, which is set to 10 by default.

We first define separate functions for the two main steps. To find the minimum of an array, we use the function `findmin` (https://docs.julialang.org/en/v1/base/collections/#Base.findmin).

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 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[4]:
```
opt_clust (generic function with 1 method)
```

The main function follows. Below, `rand(1:k)` is a uniformly chosen integer between `1` and `k` (inclusive).

In [5]:
```
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[5]:
```
mmids_kmeans (generic function with 1 method)
```

The k-means algorithm is only a heuristic. In particular, it is not guaranteed to find the global minimum of the k-means objective. However, it is guaranteed to improve the objective at every iteration, or more precisely, not to make it worse.

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

Proof idea: By the *Optimal Representatives* and *Optimal Clustering* lemmas, each step does not increase the objective.

Proof: Let $C_1', \ldots, C_k'$ be the current clusters, with representatives $\boldsymbol{\mu}_1', \ldots, \boldsymbol{\mu}_k'$. After Step 1, the new representatives are $\boldsymbol{\mu}_1'', \ldots, \boldsymbol{\mu}_k''$. By the *Optimal Representatives Lemma*, they satisfy

$$\sum_{i=1}^{k} \sum_{j \in C_i'} \|\mathbf{x}_j - \boldsymbol{\mu}_i''\|^2 \leq \sum_{i=1}^{k} \sum_{j \in C_i'} \|\mathbf{x}_j - \boldsymbol{\mu}_i'\|^2.$$

After Step 2, the new clusters are $C_1'', \ldots, C_k''$. By the *Optimal Clustering Lemma*, they satisfy

$$\sum_{i=1}^{k} \sum_{j \in C_i''} \|\mathbf{x}_j - \boldsymbol{\mu}_i''\|^2 \leq \sum_{i=1}^{k} \sum_{j \in C_i'} \|\mathbf{x}_j - \boldsymbol{\mu}_i''\|^2.$$

Combining these two inequalities gives

$$\sum_{i=1}^{k} \sum_{j \in C_i''} \|\mathbf{x}_j - \boldsymbol{\mu}_i''\|^2 \leq \sum_{i=1}^{k} \sum_{j \in C_i'} \|\mathbf{x}_j - \boldsymbol{\mu}_i'\|^2,$$

as claimed. □

The sequence of objective values is monotone and bounded from below by 0. Hence it converges (https://en.wikipedia.org/wiki/Monotone_convergence_theorem#Convergence_of_a_monotone_sequence_of_real

Exercise: Modify `mmids_kmeans` to take a tolerance `tol` as input and stop when the improvement in objective value `G` falls below the tolerance. ◁

2.3 Simulated data

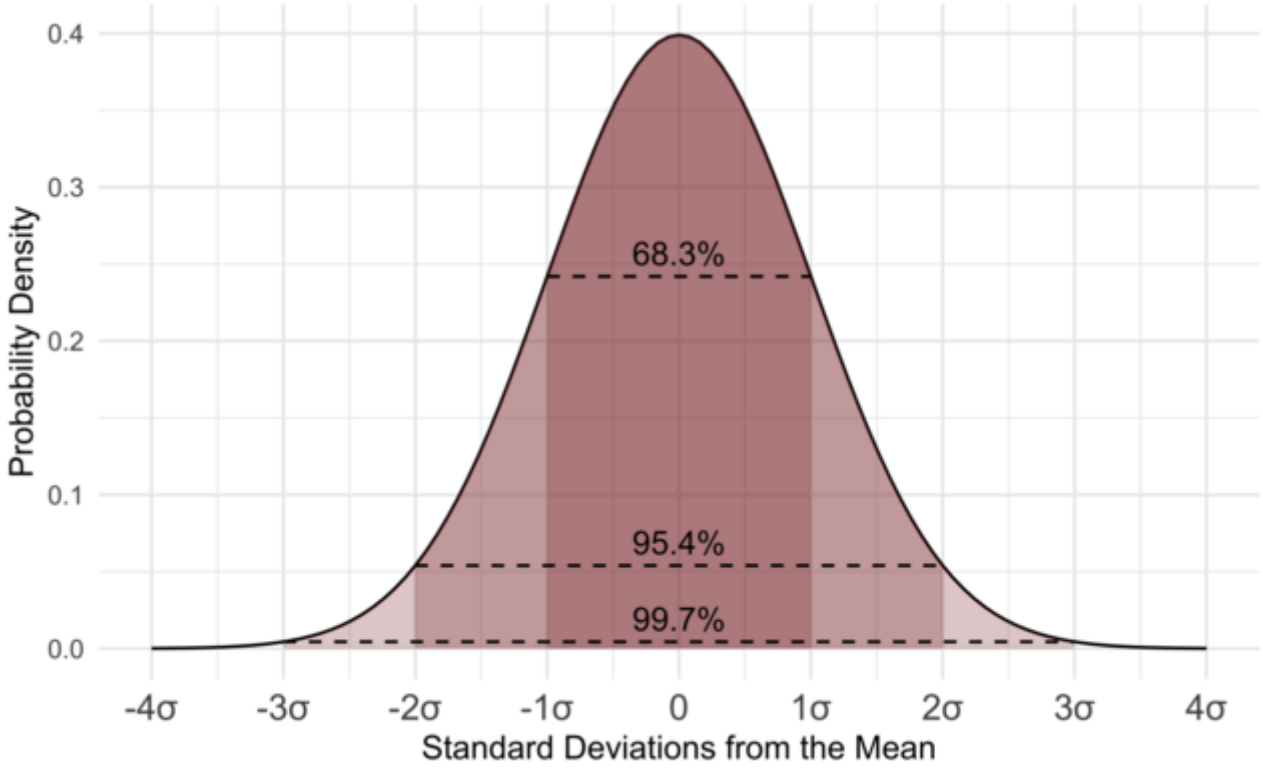
We will test our implementation of k-means on a simple simulated dataset.

Recall that a standard Normal variable X has PDF

$$f_X(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-x^2/2\right).$$

Its mean is 0 and its variance is 1.

This is what its PDF looks like:



(Source (https://upload.wikimedia.org/wikipedia/commons/thumb/4/44/Standard_Normal_Distribution.png/1599px-Standard_Normal_Distribution.png))

To construct a d-dimensional version, we take d independent standard Normal variables $X_1, X_2, \ldots, X_d$ and form the vector $\mathbf{X} = (X_1, \ldots, X_d)$. We will say that $\mathbf{X}$ is a standard Normal d-vector. By independence, its joint PDF is given by the product of the PDFs of the X_i's, that is,

$$\begin{aligned} f_{\mathbf{X}}(\mathbf{x}) &= \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi}} \exp\left(-x_i^2/2\right) \\ &= \frac{1}{\prod_{i=1}^{d} \sqrt{2\pi}} \exp\left(- \sum_{i=1}^{d} x_i^2/2 \right) \\ &= \frac{1}{(2\pi)^{d/2}} \exp(-\|\mathbf{x}\|^2/2). \end{aligned}$$

We can also shift and scale it.

Definition (Spherical Gaussian): Let $\mathbf{Z}$ be a standard Normal d-vector, let $\boldsymbol{\mu} \in \mathbb{R}^d$ and let $\sigma \in \mathbb{R}_+$. Then we will refer to the transformed random variable $\mathbf{X} = \boldsymbol{\mu} + \sigma \mathbf{Z}$ as a spherical Gaussian with mean $\boldsymbol{\mu}$ and variance σ^2. We use the notation $\mathbf{Z} \sim N_d(\boldsymbol{\mu}, \sigma^2 I)$. ◁

The following function generates n data points from two spherical d-dimensional Gaussians with variance 1, one with mean $-w\mathbf{e}_1$ and one with mean $w\mathbf{e}_1$. Below, `randn(d)` generates a `d`-dimensional spherical Gaussian.

In [6]:
```
function two_clusters(d, n, w)
    X1 = reduce(hcat, [vcat(-w, zeros(d-1)) .+ randn(d) for i=1:n])'
    X2 = reduce(hcat, [vcat(w, zeros(d-1)) .+ randn(d) for i=1:n])'
    return X1, X2
end
```

Out[6]:
```
two_clusters (generic function with 1 method)
```

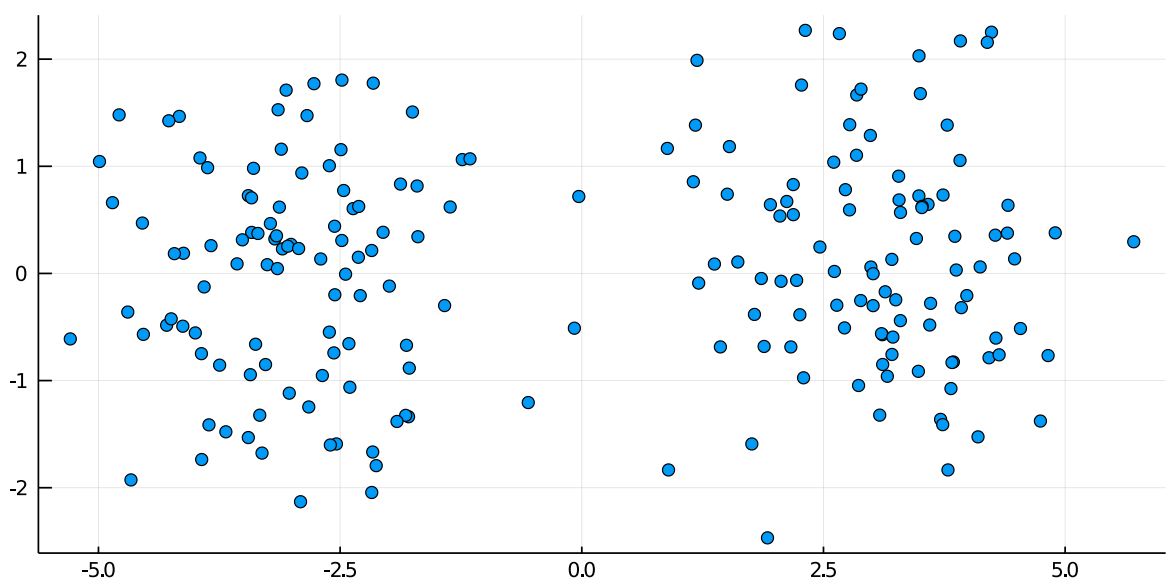
We will mix these two datasets to form an interesting case for clustering.

2.3.1 Two dimension

We start with $d = 2$.

In [7]:
```
d, n, w = 2, 100, 3.
X1, X2 = two_clusters(d, n, w)
X = vcat(X1, X2)
scatter(X[:,1], X[:,2], legend=false, size=(700,350))
```

Out[7]:



In the figure above, we observe two fairly well-separated clusters. Let's run k-means on this dataset using $k = 2$.

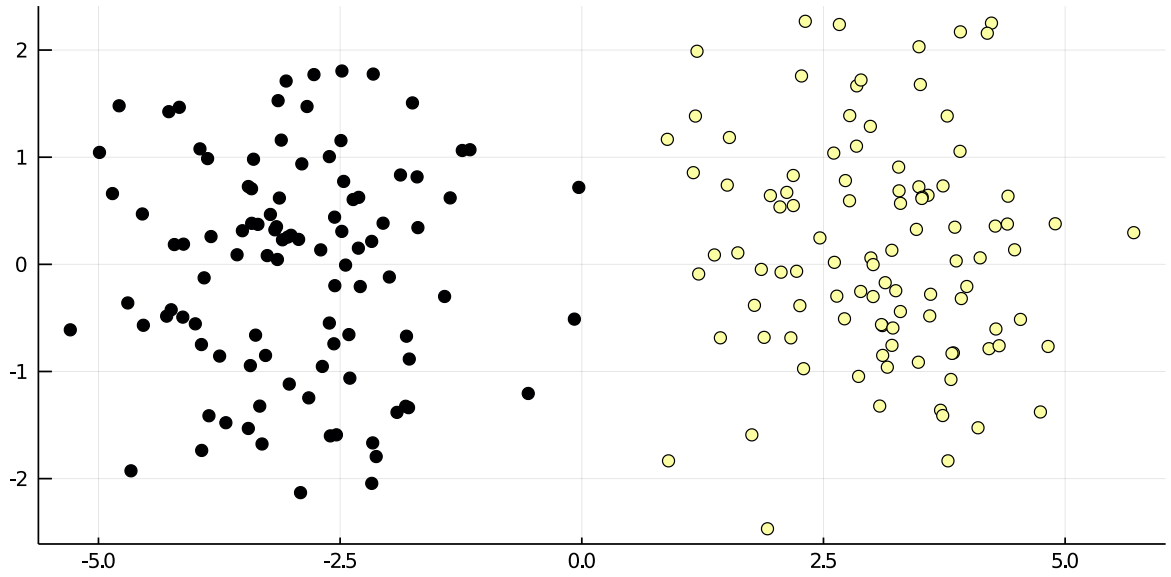
In [8]:
```
assign = mmids_kmeans(X, 2);
```

```
G = sum(dist .^ 2) = 1642.660544012831
G = sum(dist .^ 2) = 412.35579408034704
G = sum(dist .^ 2) = 412.35579408034704
G = sum(dist .^ 2) = 412.35579408034704
G = sum(dist .^ 2) = 412.35579408034704
G = sum(dist .^ 2) = 412.35579408034704
G = sum(dist .^ 2) = 412.35579408034704
G = sum(dist .^ 2) = 412.35579408034704
G = sum(dist .^ 2) = 412.35579408034704
G = sum(dist .^ 2) = 412.35579408034704
```

Our default of 10 iterations seem to have been enough for the algorithm to converge. We can visualize the result by coloring (https://docs.juliaplots.org/latest/attributes/) the points according to the assignment.

In [9]:
```
scatter(X[:,1], X[:,2], marker_z=assign, legend=false, size=(700,350))
```

Out[9]:

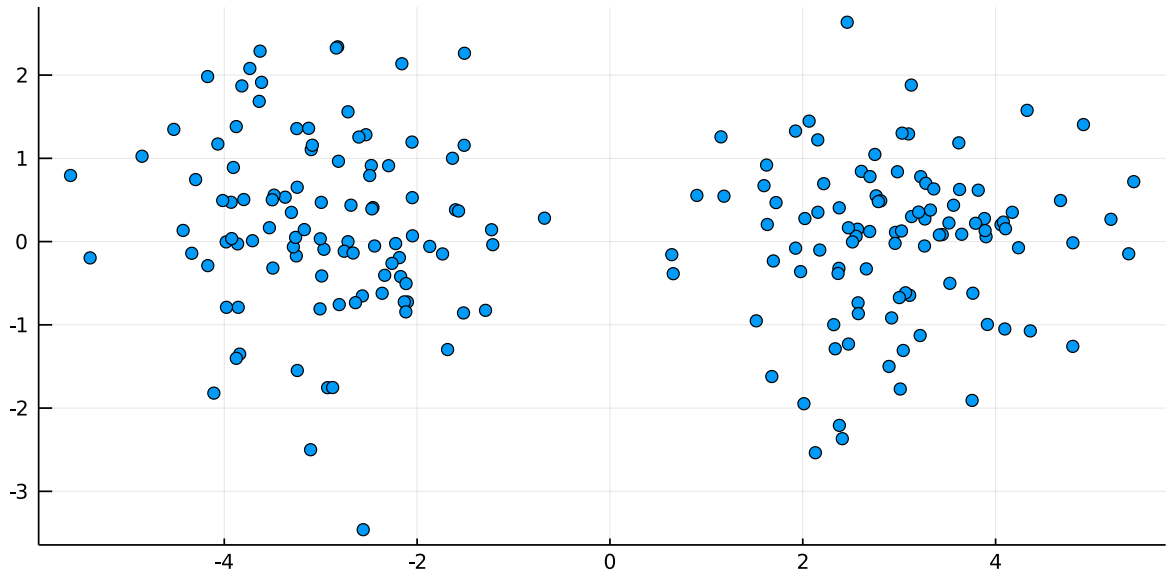


2.3.2 General dimension

Let's see what happens in higher dimension. We repeat our experiment with $d = 1000$.

In [10]:
```
d, n, w = 1000, 100, 3
X1, X2 = two_clusters(d, n, w)
X = vcat(X1, X2)
scatter(X[:,1], X[:,2], legend=false, size=(700,350))
```

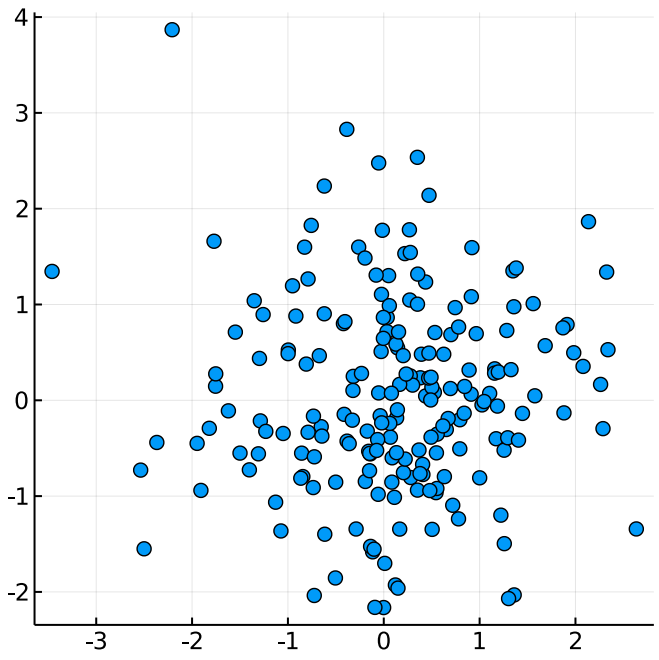
Out[10]:



This dataset is in 1000 dimensions, but we've plotted the data in only the first two dimensions. We observe well-separated clusters. If we plot in any two dimensions not including the first one instead, we see only one cluster.

In [11]:
```
scatter(X[:,2], X[:,3], legend=false, size=(350,350))
```

Out[11]:



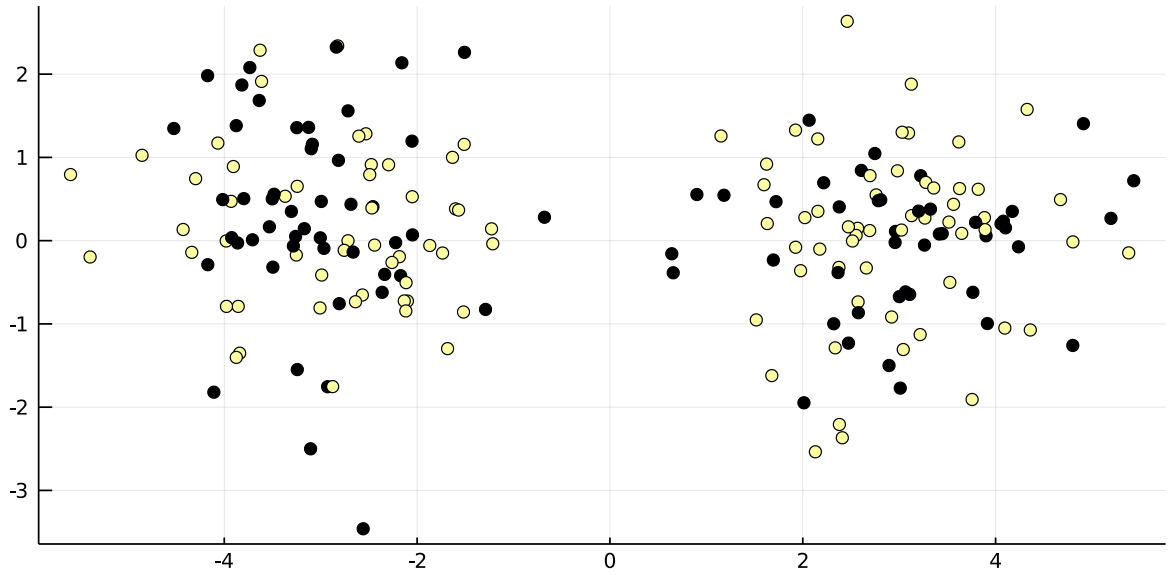
Let's see how k-means fares on this dataset.

In [12]:
```
assign = mmids_kmeans(X, 2);
```

```
G = sum(dist .^ 2) = 199775.6175486944
G = sum(dist .^ 2) = 199754.31646446598
G = sum(dist .^ 2) = 199717.27042900465
G = sum(dist .^ 2) = 199717.27042900465
G = sum(dist .^ 2) = 199717.27042900465
G = sum(dist .^ 2) = 199717.27042900465
G = sum(dist .^ 2) = 199717.27042900465
G = sum(dist .^ 2) = 199717.27042900465
G = sum(dist .^ 2) = 199717.27042900465
G = sum(dist .^ 2) = 199717.27042900465
```

In [13]:
```
scatter(X[:,1], X[:,2], marker_z=assign, legend=false, size=(700,350))
```

Out[13]:



Our attempt at clustering does not appear to have been successful. What happened? While these clusters are easy to tease apart if we know to look at the first coordinate only, in the full space the within-cluster and between-cluster distances become harder to distinguish: the noise overwhelms the signal.

The function below plots the histograms of within-cluster and between-cluster distances for a sample of size n in d dimensions with a given offset. As d increases, the two distributions become increasingly indistinguishable. Later in the course, we will develop dimension-reduction techniques that help deal with this issue.

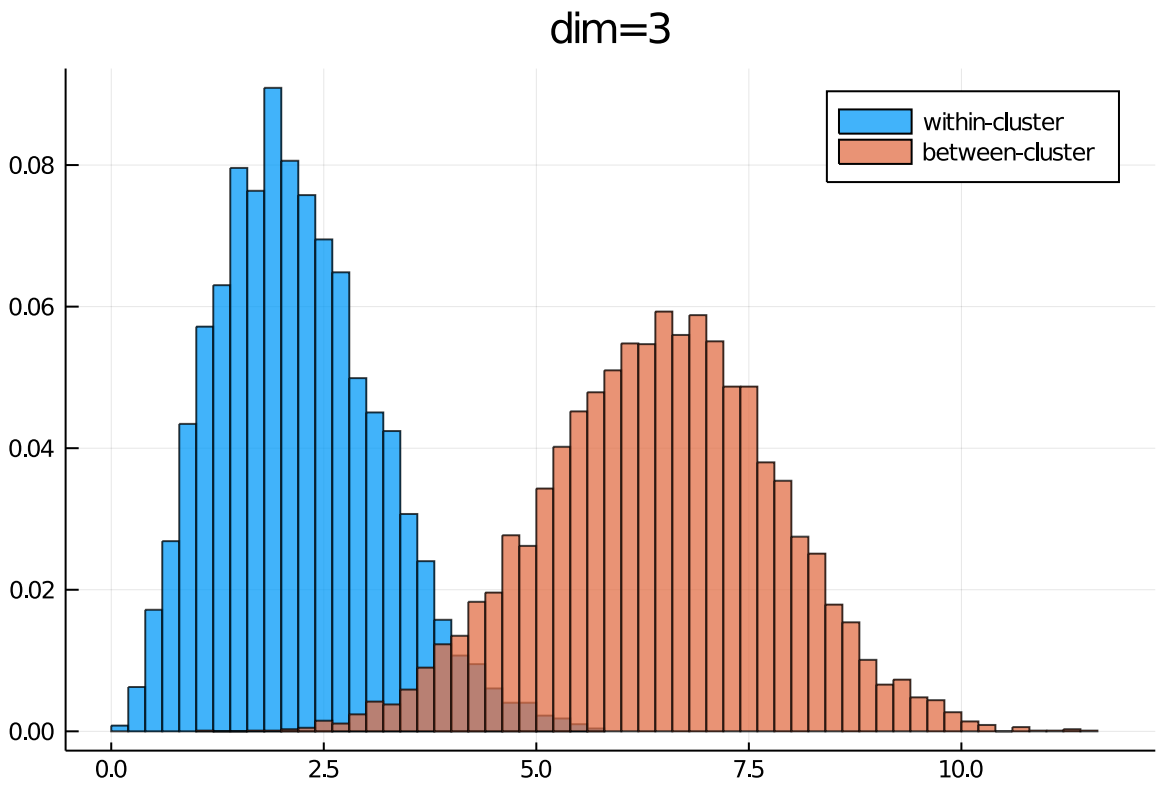
In [14]:
```
function highdim_2clusters(d, n, w)
    # generate datasets
    X1, X2 = two_clusters(d, n, w)

    # within-cluster distances for X1
    intra = vec([norm(X1[i,:] .- X1[j,:]) for i=1:n, j=1:n if j>i])
    h = histogram(intra, normalize=:probability,
        label="within-cluster", alpha=0.75, title="dim=$d") # alpha=tran
sparency

    # between-cluster distances
    inter = vec([norm(X1[i,:] .- X2[j,:]) for i=1:n, j=1:n])
    histogram!(inter, normalize=:probability,
        label="between-cluster", alpha=0.75)
end
```

Out[14]:
```
highdim_2clusters (generic function with 1 method)
```

Below we plot the results for dimensions $d = 3, 100, 1000$. What do you observe?

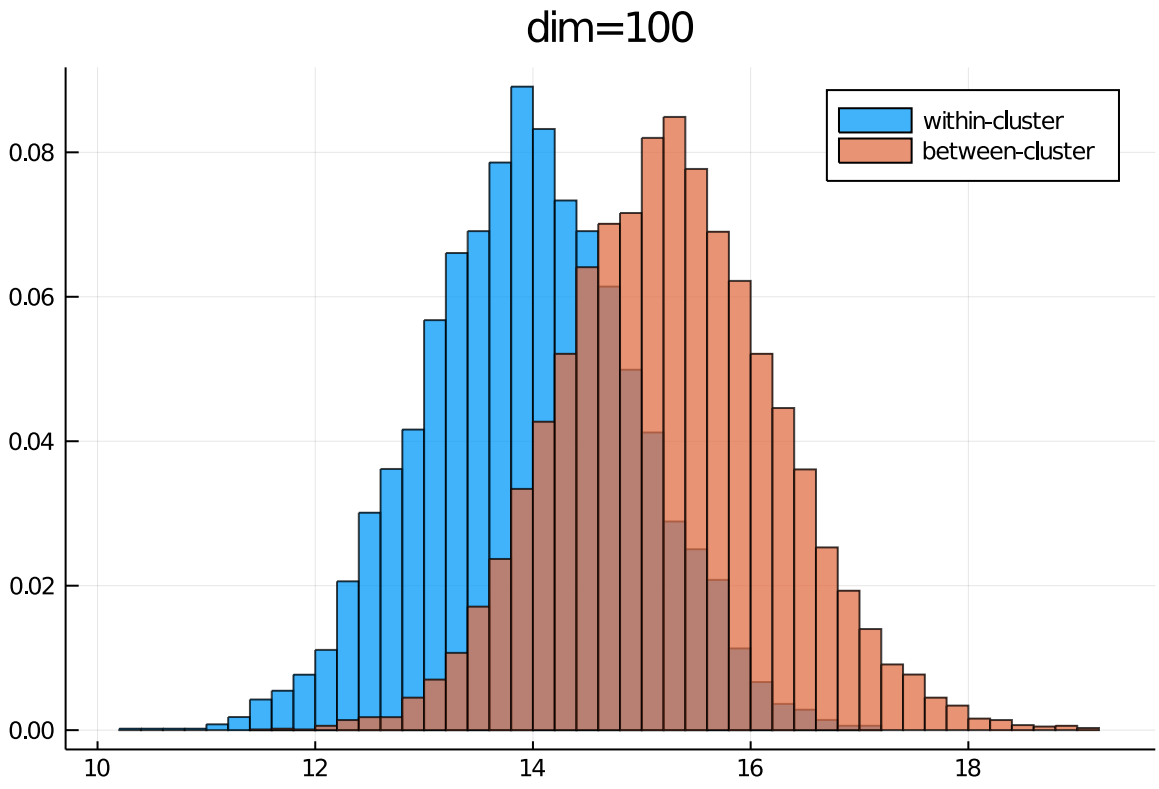
In [15]:
```
highdim_2clusters(3, 100, 3)
```

Out[15]:



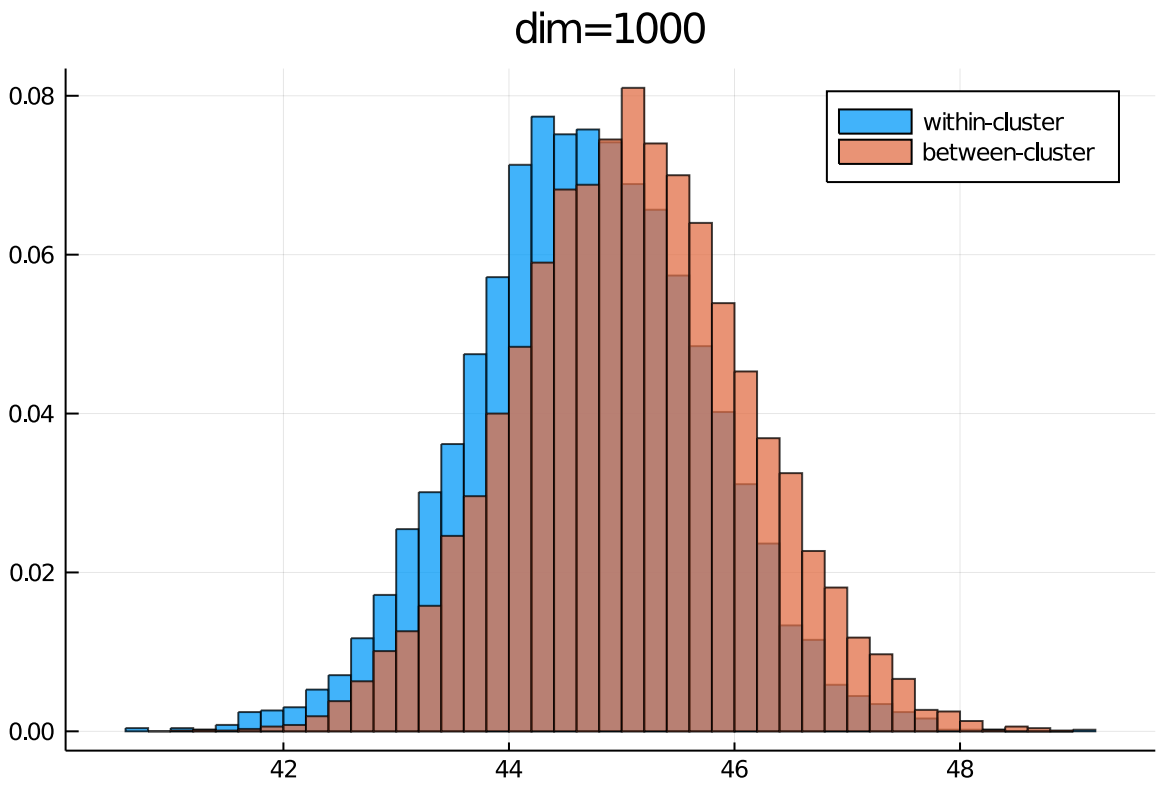
In [16]:
```
highdim_2clusters(100, 100, 3)
```

Out[16]:



In [17]:
```
highdim_2clusters(1000, 100, 3)
```

Out[17]:



Exercise: Let $Z_1 \sim \mathrm{N}(\mu_1, \sigma_1^2)$ and $Z_2 \sim \mathrm{N}(\mu_2, \sigma_2^2)$ be independent Normal variables with mean μ_1, μ_2 and variance σ_1^2 and σ_2^2 respectively. Recall that $Z_1 + Z_2$ is still Normal.

(a) What are the mean and variance of $Z_1 + Z_2$?

(b) Compute $\mathbb{E}[\|\mathbf{X}_1 - \mathbf{X}_2\|^2]$ and $\mathbb{E}[\|\mathbf{Y}_1 - \mathbf{Y}_2\|^2]$ explicitly in the claim above.◁

2.3.3 Proof of the claim [optional]

In this optional section, we give a formal statement of the phenomenon described in the previous subsection.

Claim: Let $\mathbf{X}_1, \mathbf{X}_2, \mathbf{Y}_1$ be independent spherical d-dimensional Gaussians with mean $-w_d \mathbf{e}_1$ and variance 1, where $\{w_d\}$ is a monotone sequence in d. Let $\mathbf{Y}_2$ be an indepedent spherical d-dimensional Gaussian with mean $w_d \mathbf{e}_1$ and variance 1. Then, letting $\Delta_d = \|\mathbf{Y}_1 - \mathbf{Y}_2\|^2 - \|\mathbf{X}_1 - \mathbf{X}_2\|^2$, as $d \to +\infty$

$$\frac{\mathbb{E}[\Delta_d]}{\sqrt{\mathrm{Var}[\Delta_d]}} \to \begin{cases} 0, & \text{if } w_d \ll d^{1/4} \\ +\infty, & \text{if } w_d \gg d^{1/4} \end{cases}$$

where $w_d \ll d^{1/4}$ means $w_d / d^{1/4} \to 0$.

The ratio is the statement is referred to as the signal-to-noise ratio (https://en.wikipedia.org/wiki/Signal-to-noise_ratio). The implication of this theorem is easier to understand in a simulation.

We will need the following lemmas. Recall first that the covariance of real-valued random variables W_1 and W_2 is

$$\begin{aligned} \mathrm{Cov}[W_1, W_2] &= \mathbb{E}[(W_1 - \mathbb{E}[W_1])(W_2 - \mathbb{E}[W_2])] \\ &= \mathbb{E}[W_1 W_2] - \mathbb{E}[W_1]\mathbb{E}[W_2]. \end{aligned}$$

Lemma (Variance of a Sum): If W_1 and W_2 are real-valued random variables, then

$$\mathrm{Var}[W_1 + W_2] = \mathrm{Var}[W_1] + \mathrm{Var}[W_2] + 2\,\mathrm{Cov}[W_1, W_2].$$

Proof: By definition of the variance and linearity of expectation,

$$\begin{aligned} \mathrm{Var}[W_1 + W_2] &= \mathbb{E}[(W_1 + W_2 - \mathbb{E}[W_1 + W_2])^2] \\ &= \mathbb{E}[(\{W_1 - \mathbb{E}[W_1]\} + \{W_2 - \mathbb{E}[W_2]\})^2]. \end{aligned}$$

Expanding the square gives the claim. □

Lemma: Let W be a real-valued random variable symmetric about zero, that is, such that W and $-W$ are identically distributed. Then for all odd k, $\mathbb{E}[W^k] = 0$.

Proof: By the symmetry,

$$\mathbb{E}[W^k] = \mathbb{E}[(-W)^k] = \mathbb{E}[(-1)^k W^k] = -\mathbb{E}[W^k].$$

The only way to satisfy this equation is to have $\mathbb{E}[W^k] = 0$. □

Proof idea (Claim): The only coordinate contributing to $\mathbb{E}[\Delta_d]$ is the first one by linearity of expectation, while all coordinates contribute to $\mathrm{Var}[\Delta_d]$. More specifically, a calculation shows that the former is $c_0 w^2$ while the latter is $c_1 w^2 + c_2 d$, where c_0, c_1, c_2 are constants.

Proof (Claim): Write $w := w_d$ and $\Delta := \Delta_d$ to simplify the notation. There are two steps:

(1) Expectation of Δ: By defintion, the random variables $X_{1,i} - X_{2,i}$, $i = 1, \dots, d$, and $Y_{1,i} - Y_{2,i}$, $i = 2, \dots, d$, are identically distributed. So, by linearity of expectation,

$$\begin{aligned}\mathbb{E}[\Delta] &= \sum_{i=1}^{d} \mathbb{E}[(Y_{1,i} - Y_{2,i})^2] - \sum_{i=1}^{d} \mathbb{E}[(X_{1,i} - X_{2,i})^2] \\ &= \mathbb{E}[(Y_{1,1} - Y_{2,1})^2] - \mathbb{E}[(X_{1,1} - X_{2,1})^2].\end{aligned}$$

Further, we can write $Y_{1,1} - Y_{1,2} \sim (Z_1 - w) - (Z_2 + w)$ where $Z_1, Z_2 \sim N(0,1)$ are independent, where here $\sim$ indicates equality in distribution. Hence, we have

$$\begin{aligned}\mathbb{E}[(Y_{1,1} - Y_{2,1})^2] &= \mathbb{E}[(Z_1 - Z_2 - 2w)^2] \\ &= \mathbb{E}[(Z_1 - Z_2)^2] - 4w\,\mathbb{E}[Z_1 - Z_2] + 4w^2.\end{aligned}$$

Similarly, $X_{1,1} - X_{1,2} \sim Z_1 - Z_2$ so $\mathbb{E}[(X_{1,1} - X_{2,1})^2] = \mathbb{E}[(Z_1 - Z_2)^2]$. Since $\mathbb{E}[Z_1 - Z_2] = 0$, we finally get $\mathbb{E}[\Delta] = 4w^2$.

(2) Variance of Δ: Using the observations from (1) and the independence of the coordinates we get

$$\begin{aligned}\mathrm{Var}[\Delta] &= \sum_{i=1}^{d} \mathrm{Var}[(Y_{1,i} - Y_{2,i})^2] + \sum_{i=1}^{d} \mathrm{Var}[(X_{1,i} - X_{2,i})^2] \\ &= \mathrm{Var}[(Z_1 - Z_2 - 2w)^2] + (2d-1)\,\mathrm{Var}[(Z_1 - Z_2)^2].\end{aligned}$$

By the *Variance of a Sum*,

$$\begin{aligned}\mathrm{Var}[(Z_1 - Z_2 - 2w)^2] &= \mathrm{Var}[(Z_1 - Z_2)^2 - 4w(Z_1 - Z_2) + 4w^2] \\ &= \mathrm{Var}[(Z_1 - Z_2)^2 - 4w(Z_1 - Z_2)] \\ &= \mathrm{Var}[(Z_1 - Z_2)^2] + 16w^2\mathrm{Var}[Z_1 - Z_2] \\ &\quad - 8w\,\mathrm{Cov}[(Z_1 - Z_2)^2, Z_1 - Z_2].\end{aligned}$$

Because Z_1 and Z_2 are independent, $\mathrm{Var}[Z_1 - Z_2] = \mathrm{Var}[Z_1] + \mathrm{Var}[Z_2] = 2$. Moreover, the random variable $(Z_1 - Z_2)$ is symmetric, so

$$\begin{aligned}\mathrm{Cov}[(Z_1 - Z_2)^2, Z_1 - Z_2] &= \mathbb{E}[(Z_1 - Z_2)^3] \\ &\quad - \mathbb{E}[(Z_1 - Z_2)^2]\,\mathbb{E}[Z_1 - Z_2] \\ &= 0.\end{aligned}$$

Finally,

$$\mathrm{Var}[\Delta] = 32w^2 + 2d\,\mathrm{Var}[(Z_1 - Z_2)^2]$$

Putting everything together:

$$\frac{\mathbb{E}[\Delta]}{\sqrt{\mathrm{Var}[\Delta]}} = \frac{4w^2}{\sqrt{32w^2 + 2d\,\mathrm{Var}[(Z_1 - Z_2)^2]}}.$$

Taking $d \to +\infty$ gives the claim. □