INF442 TD
Clustering high-dimensional data

Leo Liberti
liberti@lix.polytechnique.fr

X 2013
April 24, 2015

Motivation

Developing efficient methods is all about choosing the right machine representation for your data. We all know that a digital picture is encoded as a matrix of pixels, each of which is usually represented by three bytes of memory to represent what we calll "24 bit color depth". So all it boils down to, really, is a very long sequence of binary digits. However, if you picture an image like this, many of the *analog* features of pictures disappear, as they are hidden by the binary encoding.

Your brains see images as analog pieces of data, where colors are perceived as chosen from a continuous palette. In order to represent an image this way, you could think of each pixel as storing a three real numbers (encoding the intensities of red, green and blue components). The image has now conveniently become a matrix of real triplets, or, equivalently, a matrix of reals where the size of each row has been multiplied by three. If you now write the rows (or the columns) one after the other, the image turns into a vector of real numbers, or, in other words, an element of some Euclidean space $\mathbb{R}^n$.

Image databases

Now imagine a large image database, think e.g. of Google Images. You query this database by means of words (encoded as strings), and hope that relevant images will be tagged with your input string. String tags, however, are just words taken out of context. As such, they may be reasonably associated with many different meanings. A query for "spaghetti" makes us all think of pasta, but how about if you were a strictly glutein-free coder assigned to clean up a legacy code from the 1970s? In that case, "spaghetti" would really correspond to "spaghetti code", i.e. those programs involving "GOTO" commands bridging functions, loop or test body boundaries. The reverse may happen: you might look for the terms "spaghetti code" and end up with a bunch of succulent pasta-related photos because the food semantics is prevalent.

This situation calls for the application of the k-means clustering algorithm you saw in earlier lectures: how about clustering the results of your query, and presenting a representative per cluster? Chances are you will save some time by focusing on the most interesting cluster (notice that Google Images does exactly this in the area above the query results).

Setting the stage

Let us assume that all interesting meanings of an image query will appear in the first, say, m ranked items (if we simply consider the first page off of a Google Image query, $m = 10$). Let us further assume that most features of an image are preserved at a 300×300 scaling, and that the color has 24 bit depth. This calls for a vector with $n = 3 \times 300^2 = 2.7 \times 10^5$ components.

Let us try and run k-means on 10 vectors in $\mathbb{R}^{270000}$ using Mathematica (Himg is an array containing the original images):

```
VHimg = Map[Flatten[ImageData[#]] &, Himg];
```

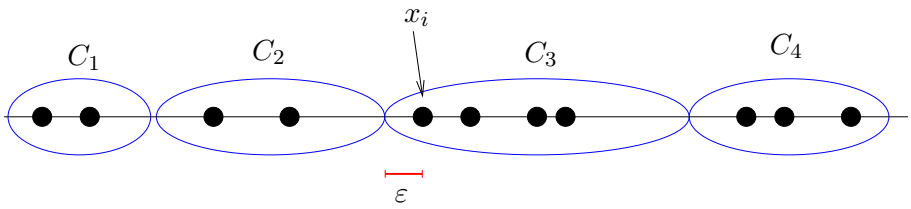
```
VHcl = Timing[ClusteringComponents[VHimg, 3, 1]]

{0.405908, {1, 2, 2, 2, 2, 2, 3, 2, 2, 2, 3}}
```

it takes a little less than half a second. Not too slow, but imagine the scale at which these queries happen: not too fast either. How can we improve on this?

Approximate clustering

Consider any clustering (in k clusters $C_1, \ldots, C_k$) of m vectors $\{x_1, \ldots, x_m\}$ in $\mathbb{R}^n$: pick any x_i and assume it is assigned to cluster C_h. Do you think it is likely that, if we offset x_i by a small $\varepsilon > 0$ in some randomly picked direction, it will change cluster? If you figure the situation in 1D, then x_i should be within ε of the cluster boundary.



How likely is this? Assume the segment of interest has length 1. Since there are k clusters, there are $2(k-1)$ cluster boundaries. So the total length of the boundaries is $2\varepsilon(k-1)$.

Exercise 1. *Prove that the total length of the boundaries is* $2\varepsilon(k-1)$.

Intuitively, we think of a "boundary" to be a very small part of the cluster length. The average cluster length is $1/k$, so ε is assumed to be much smaller than this value, which means that $2\varepsilon(k-1)$ is much smaller than 1. So the probability of a point being on a boundary is very small indeed.

Now consider that the probability is proportional to the volume of the boundaries with respect to that of the clusters, and that the volume grows exponentially in function of length (which is the unit of measure of ε) as n increases. So, whenever the clusters are in some $\mathbb{R}^n$ for large n, the probability that a clustering might be wrong in an ε-approximation is really tiny. In other words, we are justified in looking for approximations.

Exercise 2. *Prove that the volume of hypercubes of side $r > 1$ in $\mathbb{R}^n$ increases exponentially in n; prove the same result for hyperspheres of radius $r > 1$. Find a family of sets of $\mathbb{R}^n$, parametrized by the dimension n, such that their volume grows polynomially in n.*

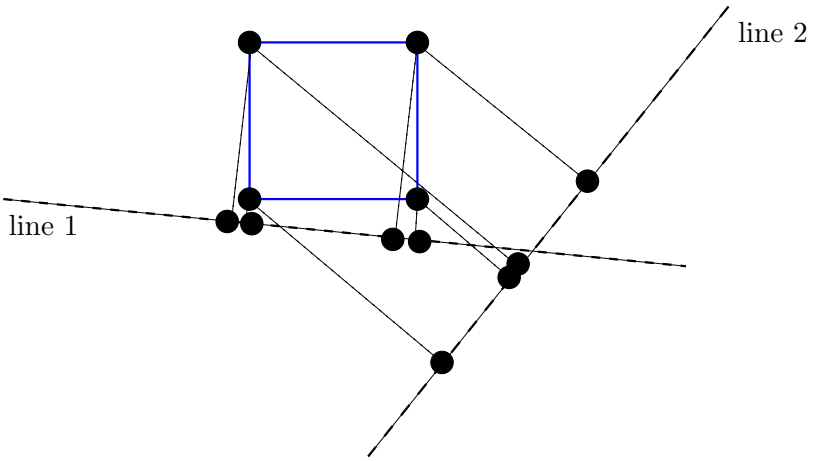
The Euclidean norm

The k-means algorithm proceeds iteratively. At each iteration, it assigns the points to a set of candidate cluster centroids[1] according to shortest Euclidean distance; then it proceeds to re-evaluate the centroids on the basis of the new assignments. It repeats until the assignment keeps changing, or until a pre-determined termination condition is verified. Most of the CPU time is spent computing Euclidean distances, which depends on the dimension n of the embedding space $\mathbb{R}^n$.

A good approximation strategy would therefore be *lose some dimensions but keep the distances approximately equal*. Is this even possible?

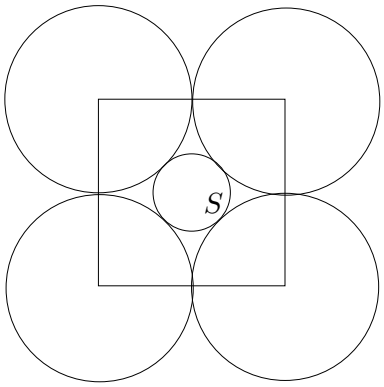
Weirdness in high dimensions

The above suggestion appears preposterous. Take a square in the plane: no matter what line you choose to project it on (so as to "lose dimensions"), it changes the pairwise distances between vertices in a seemingly arbitrary manner.



However, high dimensional geometry is somewhat magic. For example, the maximal sphere inscribed in the unit cube occupies a really tiny volume of the cube in high dimensional spaces. As a more striking example, if you draw the high-dimensional version of the 2D picture below:

[1] See the lecture notes for the definition.



which consists of spheres centered at each vertex of the unit cube with radius equal to 0.5, and a sphere S placed at the center of the cube which touches each of the surrounding spheres, S is *not* always within the unit cube in $\mathbb{R}^n$ for every n; already for $n = 5$ it starts protruding out [3].

Even weirder things happen. The phenomenon which is interesting to us is impossible to visualize and really hard to believe: most of the area of an n-dimensional sphere is concentrated in a tiny band around the equator, *for any equator*. The drawing in Fig. 1 (re-drawn from [3])

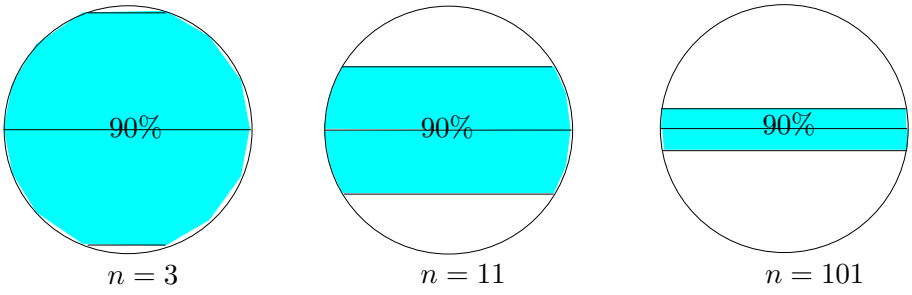


Figure 1: Concentration of measure on the sphere surface.

is a partial visual representation of the *concentration of measure* phenomenon: it emphasizes the shaded regions holding 90% of the total spherical area in different dimensions, but it lacks the insight that this phenomenon holds (concurrently) for any equator.

Surface and volume

Instead of arguing the case for surface, we argue the one for volume, i.e. we claim that most of the volume of a hypersphere in $\mathbb{R}^n$ is concentrated in *equatorial band* of height t (the shaded bands in the circles of Fig. 1 above, with t being the distance from the center of the equatorial disc to the center of the upper band disc), where, for n sufficiently high, t is surprisingly very low indeed.

How does volume help? Intuitively, you might consider that if the claim holds for the volume, then it might hold for surface area too. Let $V_n(r)$ be the volume of the n-dimensional hypersphere

of radius r, and $S_n(r)$ the surface area. Then:

$$\begin{aligned} V_n(r) &= \frac{\pi^{n/2} r^n}{\Gamma(n/2+1)} \\ S_n(r) &= \frac{2\pi^{n/2} r^{n-1}}{\Gamma(n/2)}. \end{aligned}$$

Notationally, we write V_n, S_n whenever $r = 1$. Accordingly, the volume V_n^{dt} and surface area S_n^{dt} of an infinitesimal equatorial band of height dt in an n-dimensional hypersphere of radius 1 are:

$$\begin{aligned} V_n^{dt} &= V_{n-1}(1-t^2)^{\frac{n-1}{2}} dt \\ S_n^{dt} &= S_{n-1}(1-t^2)^{\frac{n-2}{2}} dt. \end{aligned}$$

Their ratio is V_{n-1}/S_{n-1}, which is equal to $\frac{n-1}{4}\sqrt{1-t^2}$.

Exercise 3. *Prove that* $V_{n-1}^{dt}/S_{n-1}^{dt} = \frac{n-1}{4}\sqrt{1-t^2}$.

So V_n^{dt} grows like $O(n\sqrt{1-t^2}S_n^{dt})$, which shows that, as long as we can show that the rate of increase of the equatorial band volume with respect to V_n is more than linear in n for fixed t, our intuition is justified. As it happens, it turns out that this rate of increase is exponential.

We compute the volume of the equatorial band by subtracting the volume of the "polar caps" (drawn in white in the above drawings). Let us compute their n-dimensional volume: let $t_0 \geq 0$ be the distance between the center of the equatorial (great) circle and the small circle which delimits the equatorial band above the equator. Then the volume we are looking for is given by

$$P_n = \int_{t_0}^{1} (1-t^2)^{\frac{n-1}{2}} V_{n-1} dt.$$

Exercise 4. *Prove that the above formula for* P_n *is valid.*

Since V_{n-1} is independent of t, we get

$$P_n = V_{n-1} \int_{t_0}^{1} (1-t^2)^{\frac{n-1}{2}} dt.$$

We compute an upper bound $\bar{P}_n$ for P_n by noticing that $e^x \geq x+1$ for all scalars x, and by evaluating the integral up to ∞:

$$\bar{P}_n = V_{n-1} \int_{t_0}^{\infty} e^{-\frac{n-1}{2}t^2} dt \leq \frac{V_{n-1}}{n-1} e^{-\frac{(n-1)t_0^2}{2}}$$

In actual fact, the integral is *not* a negative exponential, but it can be expressed analytically in function of a *complementary error function* $\mathsf{erfc}(x)$ (see the Wikipedia page `http://en.wikipedia.org/wiki/Error_function`), which has the property that $\mathsf{erfc}(x) \leq e^{-x^2}$.

Finally, we can give an upper bounding estimate $\bar{p}$ for the ratio p between the volume of the polar caps to the total volume of the hypersphere. We have $p = \frac{2P_n}{V_n} \leq \frac{2\bar{P}_n}{V_n} = \bar{p}$, where

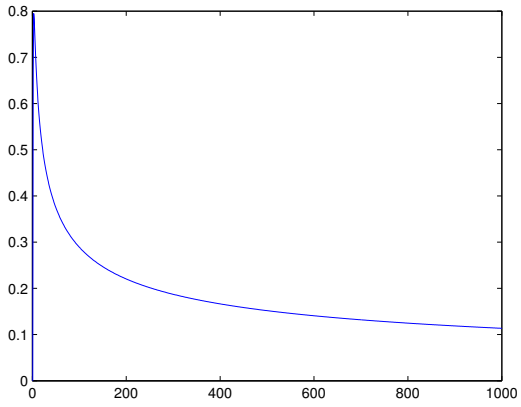
$$\bar{p} = \frac{n}{\sqrt{\pi}} e^{-\frac{n-1}{2}t^2}.$$

Exercise 5. *Prove the validity of the formula for* $\bar{p}$.

The expression for $\bar{p} = \bar{p}(n,t)$ above shows that $\bar{p}$ decreases exponentially as both n and t increase. We can invert $\bar{p}(t)$ and derive $t(\bar{p})$ as:

$$t = \sqrt{\frac{2}{1-n} \ln\left(\frac{\sqrt{\pi}}{n}\bar{p}\right)}.$$

Setting a target $\bar{p}$ value to 0.9, the plot of t (the height of half an equatorial band containing at most 90% of the hypersphere volume) w.r.t. the dimension n is shown below.



This plot substantiates the claim that most of the volume of the hypersphere is concentrated around a narrow equatorial band. As we mentioned at the beginning of this section, this also applies to the surface area.

Low distortion approximations...

Let X be our image database (with $|X| = m$), represented as a finite set of vectors in $\mathbb{R}^n$. Since we mean to "lose dimensions", we are looking for a mapping $A : \mathbb{R}^n \to \mathbb{R}^\ell$, where ℓ is hopefully much smaller than n, such that there is some $\varepsilon > 0$ tolerance ensuring the following condition:

$$\forall x, y \in X \quad (1-\varepsilon)\|x-y\|_2^2 \leq \|Ax - Ay\|_2^2 \leq (1+\varepsilon)\|x-y\|_2^2. \tag{1}$$

In other words, the image of X under A roughly preserves Euclidean pairwise distances. If Eq. (1) holds, A is called a *low distortion approximation* of X.

...are actually very common

It turns out that such approximations are not hard to come by. In fact, in high dimensions, it suffices to sample random $\ell \times n$ matrices, component by component, from a standard normal distribution.

Specifically, we first decide on a tolerance $\varepsilon \in (0, 0.5)$, and, for technical reasons, we set ℓ to the integer rounding of $c\varepsilon^{-2} \log m$, where c is a certain constant (according to [4], $c \approx 1.8$ on the basis of empirical tests). Note that ℓ does not depend on the dimension n of the original space, but

only on ε and on the number of vectors m. It is quadratic in $1/\varepsilon$, but it is logarithmic in m, which means that it scales very well with respect to the growth of the database. For $m = 10$ and $\varepsilon = 0.1$ we obtain[2] $\ell = 432$: a considerable dimensional reduction with respect to $n = 270000$. If we can build such an A, we will certainly be able to reduce our k-means running time considerably.

The Johnson-Lindenstrauss operator

We construct A as follows:

```
void JLOperator(double** A, int l, int n) {
  for(int i = 0; i < l; i++) {
    for(int j = 0; j < m; j++) {
      A[i][j] = StdNormalRandom() / sqrt(l);
    }
  }
}
```

so each component of A is drawn from a standard normal distribution $N(0, 1)$, and then scaled by $\sqrt{\ell}$. A is called a *Johnson-Lindenstrauss operator* because this type of low-distortion approximation was introduced by Johnson and Lindenstrauss [2] as a technical lemma (henceforth called the *Johnson-Lindenstrauss Lemma*) necessary to prove another theorem.

The scaling is due to the same reason that the diagonal in a square is $\sqrt{2}$ and a diagonal in an n-dimensional cube is $\sqrt{n}$. Now let us focus on the normal distribution part. Pick any unit vector u in $\mathbb{R}^n$, and consider $\|Au\|_2^2$. Since u is a unit vector, $\|u\|_2 = 1$. We show that $\mathsf{E}(\|Au\|_2^2)$, the expected value of $\|Au\|_2^2$ with respect to the random variable components of A, is also 1. In fact we claim that $\mathsf{E}(\|Au\|_2^2) = \mathsf{E}(\|u\|_2^2)$.

Let $Au = v = (v_1, \ldots, v_\ell)$. Then for each $i \leq \ell$ we have $v_i = \sum_{j \leq n} A_{ij} u_j$, so

$$\mathsf{E}(v_i) = \sum_{j \leq n} \mathsf{E}(A_{ij} u_j) = \sum_{j \leq n} \mathsf{E}(A_{ij}) u_j = \sum_{j \leq n} 0 u_j = 0,$$

and

$$\mathsf{Var}(v_i) = \sum_{j \leq n} \mathsf{Var}(A_{ij} u_j) = \sum_{j \leq n} \mathsf{Var}(A_{ij}) u_j^2 = \sum_{j \leq n} \frac{u_j^2}{\ell} = \frac{1}{\ell} \|u\|_2^2.$$

Exercise 6. *Prove that* $\mathsf{E}(A_{ij} u_j) = \mathsf{E}(A_{ij}) u_j$ *for each* i, j.

Exercise 7. *Why is* $\mathsf{E}(A_{ij}) = 0$ *for all* i, j*?*

Now we have $\frac{1}{\ell} = \mathsf{Var}(v_i) = \mathsf{E}(v_i^2 - (\mathsf{E}(v_i))^2) = \mathsf{E}(v_i^2 - 0) = \mathsf{E}(v_i^2)$. Hence:

$$\mathsf{E}(\|Au\|_2^2) = \mathsf{E}(\|v\|_2^2) = \mathsf{E}\left(\sum_{i \leq \ell} v_i^2\right) = \sum_{i \leq \ell} \mathsf{E}(v_i^2) = \sum_{i \leq \ell} \frac{\|u\|_2^2}{\ell} = \|u\|_2^2,$$

as claimed.

Exercise 8. *In the paragraph above, we used the fact that* $\mathsf{Var}(v_i) = \frac{1}{\ell}$ *for all* i*. However, we had proved that in fact* $\mathsf{Var}(v_i) = \frac{1}{\ell}\|u\|_2^2$*. Why are these the same?*

[2]The result of the computation is 414; we obtain 432 by using more digits in the decimal expansion of c.

We showed previously that $\bar{p}$ decreases exponentially as e^{-n}. Since the argument works for any equator, we choose the equator given by setting to zero the last component of the unit vector u, namely $u_n = 0$. Then the polar caps area can be expressed as:

$$\mathcal{A}_t^n = \{u \in \mathbb{R}^n \mid \|u\|_2 = 1 \wedge |u_n| \geq t\}.$$

Letting $\mu(\cdot)$ be the spherical surface measure function, we have that $\mu(\mathcal{A}_t^n)$ is $O(e^{-\frac{n-2}{2}t^2})$. Now, however, we are not considering the measure on the unit sphere in $\mathbb{R}^n$, but a different family of sets:

$$\mathcal{B}_t^n = \{u \in \mathbb{R}^n \mid \|u\|_2 = 1 \wedge \left|\|Au\|_2^2 - 1\right| \geq t\}.$$

We claim that the Lebesgue measure λ in $\mathbb{R}^\ell$ is such that $\lambda(\mathcal{B}_t^n)$ is $O(e^{-\frac{\ell-2}{2}t^2})$, and in particular $O(e^{-\ell})$ for fixed t.

Exercise 9. *Prove the claim (this is a difficult exercise).*

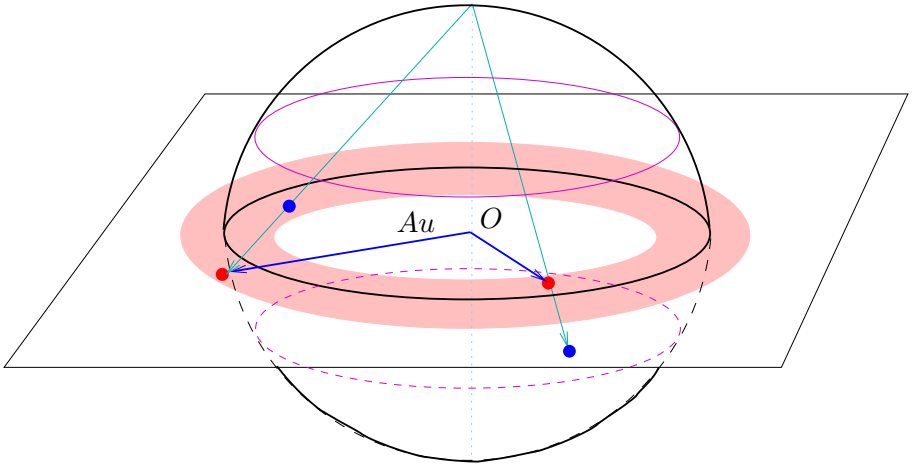


Figure 2: Stereographic projection from the sphere surface to a subspace. The blue dots are on the sphere, the red dots are on the subspace.

At this point we look back at Fig. 1, where the equator is the mean, and the area in the band around it is a probability of sampling points within a band of thickness t of the mean. If we want a 90% chance of sampling points at most t far from the mean, we notice that, as n increases, we can stay closer and closer to the mean. In other words, the variance of this distribution gets smaller and smaller as n grows. The distribution in question is over the spherical surface, but by the above argument we can map it onto the relevant distribution over the projected subspace W. We now remark that an equatorial band of height t on the sphere corresponds to a band of width t around $\|Au\|_2^2 = 1$. Thus, concentration of measure on the sphere implies that $\|Au\|_2^2$ is sharply concentrated around its mean.

Since A is a matrix, its action on X is linear. Hence, for $x, y \in X$ we have:

$$\|Ax - Ay\|_2^2 = \|A(x-y)\|_2^2 = \|x-y\|_2^2 \|Au\|_2^2 \tag{2}$$

where $u = \frac{x-y}{\|x-y\|_2}$, which ensures $\|u\|_2 = 1$. By our reasoning above, the distribution of $\|Ax - Ay\|_2^2$ is very close to its mean $\|x-y\|_2^2$ with high probability.

Exercise 10. *Why are we claiming that* $E(\|Ax - Ay\|_2^2) = \|x - y\|_2^2$*?*

The technical choice of ℓ (obtained by a reasoning not dissimilar to the derivation of the formula for $\bar{p}$, but with different bounding techniques) ensures that Eq. (1) holds with probability more or less $1 - e^{\ell/\varepsilon^2}$, which tends to 1 as ℓ increases, assuming ε is constant.

Random sampling from a standard normal distribution

We still have to explain how to implement the function `StdNormalRandom()` in the `JLOperator()` code given above. The trick is to sample from two independently uniformly distributed random variables U_1, U_2 from $(0, 1)$ and, via a transformation to polar coordinates, obtain two independently normally distributed random variables Z_0, Z_1. The following picture, obtained from the Wikipedia page for the *Box-Muller transform*, illustrates the change of coordinates graphically, and shows that the marginal distributions are normal. In Fig. 3, the samples from U_1, U_2 are drawn in the unit

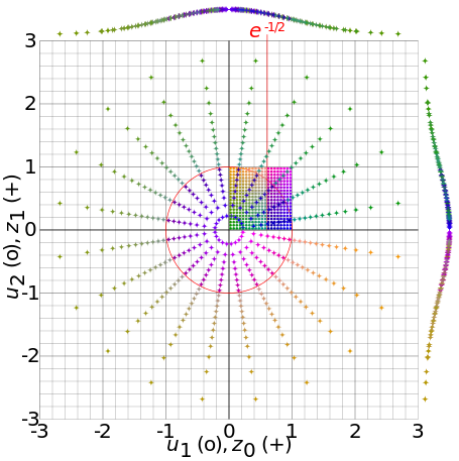


Figure 3: Picture taken from Wikipedia ⟨`http://en.wikipedia.org/wiki/Box-Muller_transform`⟩.

square cornered at the origin, and marked with different colors. Each of these points are mapped (with the same color) through the two coordinate changes[3]:

$$Z_1 = R\cos\Theta = \sqrt{-2\ln U_1}\cos(2\pi U_2) \quad (3)$$

$$Z_2 = R\sin\Theta = \sqrt{-2\ln U_1}\sin(2\pi U_2), \quad (4)$$

where R, Θ are the variable names for the intermediate coordinate change $R^2 = -2\ln U_1$ and $\Theta = 2\pi U_2$.

Exercise 11. *Show that* Z_0, Z_1 *are independent normally distributed random variables.*

So in order to yield a sample from $N(0, 1)$ we really must compute two samples from two identically independently normally distributed random variables Z_1, Z_2.

[3]Note that Z_1 is shown as z_0 and Z_2 as z_1 in the picture.

In practice

In practice, the overall procedure is:

1. sample the components A_{ij} of A from a standard normal distribution $N(0,1)$ as described above, using a uniform pseudorandom number generator;
2. multiply A by the scalar $\frac{1}{\sqrt{\ell}}$;
3. compute the smaller-dimensional set $Y = AX \subseteq \mathbb{R}^\ell$ and either trust that the probability guarantee holds, or verify that Eq. (1) holds;
4. if it does not, try again with another randomly sampled matrix A;
5. compute the projected point set $Y = AX$;
6. run k-means on Y.

As shown below (again, using Mathematica), the clustering is the same, but it is computed in a fraction of the time.

```
Get["Projection.m"];
VKimg = JohnsonLindenstrauss[VHimg, 0.1];
VKcl = Timing[ClusteringComponents[VKimg, 3, 1]]

{0.002232, {1, 2, 2, 2, 2, 2, 3, 2, 2, 2, 3}}
```

Extensions

Sparser projection matrices

Achlioptas [1] proved that it suffices to sample projection matrix coefficients A_{ij} from very coarse approximations of the normal distribution. For example, the following is shown to work (with some minor differences in the definition of ℓ):

$$A_{ij} = \sqrt{\frac{3}{\ell}} \begin{cases} +1 & \text{with probability } \frac{1}{6} \\ 0 & \text{with probability } \frac{2}{3} \\ -1 & \text{with probability } \frac{1}{6}. \end{cases} \tag{5}$$

Such a choice yields a very sparse A, which in turn yields computational savings in the matrix times vector multiplication phase.

Other types of projection

A very common type of projection method is Principal Component Analysis (PCA). It should be seen as a heuristic, since it does not provide an approximation ratio. The aim of PCA is to project a set of points in a certain Euclidean space to a Euclidean space of given dimension ℓ. So the input to PCA is a set X of m points in $\mathbb{R}^n$, and the target dimension $\ell \in \mathbb{N}$.

Exercise 12. *Describe the PCA algorithm (look for information online), justifying each step.*

References

[1] Dimitris Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary choins. *Journal of Computer and System Sciences*, 66:671–687, 2003.

[2] W. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In G. Hedlund, editor, *Conference in Modern Analysis and Probability*, volume 26 of *Contemporary Mathematics*, pages 189–206, Providence, 1984. AMS.

[3] J. Matoušek. *Lectures on Discrete Geometry*. Springer, New York, 2002.

[4] S. Venkatasubramanian and Q. Wang. The Johnson-Lindenstrauss transform: An empirical study. In *Algorithm Engineering and Experiments*, volume 13 of *ALENEX*, pages 164–173, Providence, 2011. SIAM.