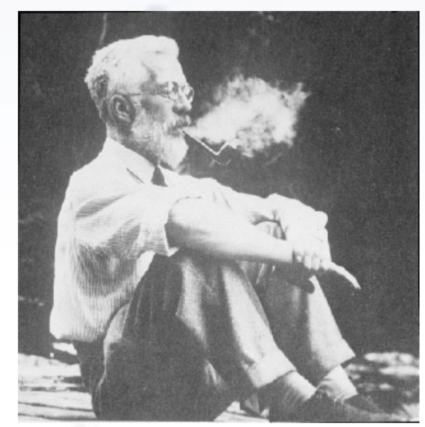
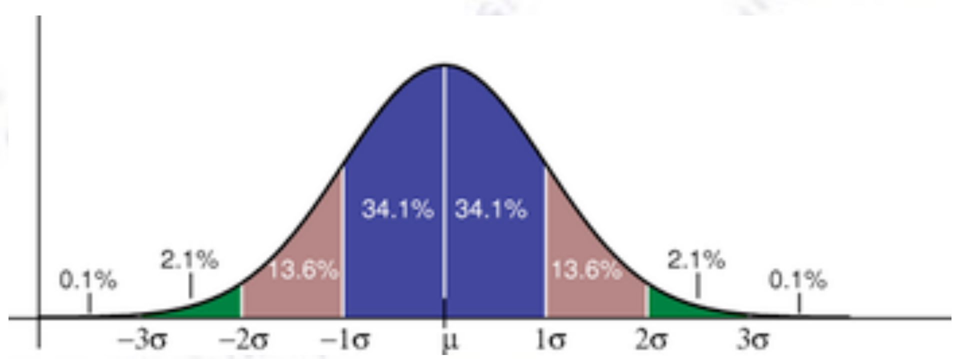
# Applied Machine Learning

## The t-SNE and UMAP dimensionality reduction algorithms

Troels C. Petersen (NBI)

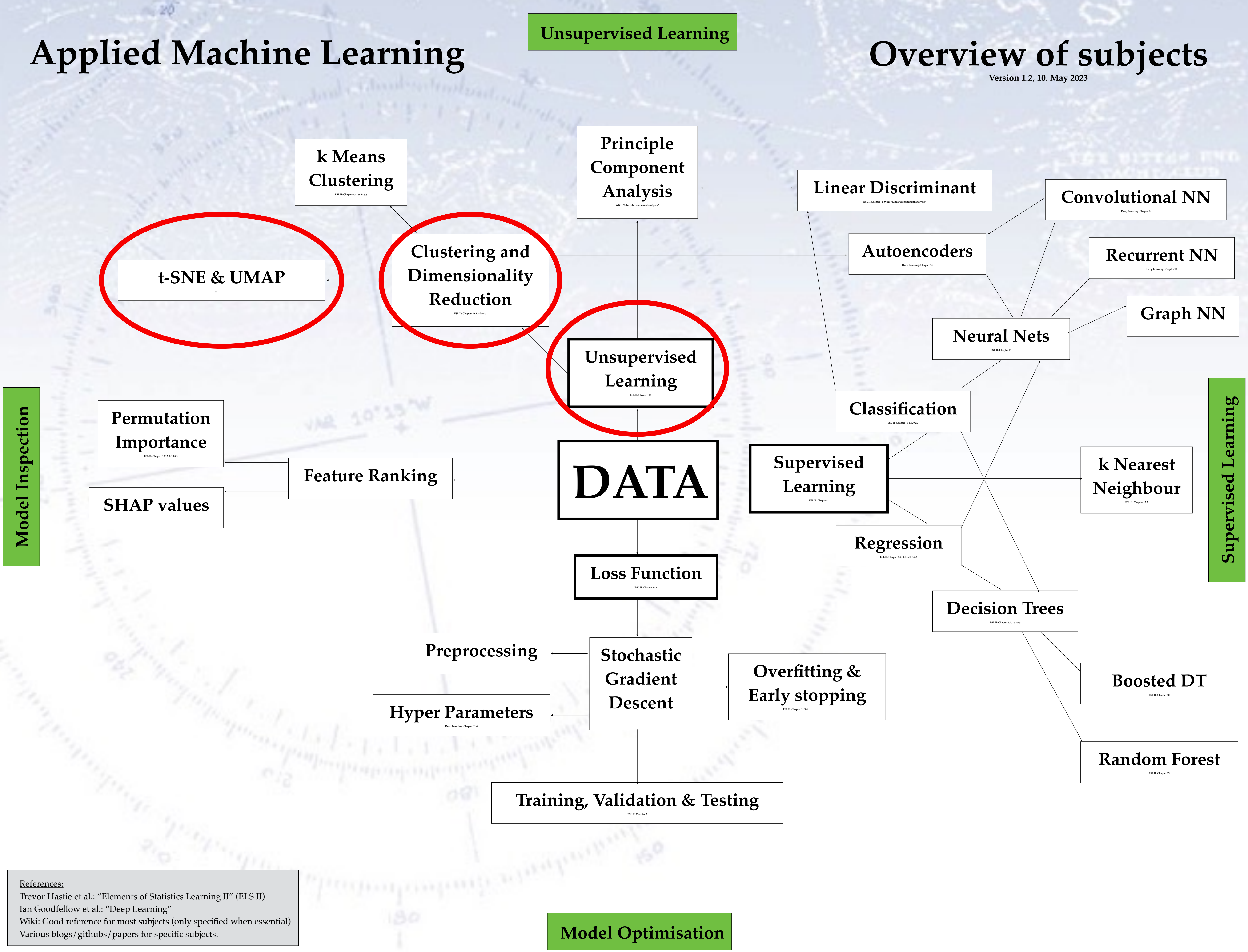



*"Statistics is merely a quantisation of common sense - Machine Learning is a sharpening of it!"*

# Applied Machine Learning

# Overview of subjects

Version 1.2, 10. May 2023





References:
Trevor Hastie et al.: "Elements of Statistics Learning II" (ELS II)
Ian Goodfellow et al.: "Deep Learning"
Wiki: Good reference for most subjects (only specified when essential)
Various blogs/githubs/papers for specific subjects.

# Supervised Vs. Unsupervised

Data is great, and data **with labels** is even greater.

But what do you do, when there are no labels?

As we saw earlier, one possible solution is to cluster the data. This may or may not work, and the hard part is knowing if it does.

"How can I verify my clustering on this 13-dimensional data, when there is no way of plotting it?", as a very relevant question asked.

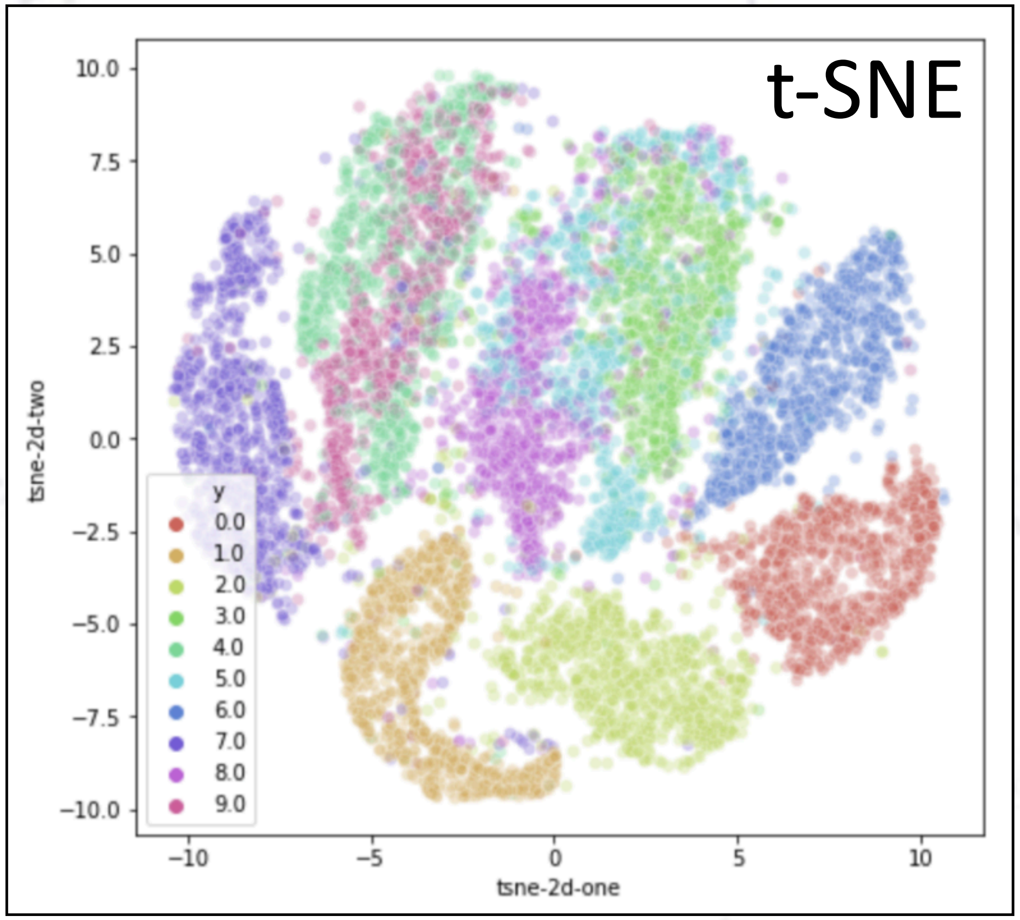
The answer is **dimensionality reduction**, i.e. taking high dimensional data, and projecting it into fewer (2?) dimensions, **without loosing all the information contained in the higher dimensions**.

PCA is a linear version of this, but there are also non-linear methods.

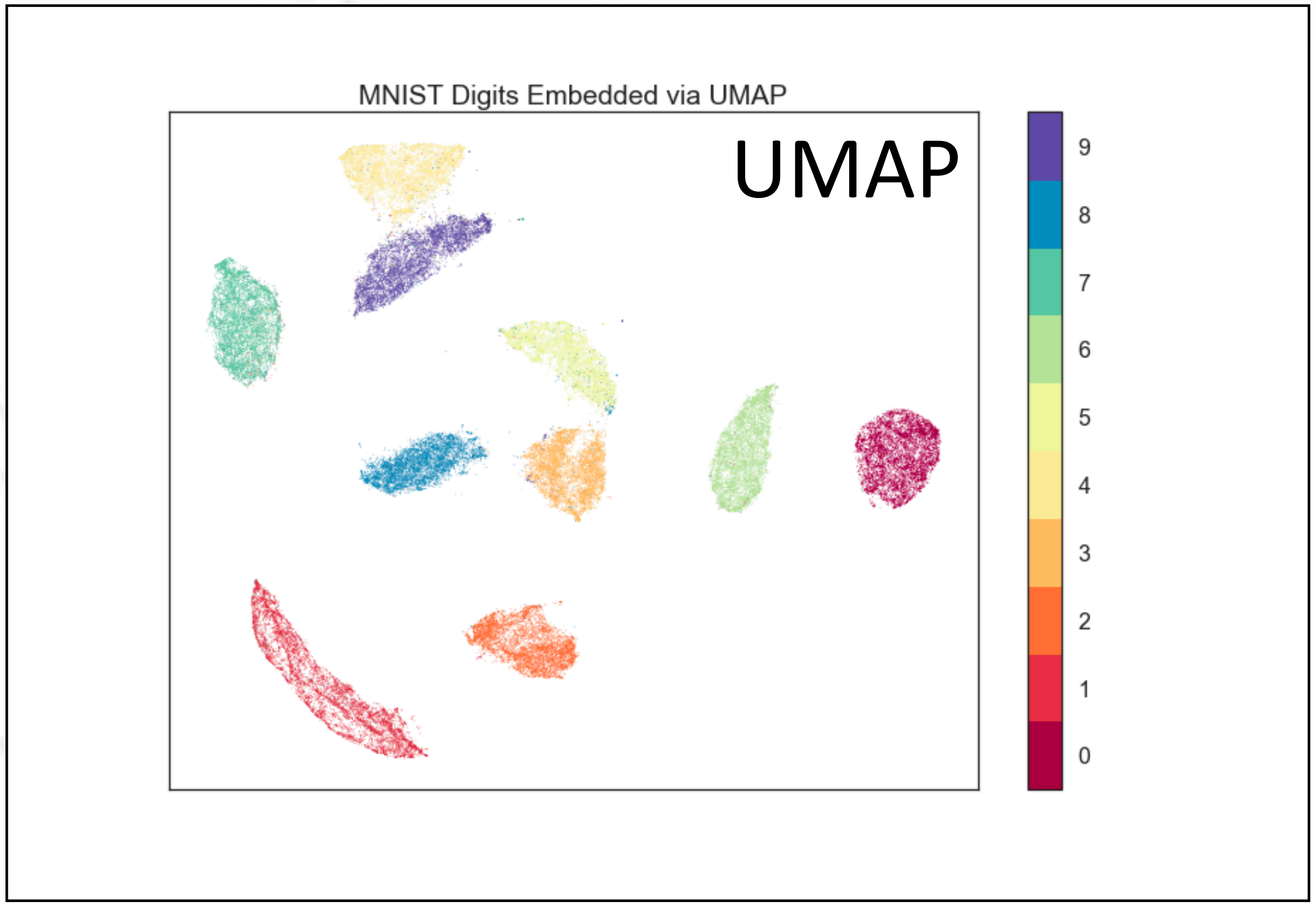
# t-SNE & UMAP

High dimensionality has always been a curse - it is extremely hard to make sense of, and requires a lot of work and domain knowledge to boil down to few dimensions without loosing a lot of information.

PCA has long reigned the linear case, and k-means the clustering, but two new(er) non-linear and powerful candidates are around: t-SNE and UMAP. Below are their performance on the MNIST data set.




Source: Towards data science (PCA and t-SNE)




Source: UMAP GitHub page: https://github.com/lmcinnes/umap

# t-SNE Pro's and Con's

**Pro:** In the words of the t-Distributed stochastic neighbour embedding (t-SNE) paper, the t-SNE algorithm… "…*minimises the divergence between two distributions: a distribution that measures pairwise similarities of the input objects and a distribution that measures pairwise similarities of the corresponding low-dimensional points in the embedding*".

The great thing about this is, that there are no assumptions about distributions, relationships, or number of clusters. The algorithm is non-linear, which gives it a clear edge over e.g. PCA.

**Con:** However, computationally it is a "heavy" (ugly?) algorithm, since t-SNE scales **quadratically** in the number of objects N. This limits its applicability to data sets with only a few thousand input objects; beyond that, learning becomes too slow to be practical (and the memory requirements become too large)".

In real life, the t-SNE algorithm has especially had its impact in (a)DNA research, where the number of cases is typically not that large.

# UMAP

UMAP builds on using **Riemannian manifolds!** Within differential geometry, this allows the definition of angles, hyper-area, and curvature in high dimensionality.


**Abstract**

UMAP (Uniform Manifold Approximation and Projection) is a novel manifold learning technique for dimension reduction. UMAP is constructed from a theoretical framework based in Riemannian geometry and algebraic topology. The result is a practical scalable algorithm that is applicable to real world data. The UMAP algorithm is competitive with t-SNE for visualization quality, and arguably preserves more of the global structure with superior run time performance. Furthermore, UMAP has no computational restrictions on embedding dimension, making it viable as a general purpose dimension reduction technique for machine learning.


UMAP paper, arXiv 1802.03426, Sep. 2020

The paper is quiet mathematical with (10) definitions, lemmas, and proofs in the appendix. I find it a bit hard to read, but like their discussion of scaling and cons.

# UMAP

As in the t-SNE case, UMAP tries to find a metric in both the original (large) space X, and the lower dimension output space Y, which can be (topologically) matched:

> At a high level, UMAP uses local manifold approximations and patches together their local fuzzy simplicial set representations to construct a topological representation of the high dimensional data. Given some low dimensional representation of the data, a similar process can be used to construct an equivalent topological representation. UMAP then optimizes the layout of the data representation in the low dimensional space, to minimize the cross-entropy between the two topological representations.

UMAP paper, arXiv 1802.03426, Sep. 2020

However, the metrics in X and Y used by UMAP and t-SNE differ:

For t-SNE these metrics are as follows:

$$v_{j|i} = \exp(-\left\|x_i - x_j\right\|_2^2 / 2\sigma_i^2)$$

$$w_{ij} = \left(1 + \left\|y_i - y_j\right\|_2^2\right)^{-1}$$

For UMAP they are:

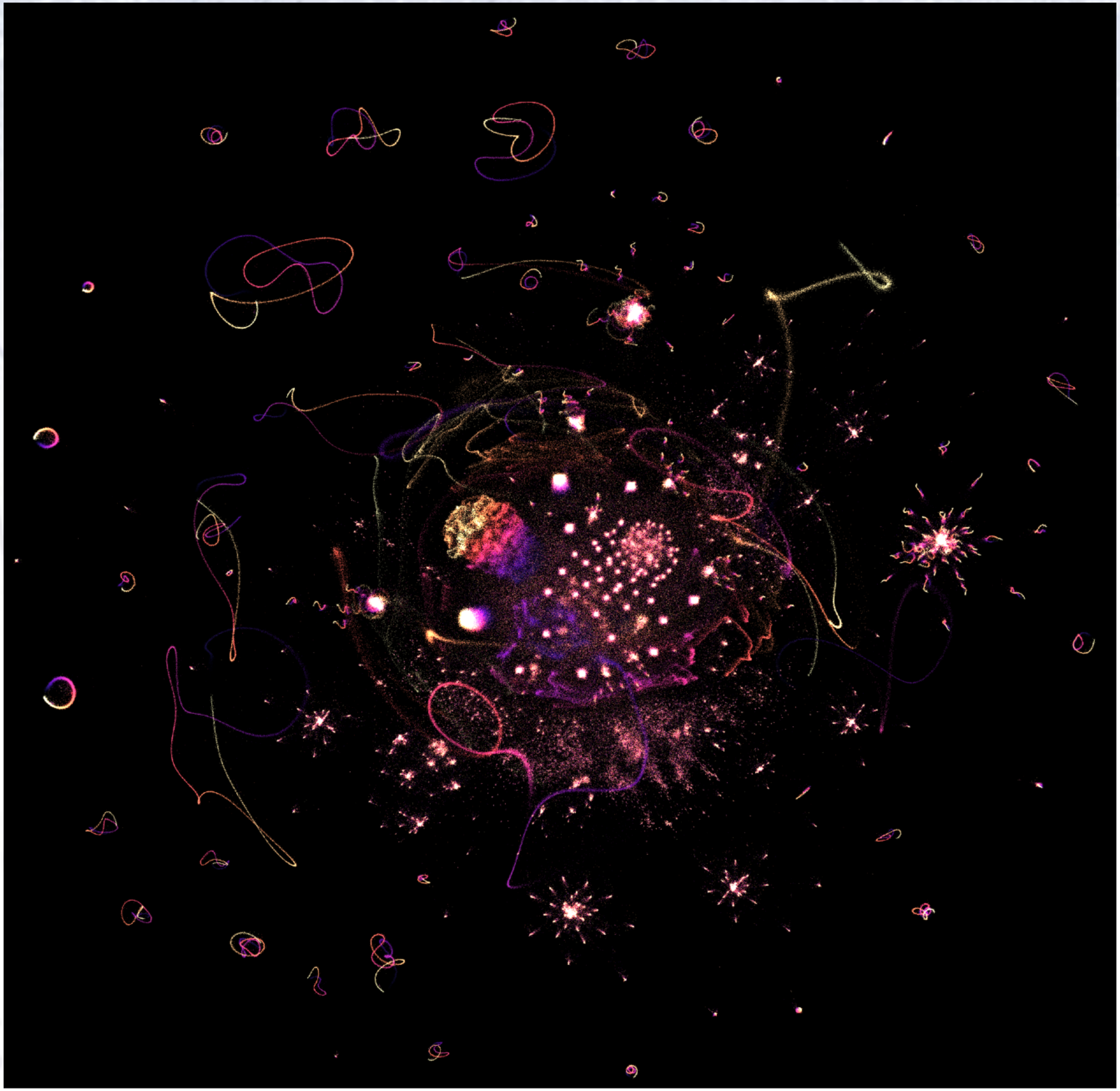
$$v_{j|i} = \exp[(-d\left(x_i, x_j\right) - \rho_i)/\sigma_i]$$

$$w_{ij} = \left(1 + a\left\|y_i - y_j\right\|_2^{2b}\right)^{-1}$$

# First million integers in UMAP

Prime factorising the first million integers, and drawing them (artfully) gives the following image.

I find it quite visually pleasing, and a cool interplay between mathematics, ML, and art.

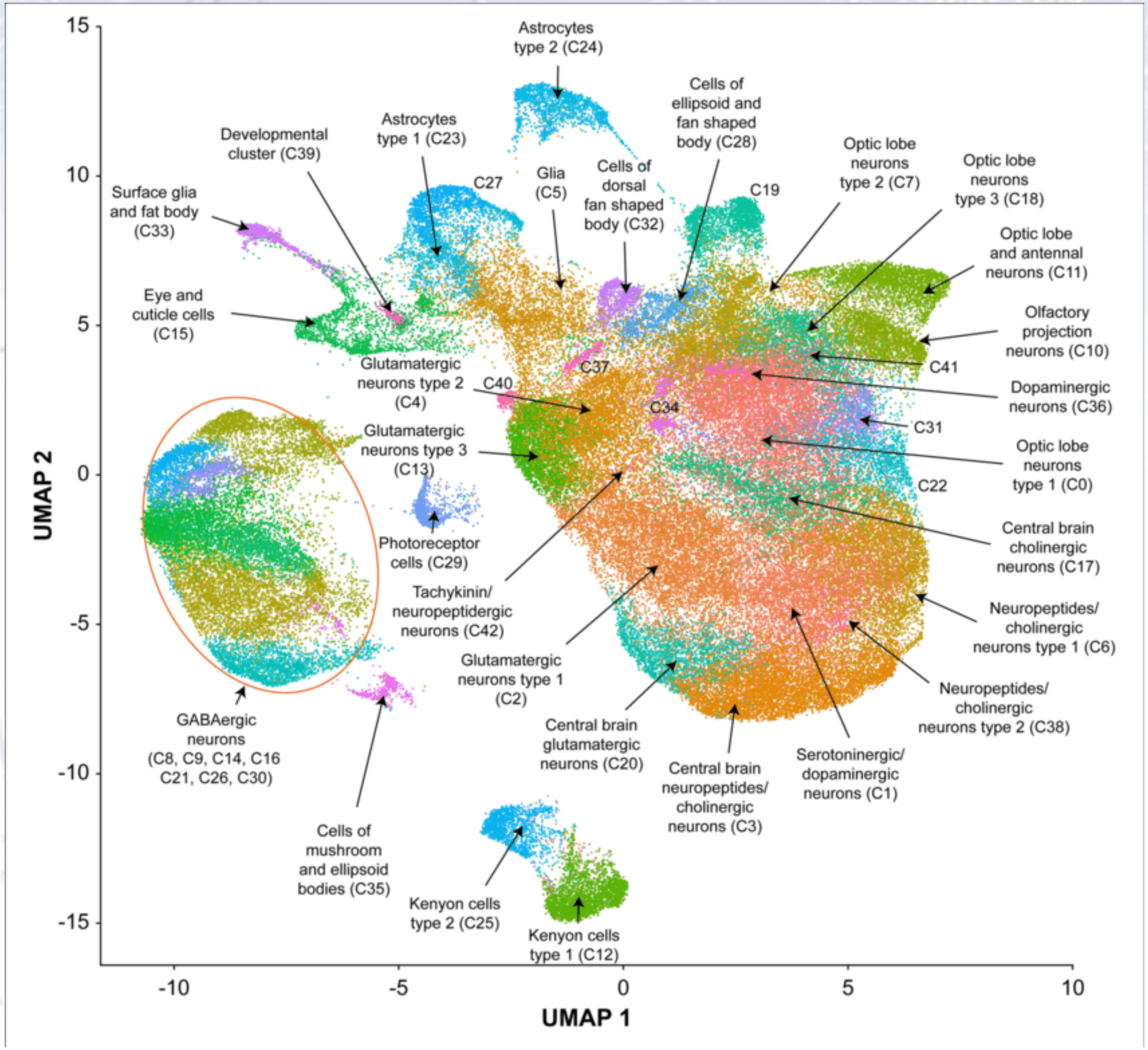


# Example use cases…

# Differentiating cells types

UMAP of different cell types. The labelling comes from known cells, but might be based on very little data.

The unsupervised clustering gives a quite clear pattern, and ability to determine cell type without having a large training sample.

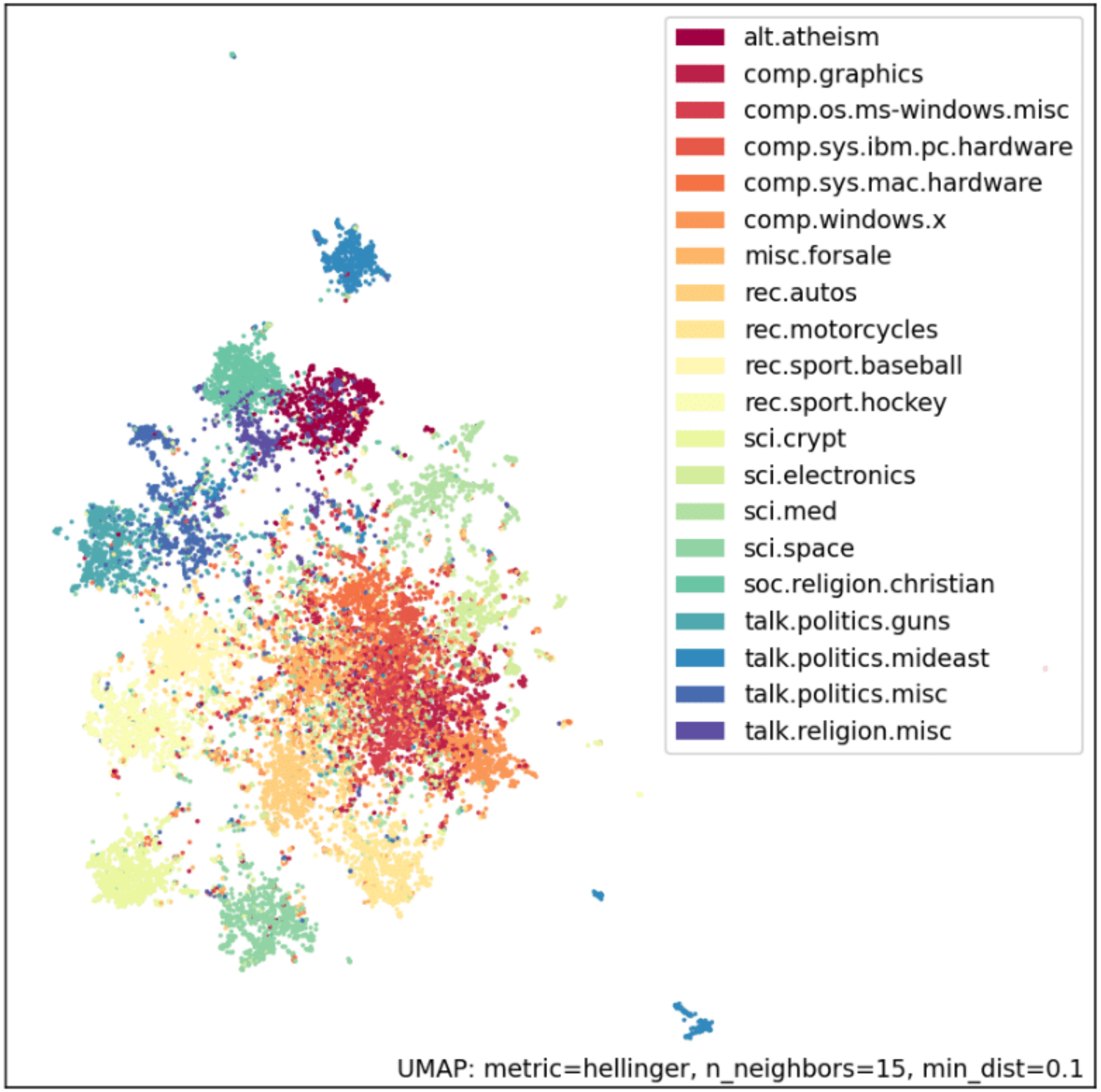



From: Developmental Alcohol Exposure in Drosophila: Effects on Adult Phenotypes and Gene Expression in the Brain

# Mapping news group discussions

UMAP showing the differences between different news group discussion fora.

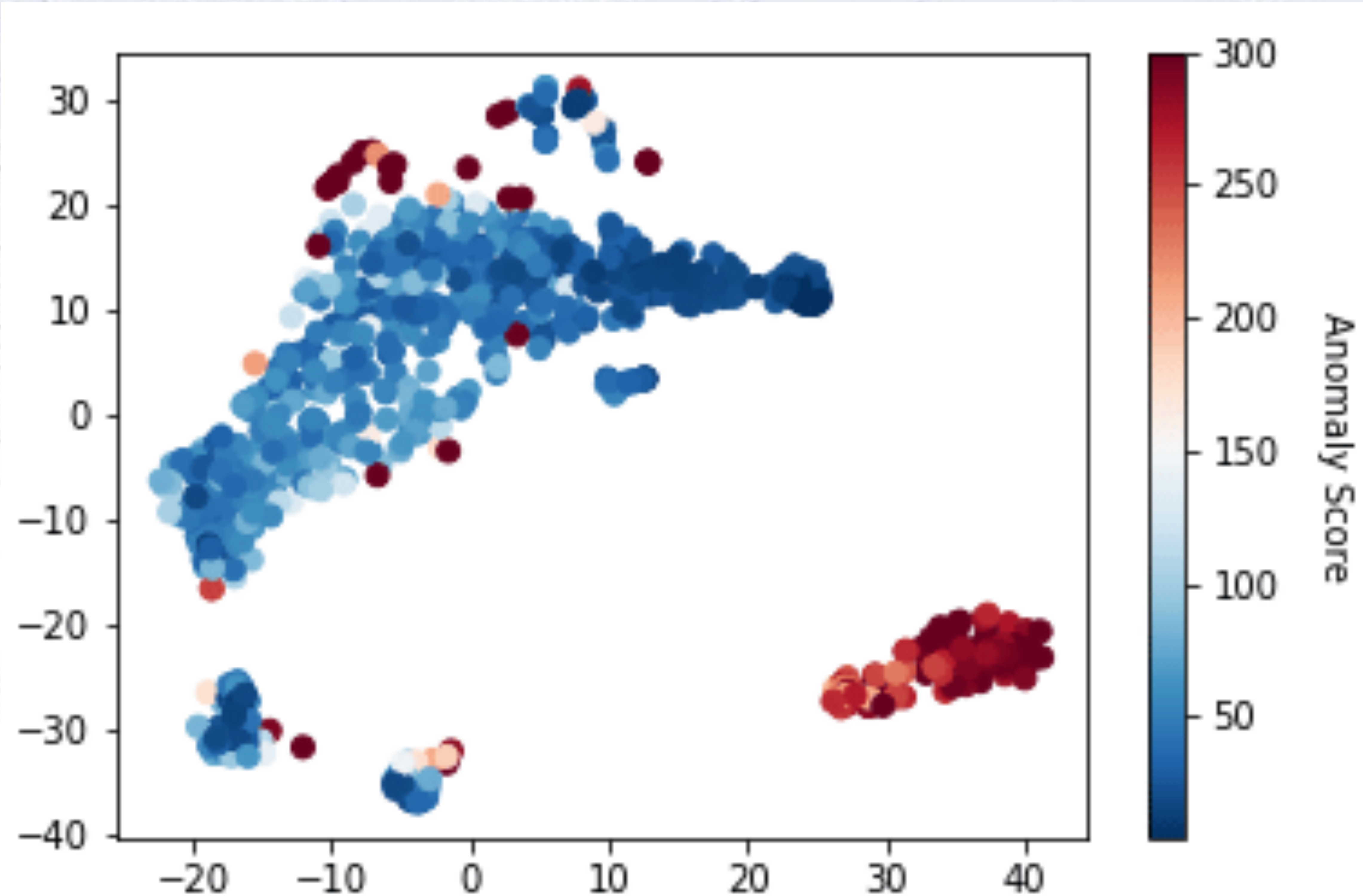
The ability to cluster fairly well would allow editors to direct text to the relevant news group.




From: Vec2GC -- A Graph Based Clustering Method for Text Representations

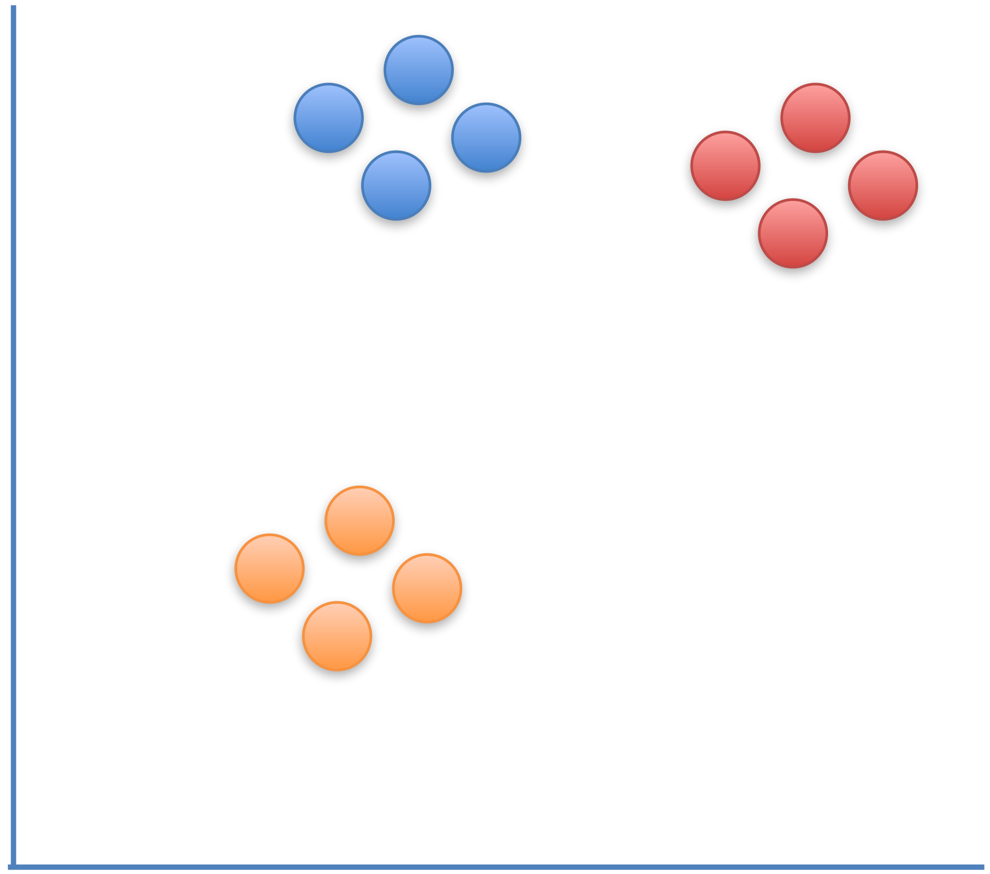
# Anomaly Detection (t-SNE)

With the below map, it is not hard to divide the data into at least 4-6 groups…



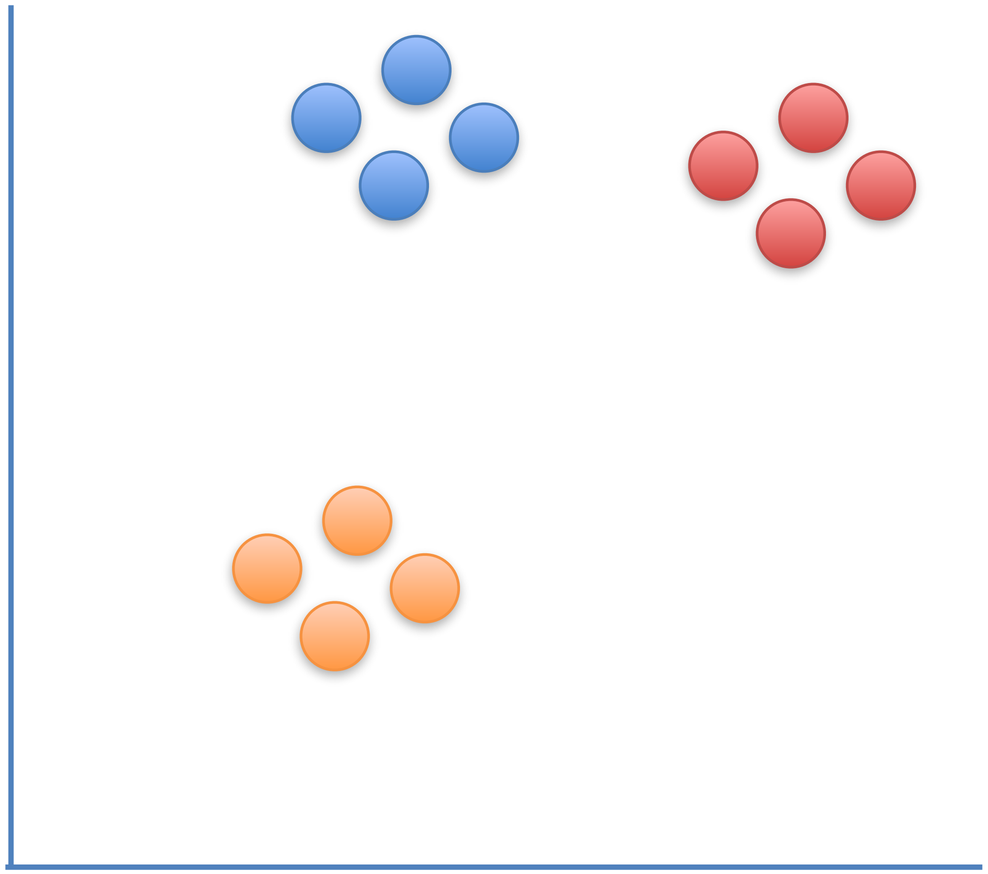
# Details of t-SNE

Here's a basic 2-D scatter plot.



Here's a basic 2-D scatter plot.

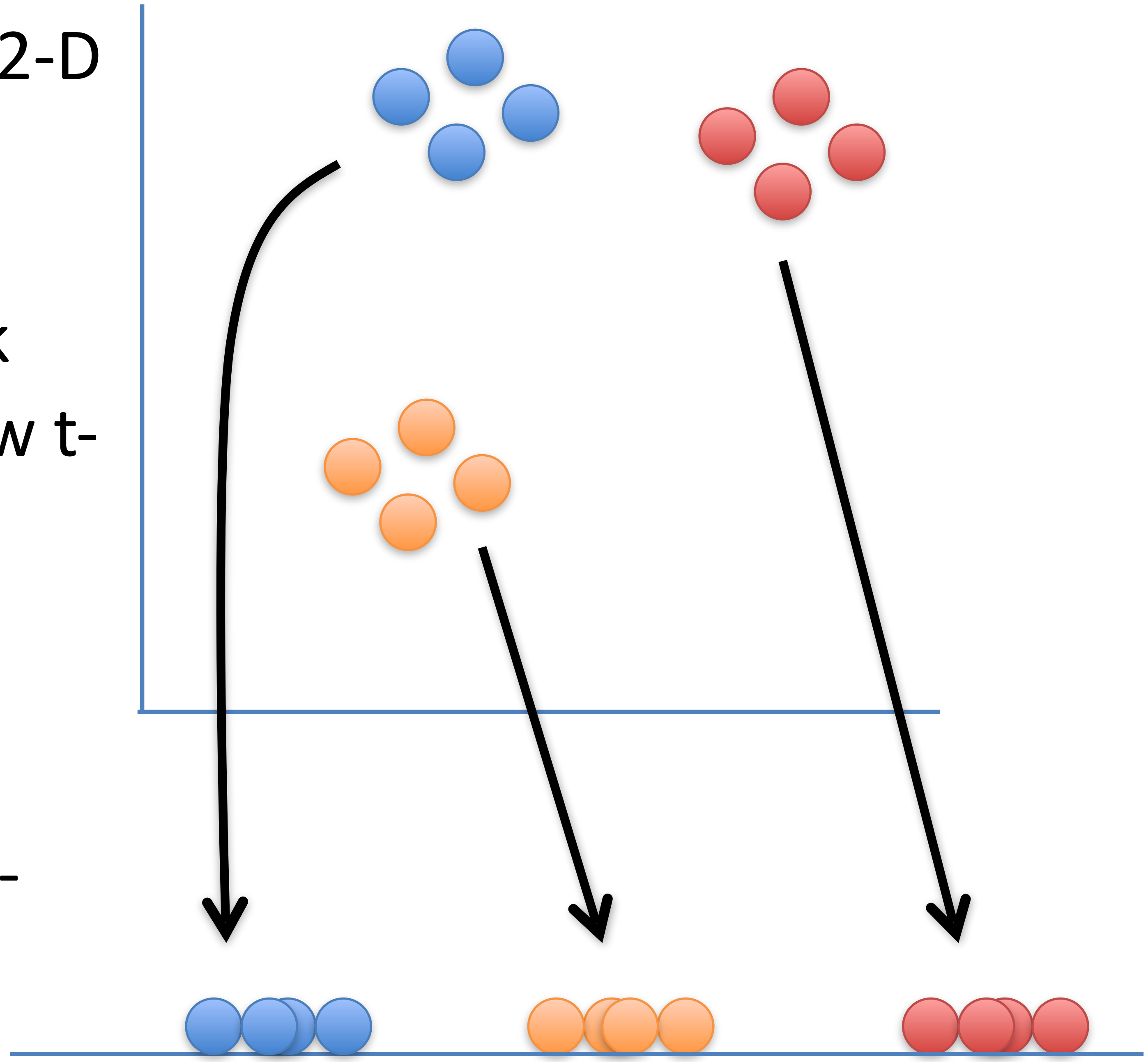
Let's do a walk through of how t-SNE would transform this graph…

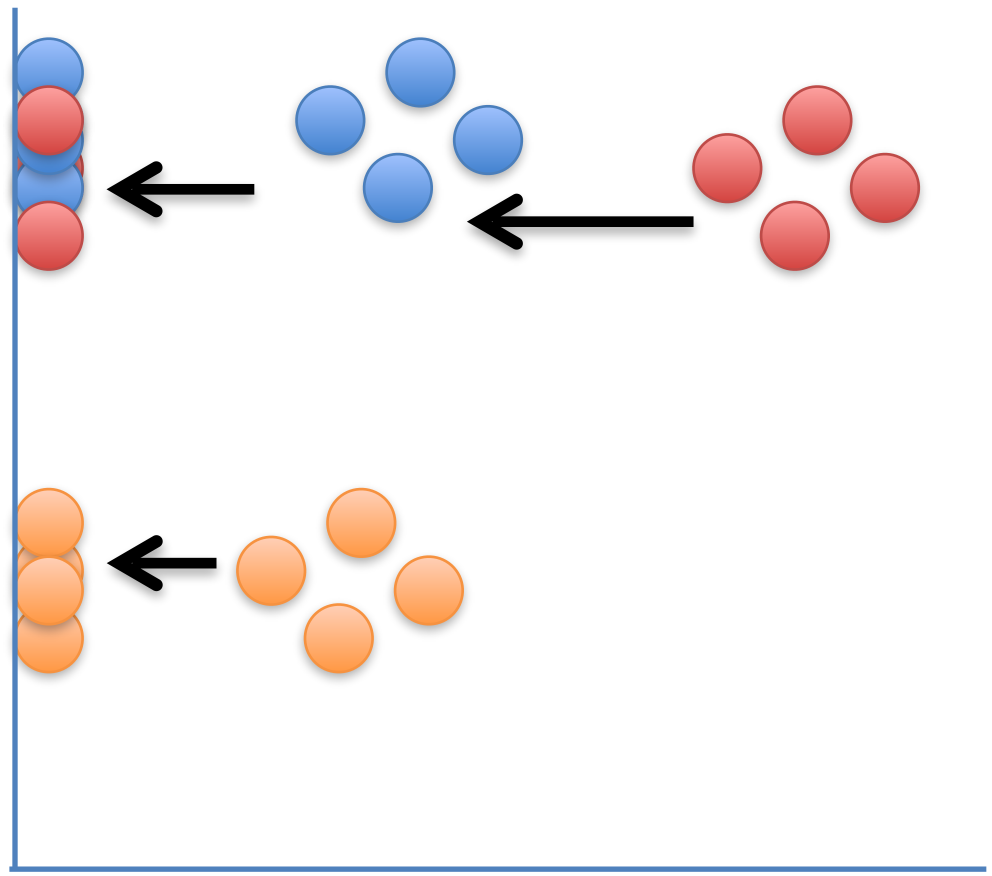


Here's a basic 2-D scatter plot.

Let's do a walk through of how t-SNE would transform this graph…

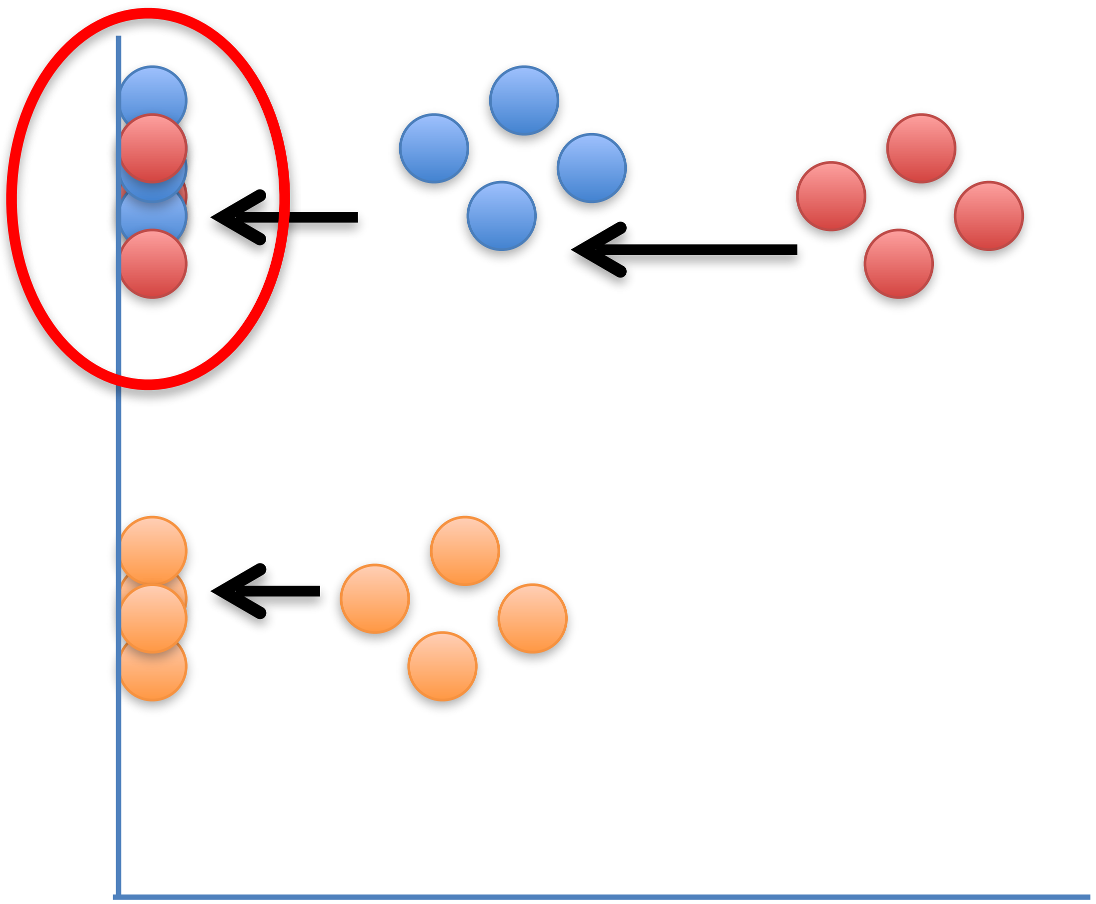
…into a flat, 1-D plot on a number line.

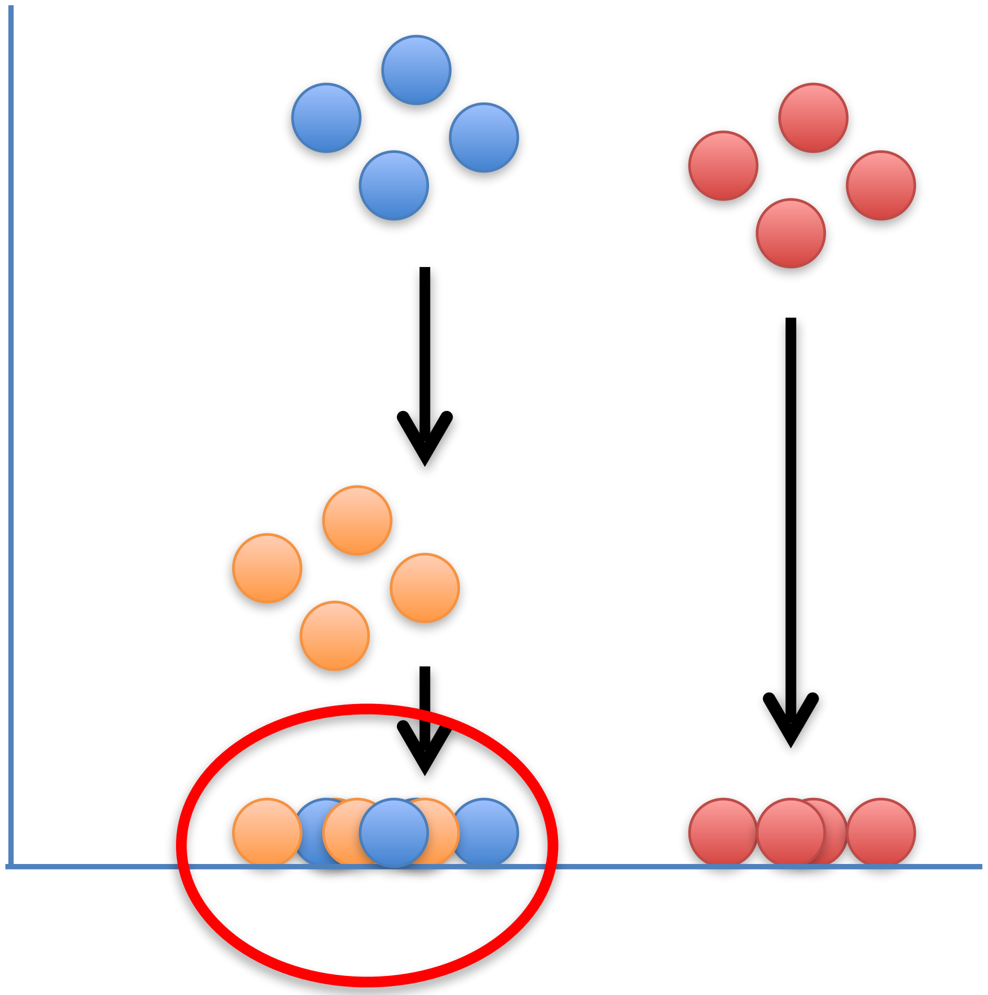




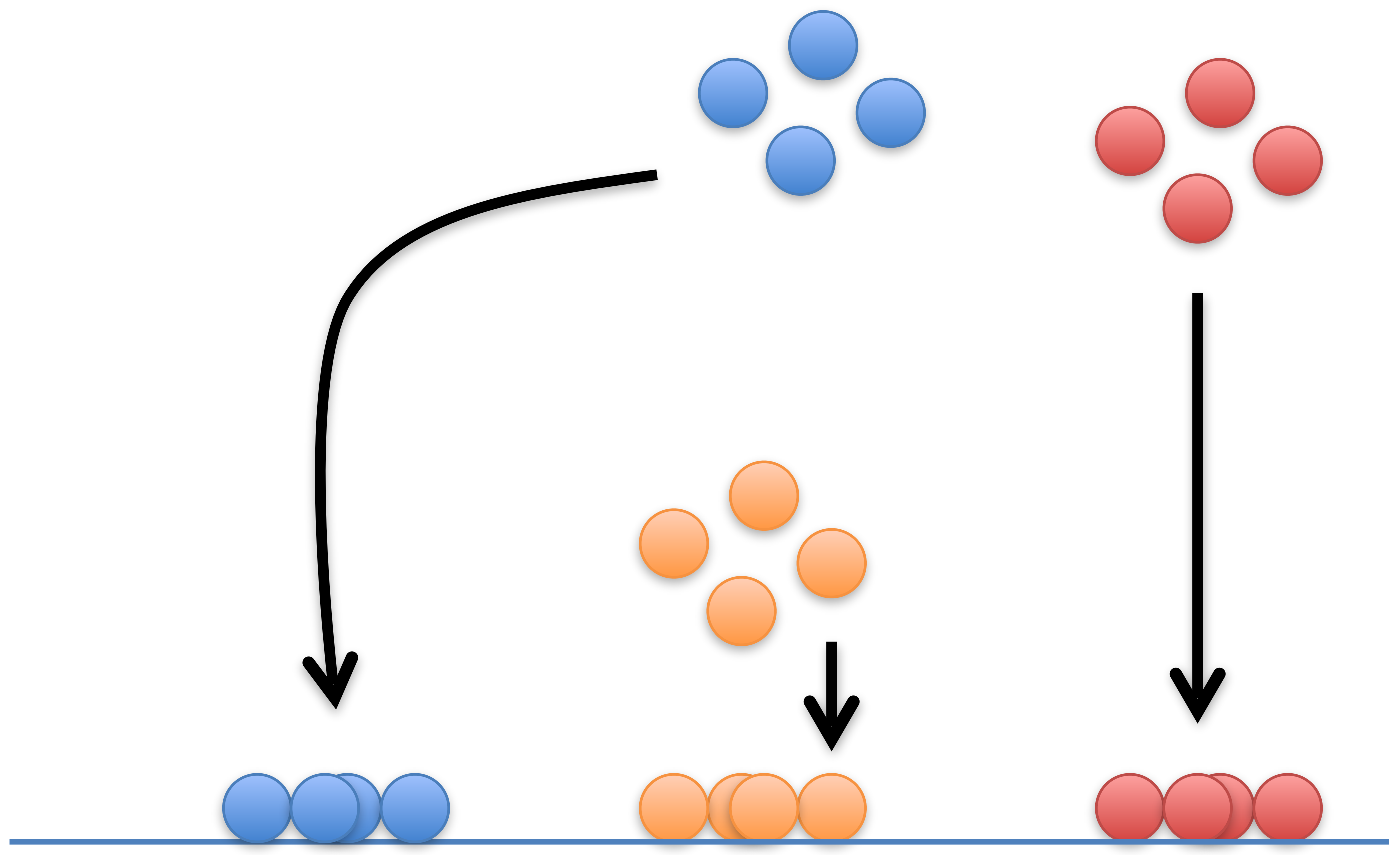
NOTE: If we just projected the data onto one of the axes, we'd just get a big mess that doesn't preserve the original clustering.

Instead of two distinct clusters, we just see a mishmash.

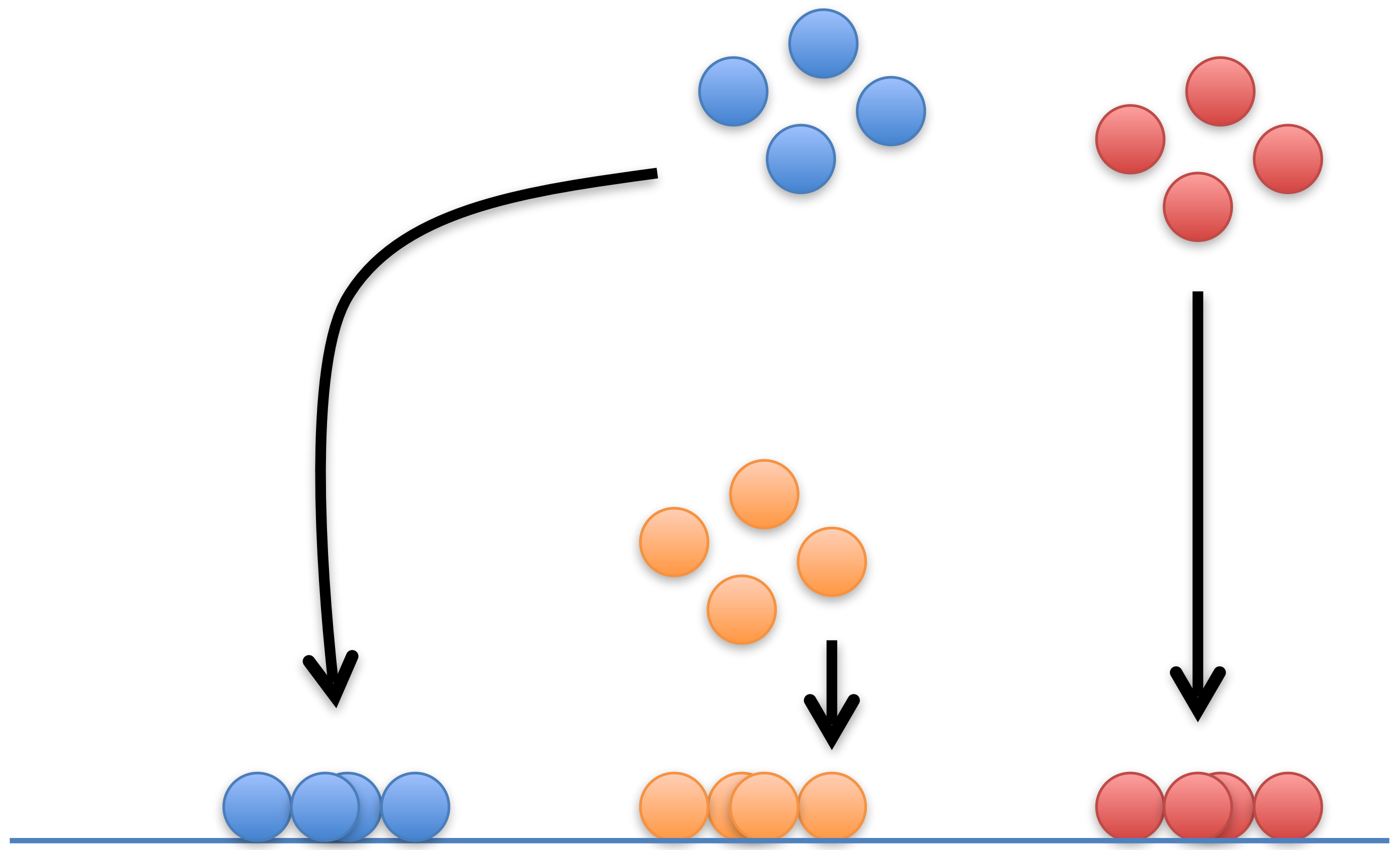




Same here...

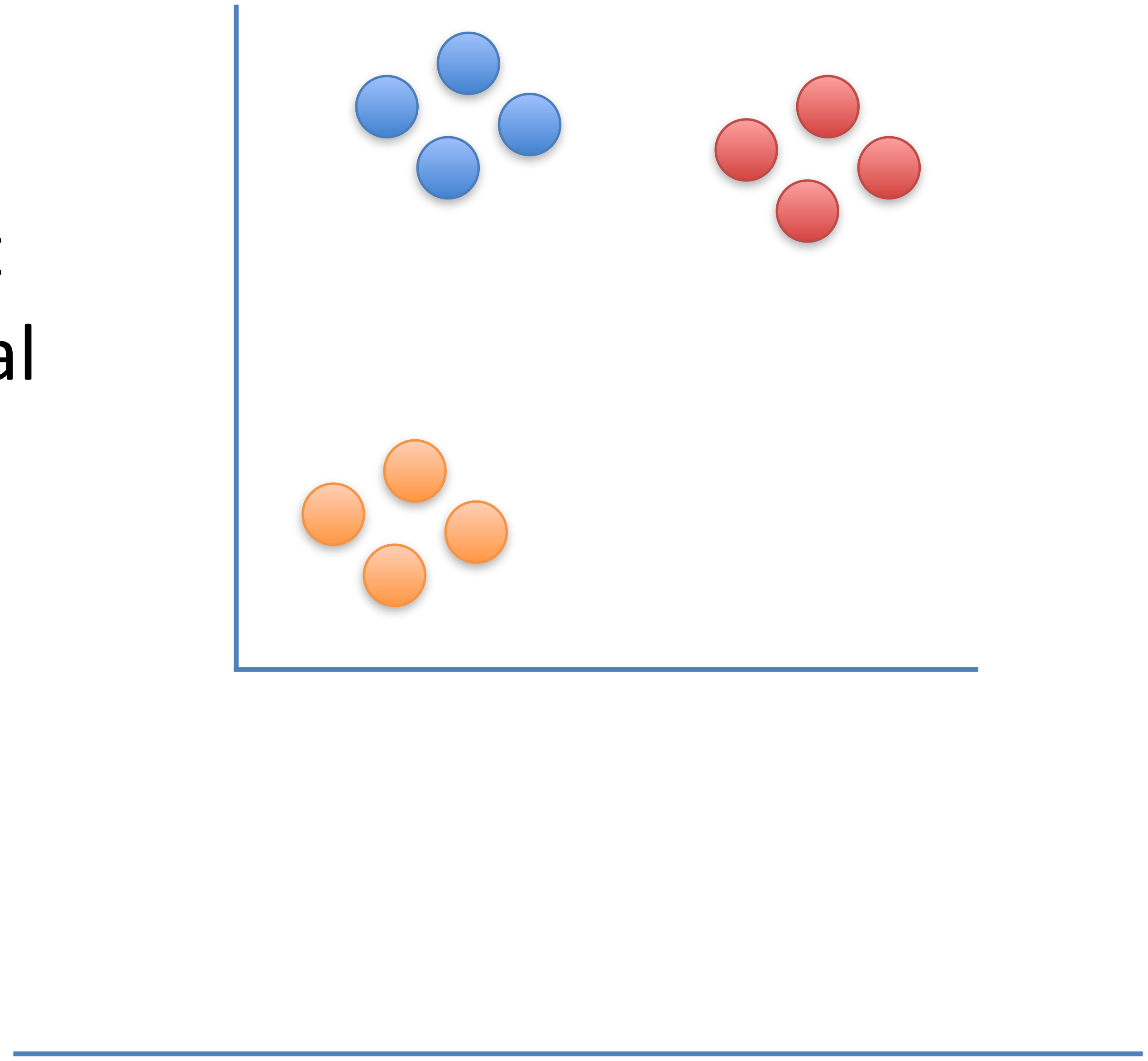


What t-SNE does is find a way to project data into a low dimensional space (in this case, the 1-D number line) so that the clustering in the high dimensional space (in this case, the 2-D scatter plot) is preserved.



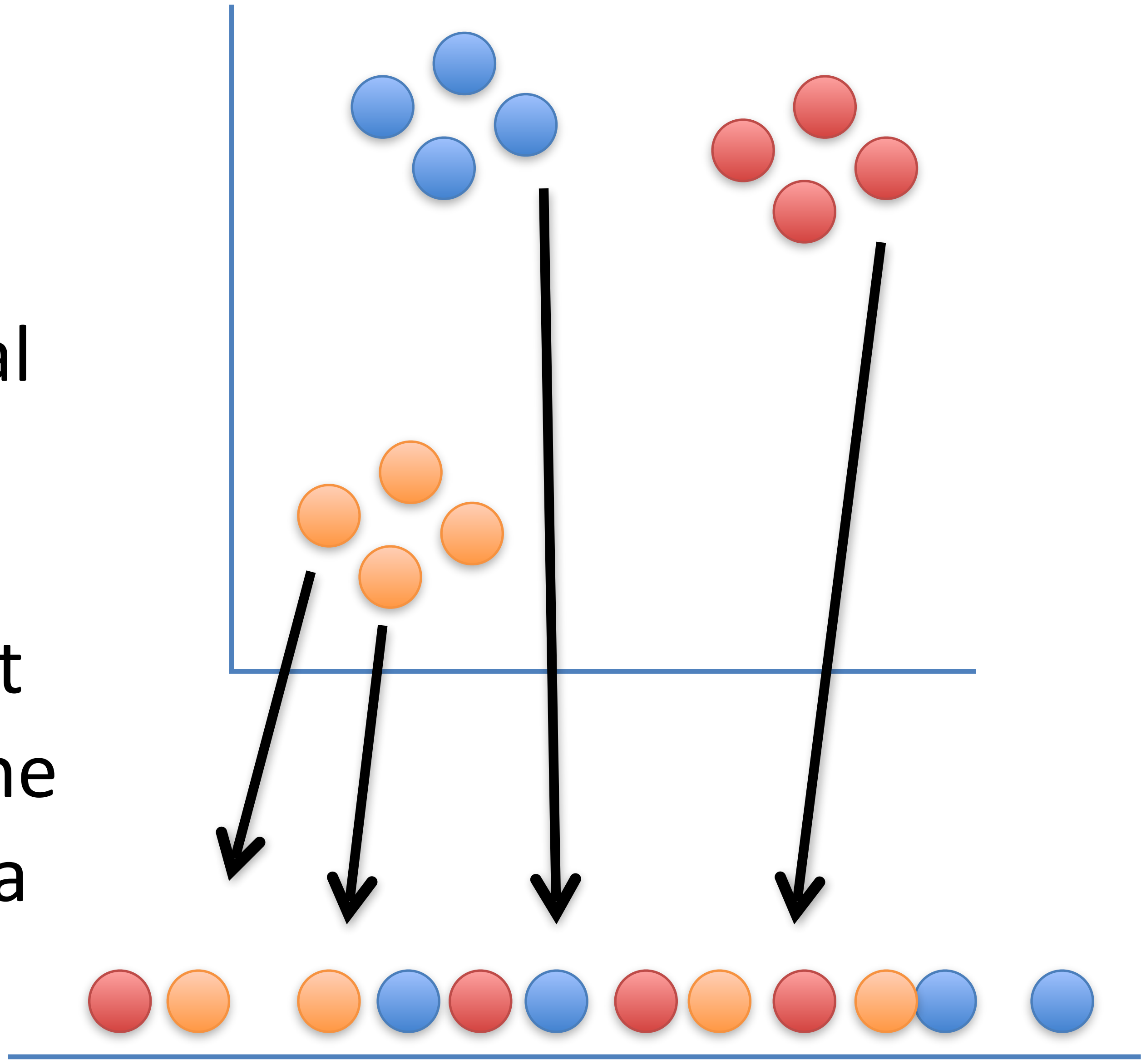
So let's step through the basic ideas of how t-SNE does this.

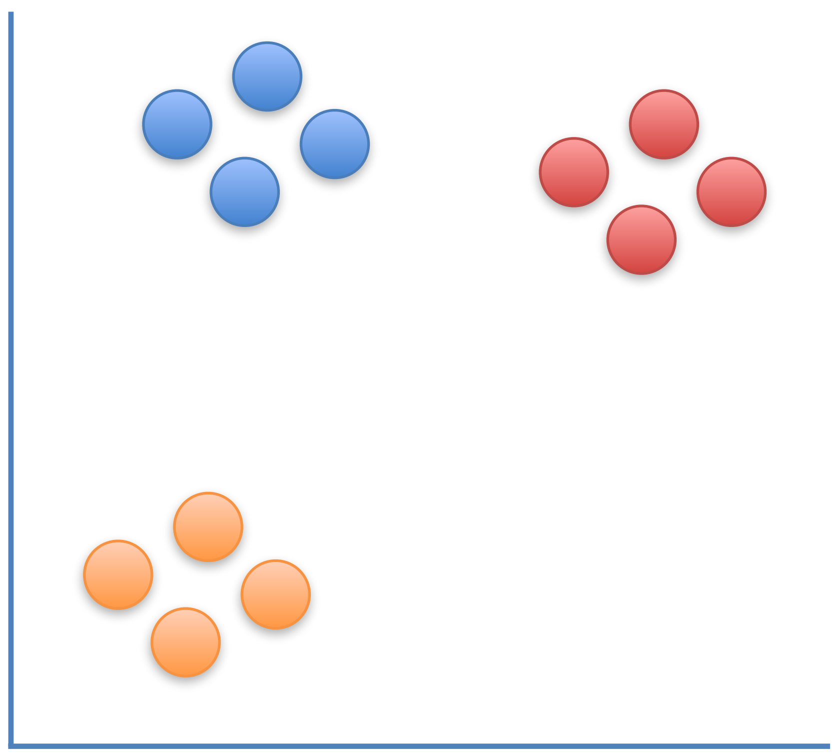
We’ll start start with the original scatter plot…

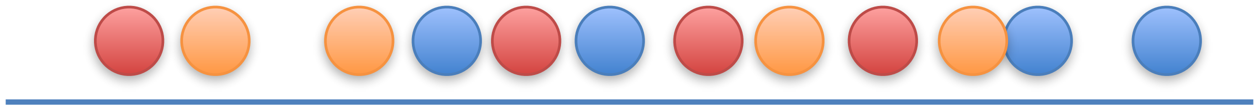


We'll start start with the original scatter plot…

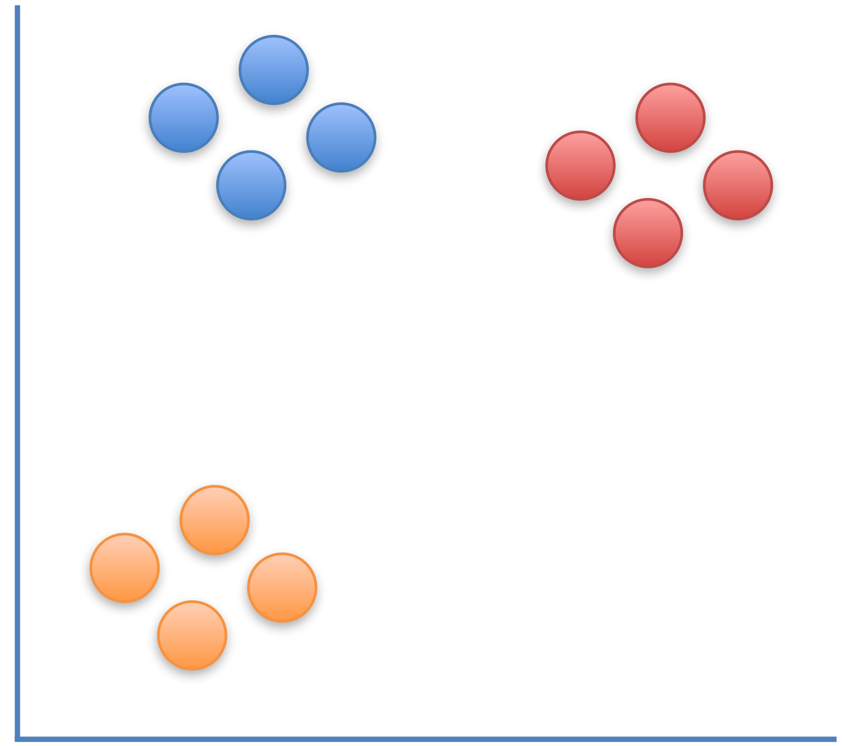
… then we'll put the points on the number line in a random order.



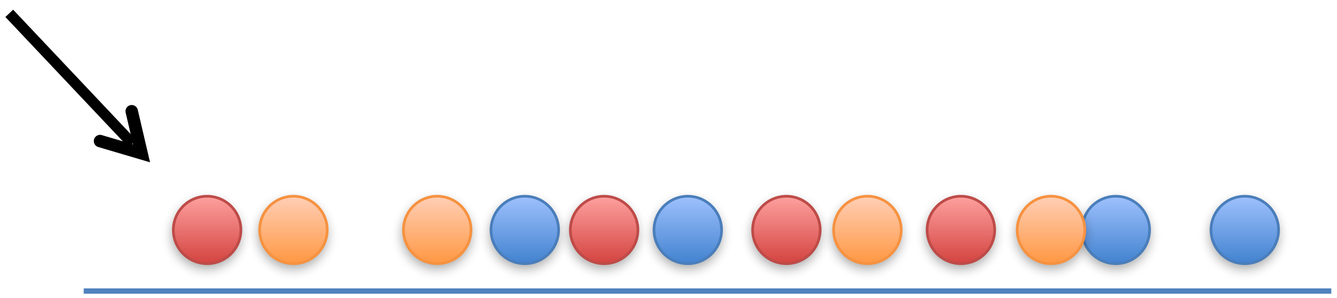


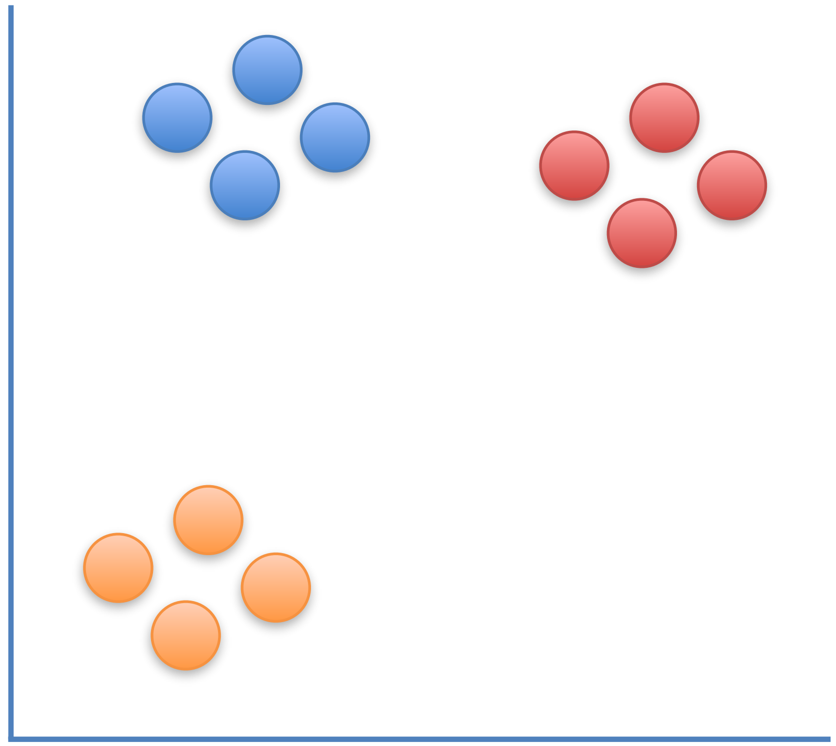


From here on out, t-SNE moves these points, a little bit at a time, until it has clustered them.



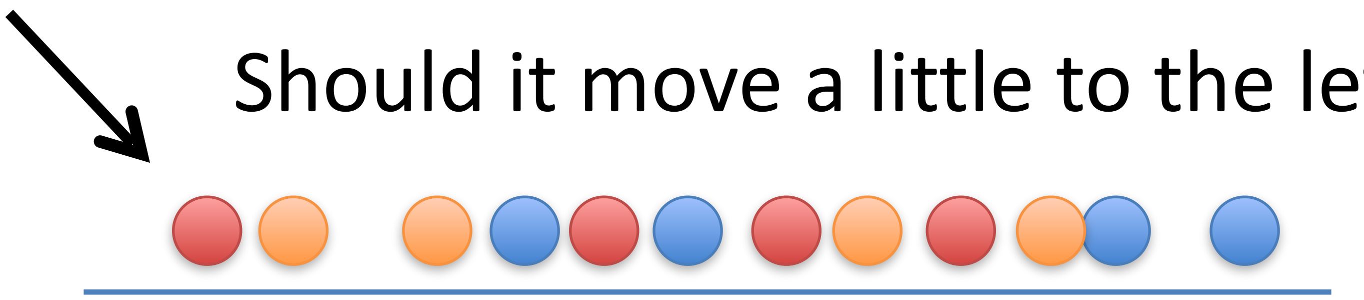
Let’s figure out where to move this first point…

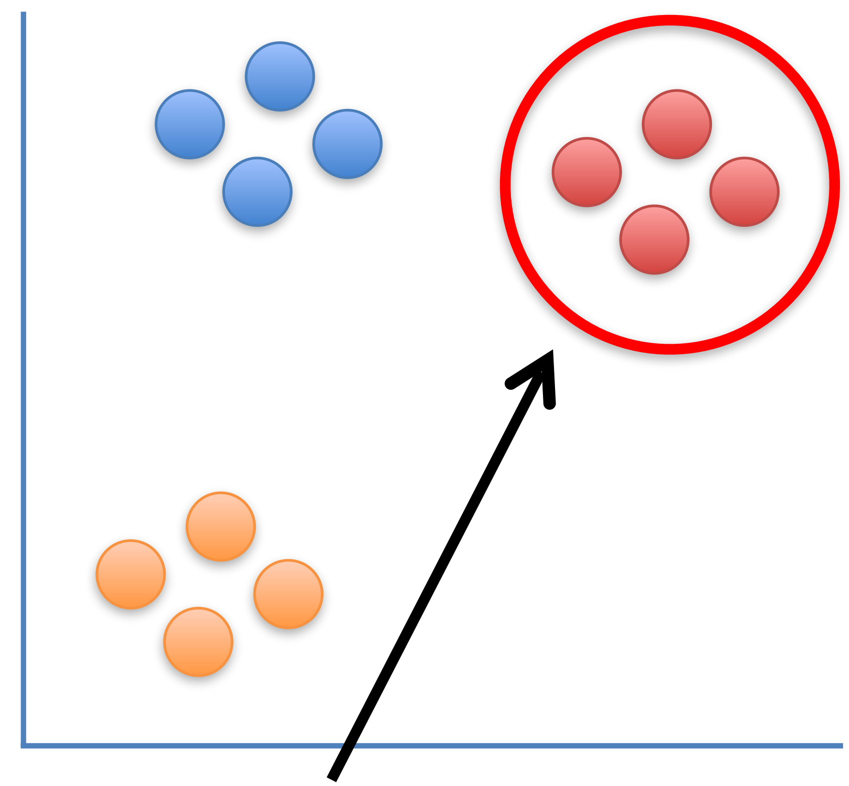




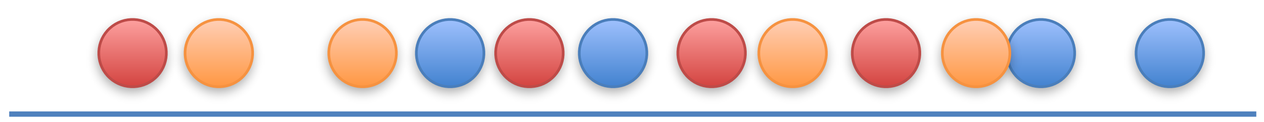
Let’s figure out where to move this first point...

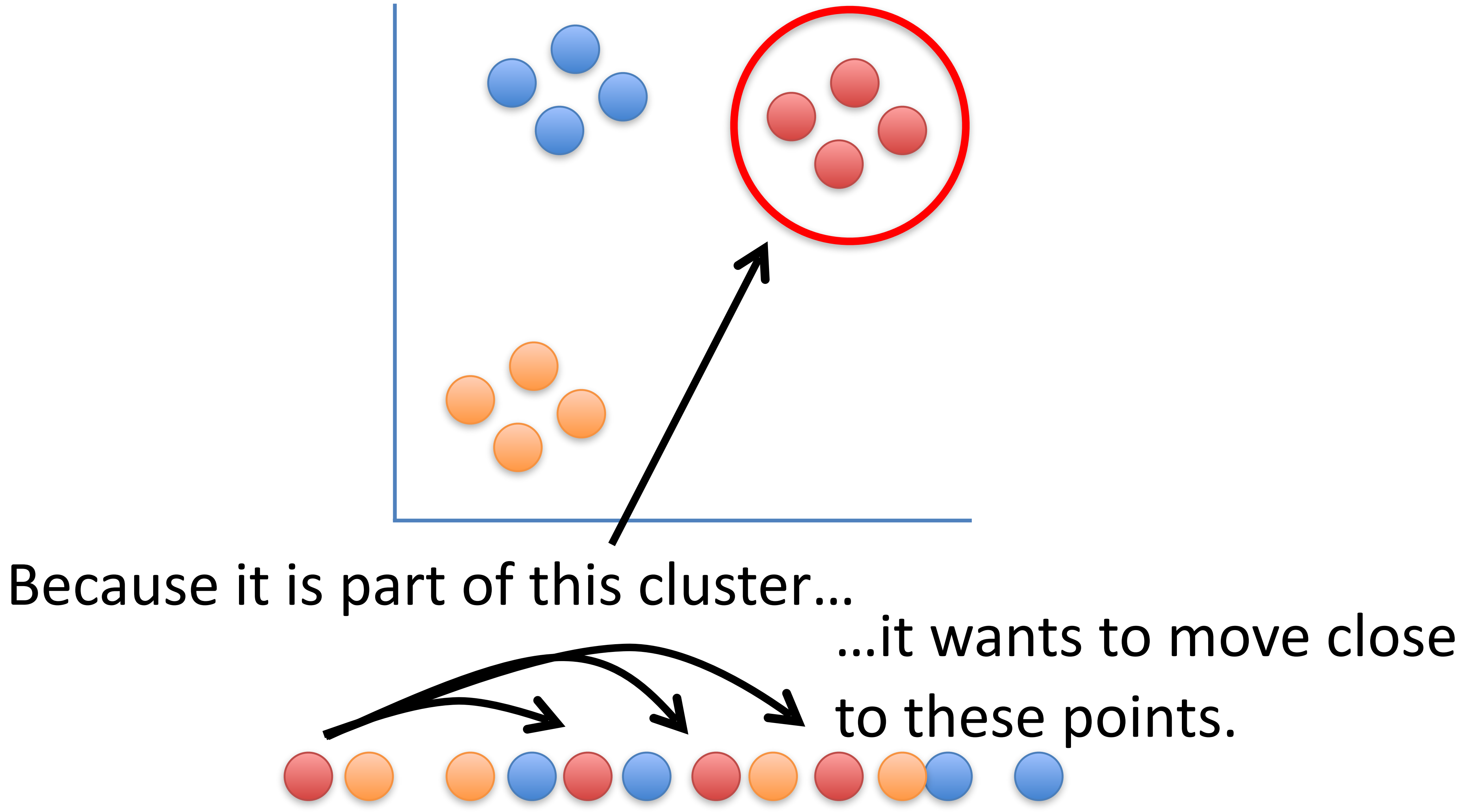
Should it move a little to the left or to the right?



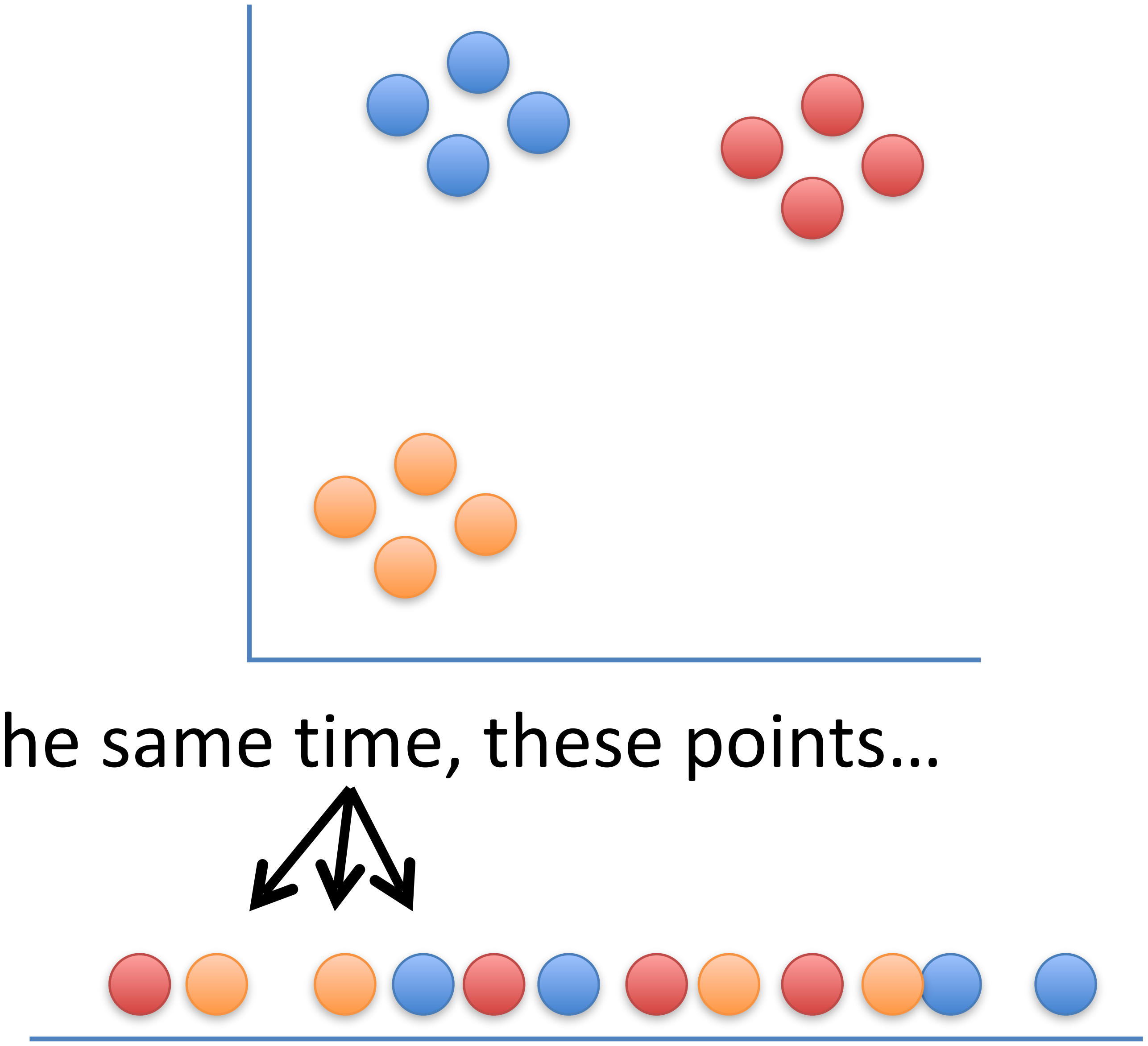


Because it is part of this cluster...

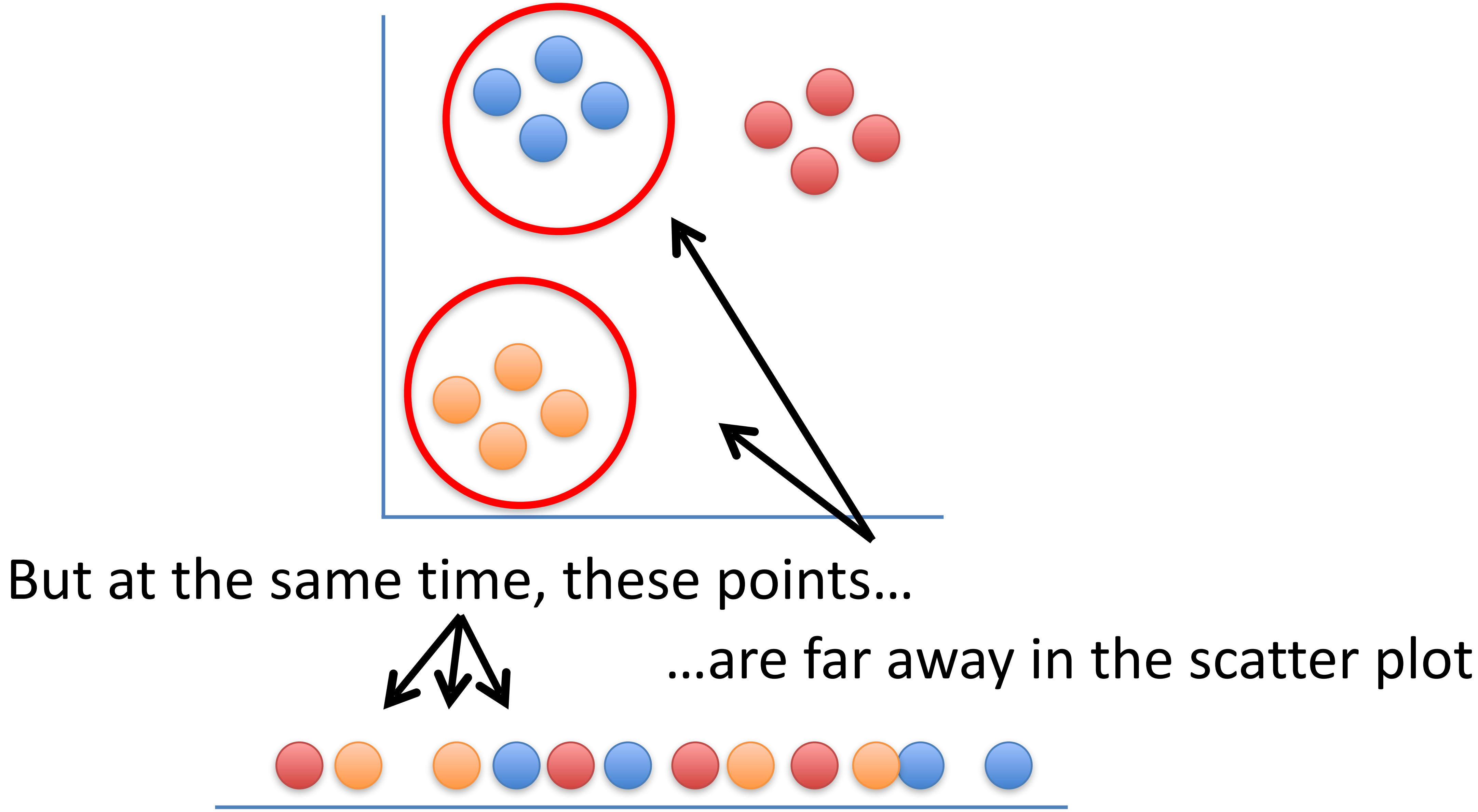




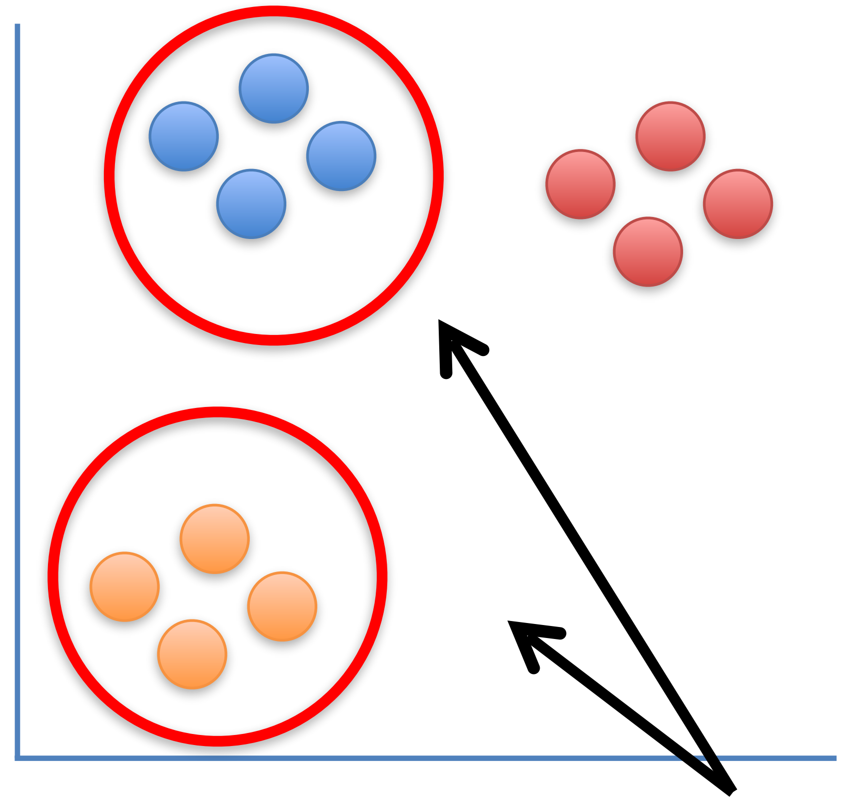
Because it is part of this cluster…
…it wants to move closer
to these points.



But at the same time, these points…

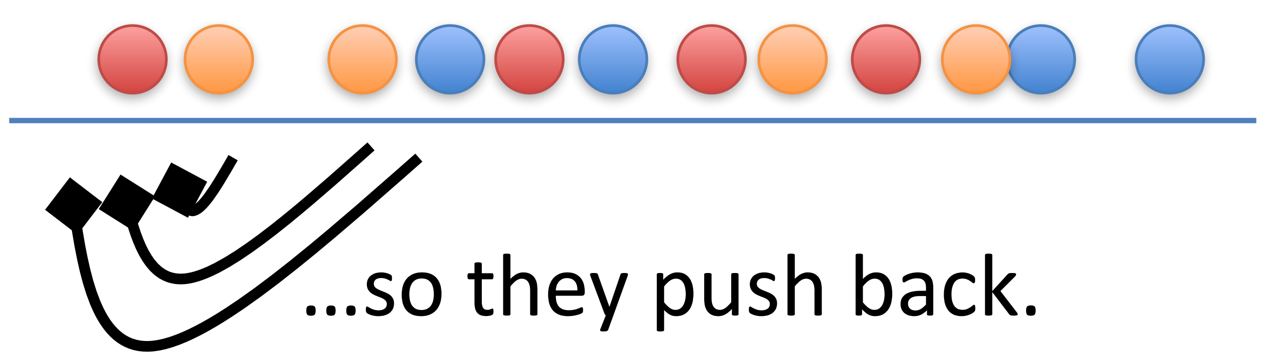


But at the same time, these points…
…are far away in the scatter plot.

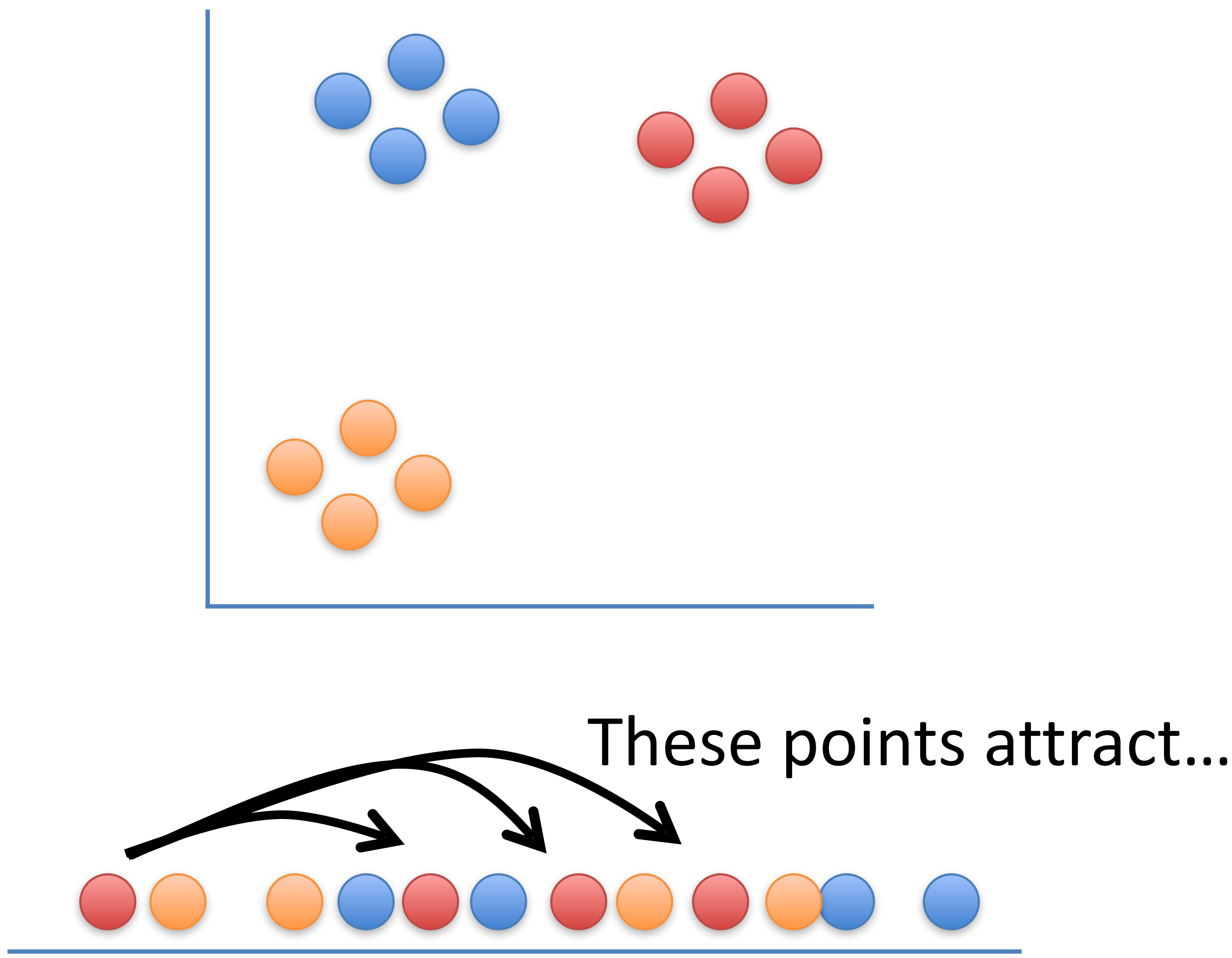


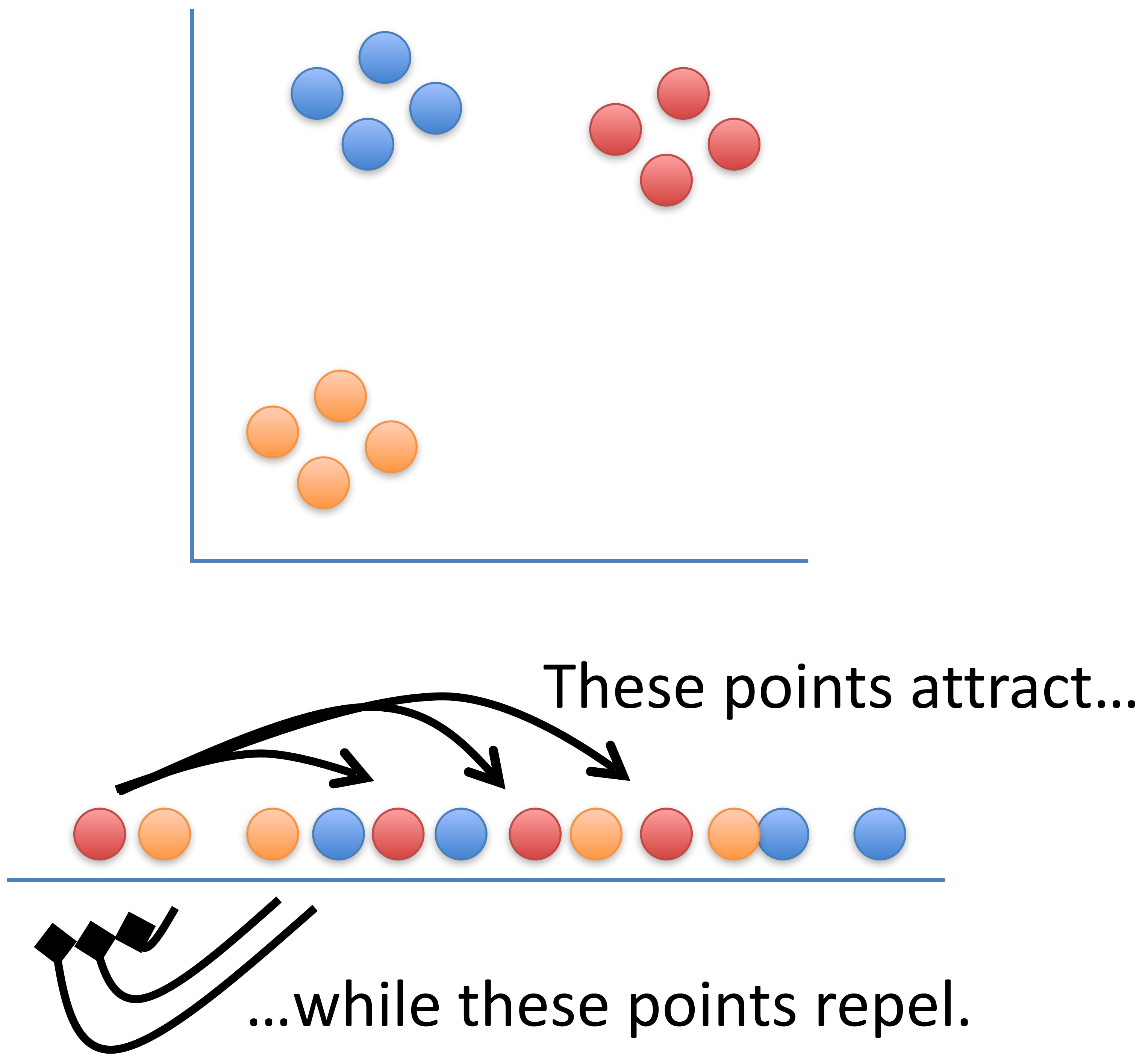
But at the same time, these points…

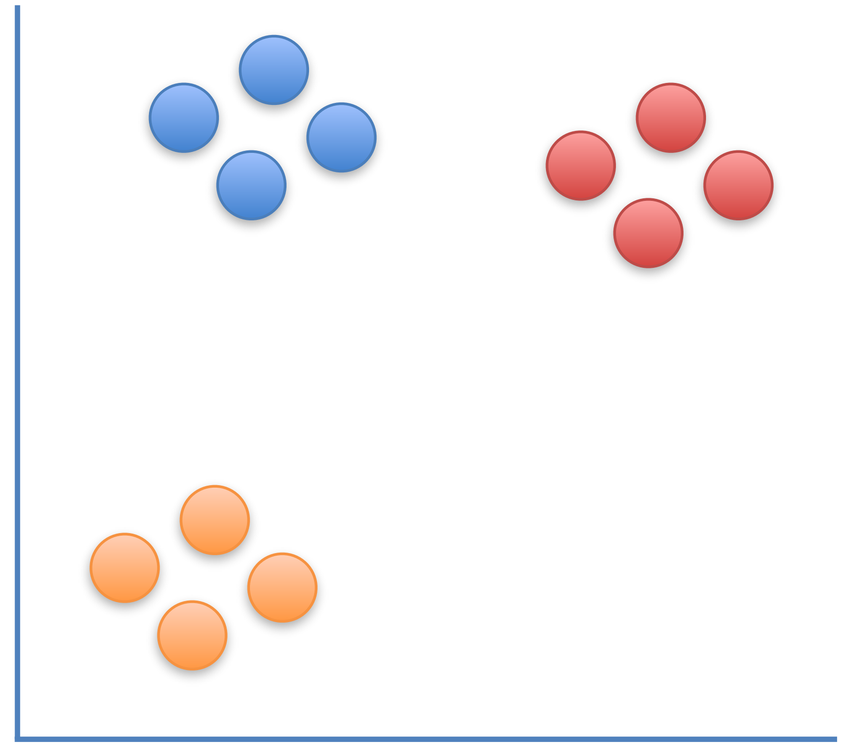
…are far away in the scatter plot.



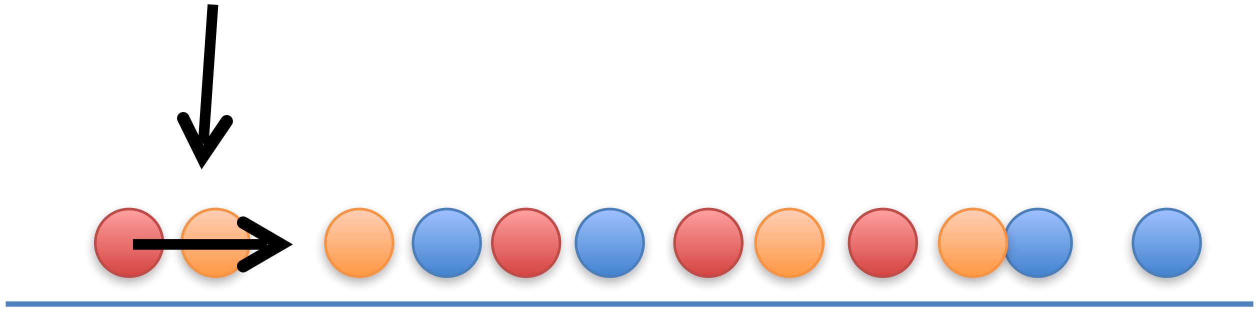
…so they push back.

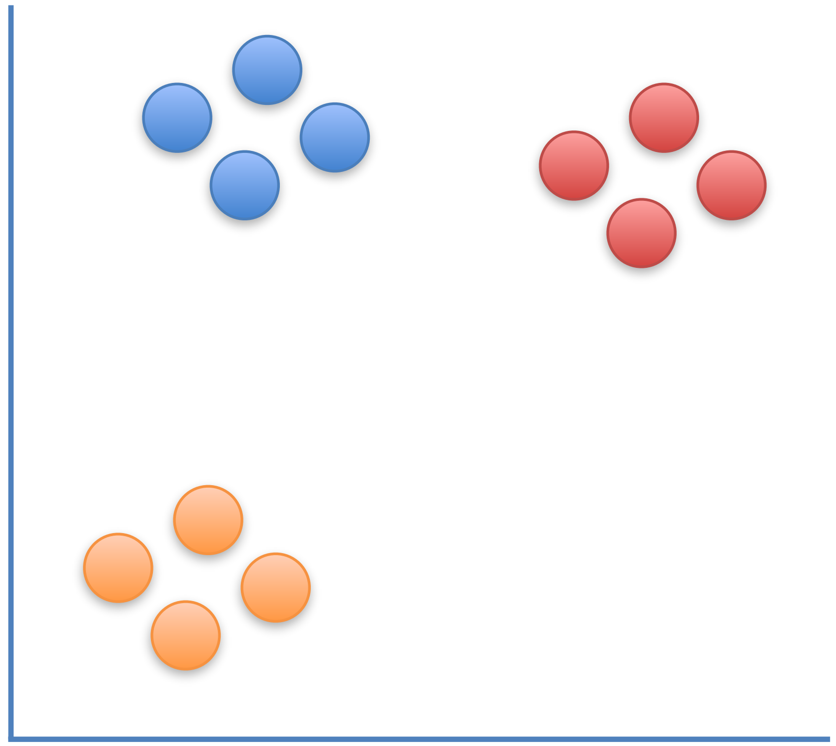

These points attract…


These points attract…
…while these points repel.

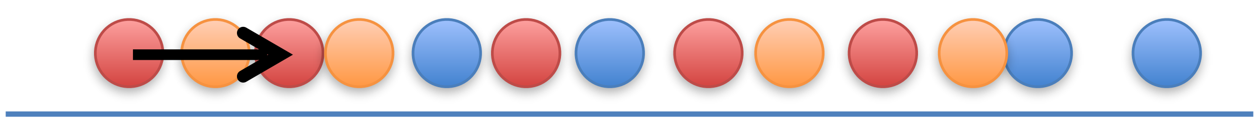


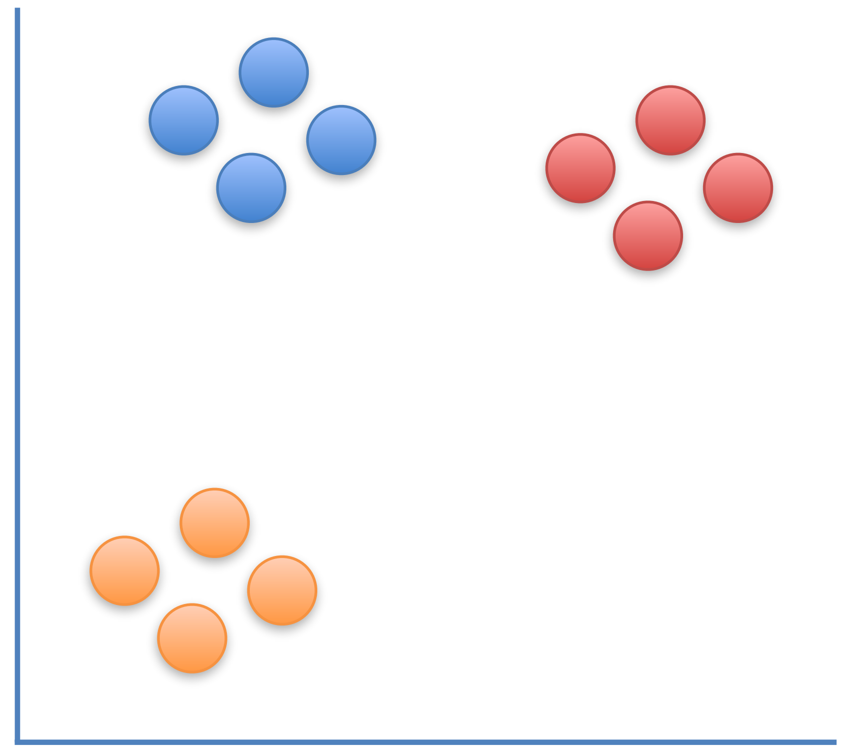
In this case, the attraction is strongest, so the point moves a little to the right.



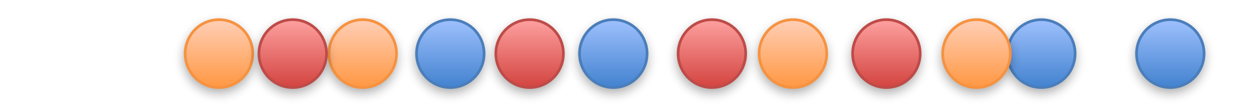


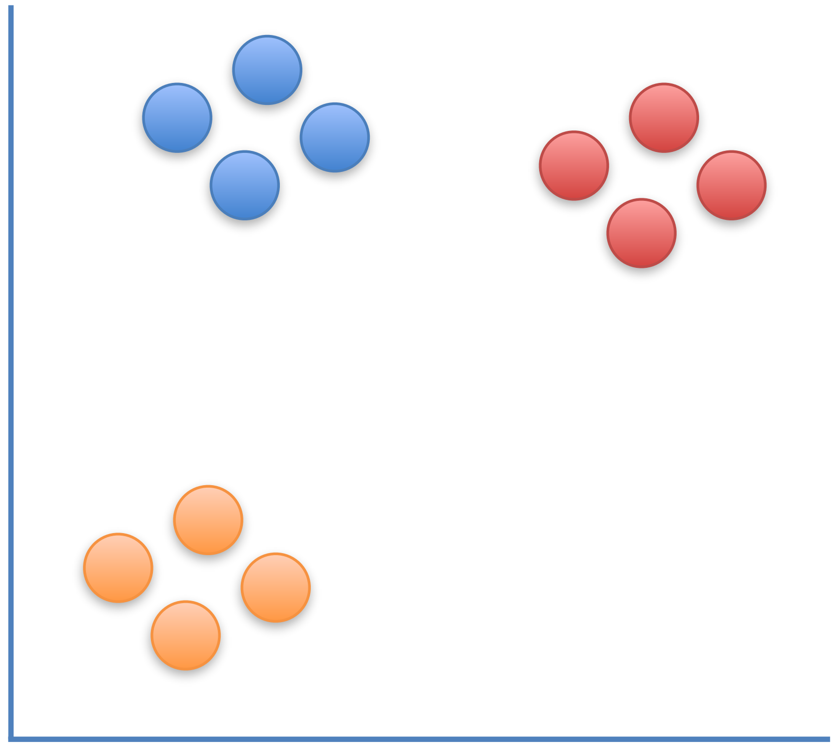
In this case, the attraction is strongest, so the point moves a little to the right.



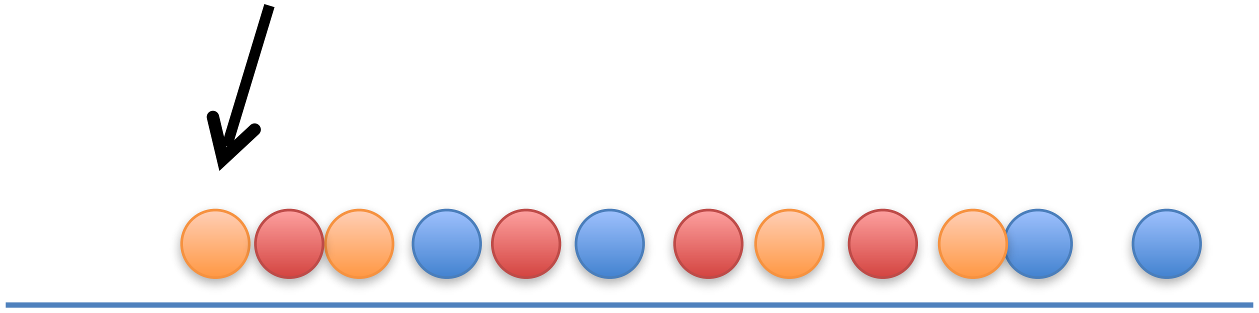


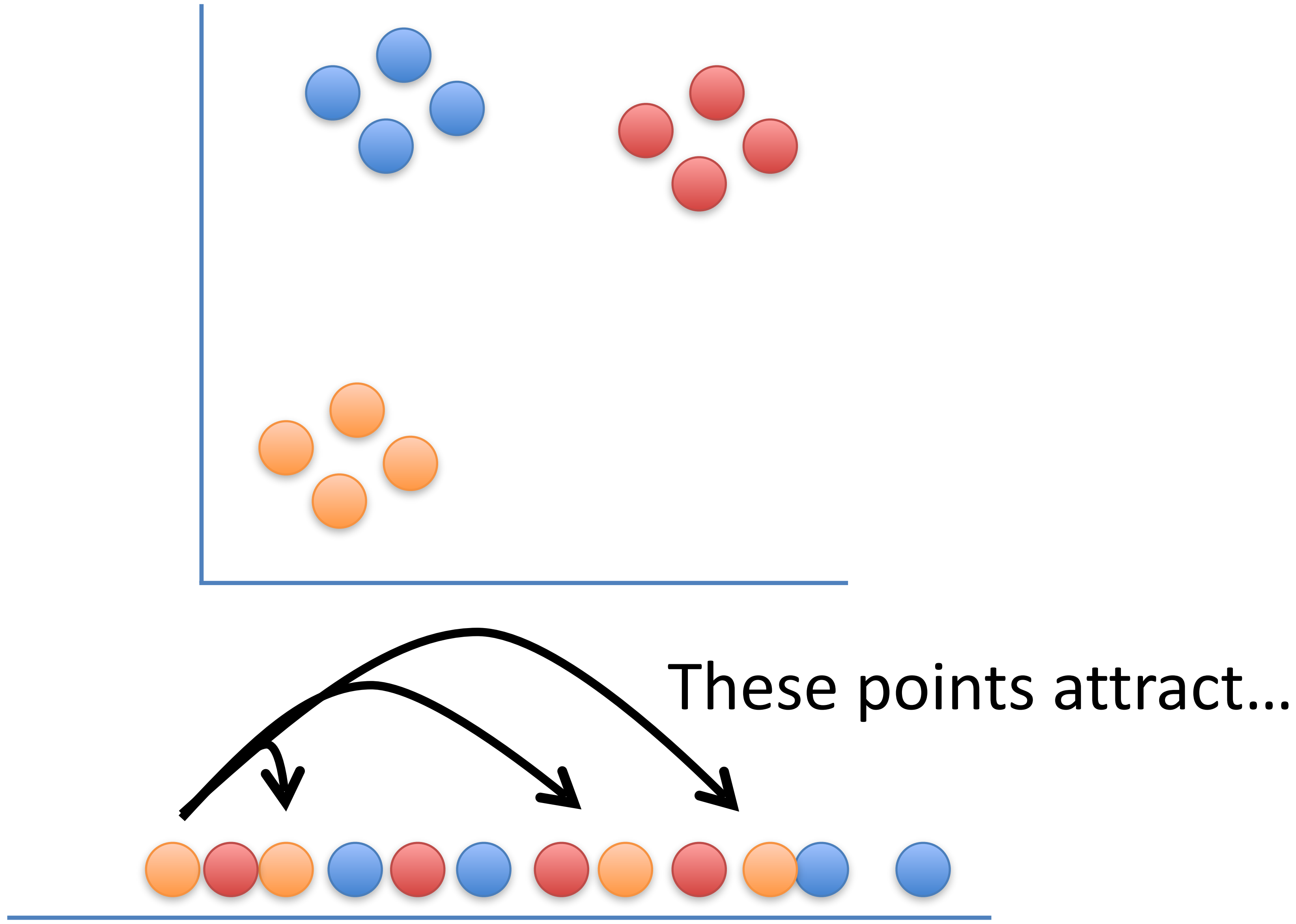
BAM!

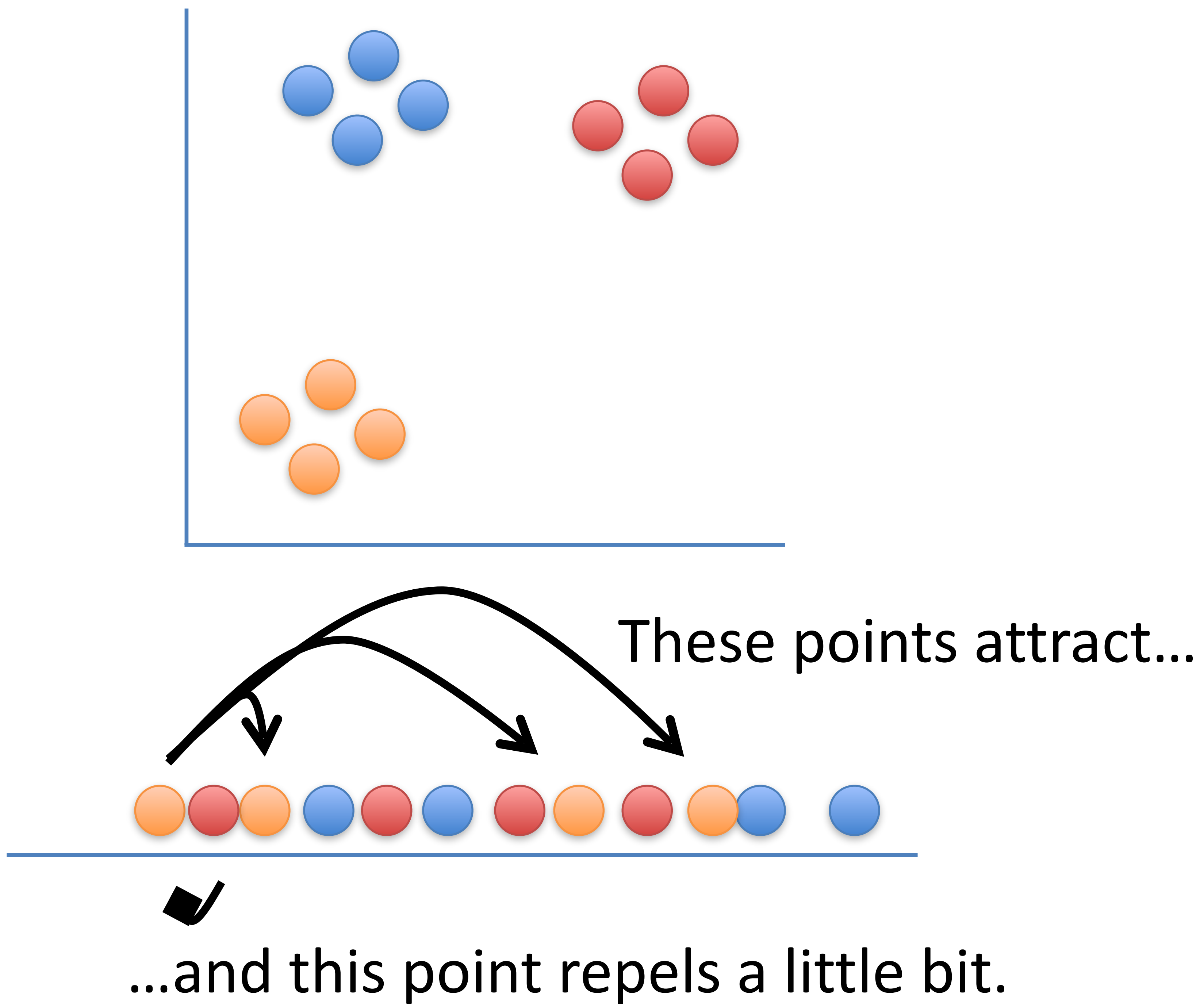


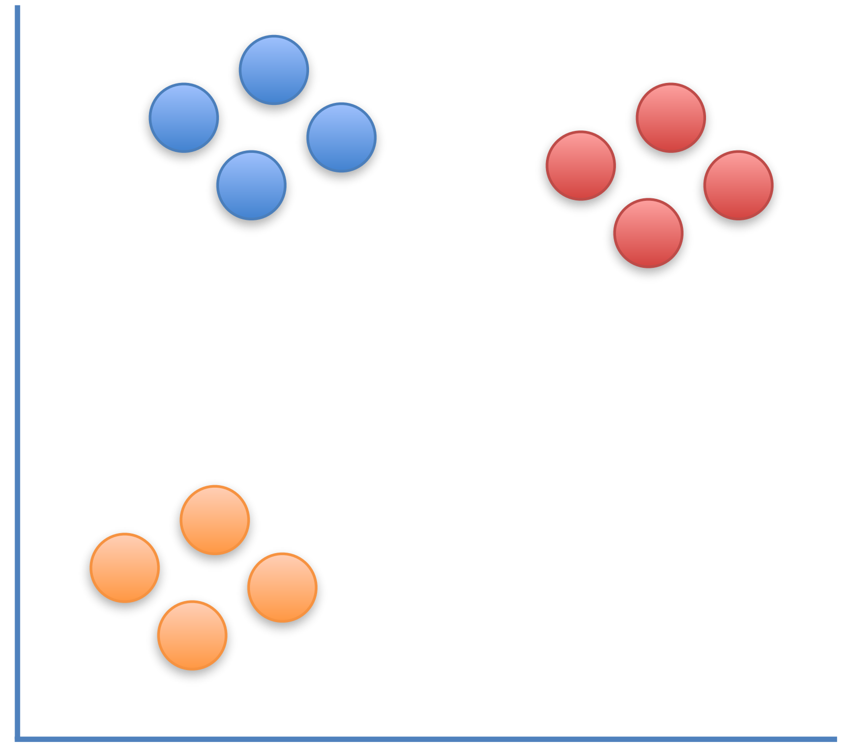


Now let’s move this point a little bit…

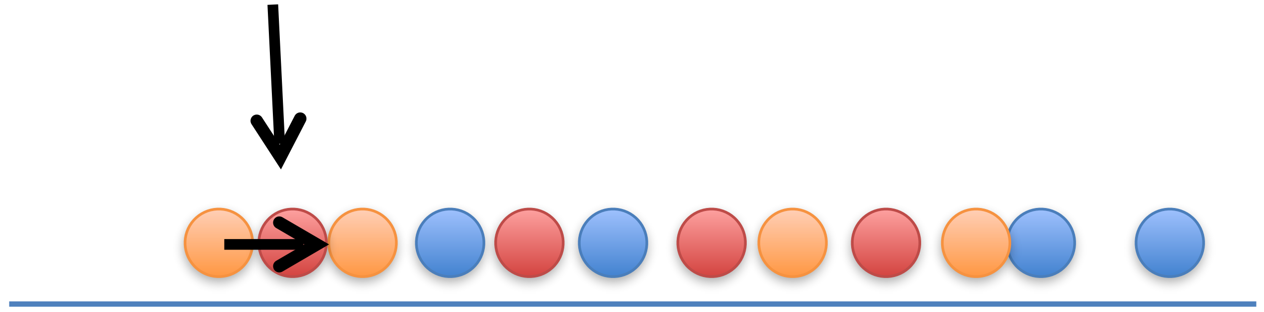


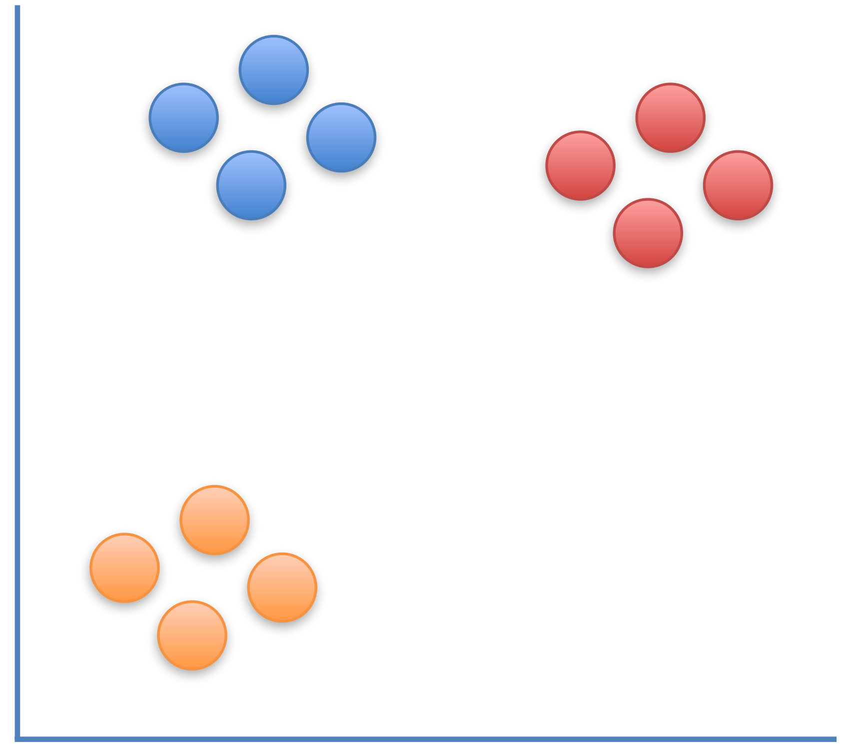

These points attract...


These points attract...
...and this point repels a little bit.

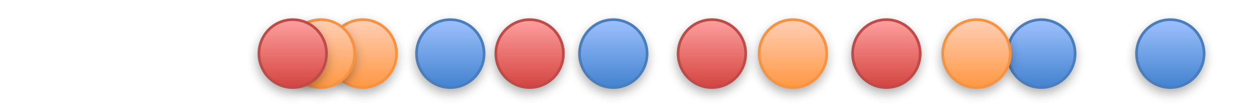


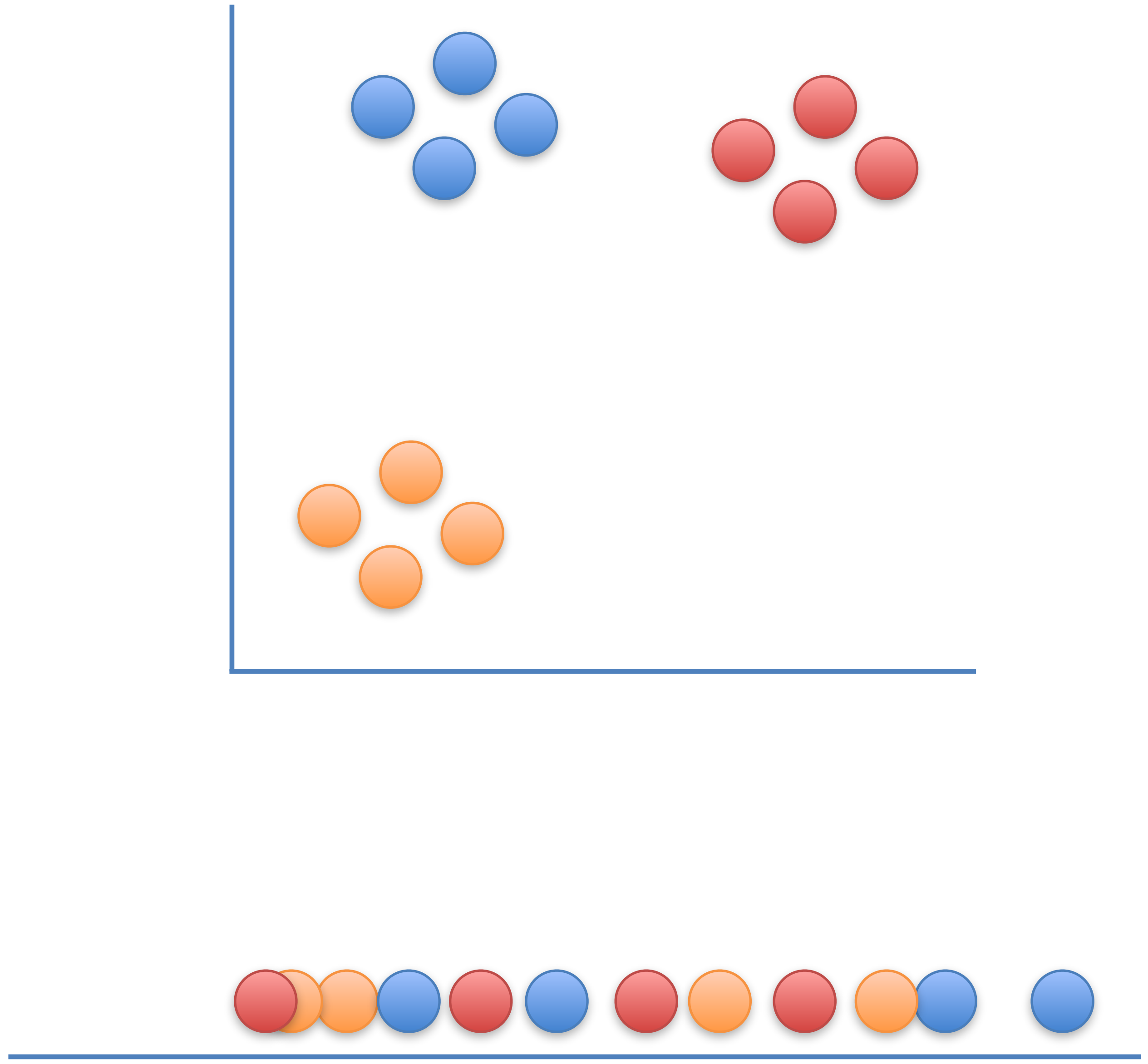
So it moves a little to closer to the other orange points.



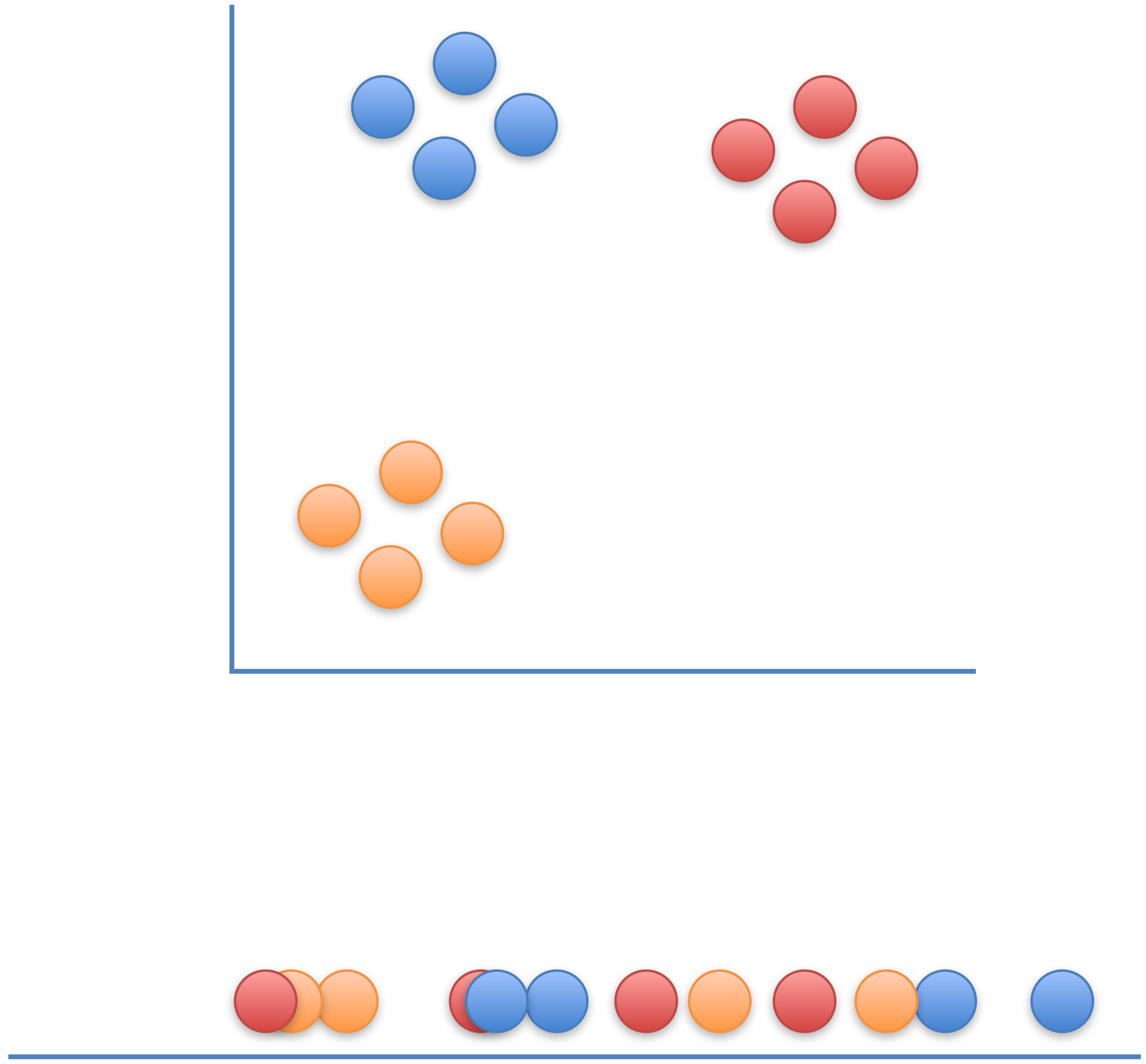


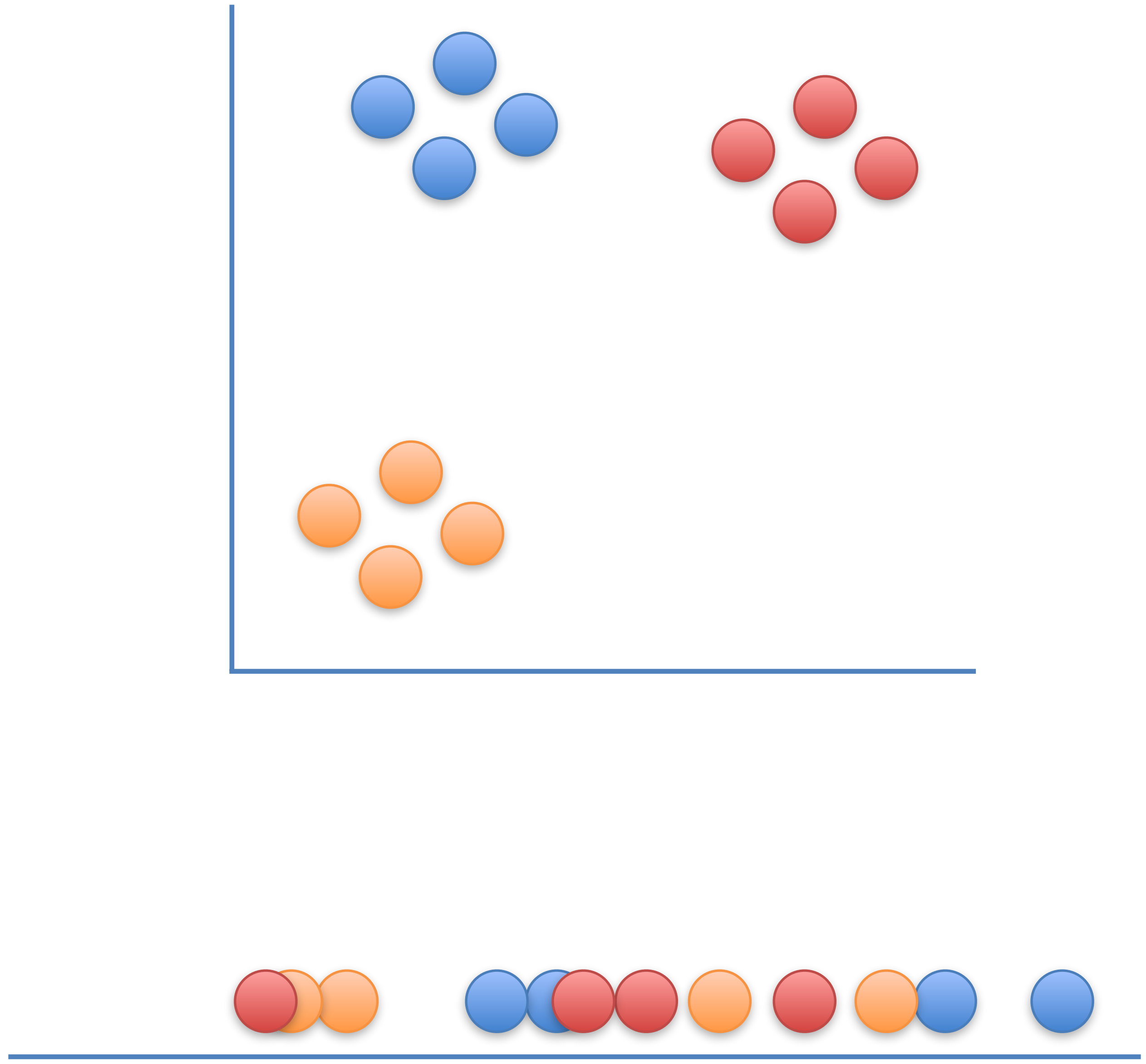
Double BAM!

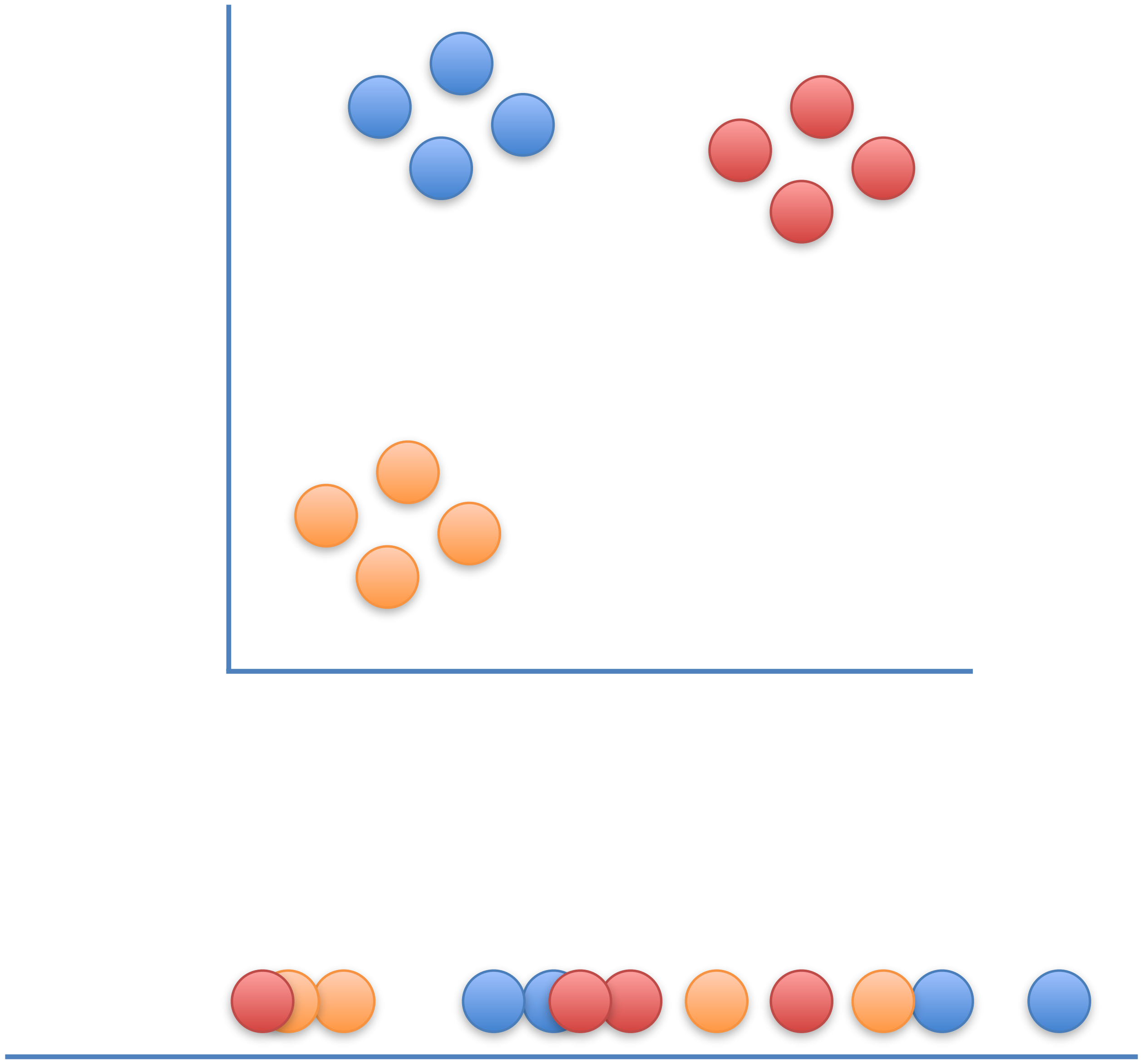


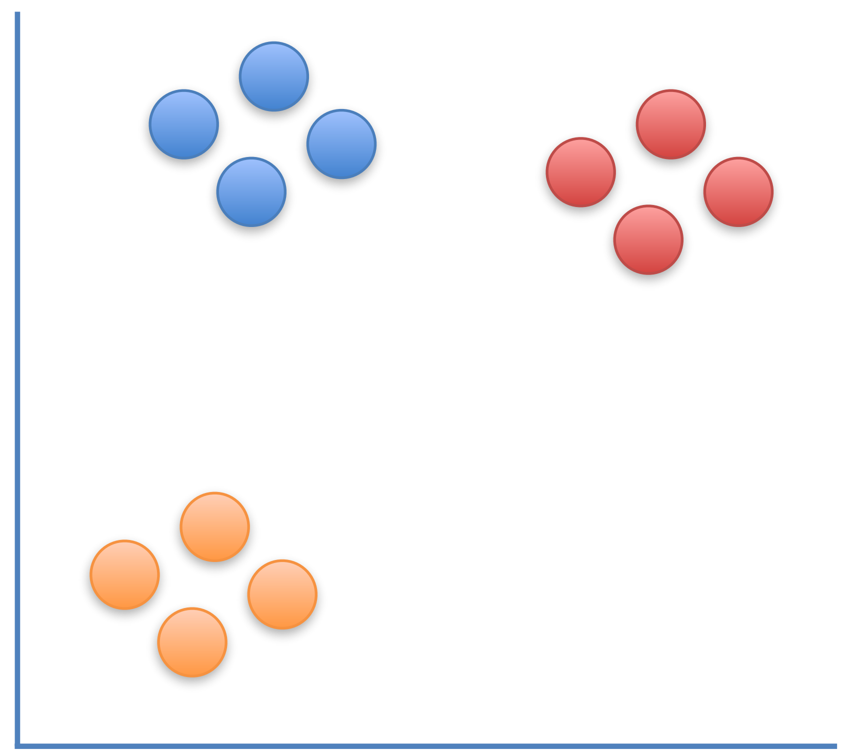


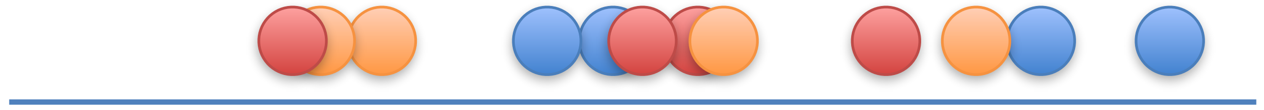
At each step, a point on the line is attracted to points it is near in the scatter plot, and repelled by points it is far from...

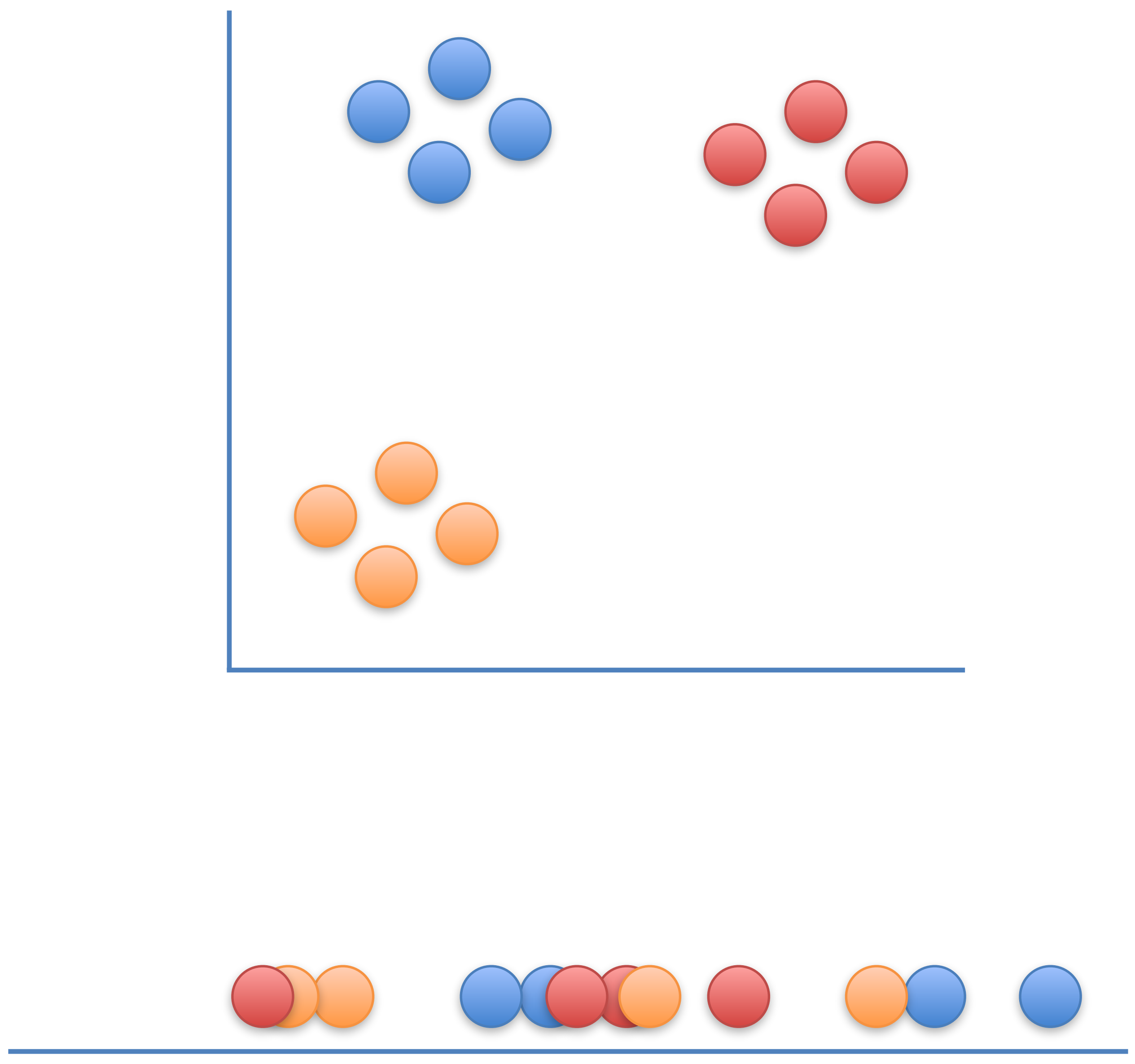


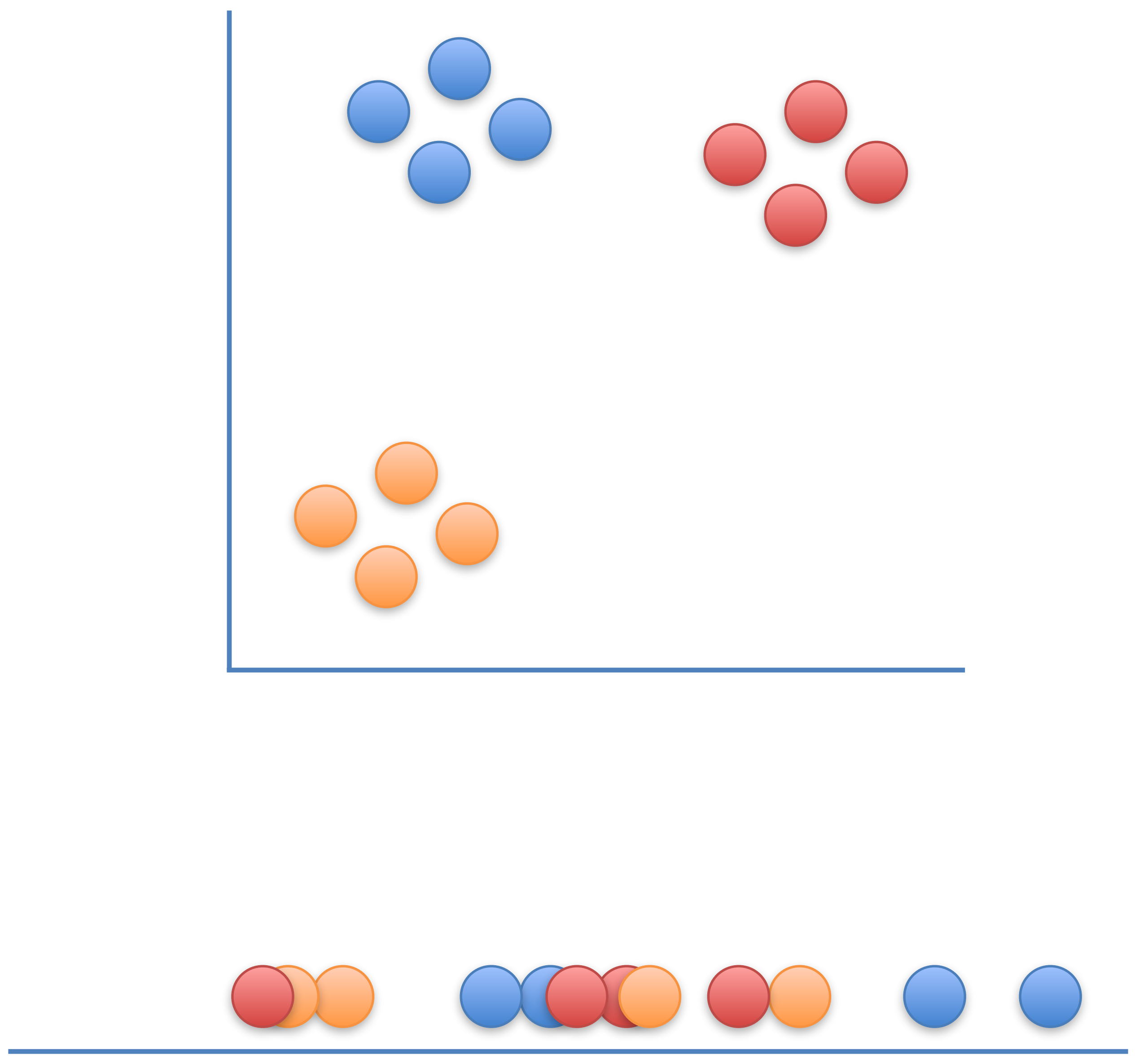


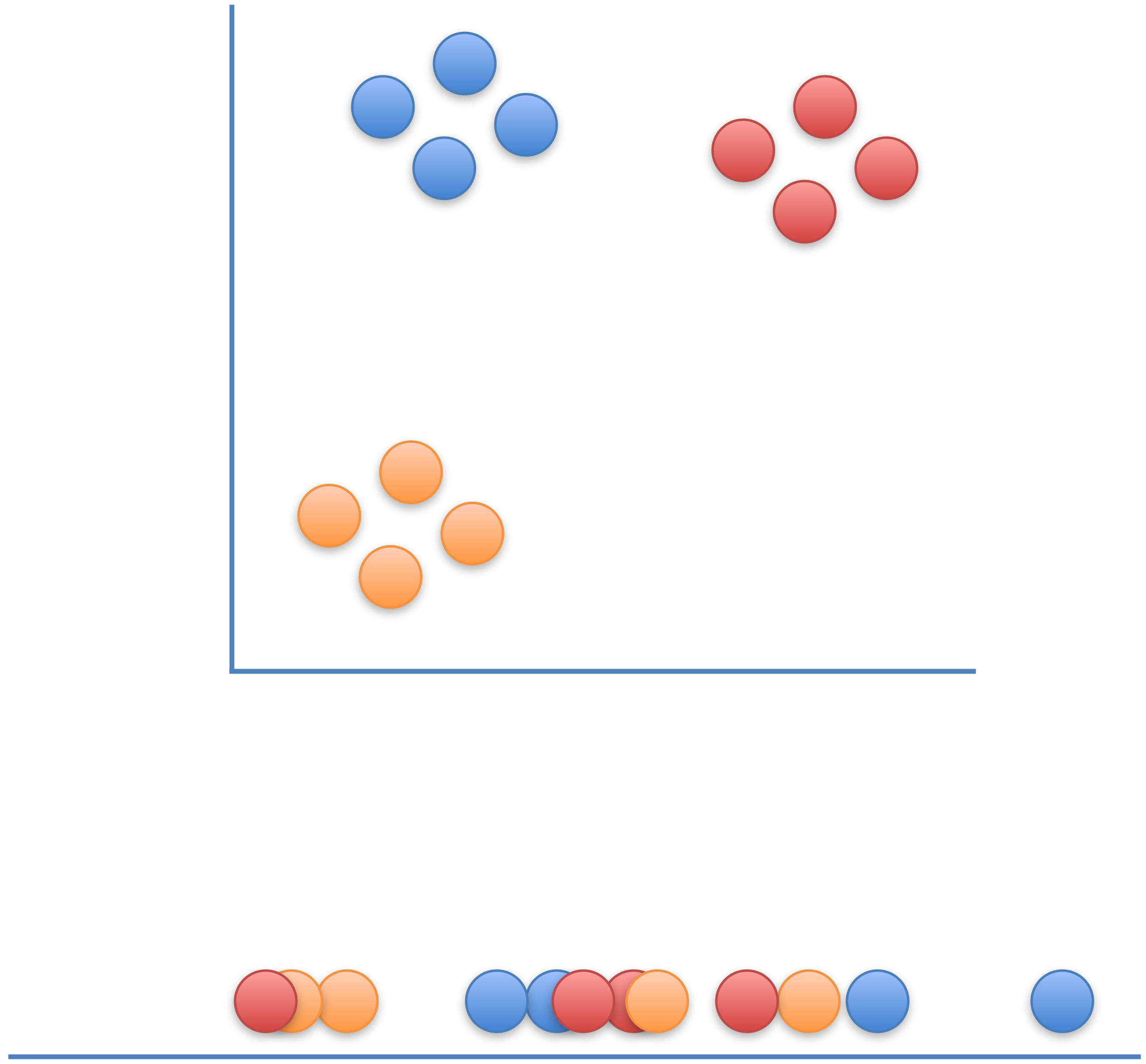


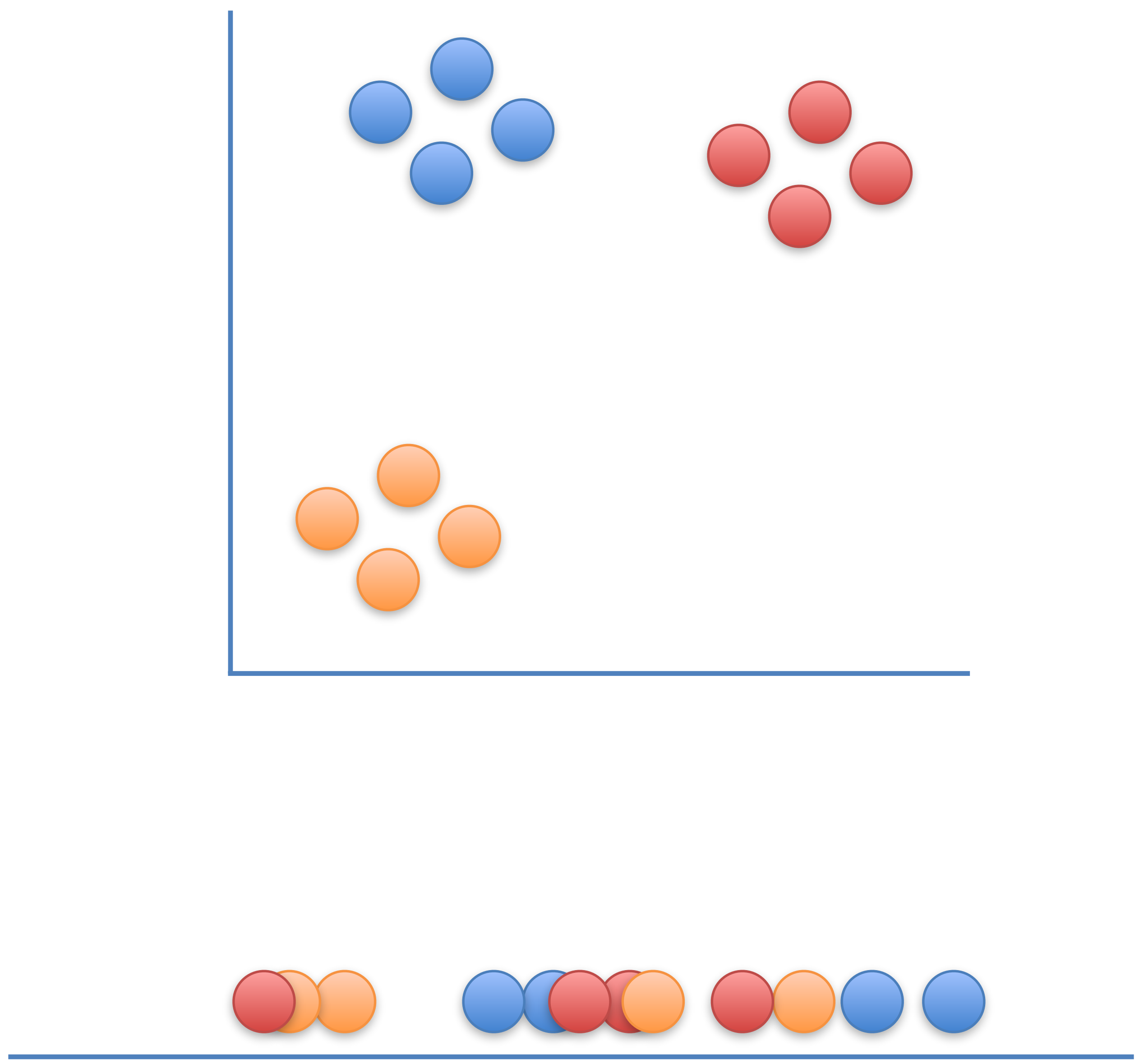


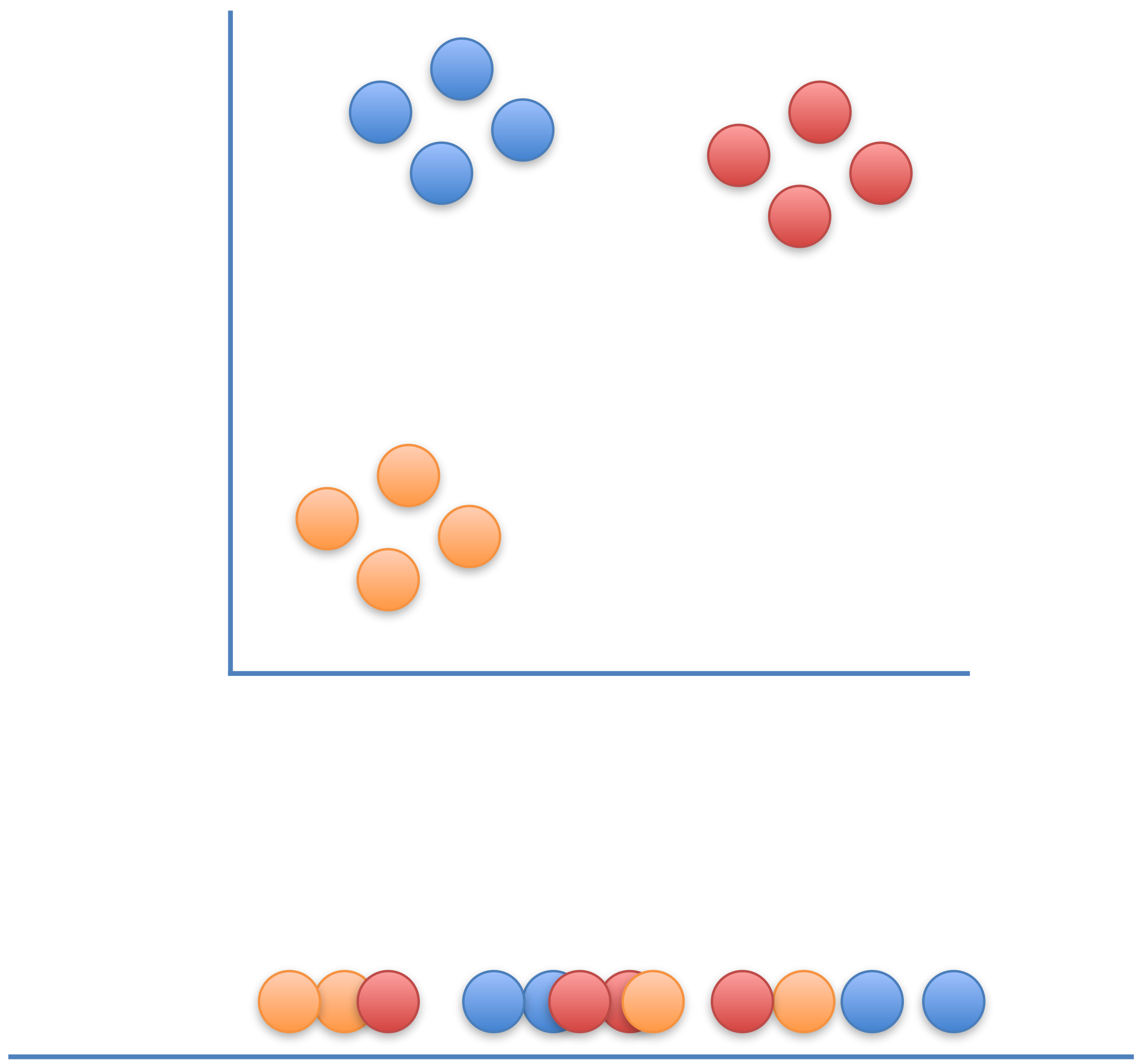


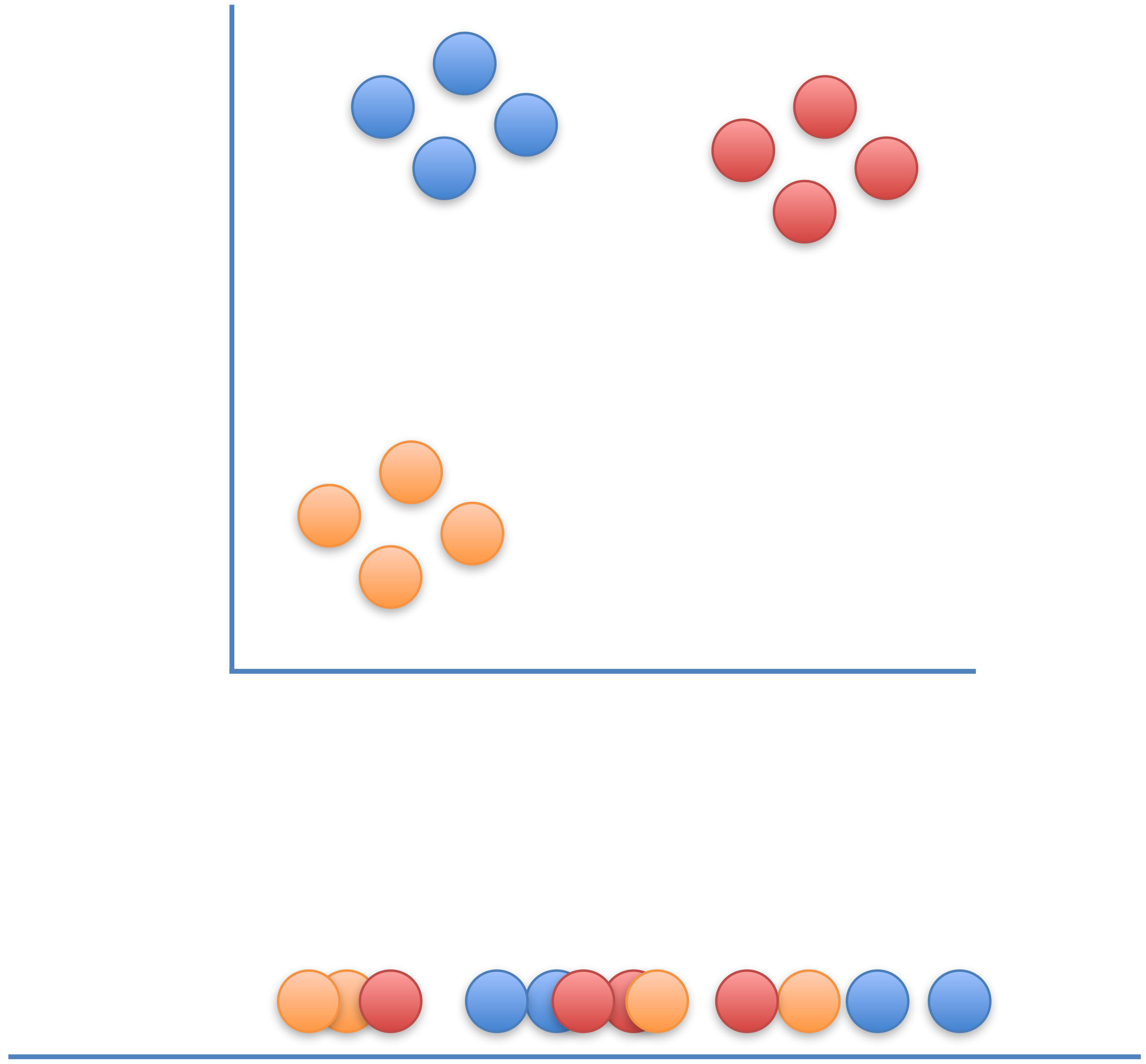


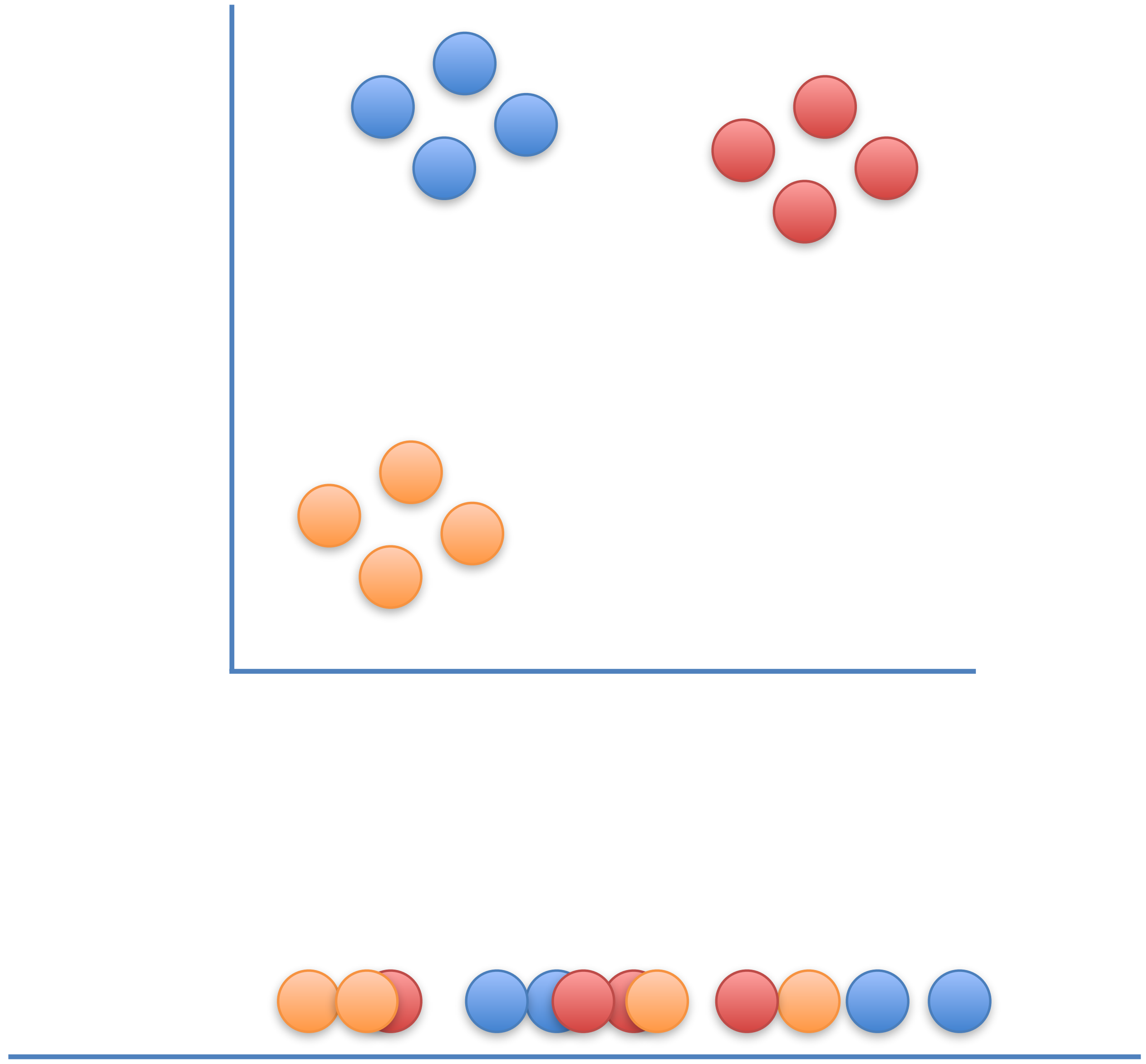


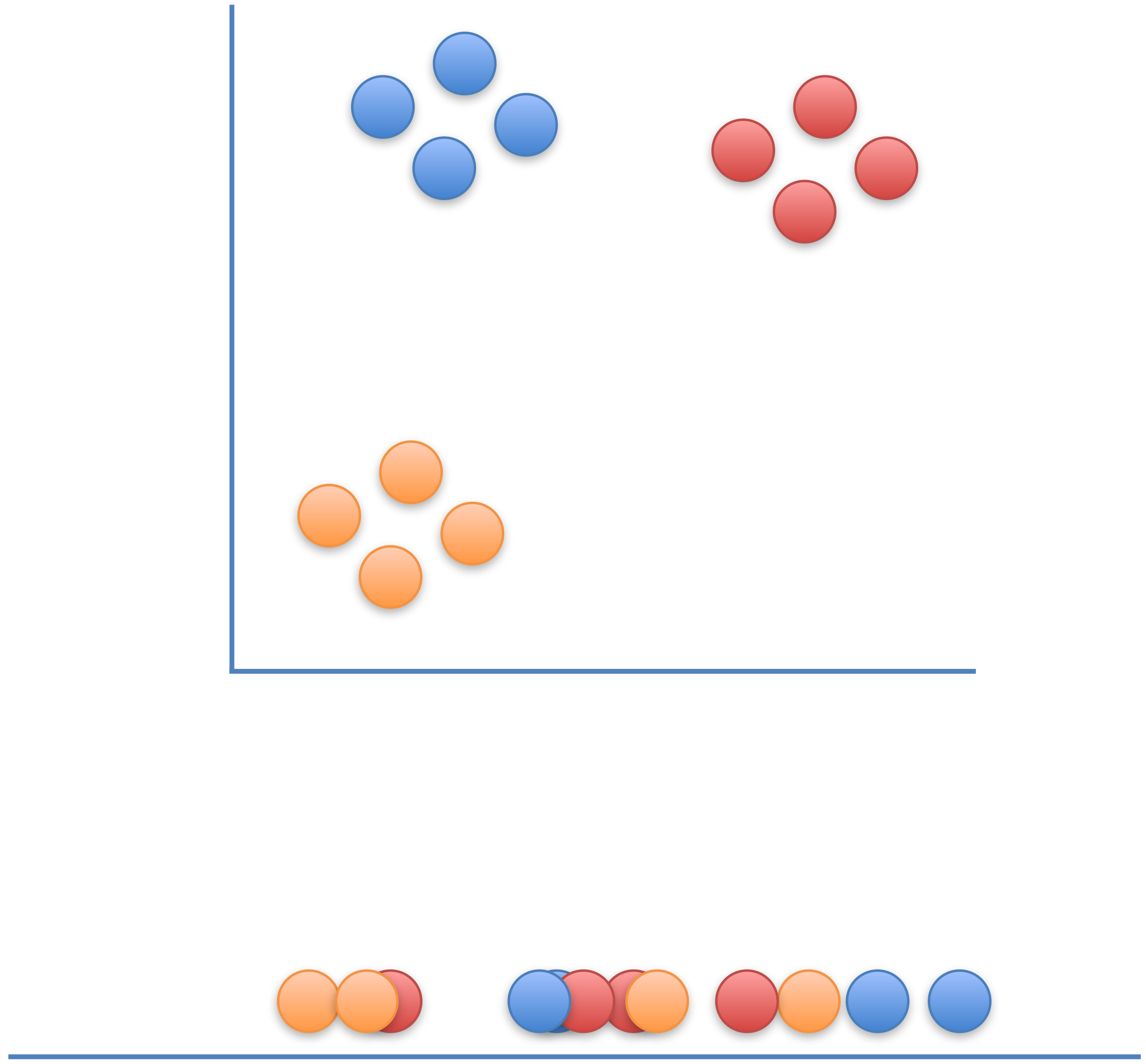


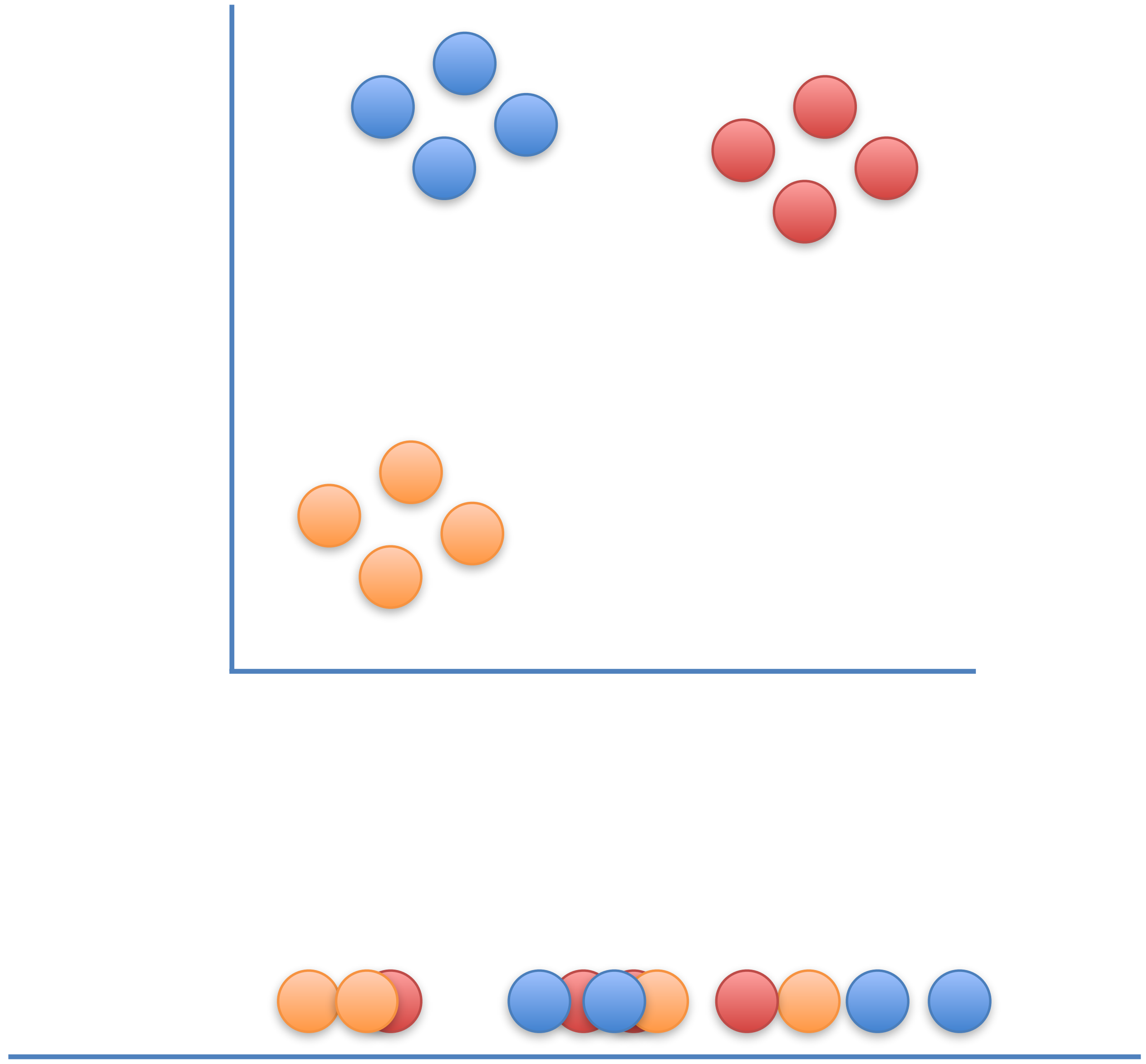


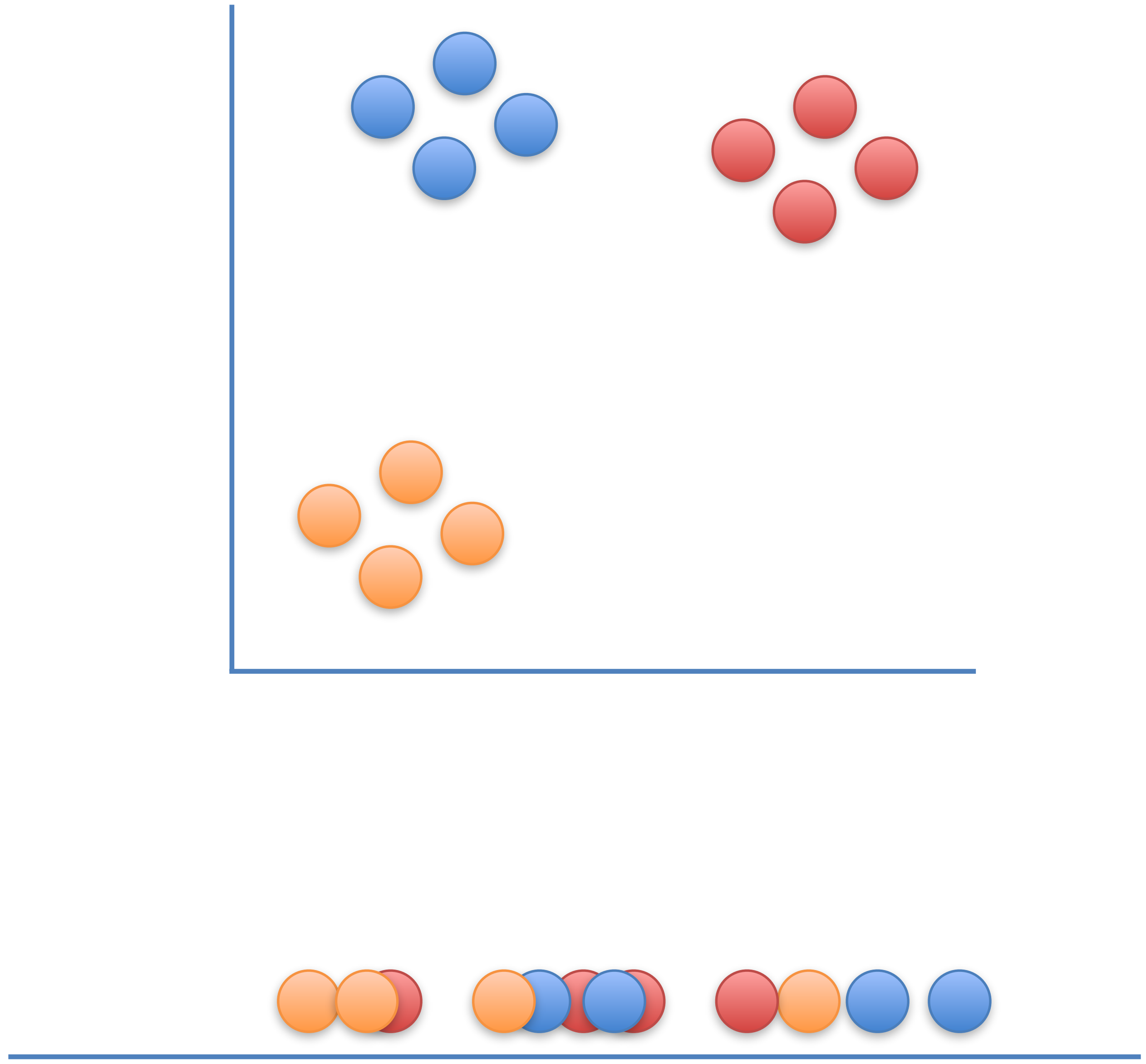


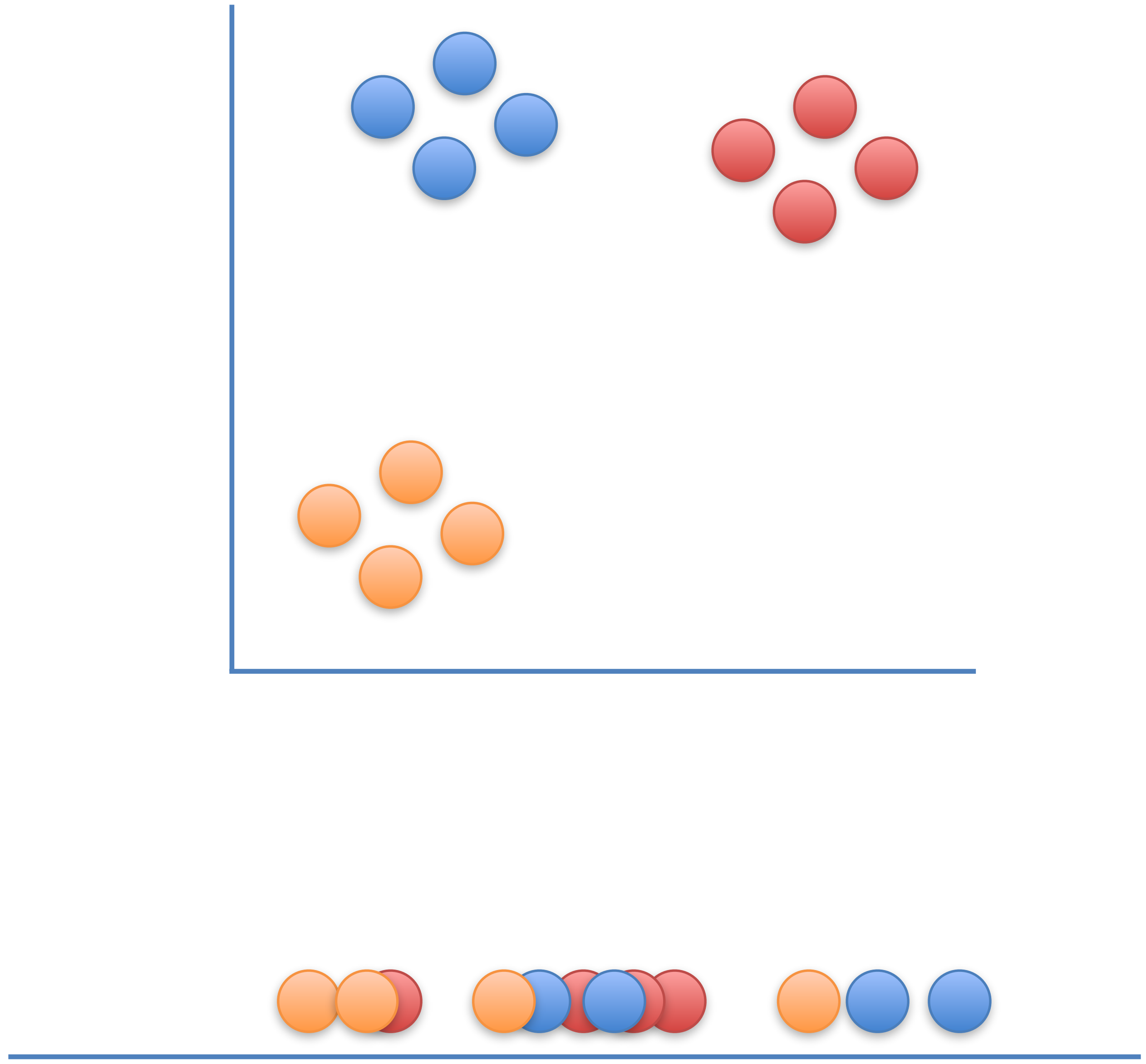


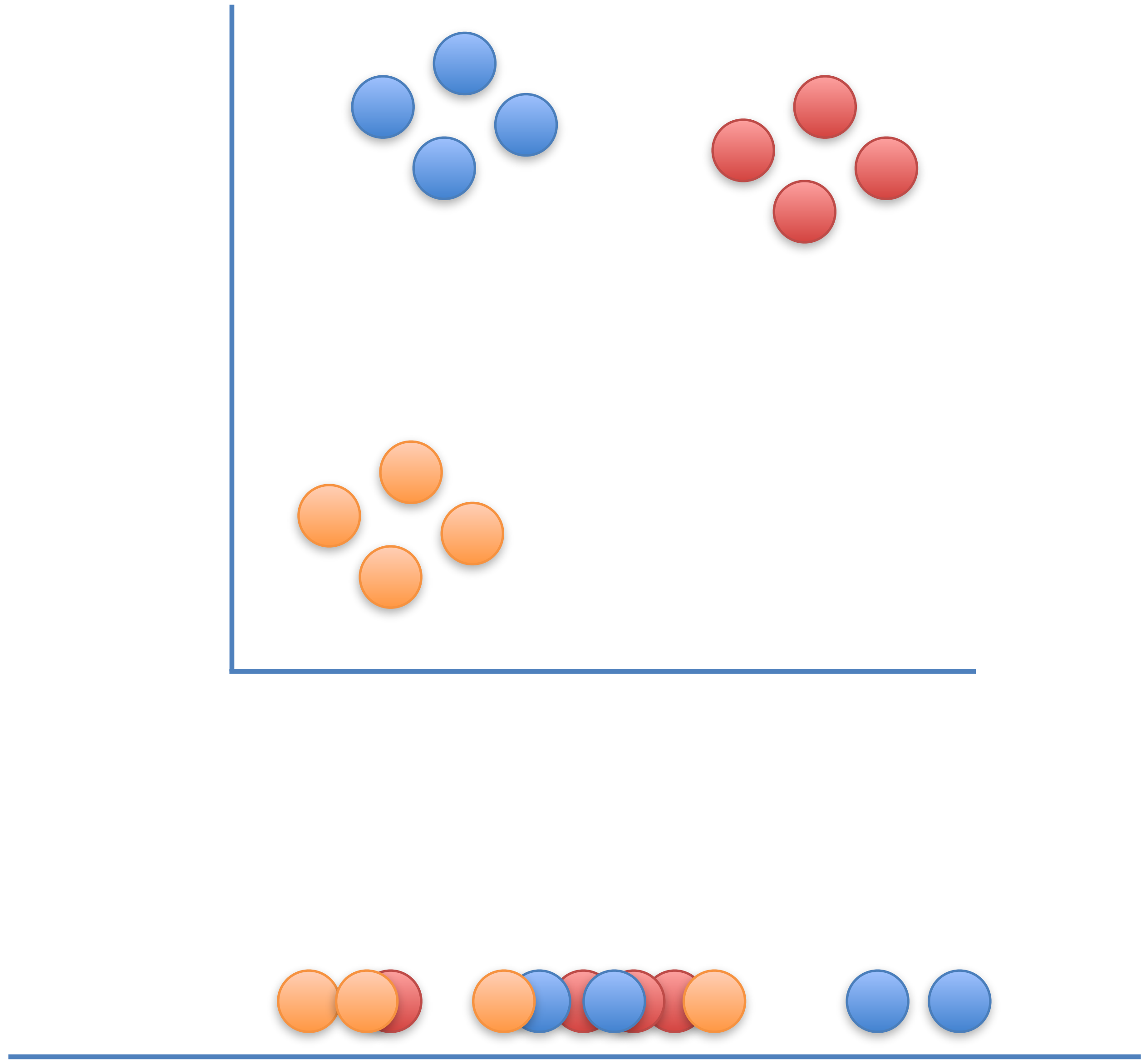


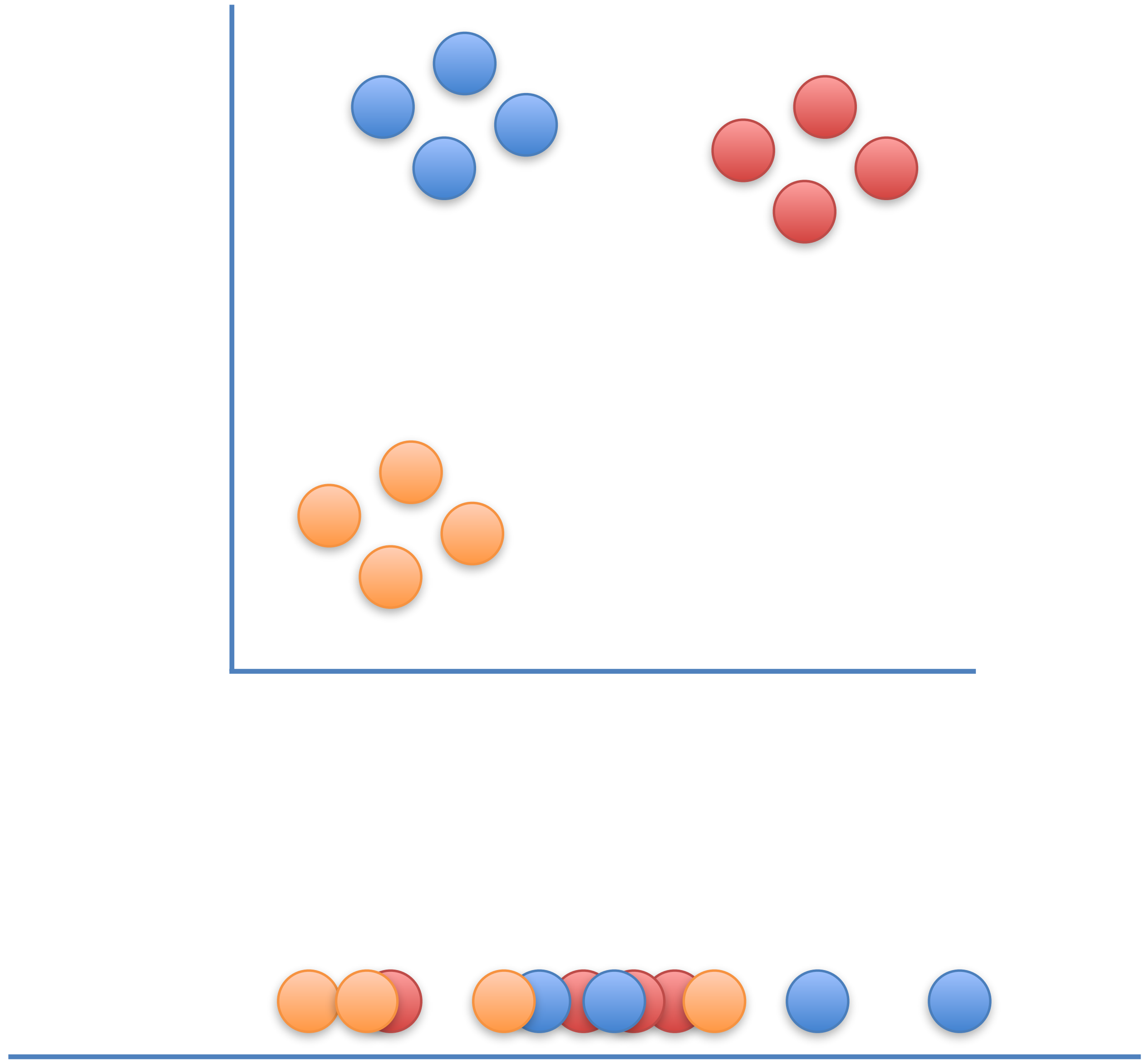


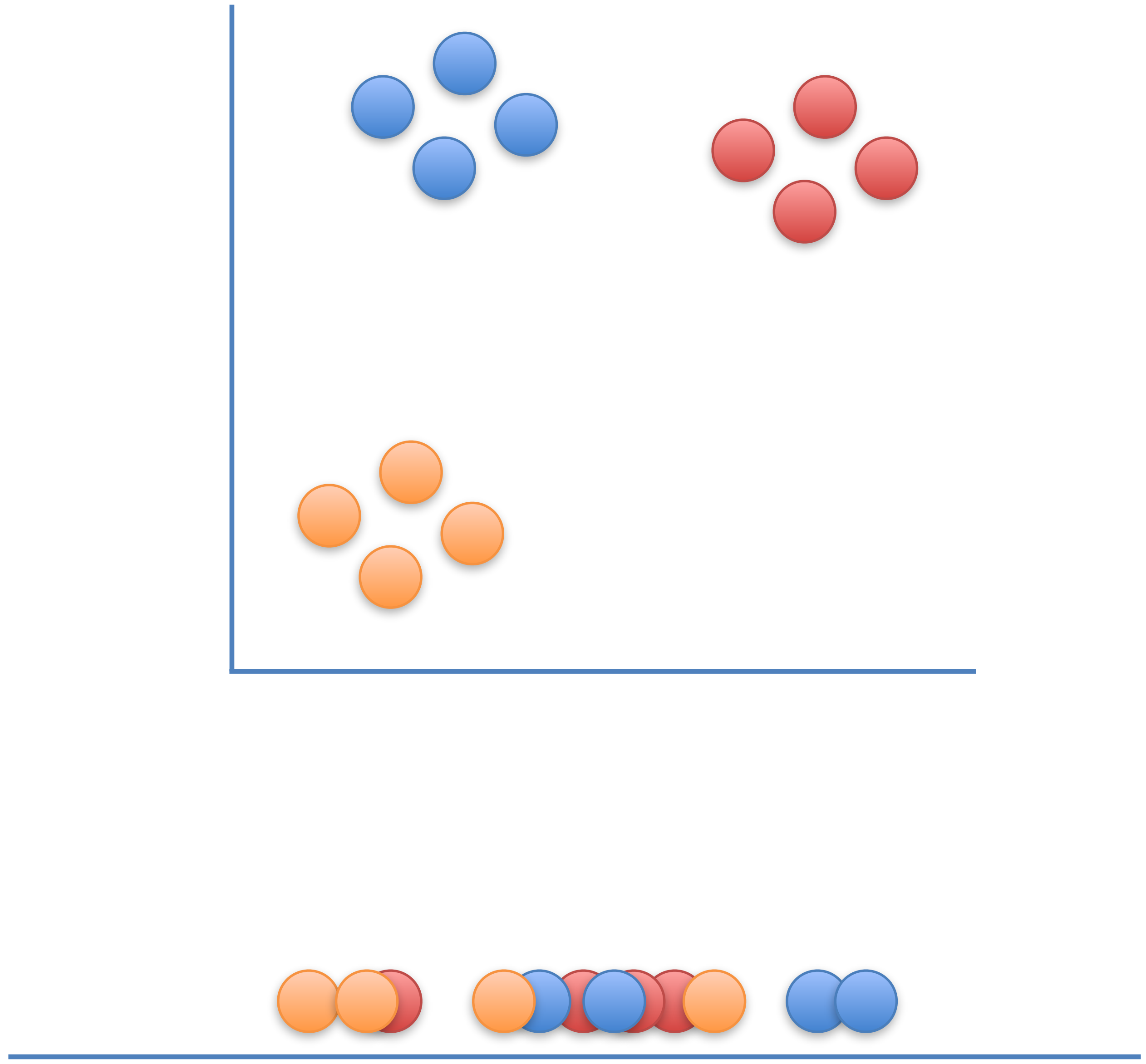


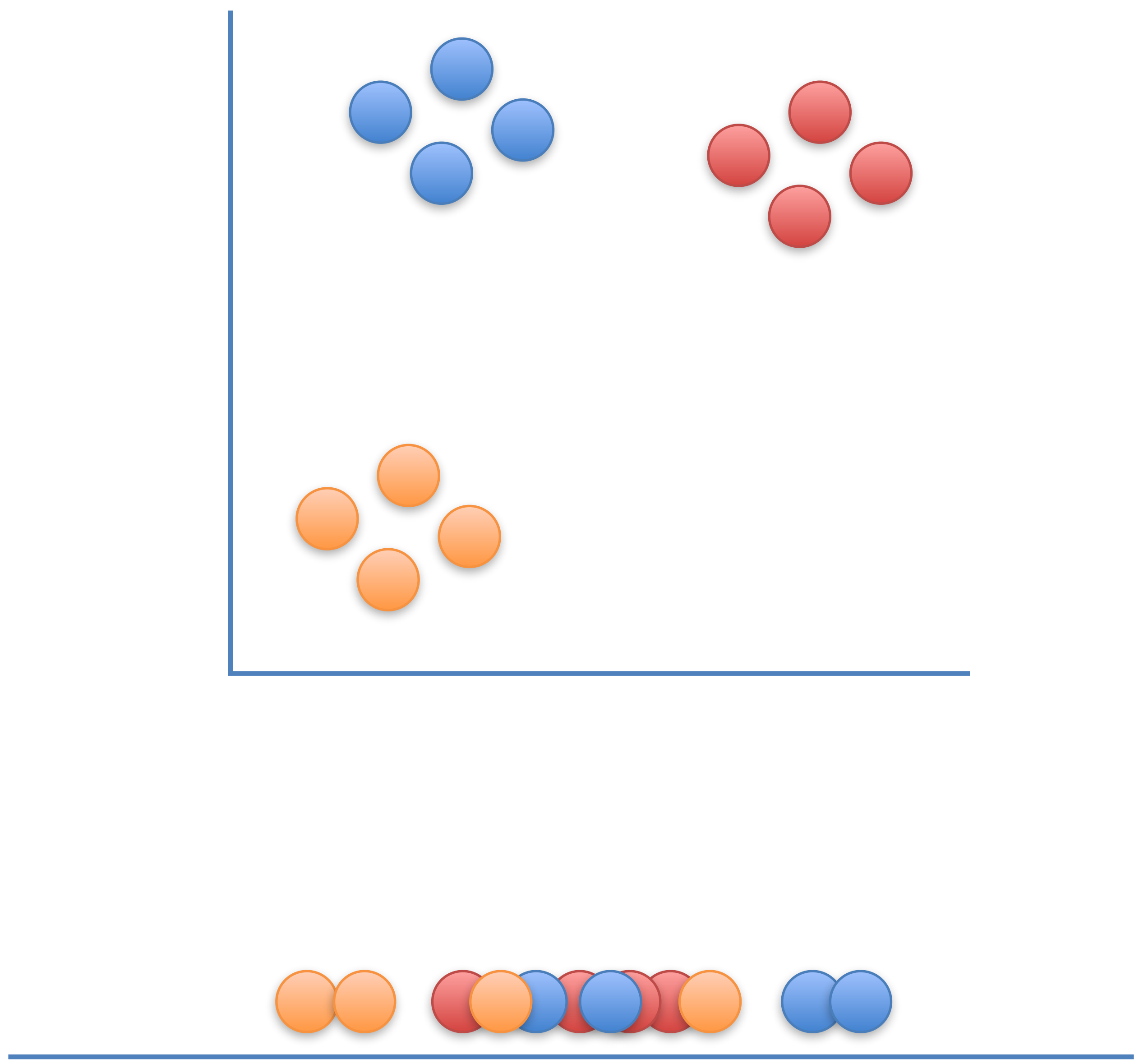


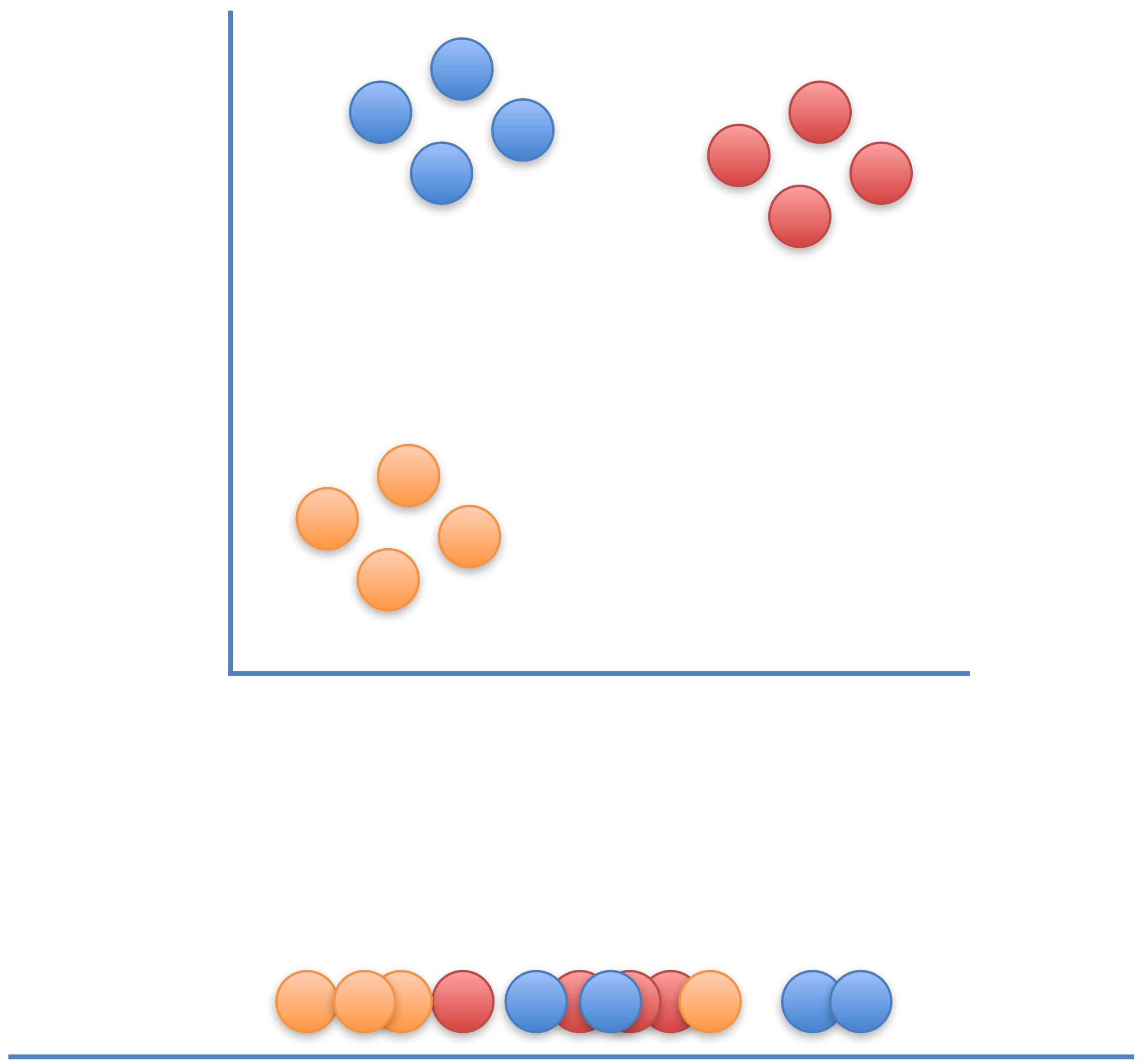


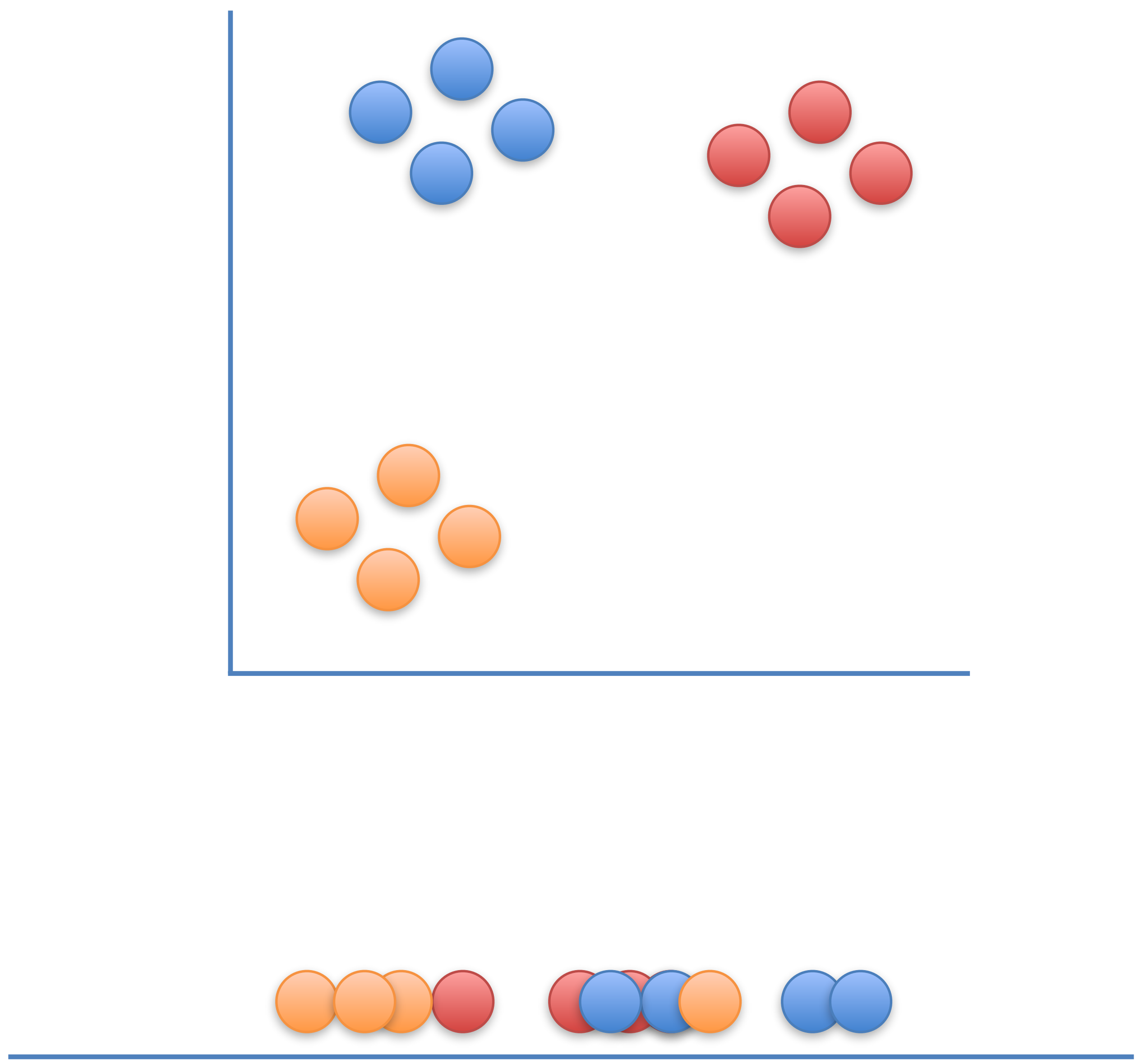


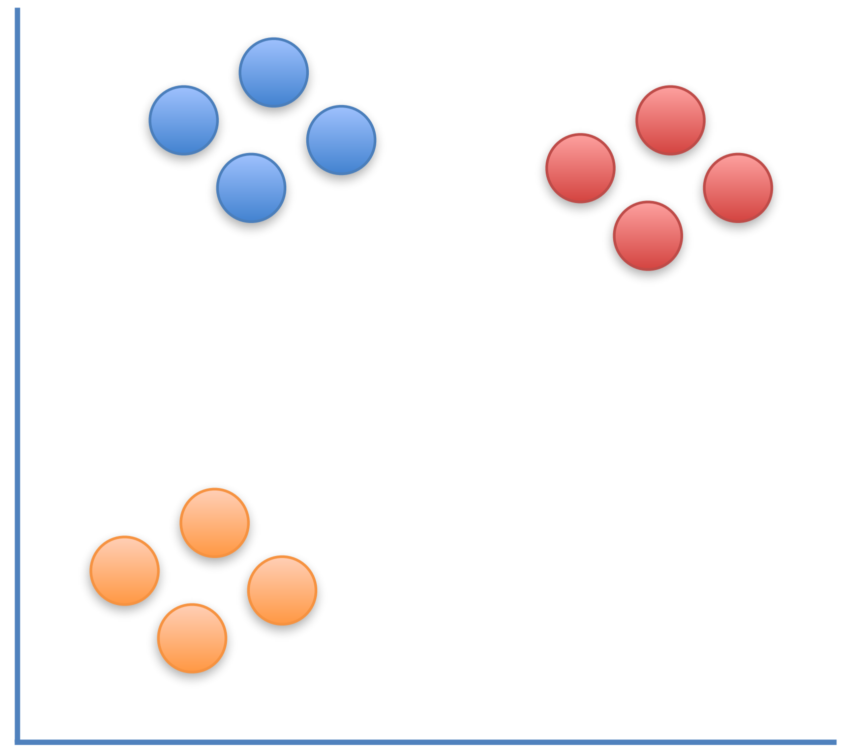


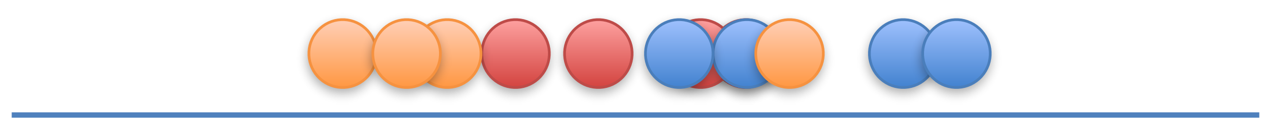


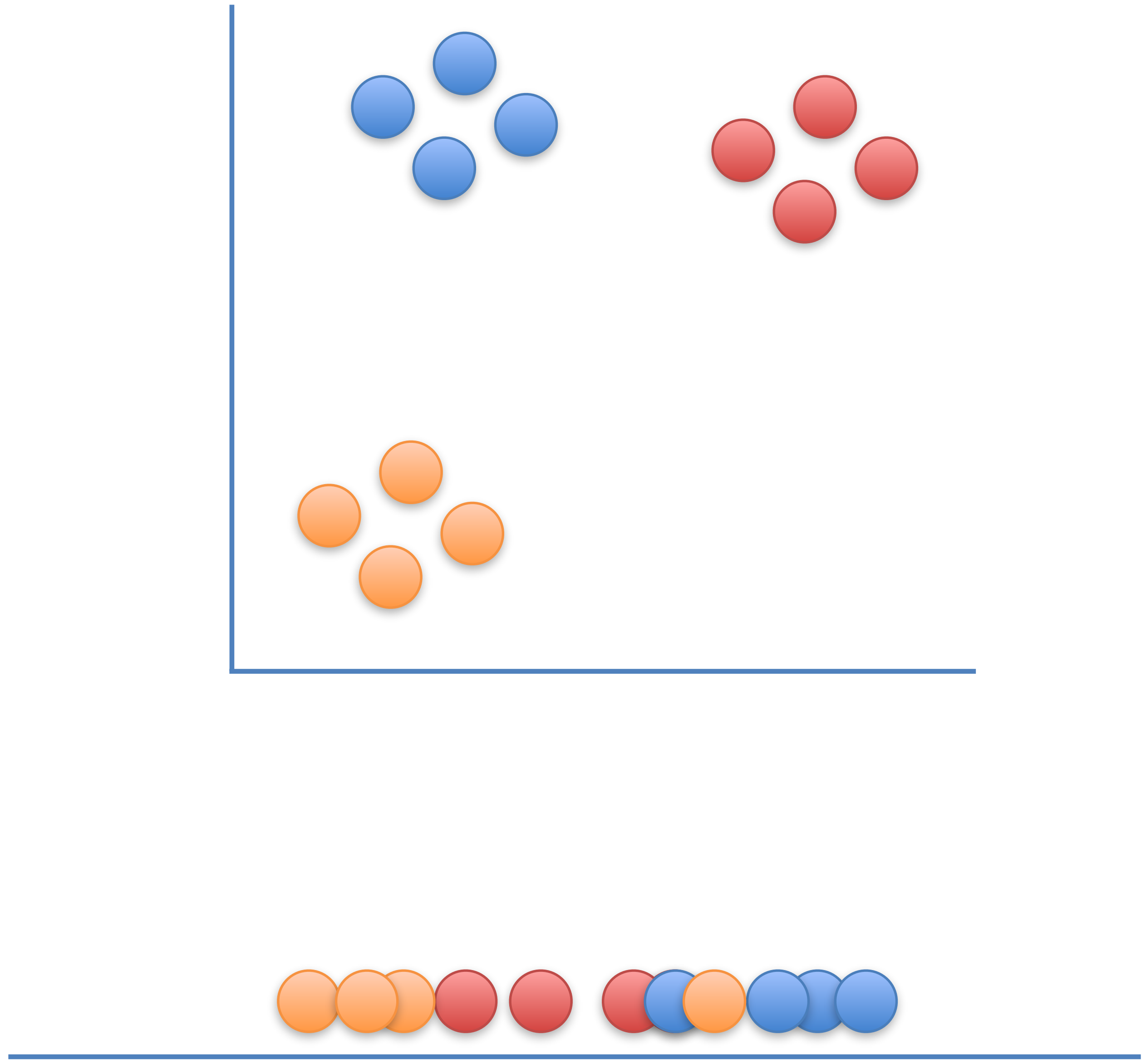


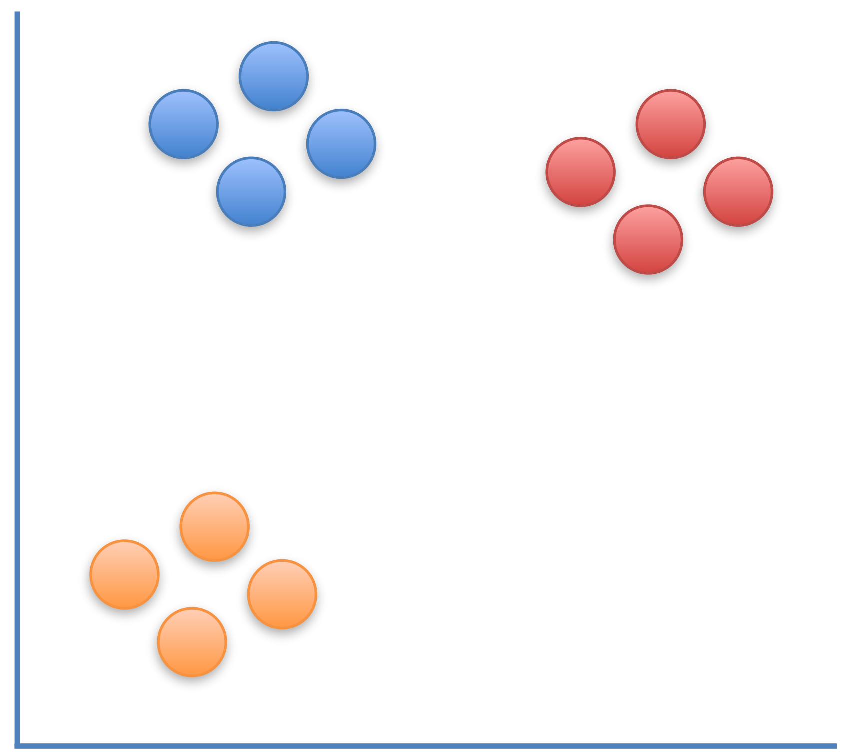


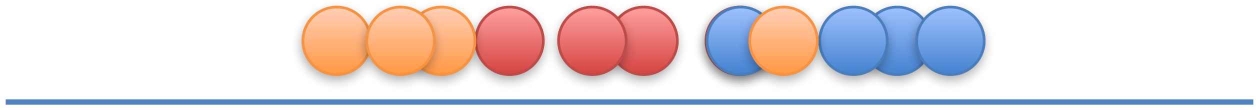


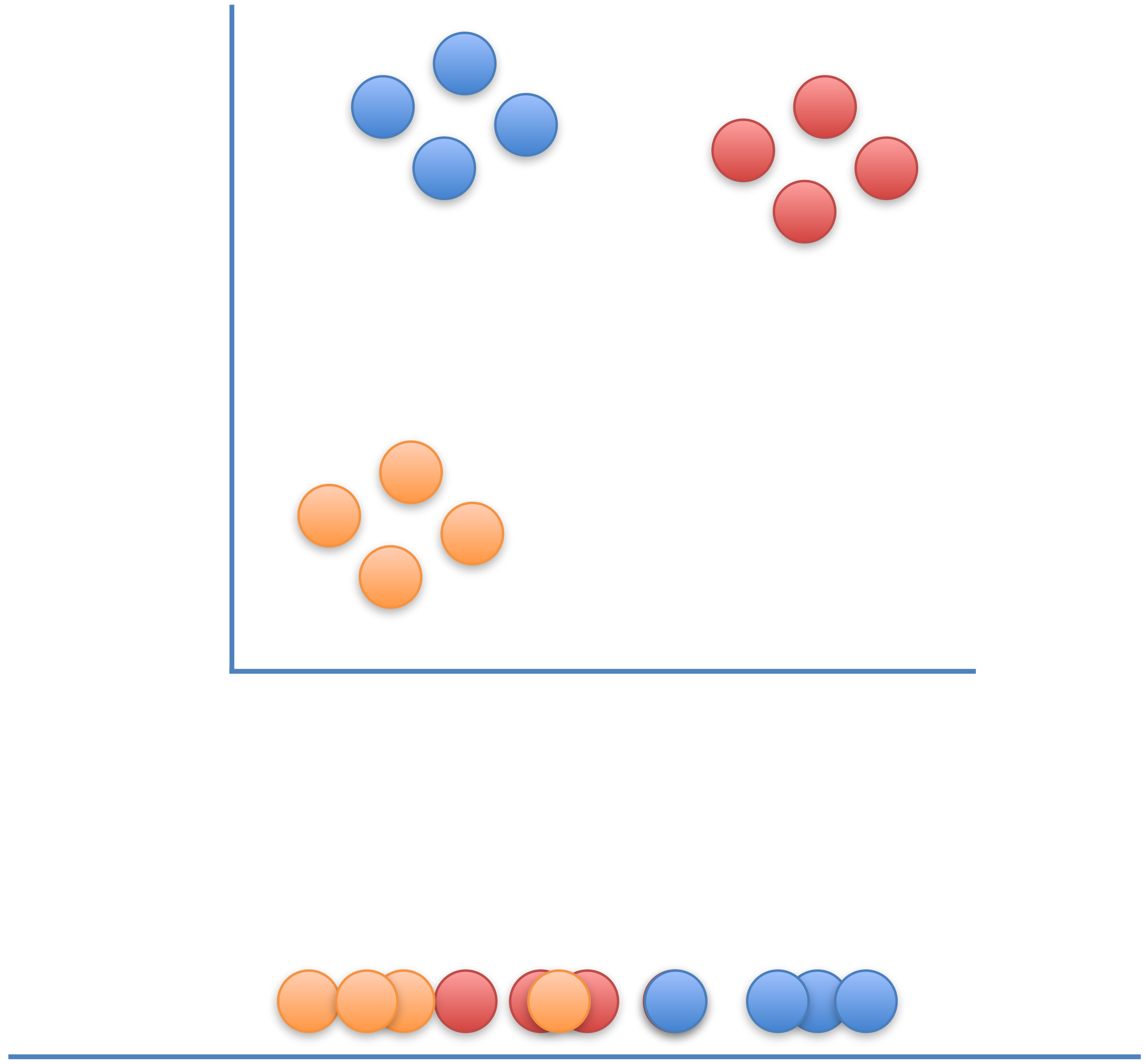


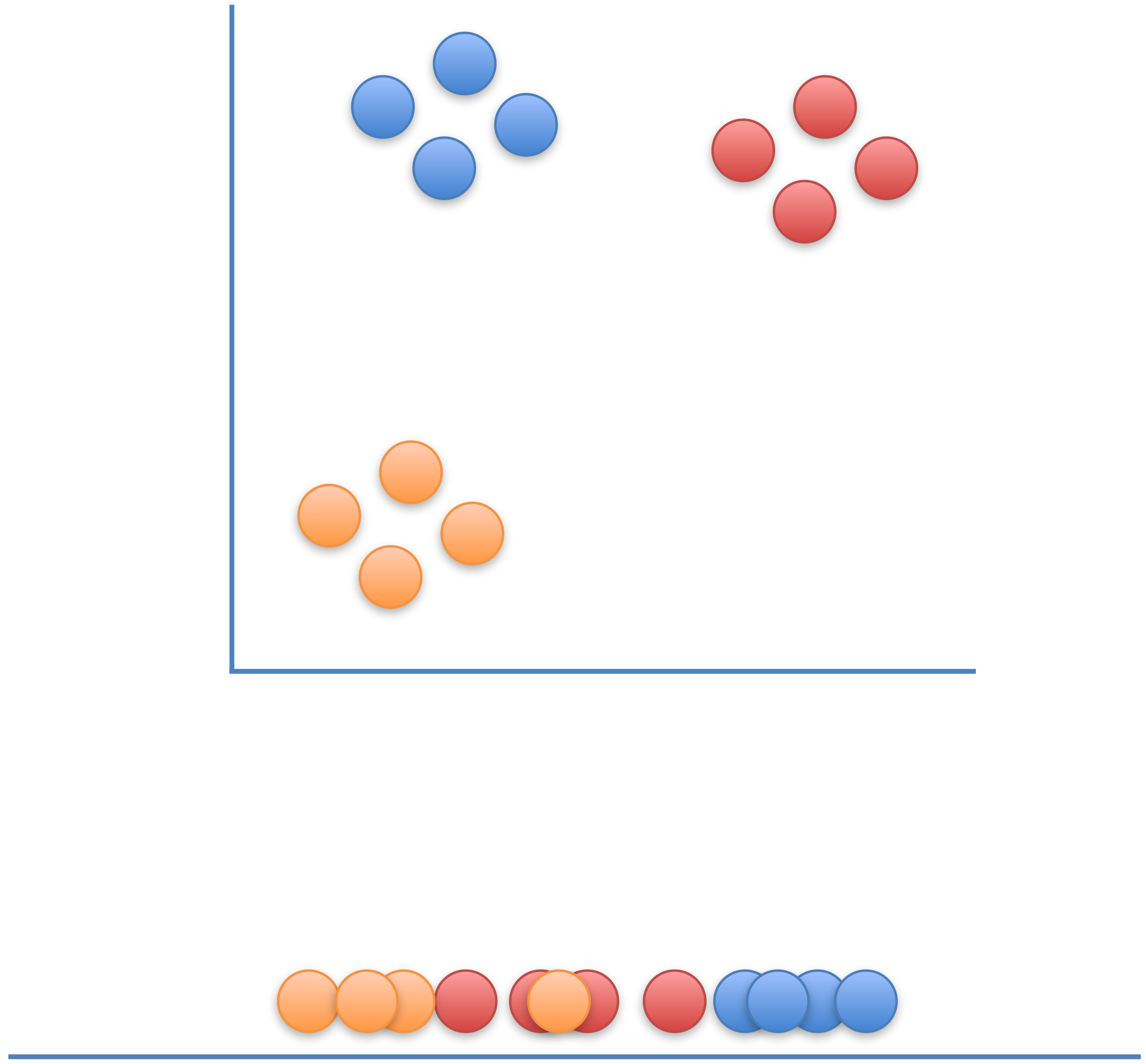


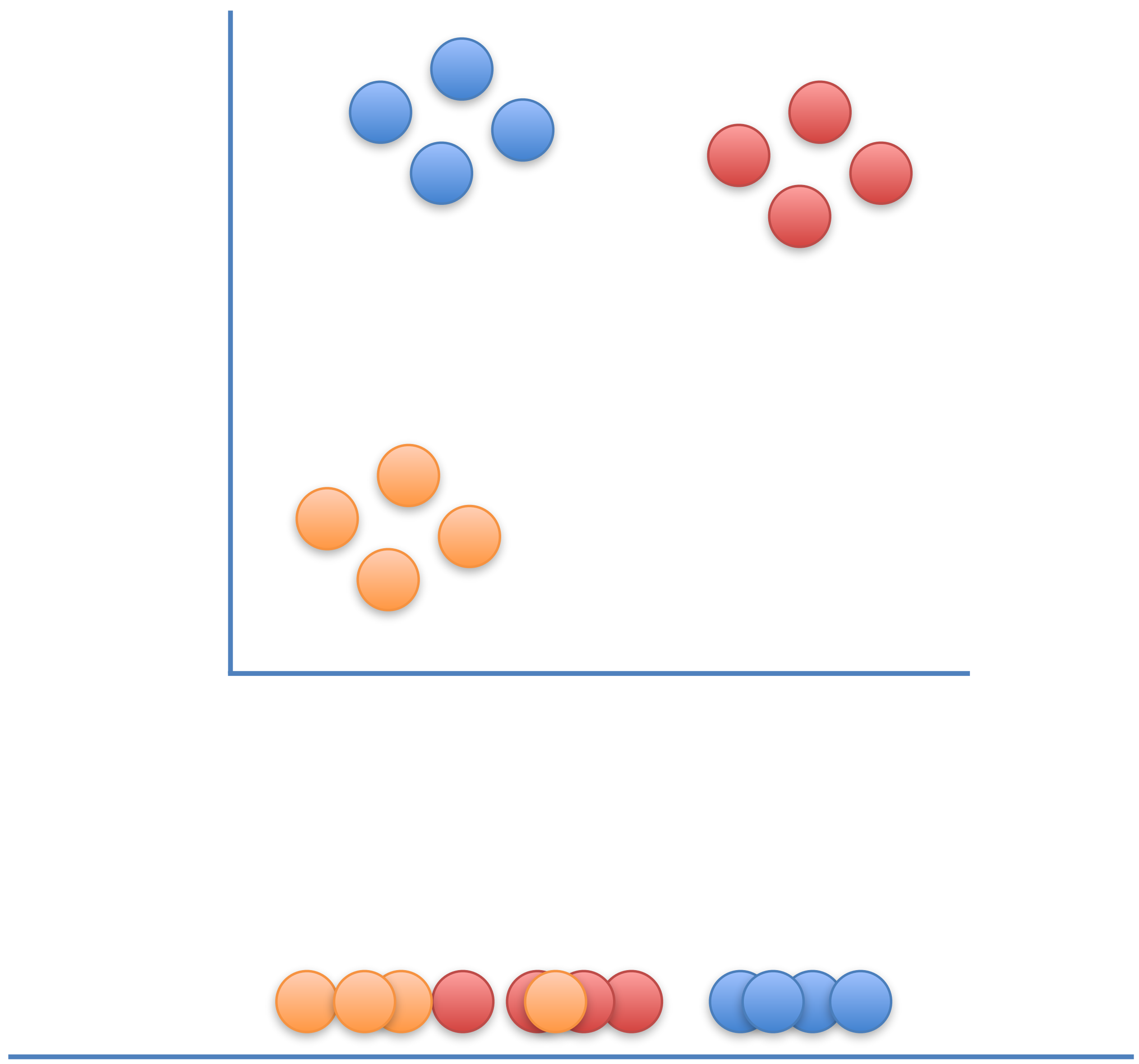


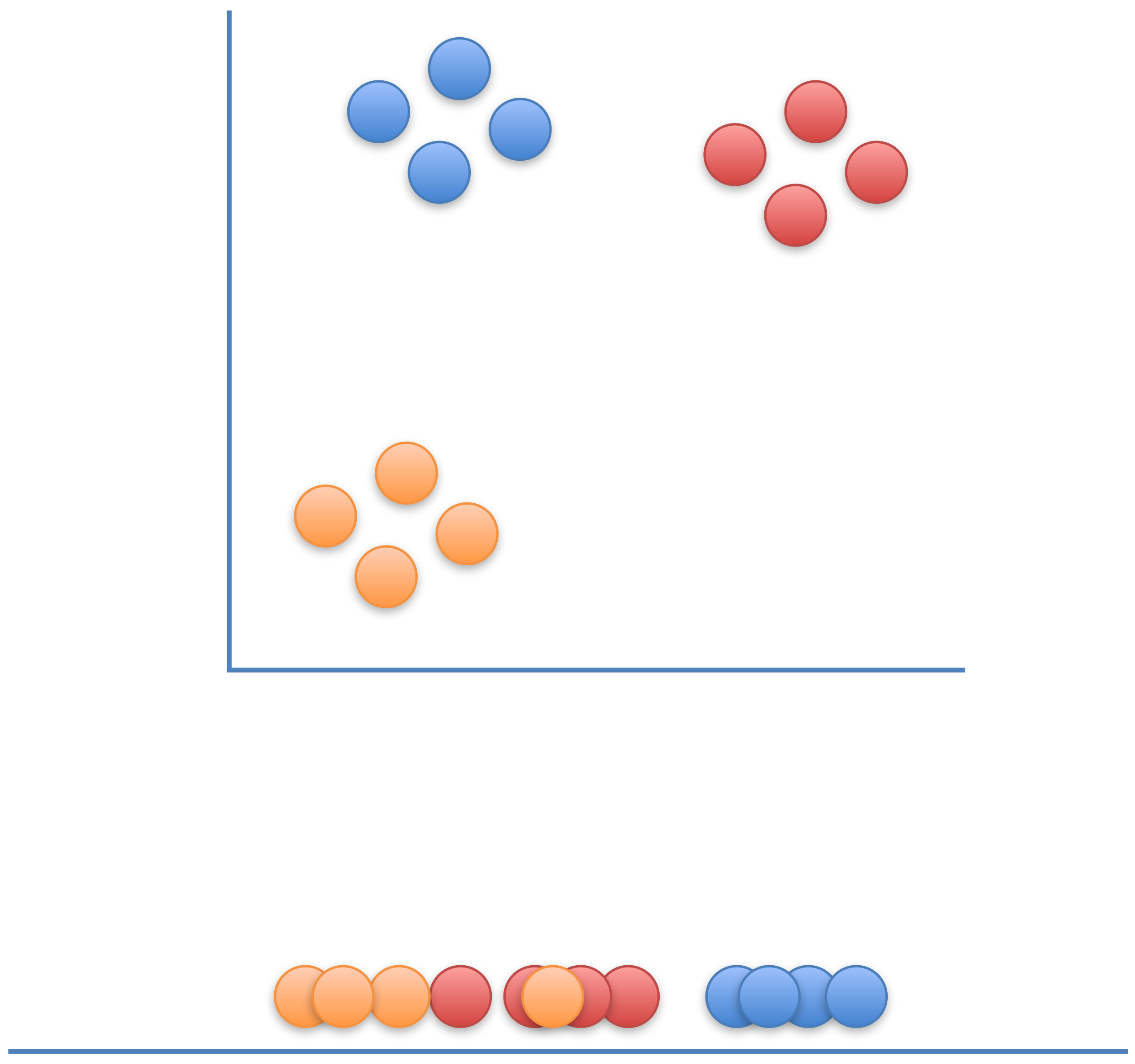


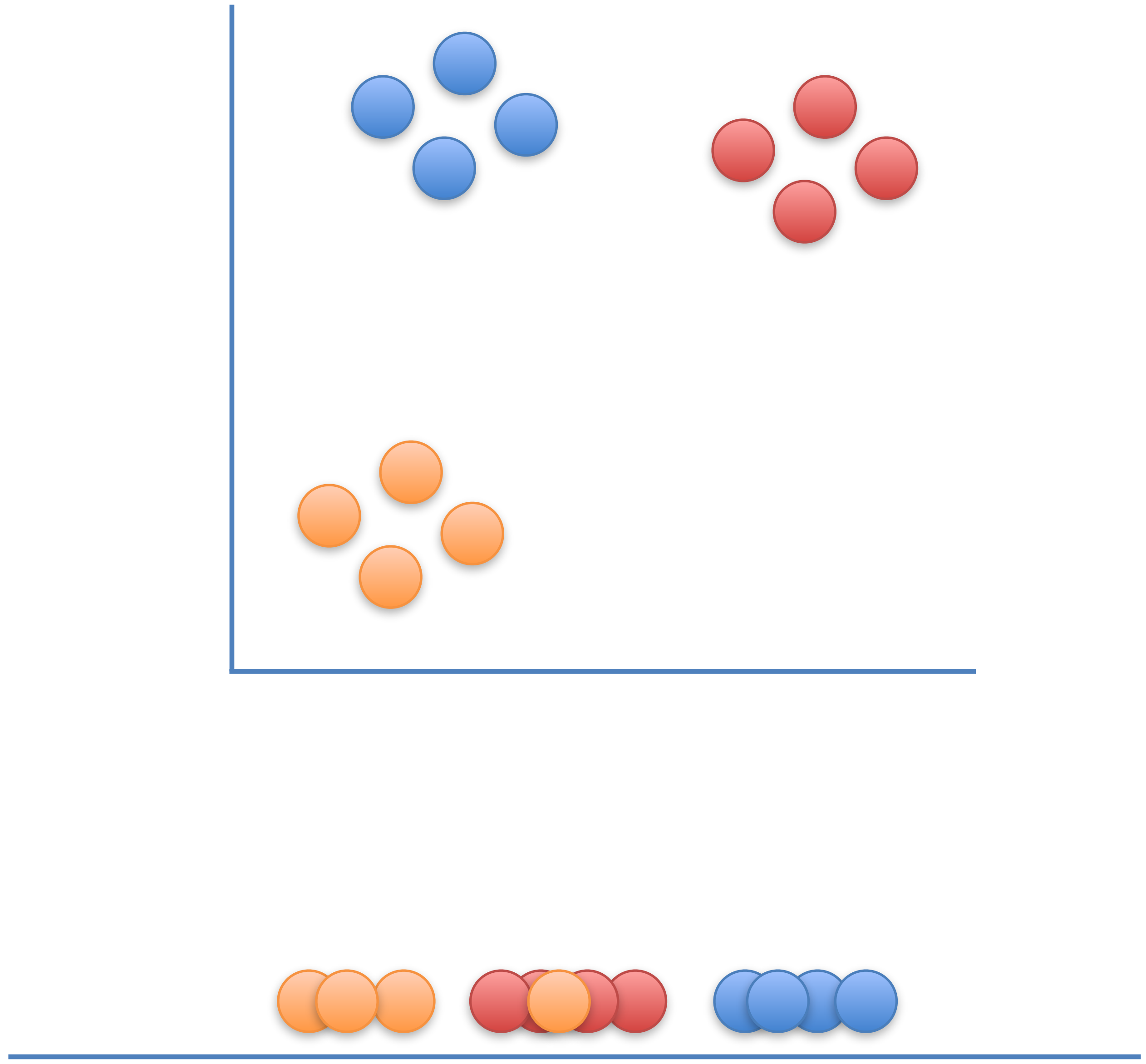


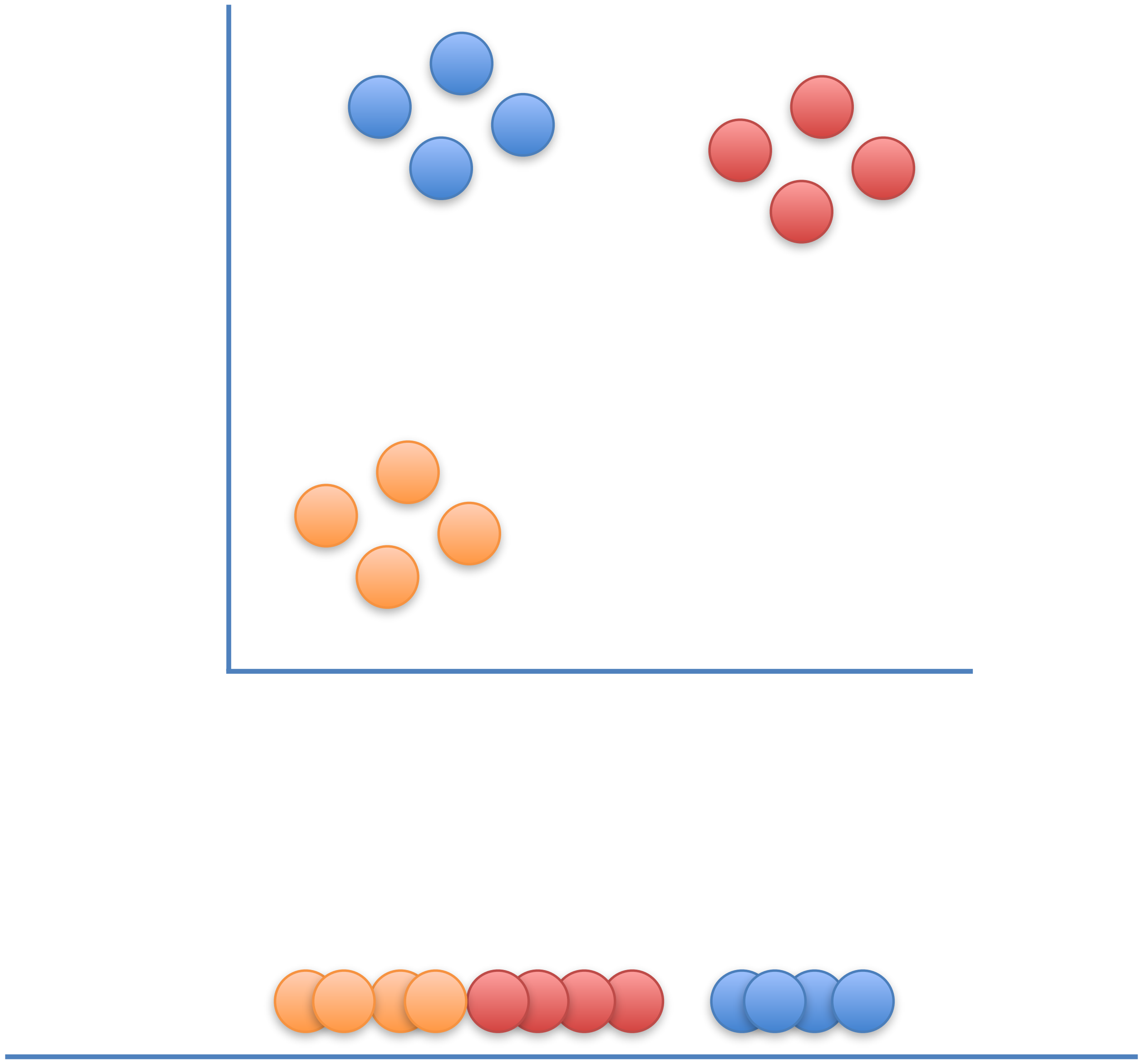


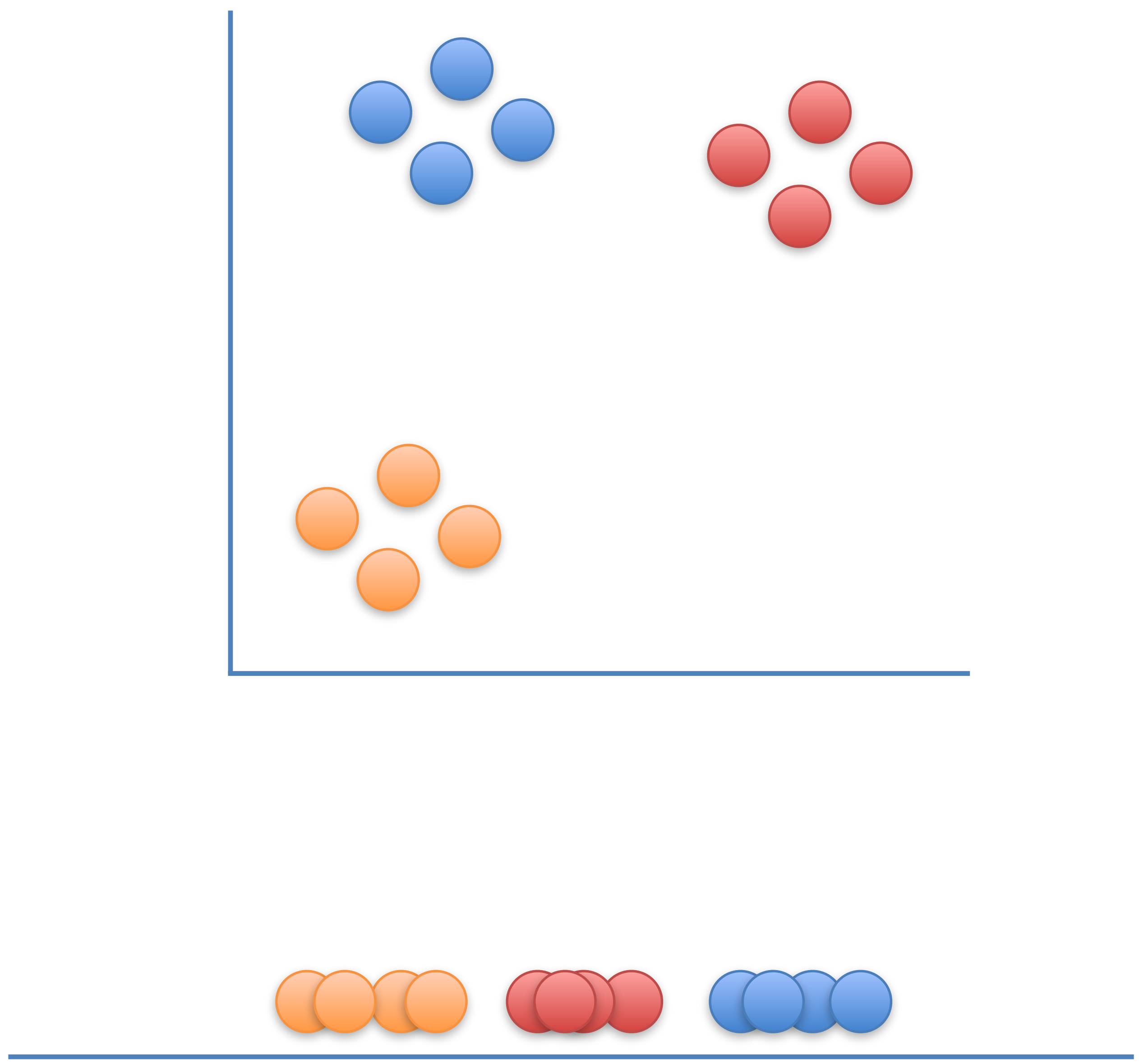


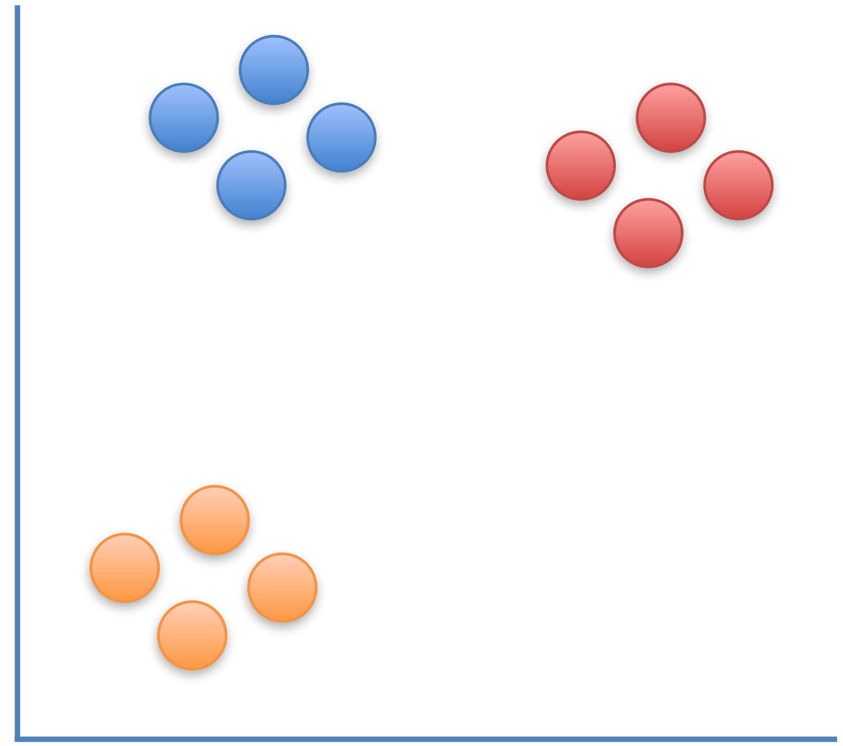




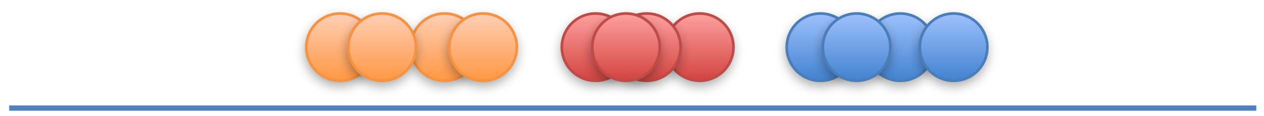


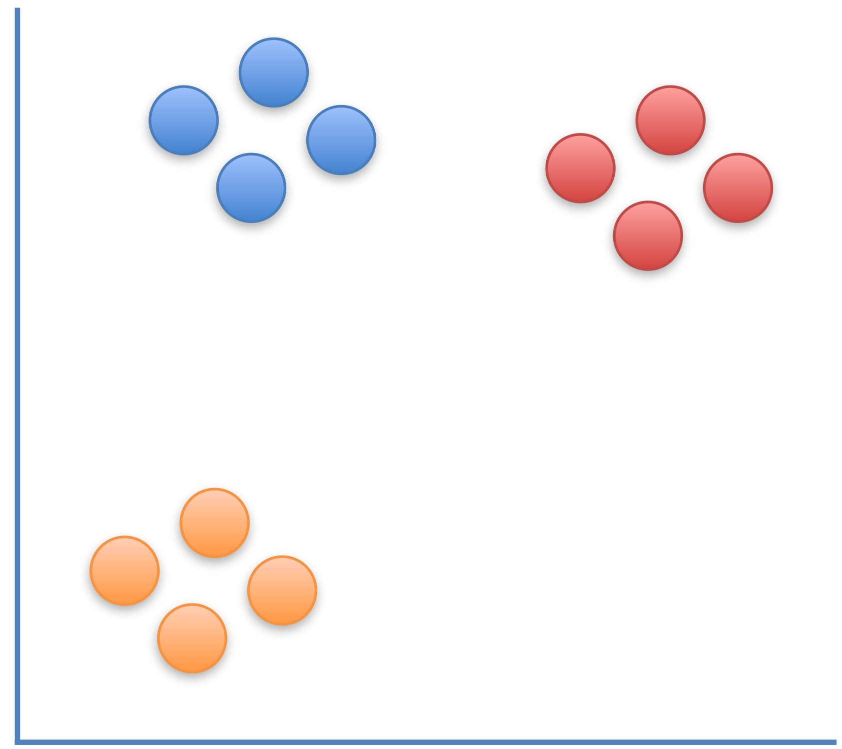




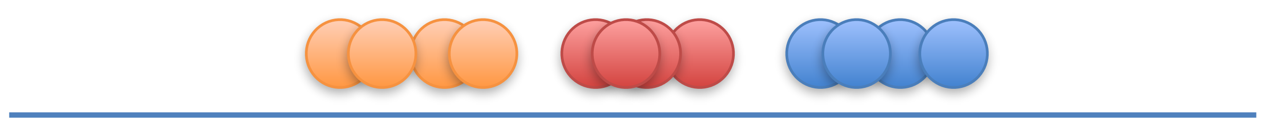


Triple BAM!!!!!

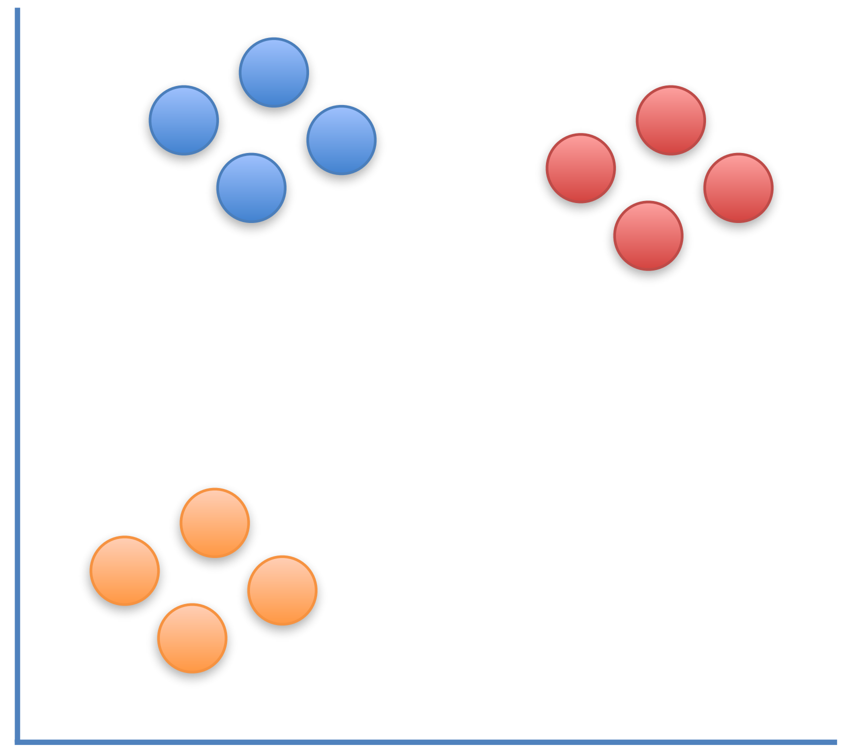




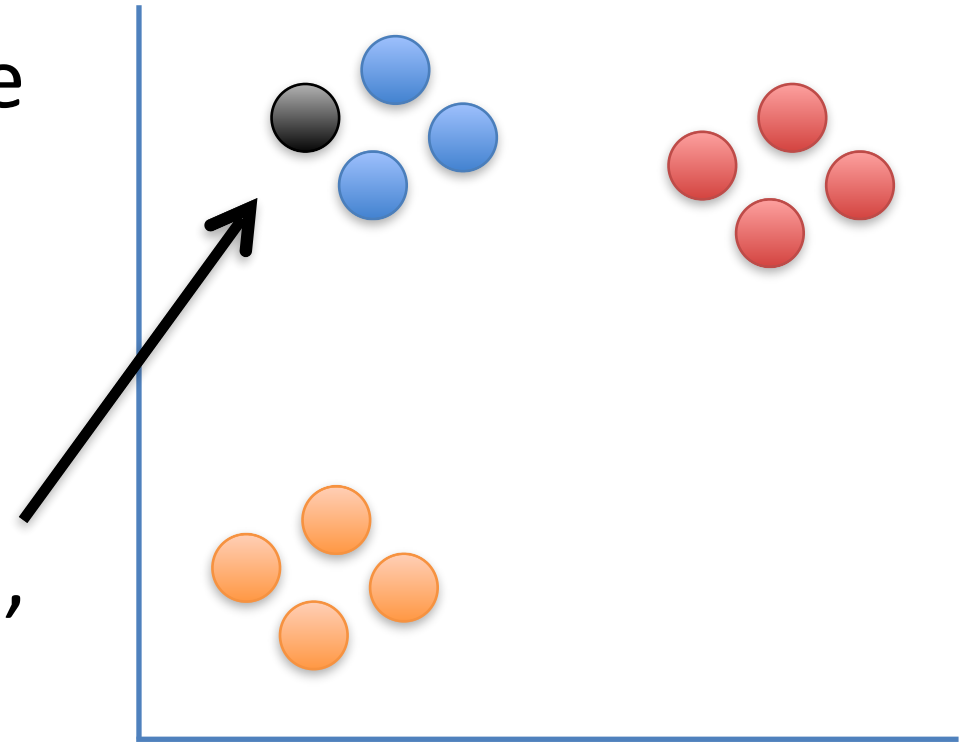
Now that we've seen the what t-SNE tries to do, let's dive into the nitty-gritty details of how it does what it does.



Step 1: Determine the "similarity" of all the points in the scatter plot.

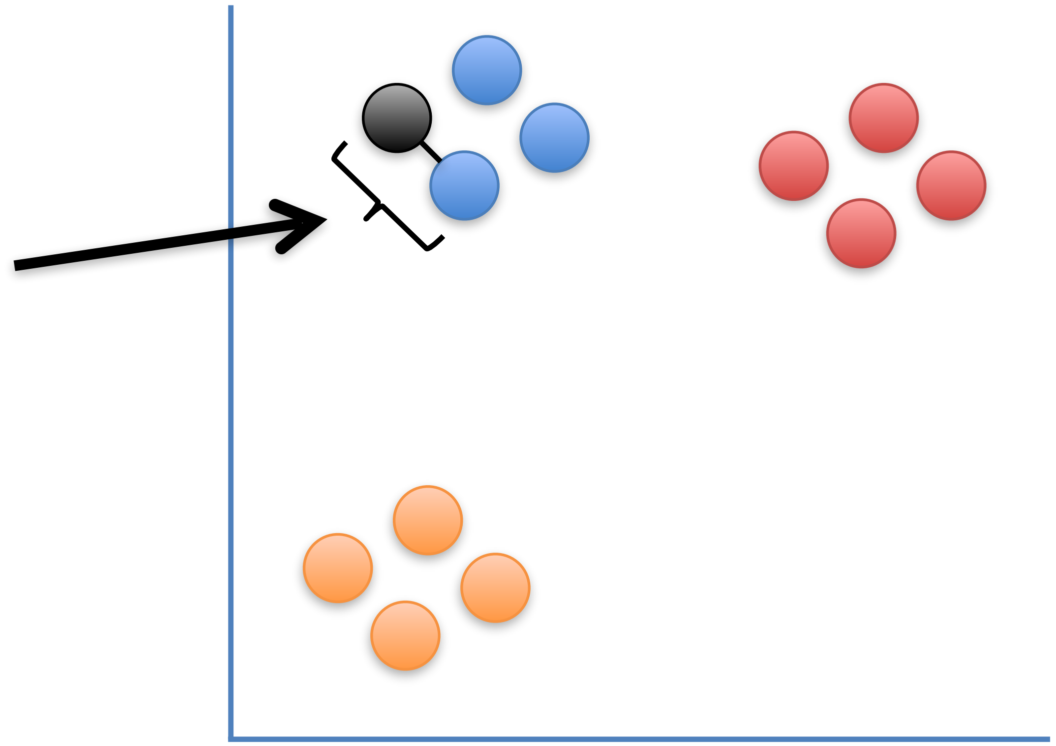


Step 1: Determine the "similarity" of all the points in the scatter plot.

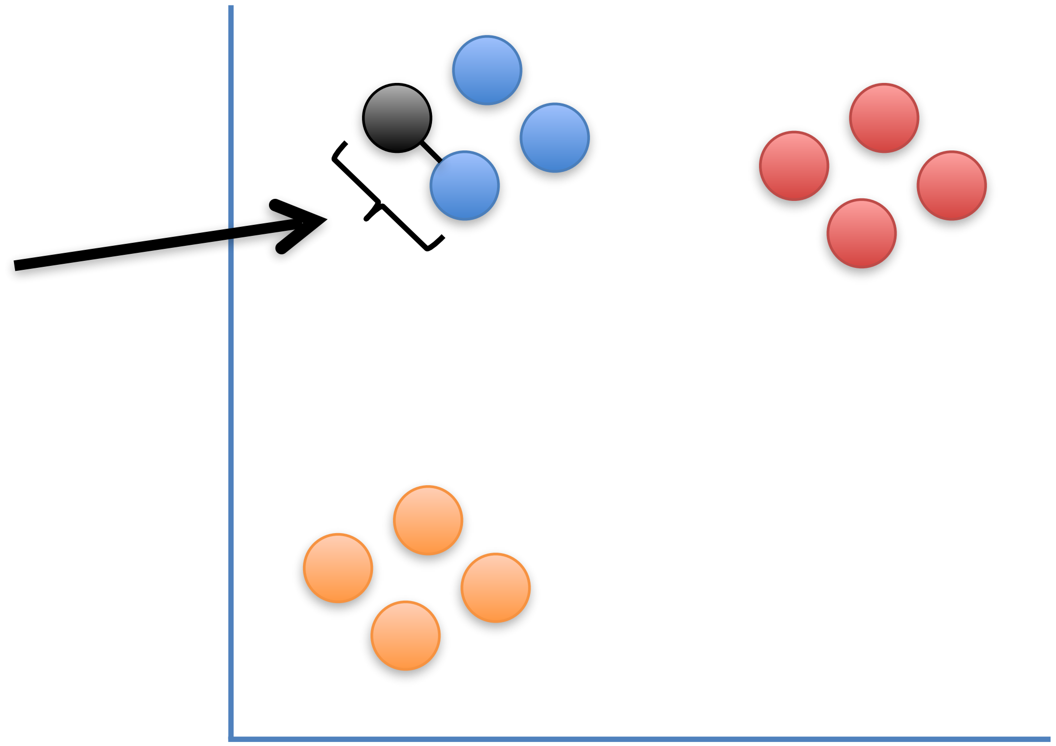


For this example, let's focus on determining the similarities between this point and all of the other points.

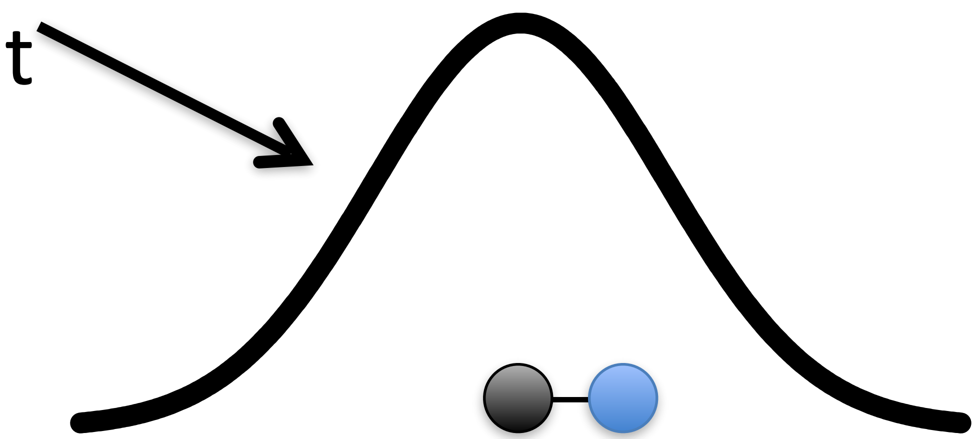
First, measure the distance between two points…



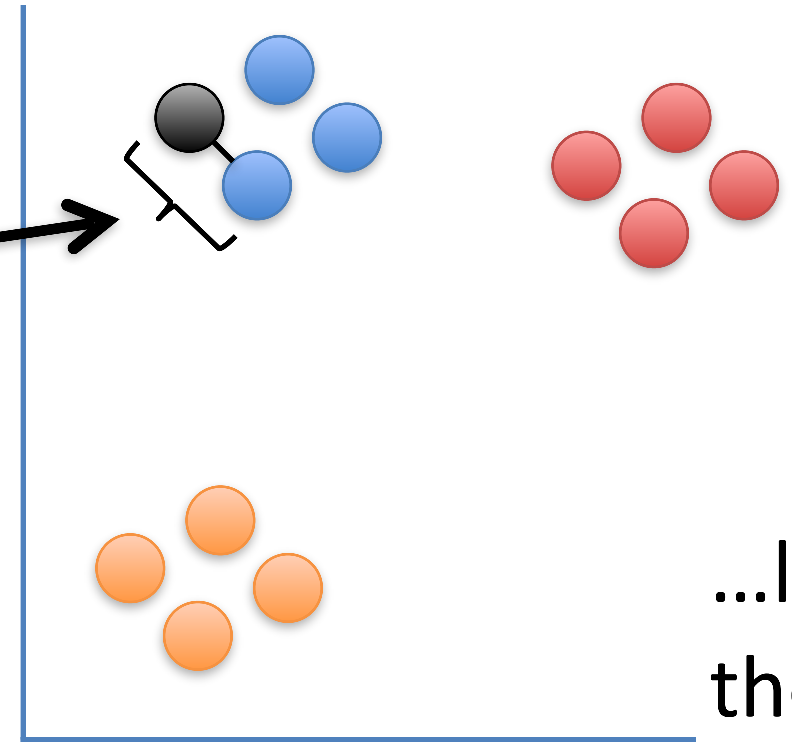
First, measure the distance between two points…



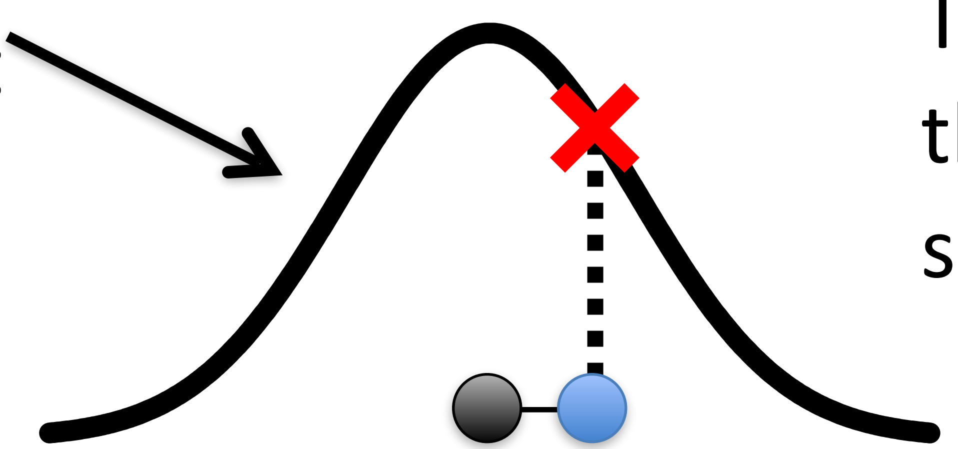
Then plot that distance on a normal curve that is centered on the point of interest…



First, measure the distance between two points...

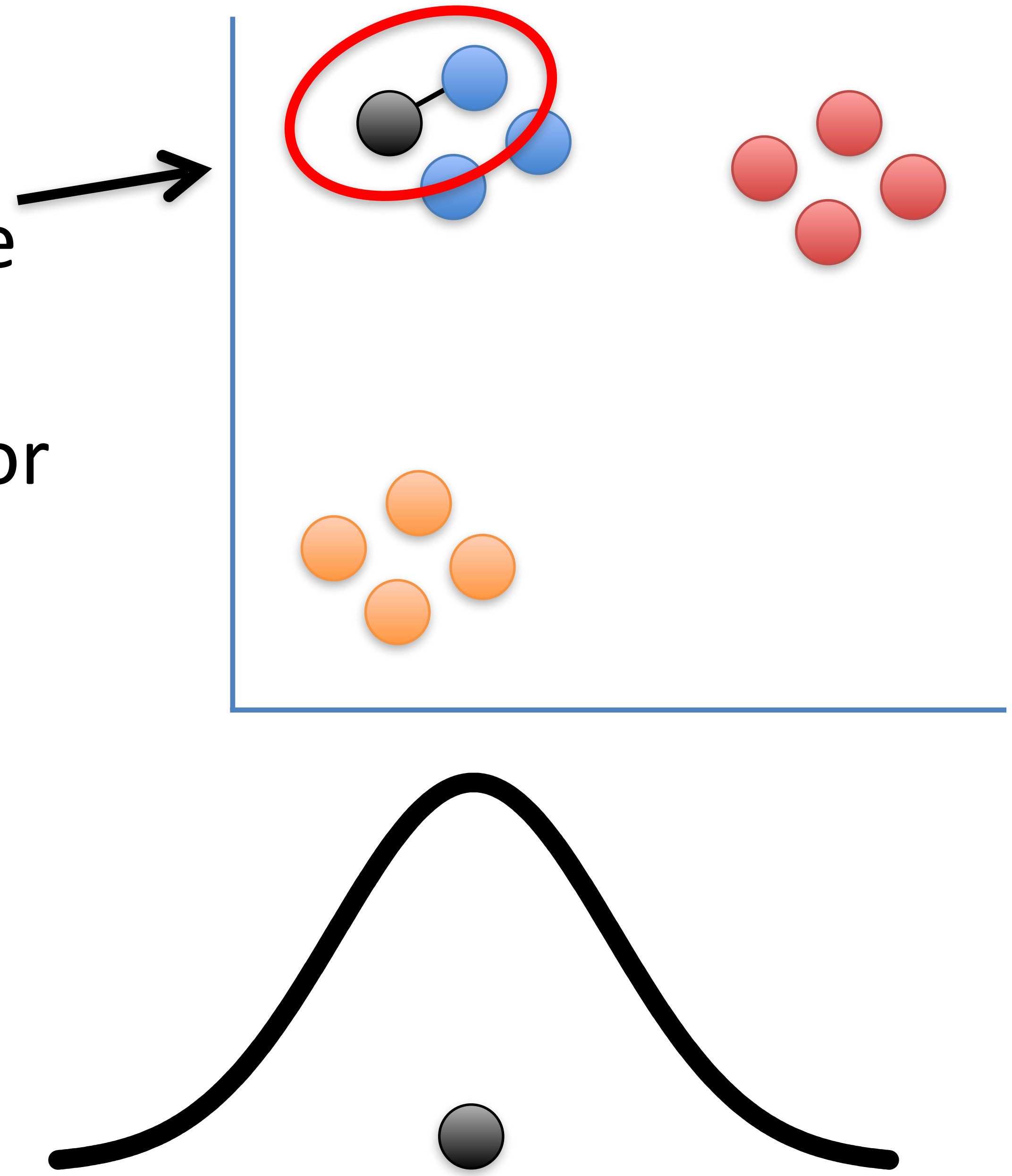


Then plot that distance on a normal curve that is centered on the point of interest...

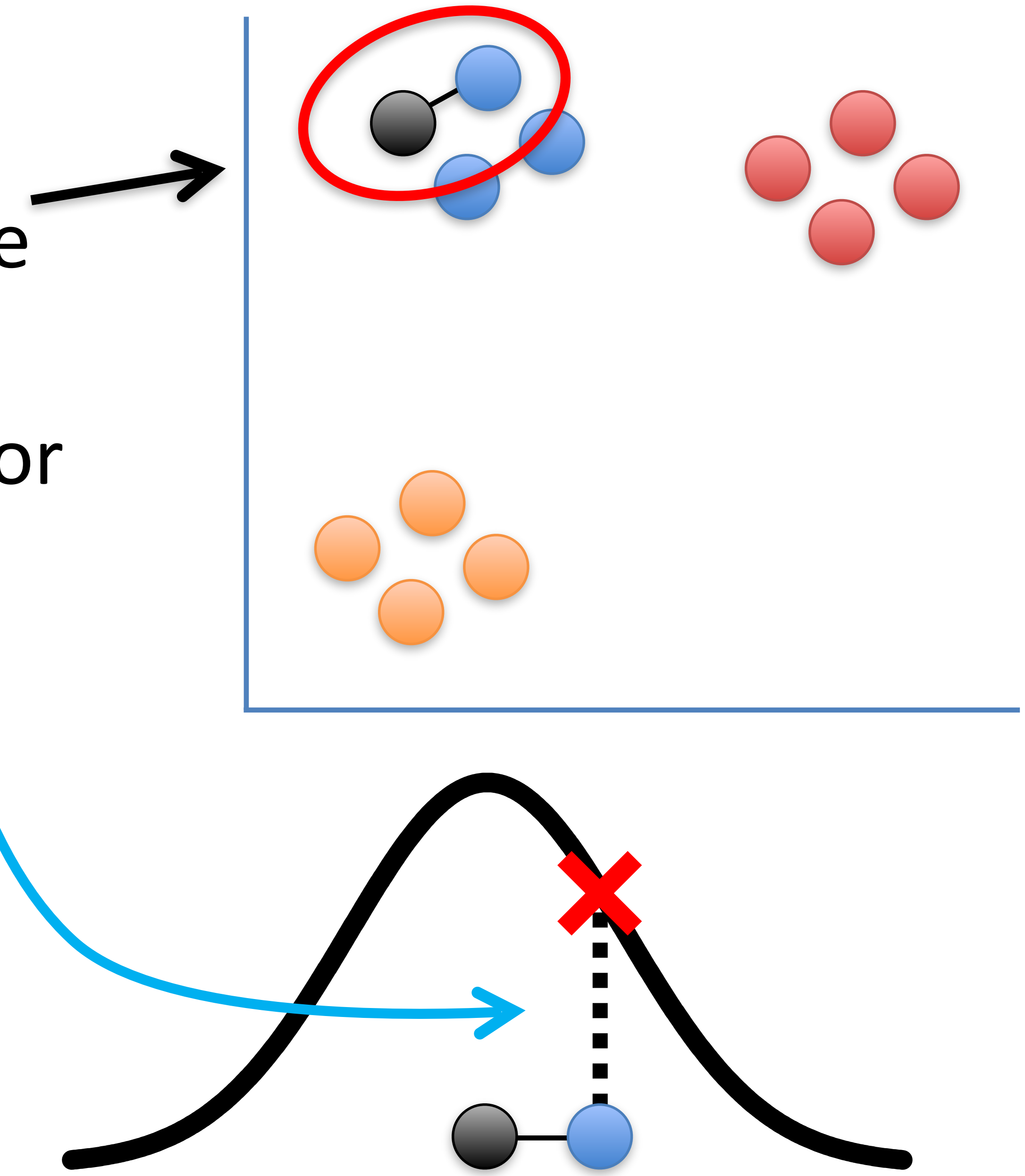


...lastly, draw a line from the point to the curve. The length of that line is the "unscaled similarity".

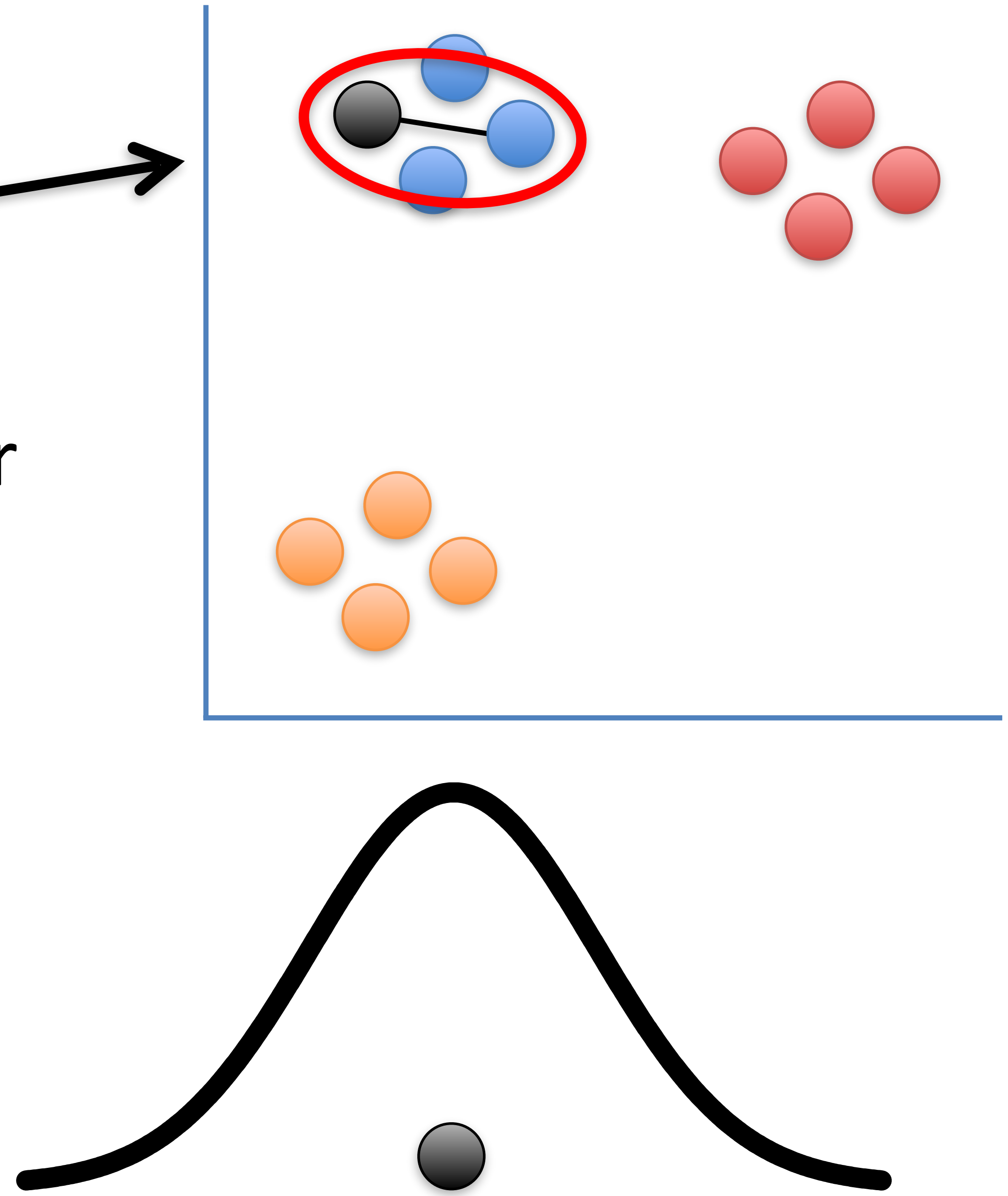
Now we calculate the "unscaled similarity" for this pair of points.



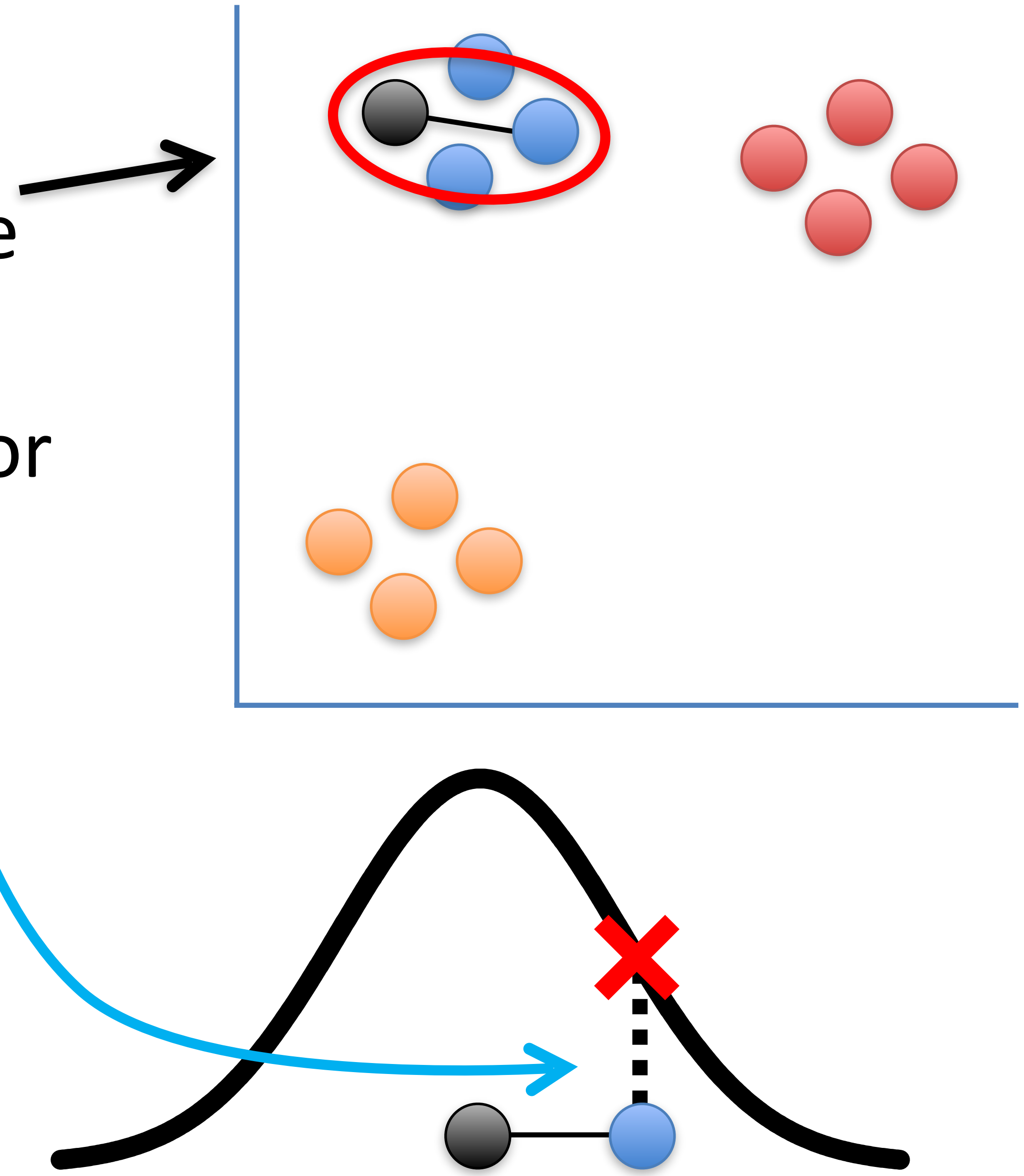
Now we calculate the "unscaled similarity" for this pair of points.



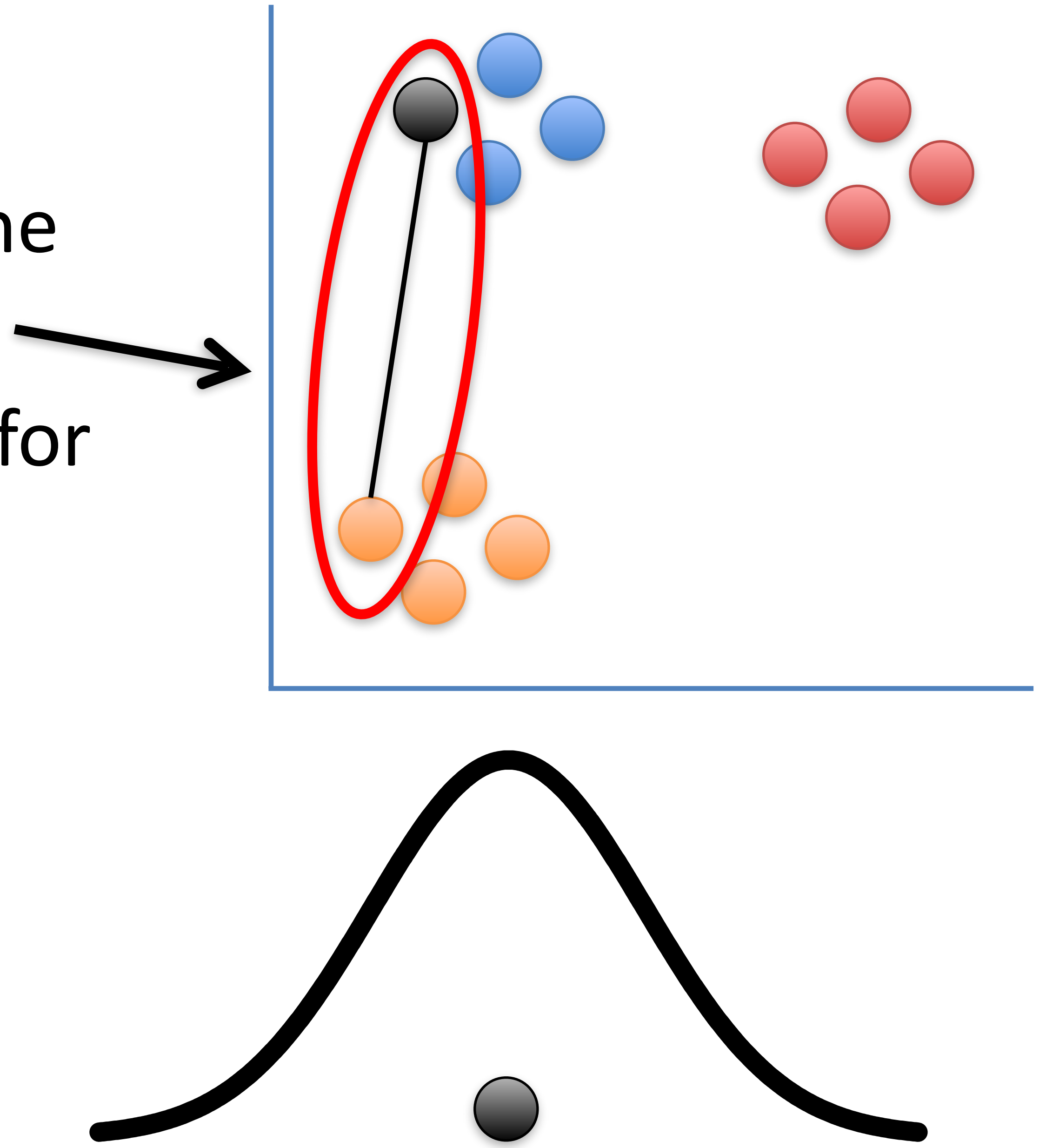
Now we calculate the "unscaled similarity" for this pair of points.



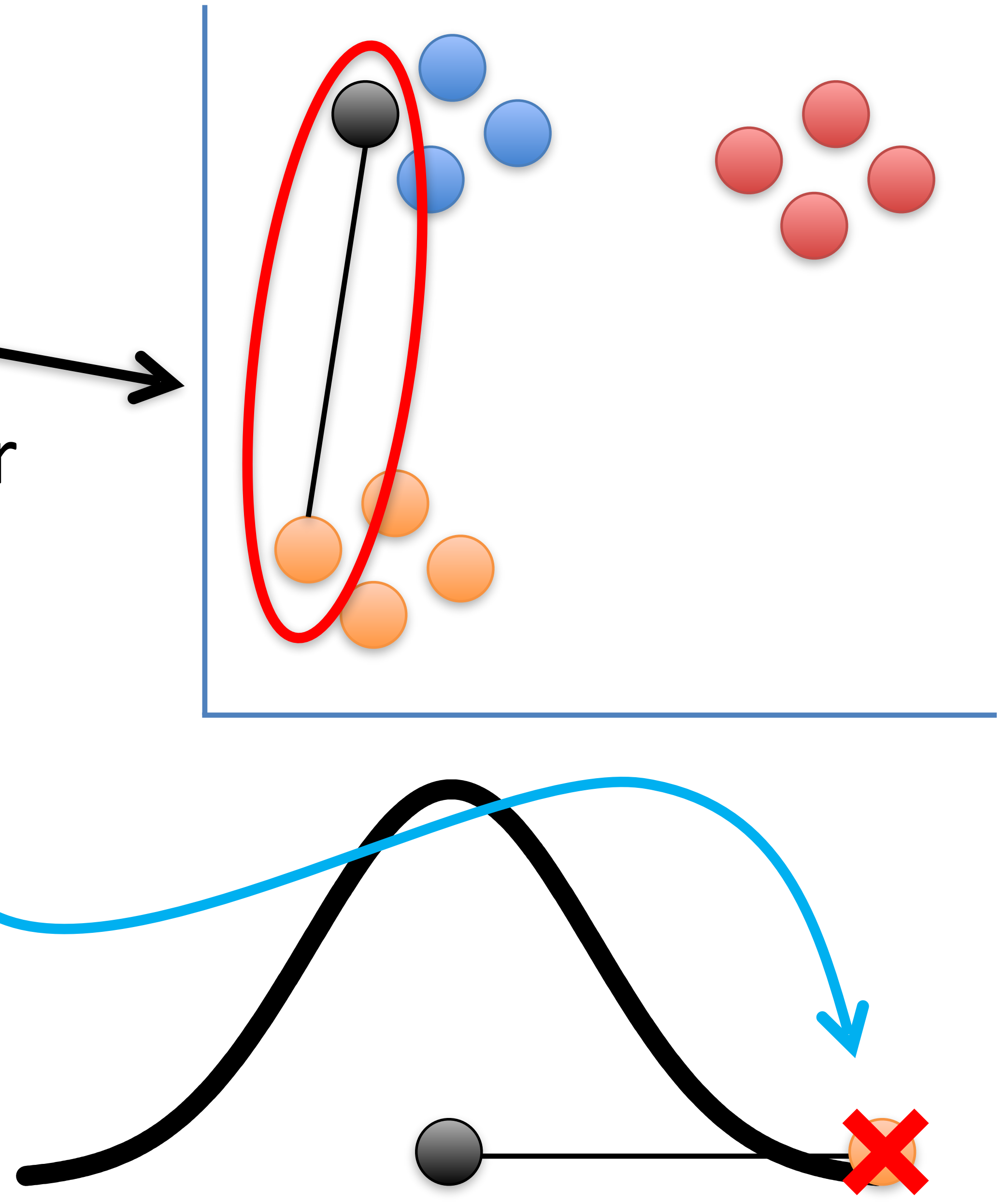
Now we calculate the “unscaled similarity” for this pair of points.



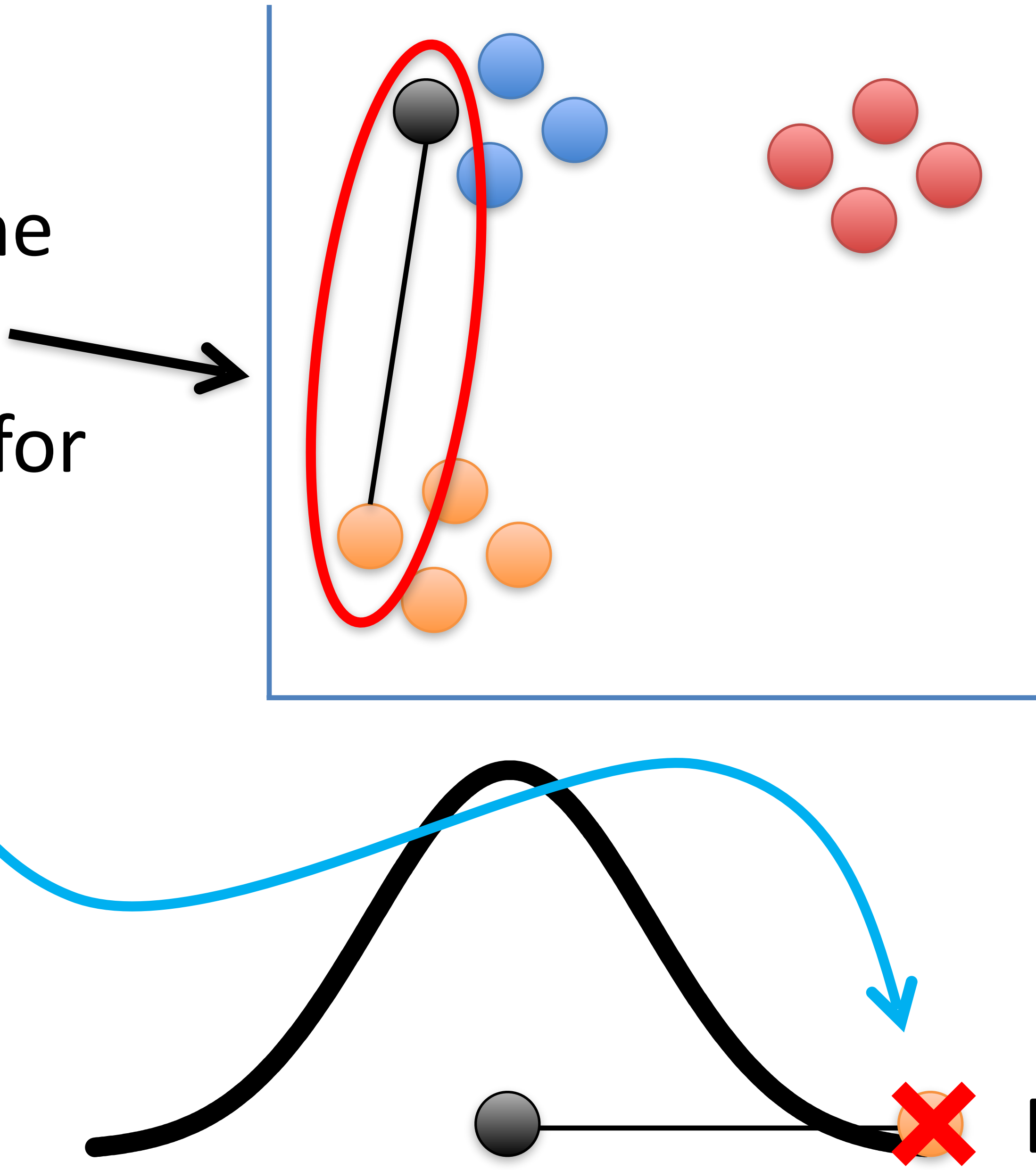
Now we calculate the "unscaled similarity" for this pair of points.



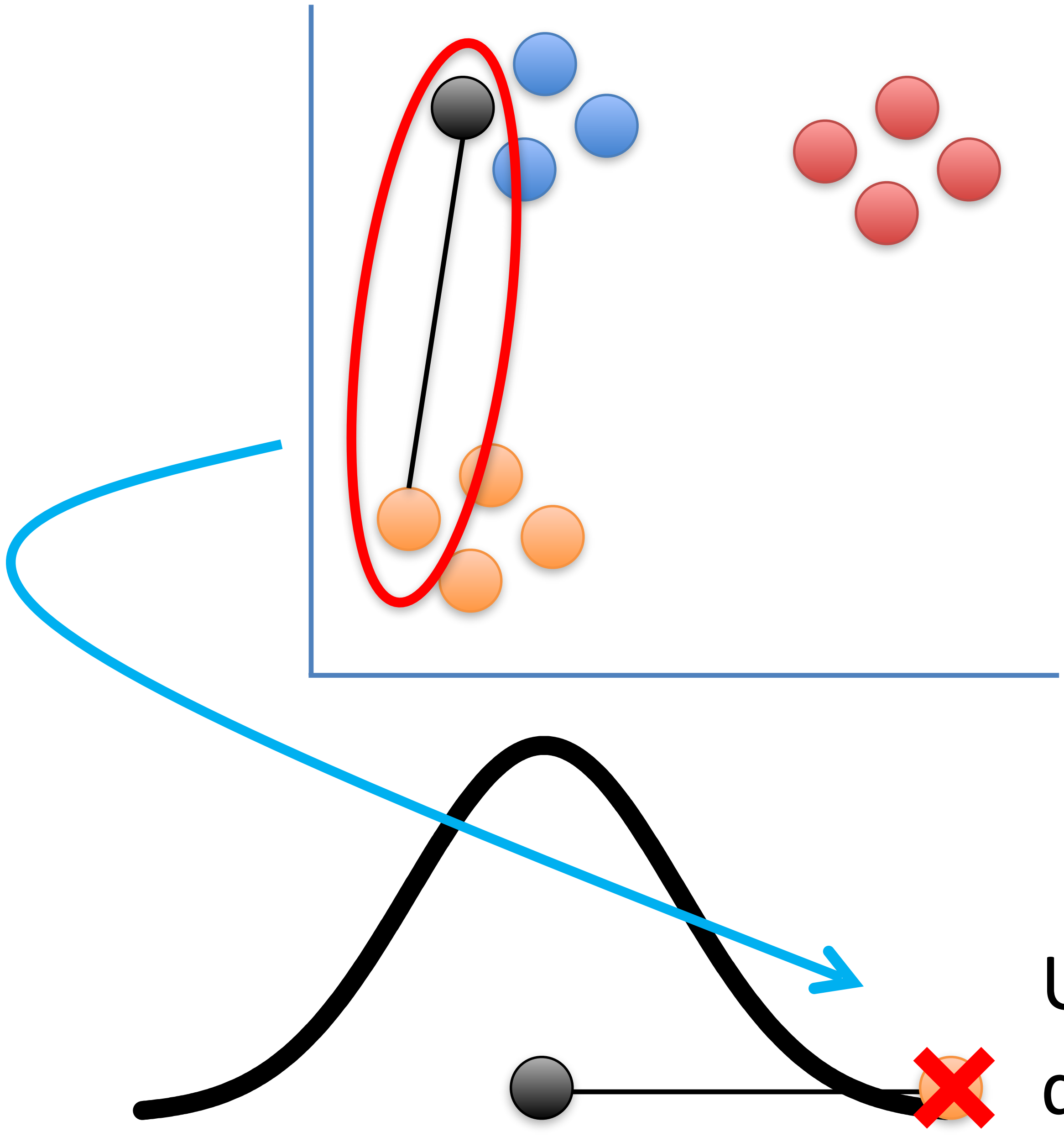
Now we calculate the "unscaled similarity" for this pair of points.



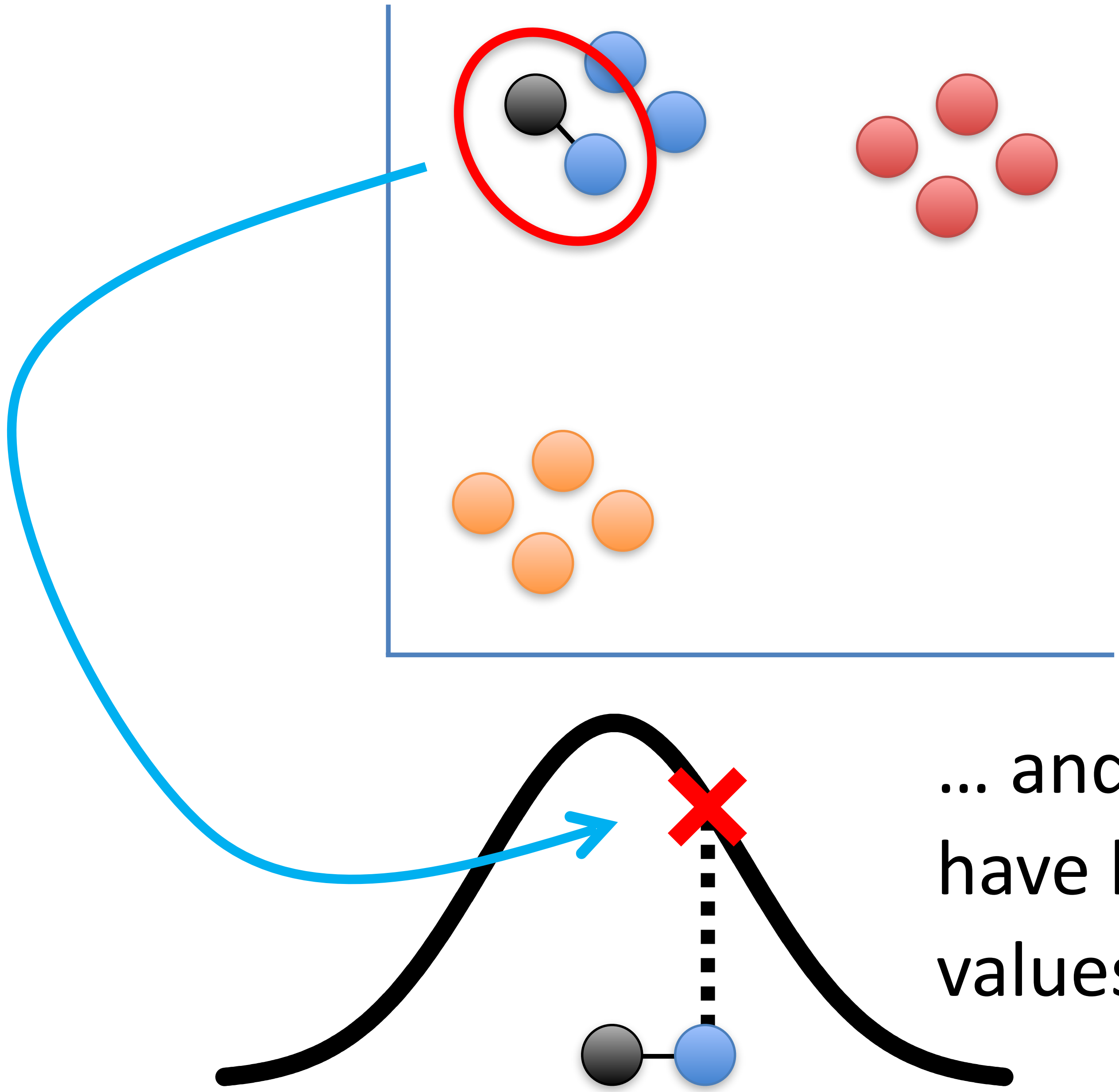
Now we calculate the "unscaled similarity" for this pair of points.



Etc. etc...

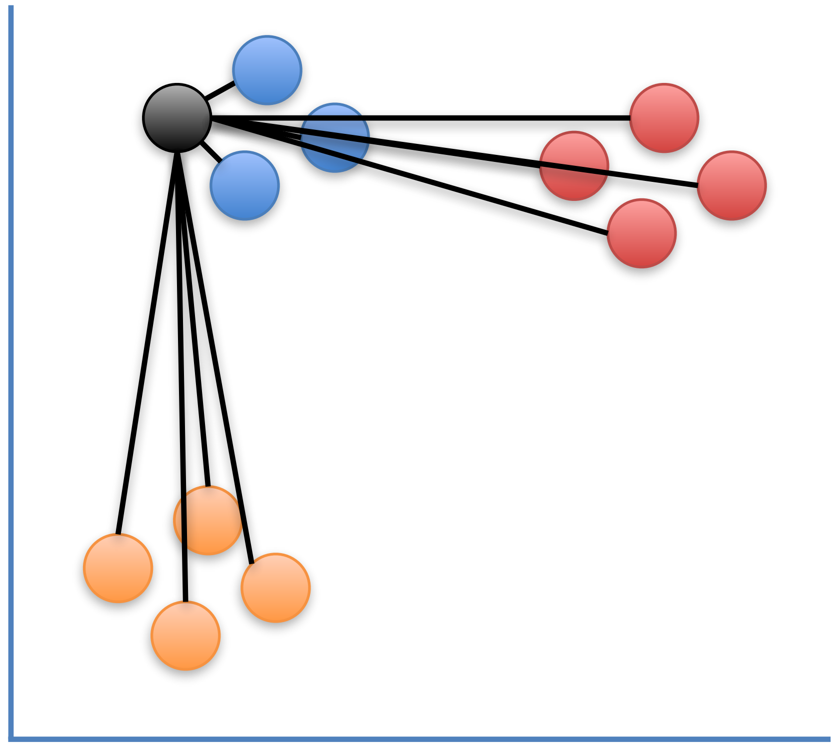


Using a normal distribution means that distant points have very low similarity values….



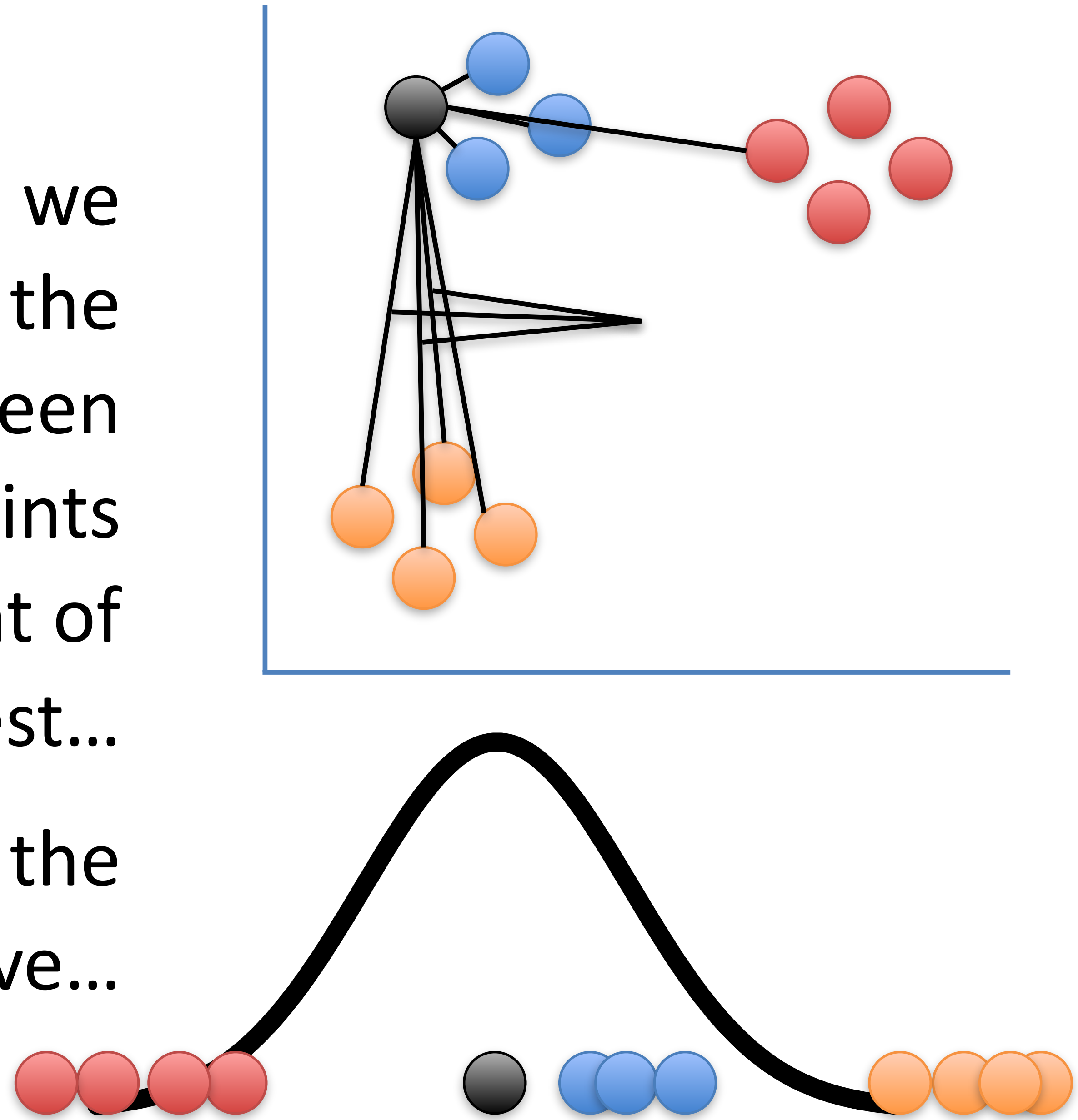
… and close points have high similarity values.

Ultimately, we measure the distances between all of the points and the point of interest…

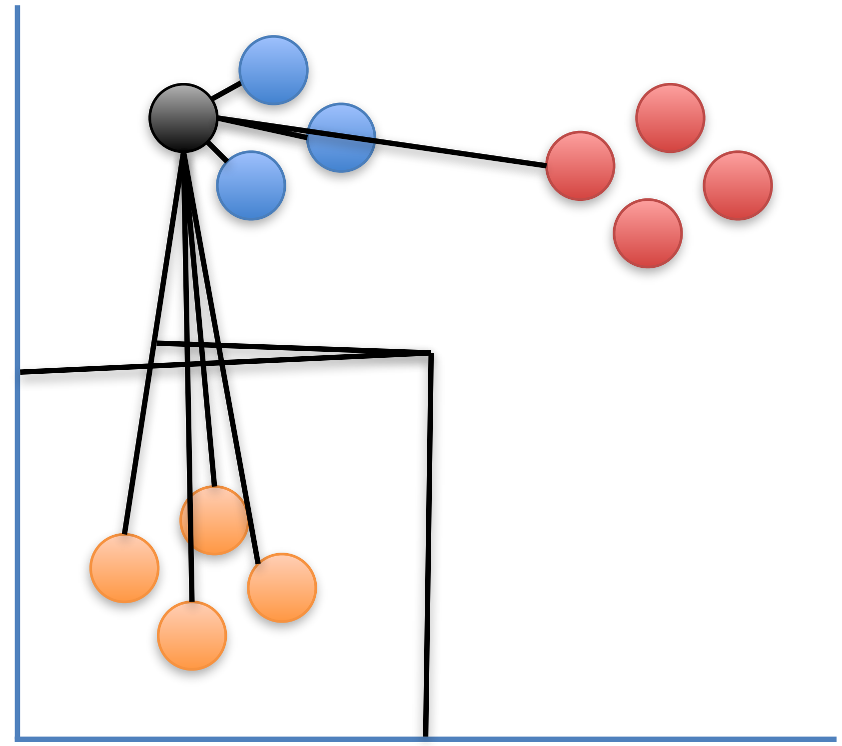


Ultimately, we measure the distances between all of the points and the point of interest…

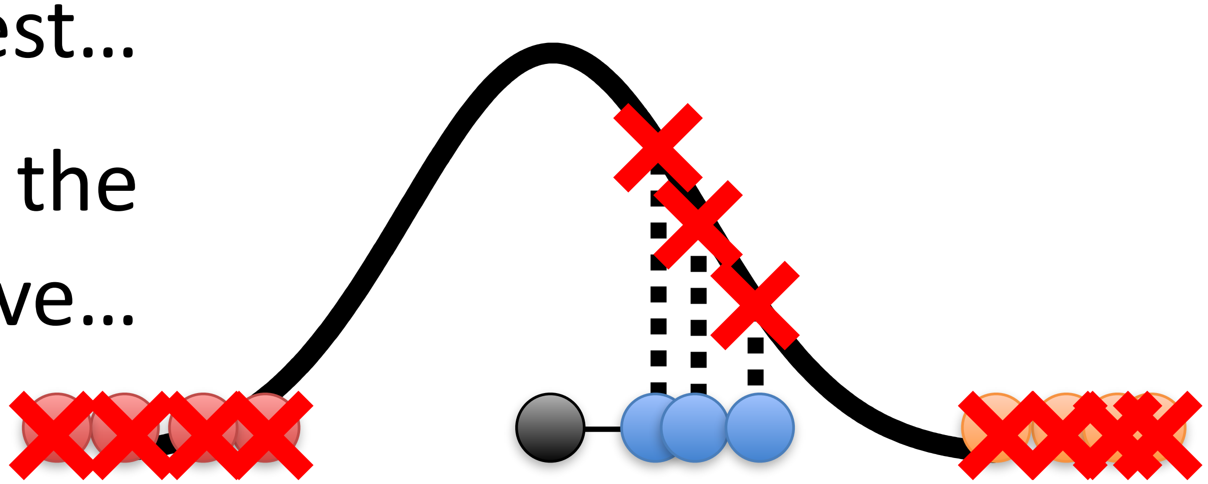
Plot them on the normal curve…



Ultimately, we measure the distances between all of the points and the point of interest…



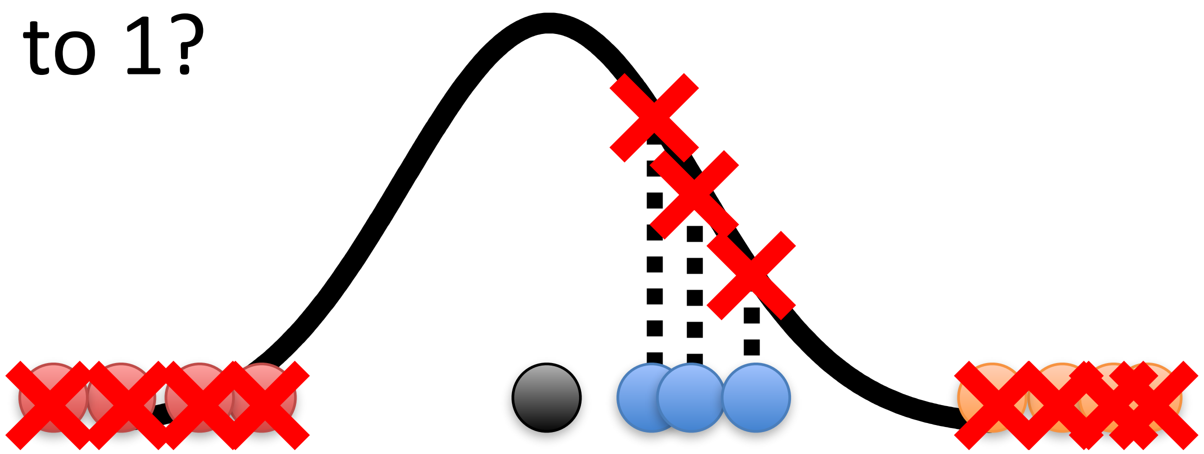
Plot them on the normal curve…



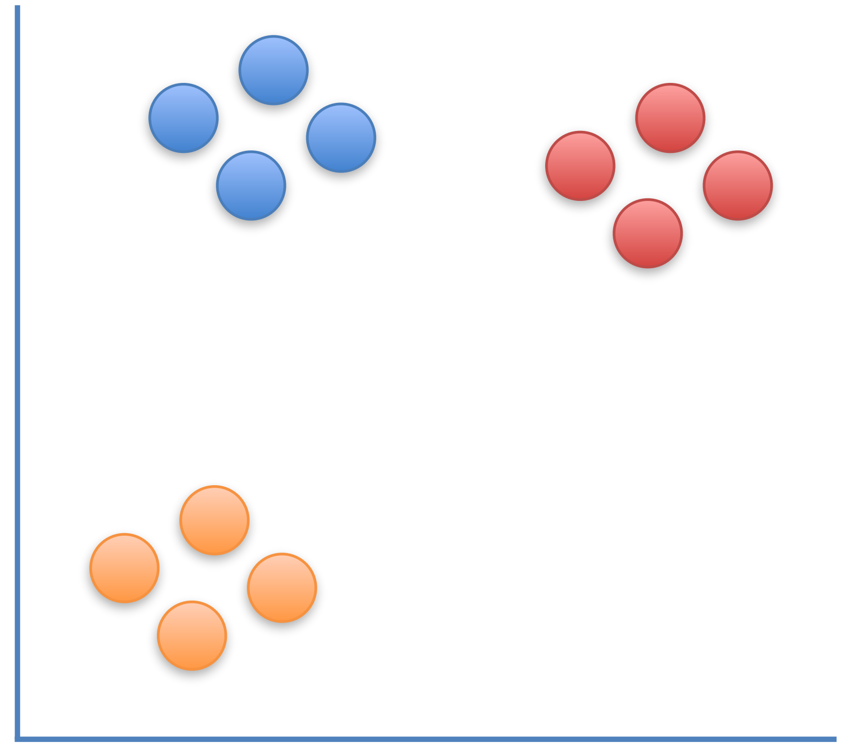
…and then measure the distances from the points to the curve to get the unscaled similarity scores with respect to the point of interest.

The next step is to scale the unscaled similarities so that they add up to 1.

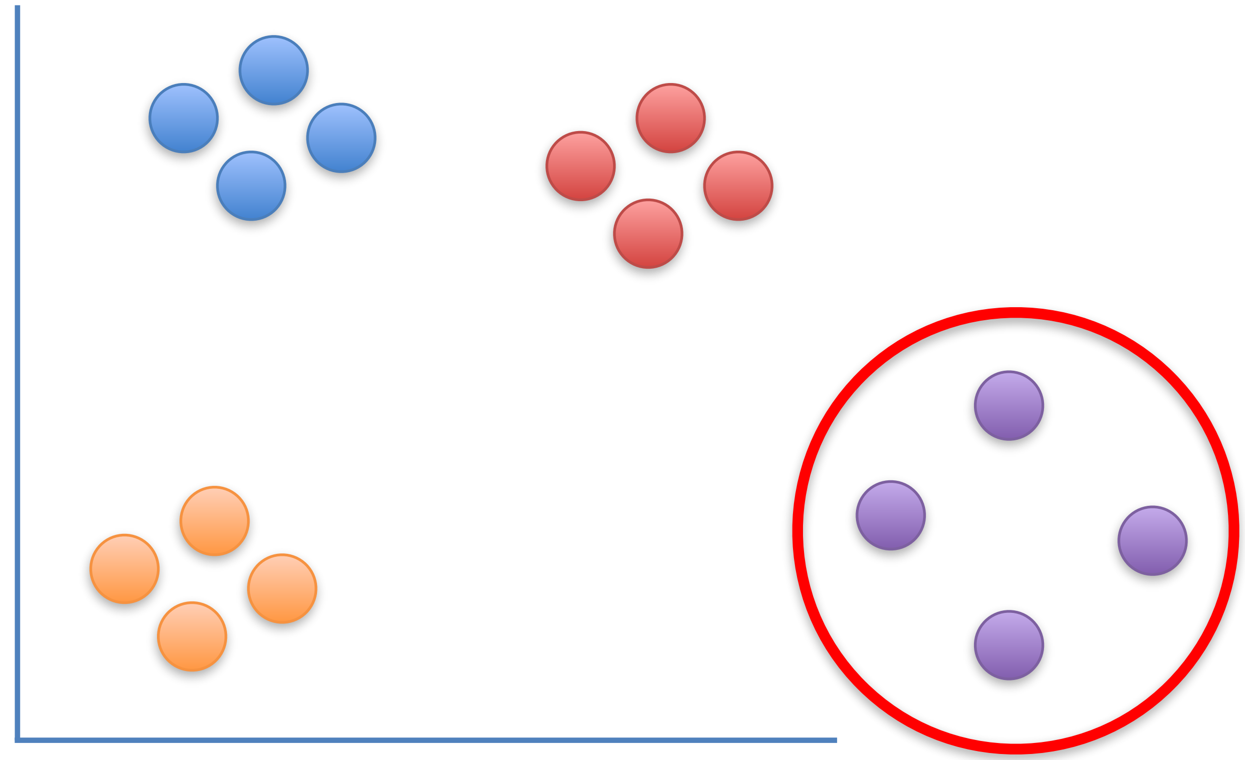
Umm… Why do the similarity scores need to add up to 1?



It has to do with something I didn't tell you earlier…

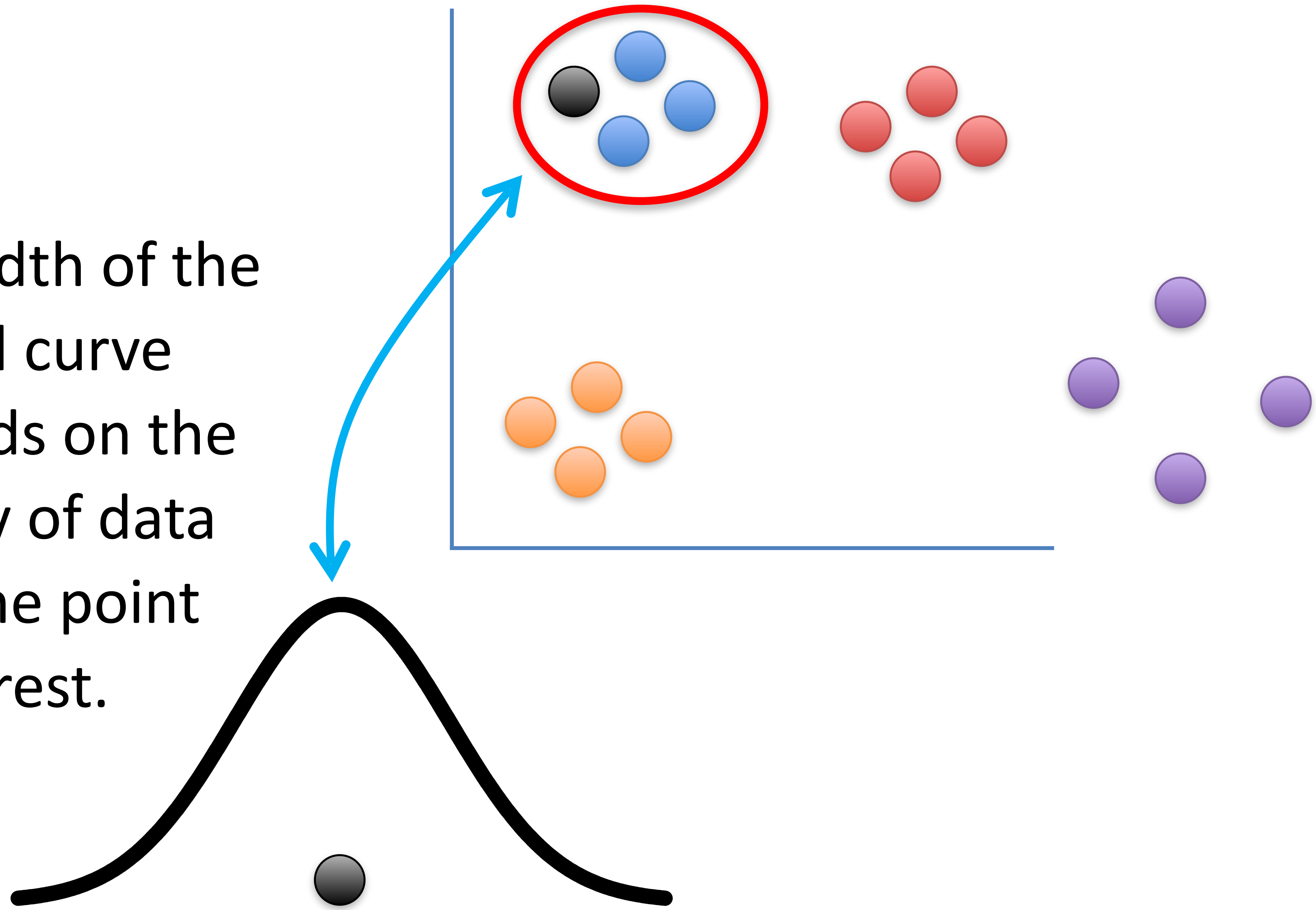


It has to do with something I didn't tell you earlier…

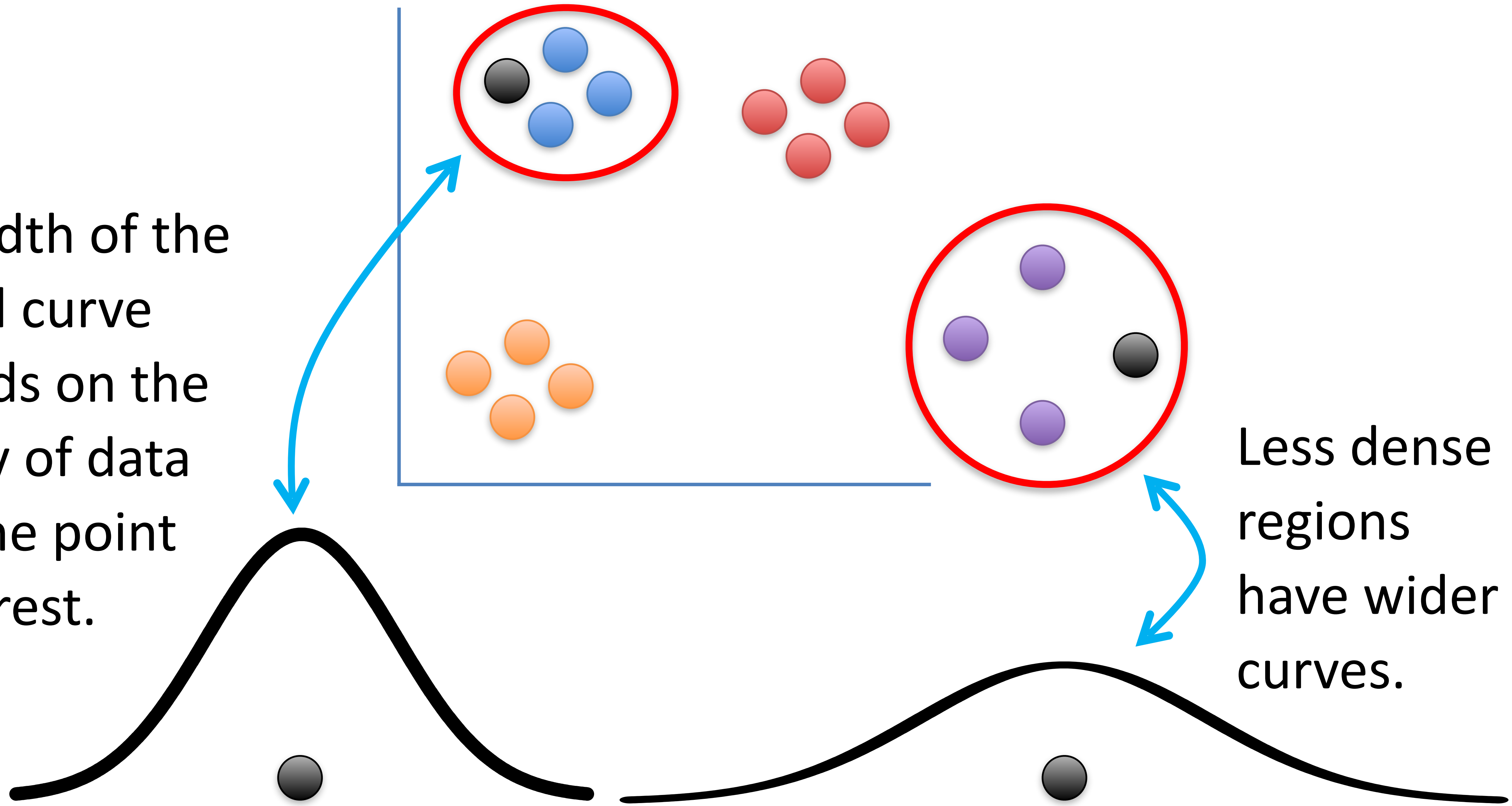


…and to illustrate the concept, I need to add a cluster that is half as dense as the others.

The width of the normal curve depends on the density of data near the point of interest.

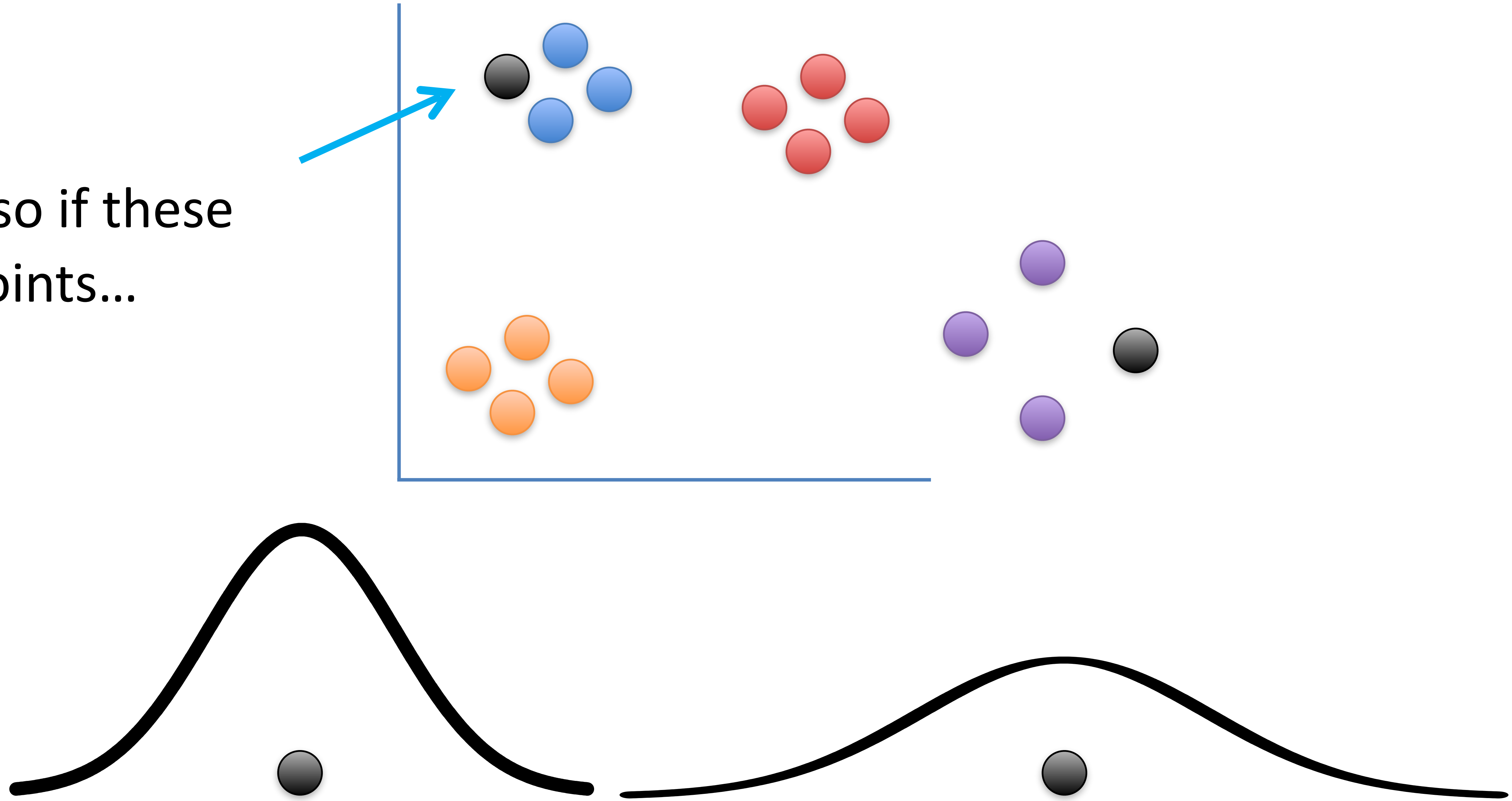


The width of the normal curve depends on the density of data near the point of interest.

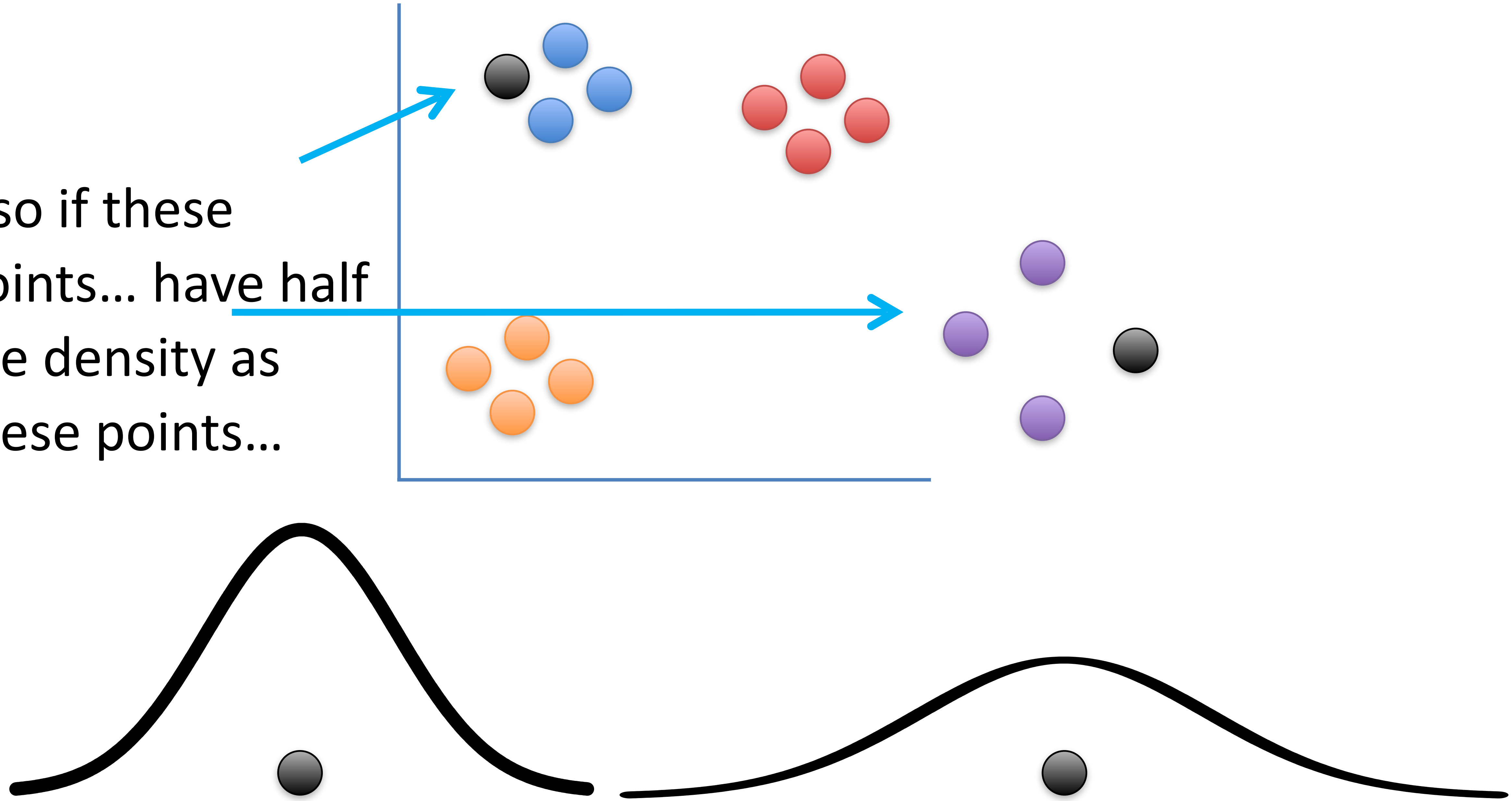


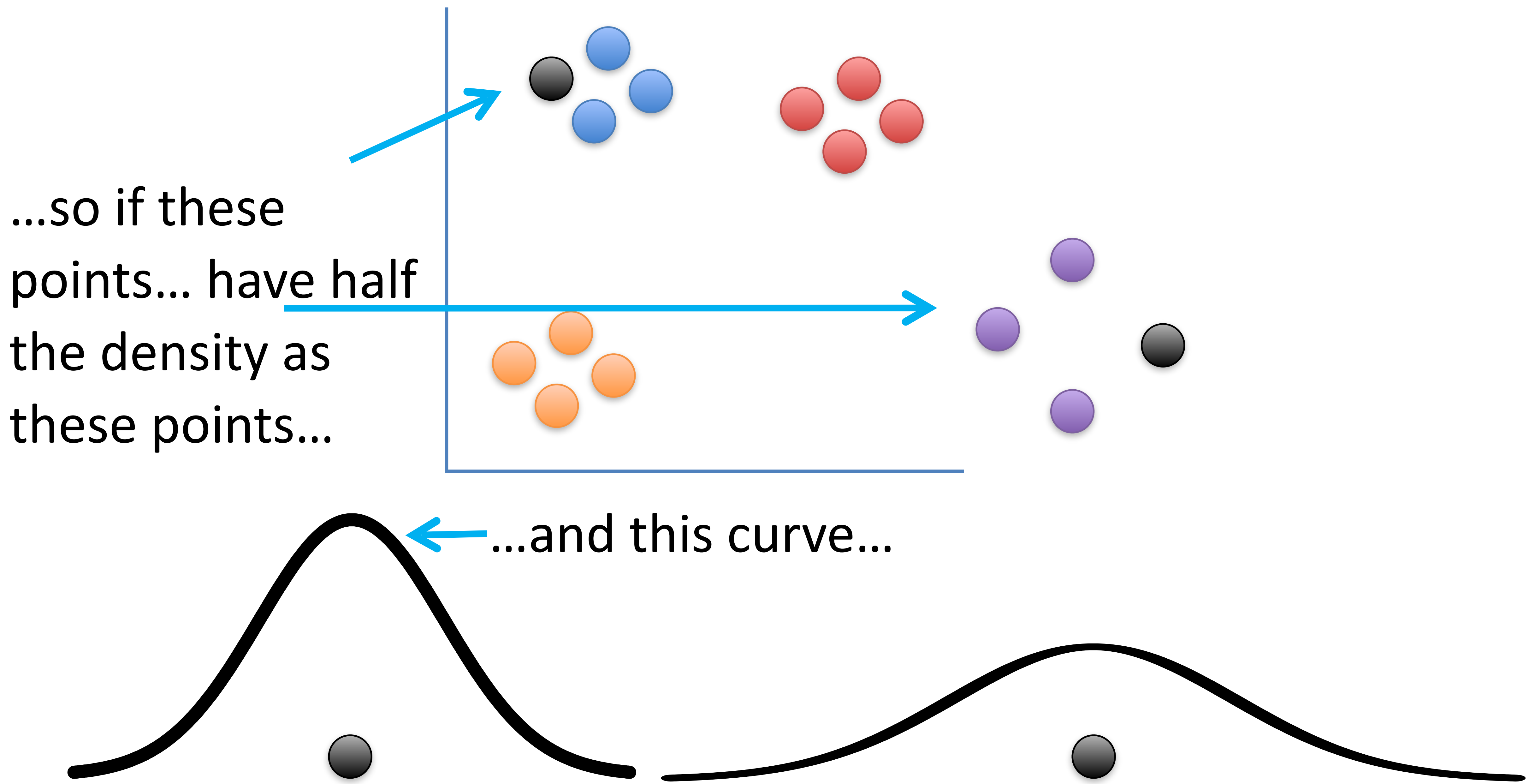
Less dense regions have wider curves.

...so if these points...

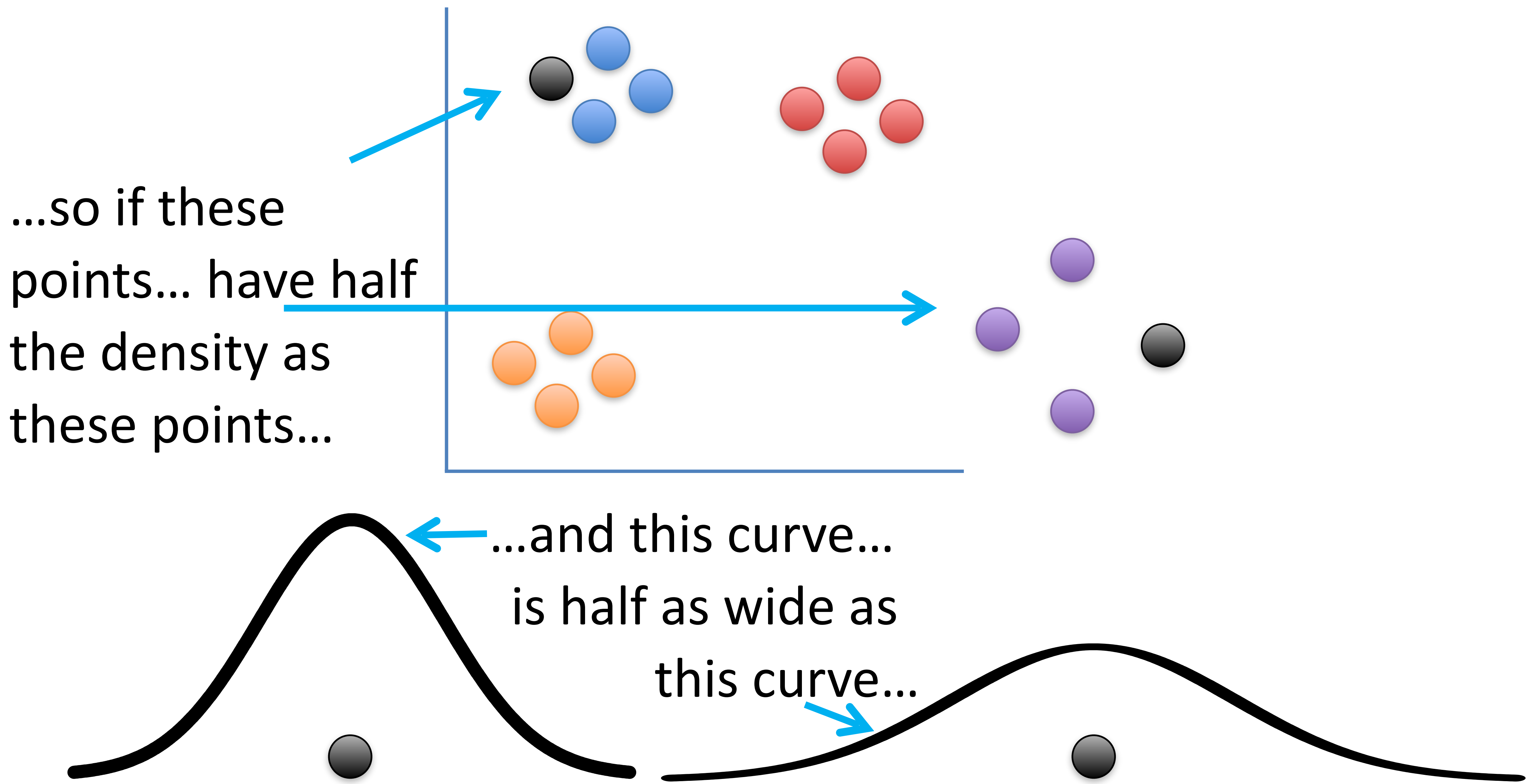


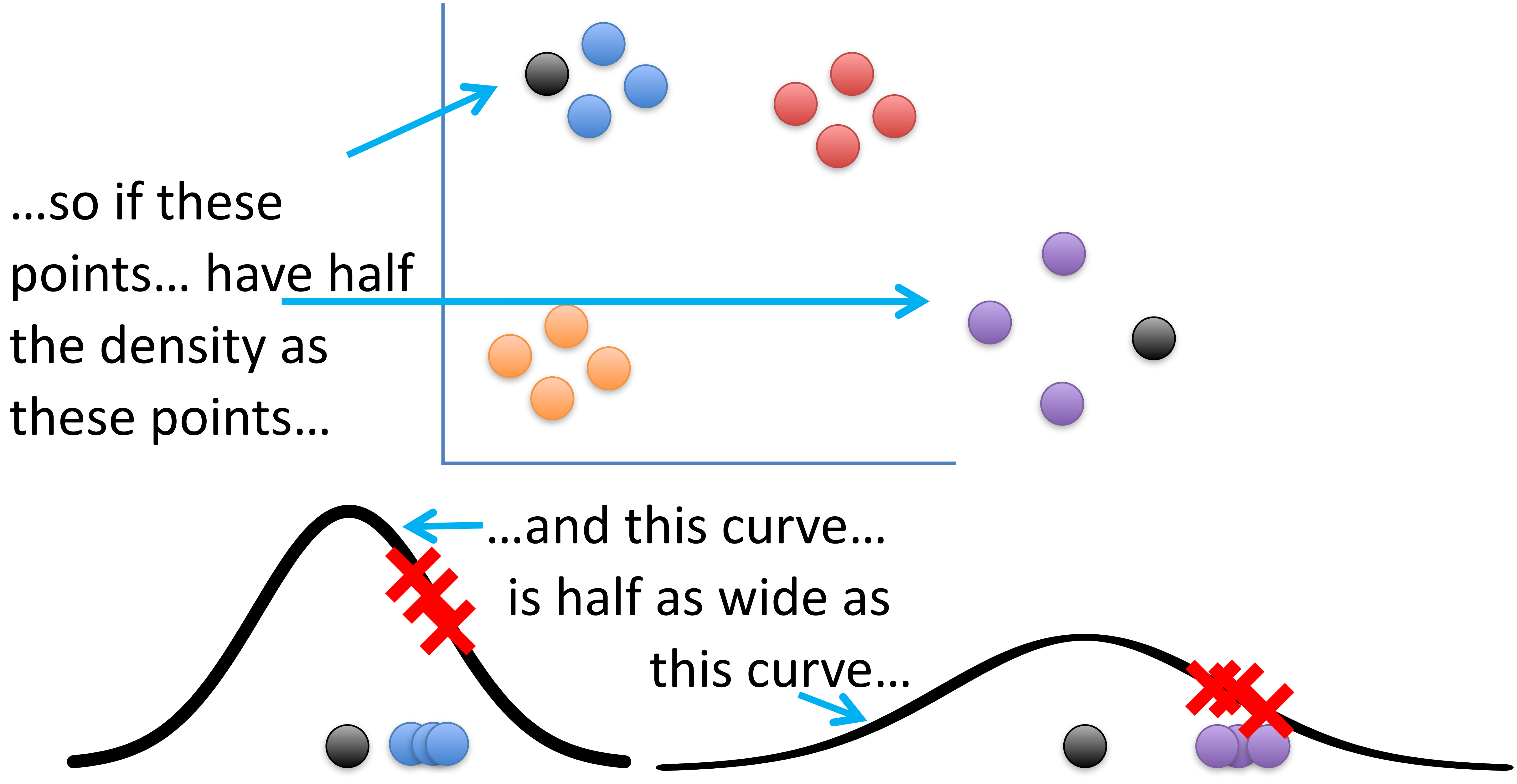
…so if these points… have half the density as these points…





...so if these points... have half the density as these points...
...and this curve...


...so if these
points... have half
the density as
these points...
...and this curve...
is half as wide as
this curve...

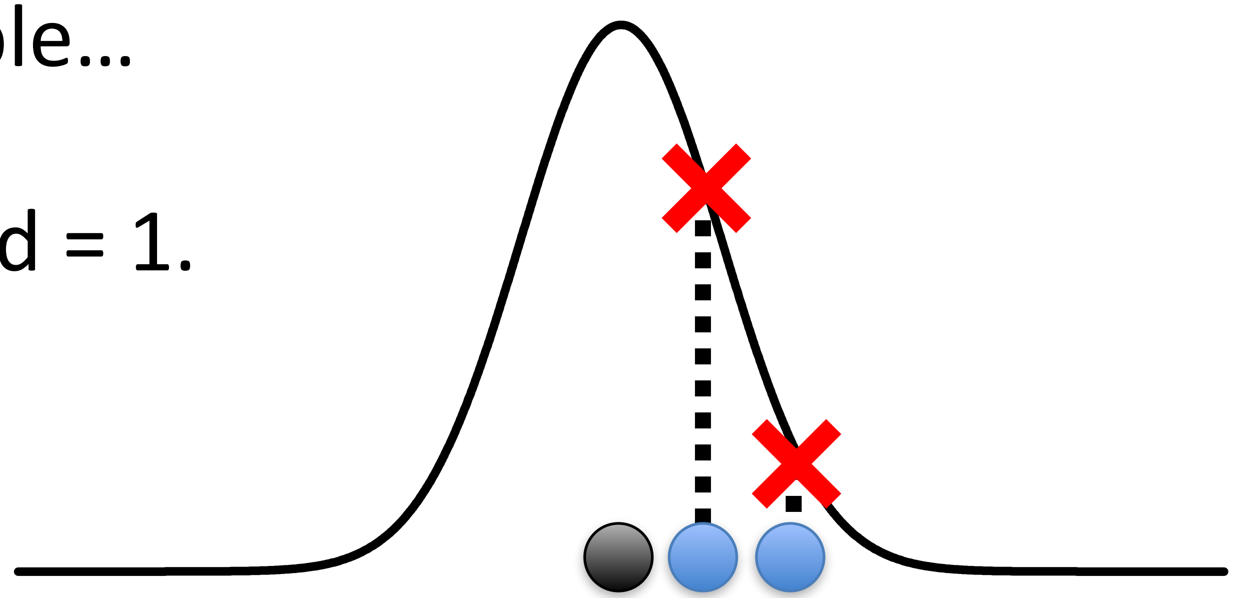



...then scaling the similarity scores will make them the same for both clusters.

Here's an example…

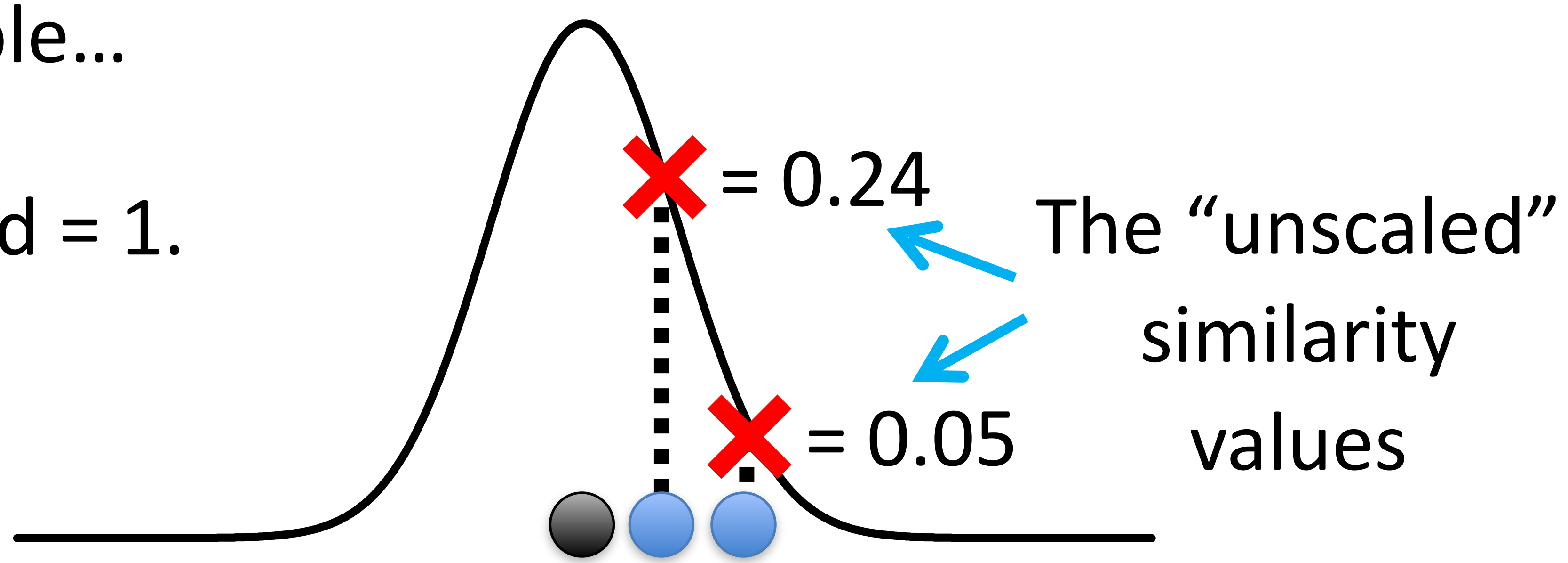
Here's an example…

This curve has a std = 1.



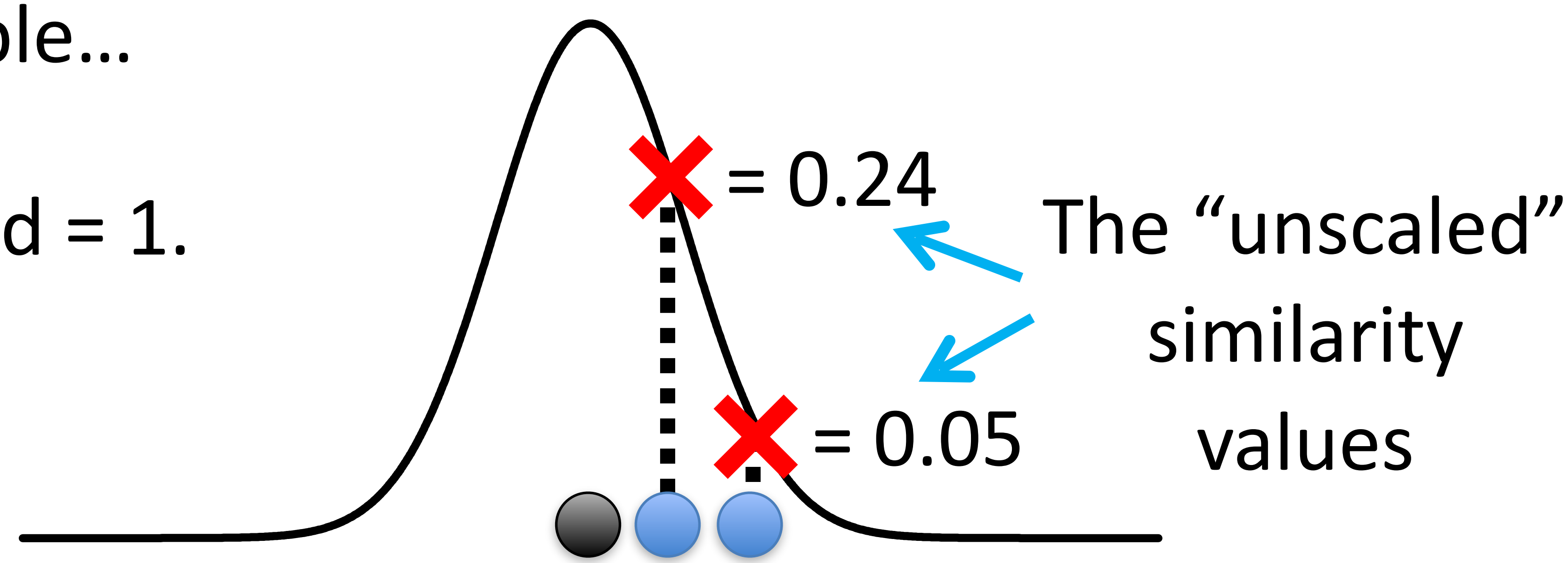
Here's an example…

This curve has a std = 1.

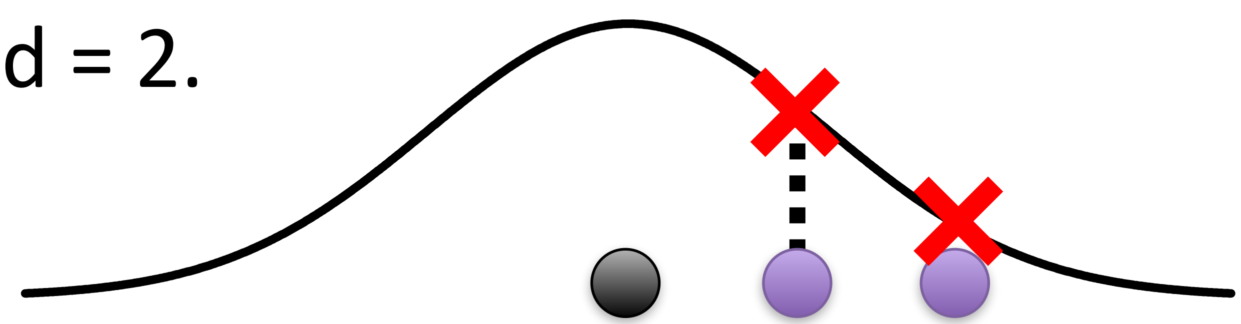


Here’s an example…

This curve has a std = 1.



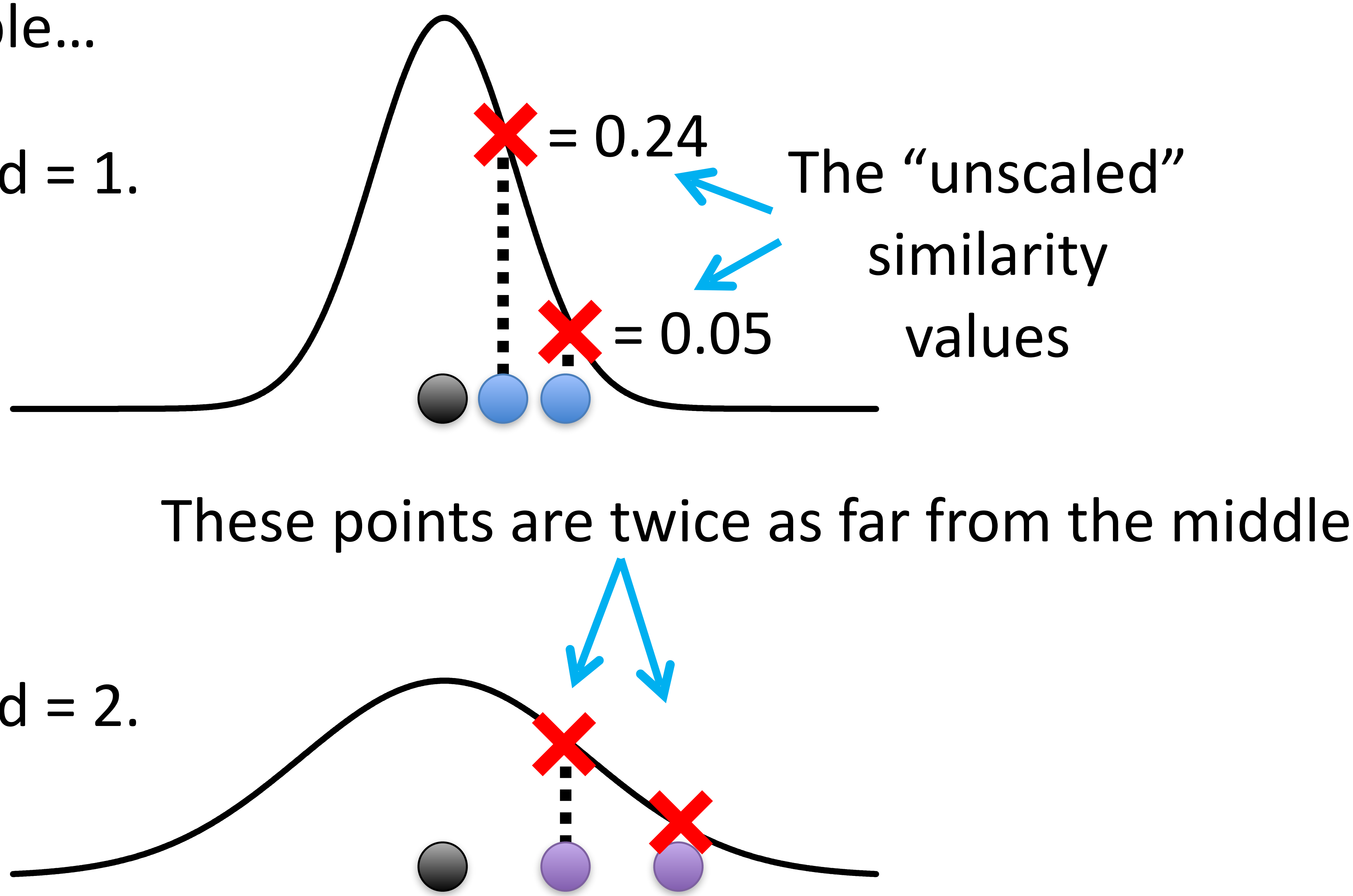

This curve has a std = 2.



Here’s an example…

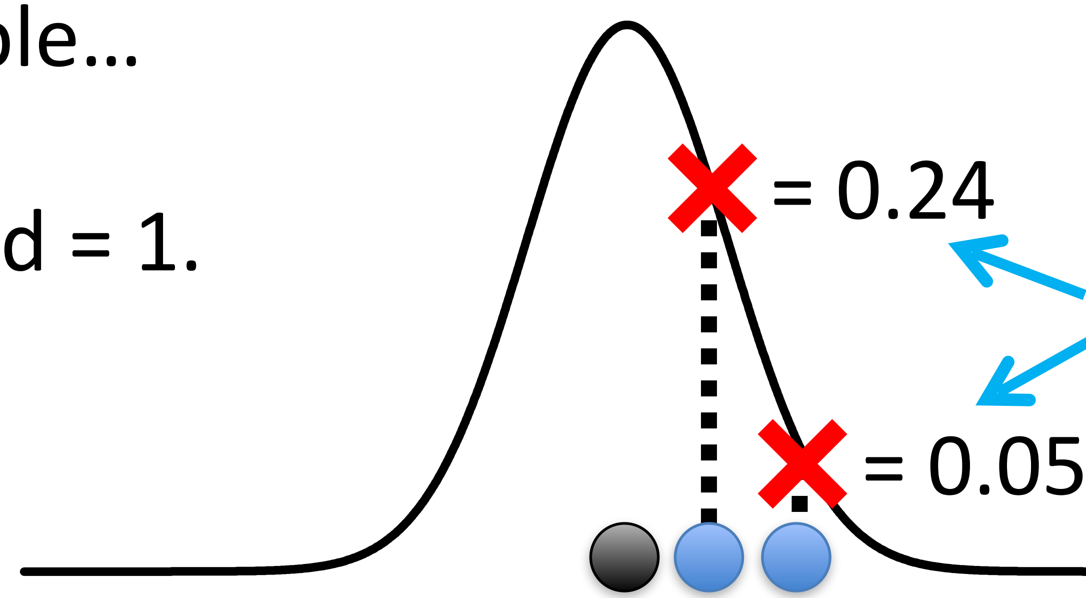
This curve has a std = 1.

This curve has a std = 2.



Here’s an example…

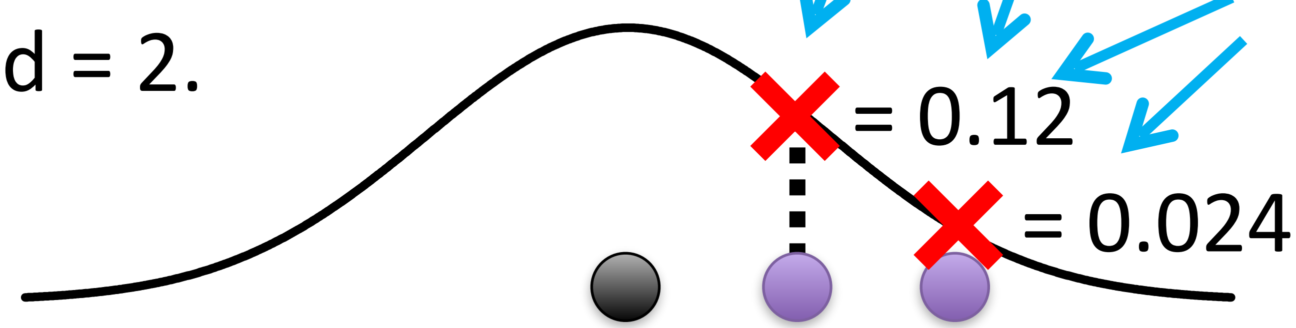
This curve has a std = 1.




The “unscaled” similarity values

These points are twice as far from the middle.

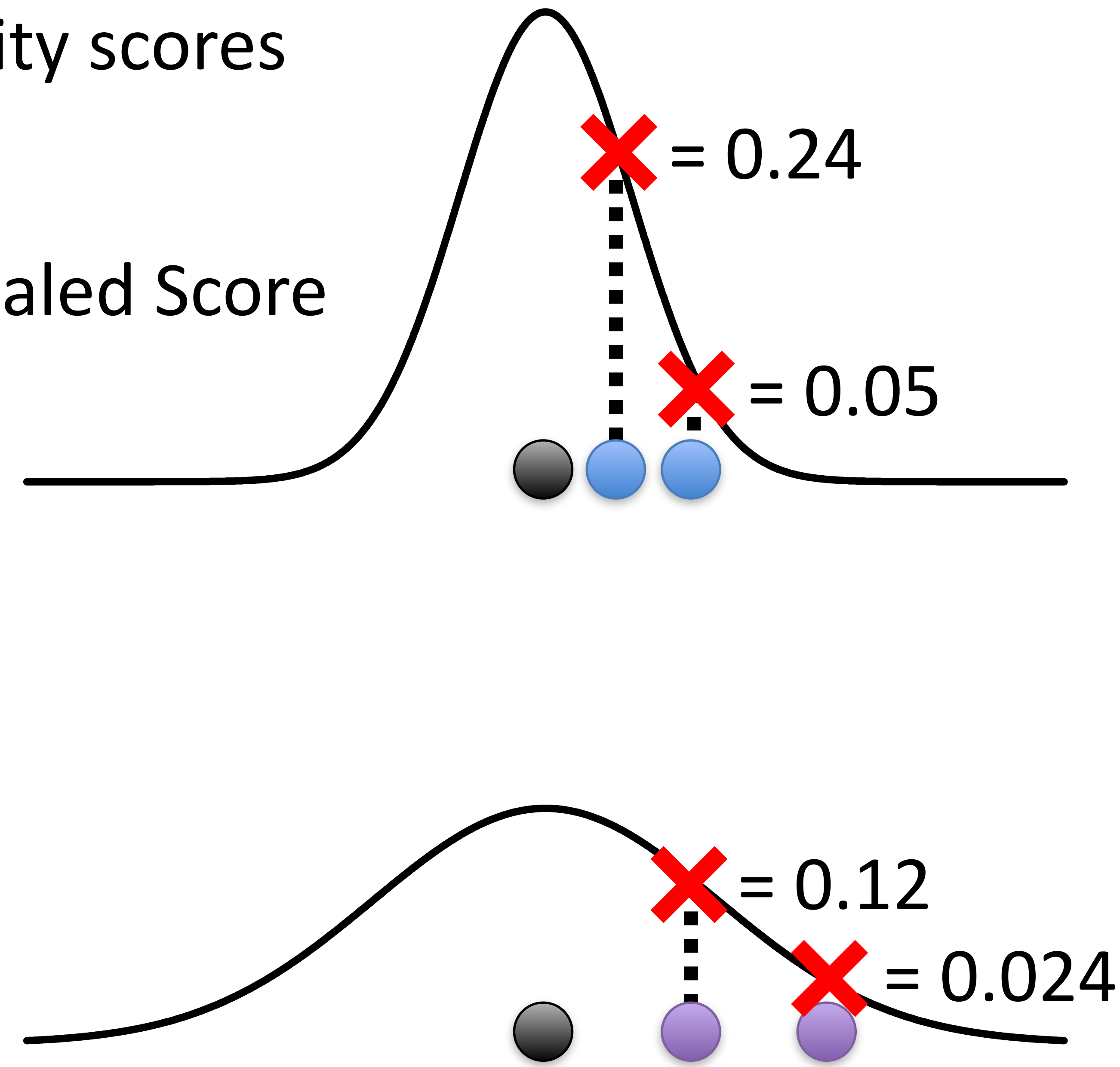
This curve has a std = 2.




The “unscaled” similarity values are half of the other ones.

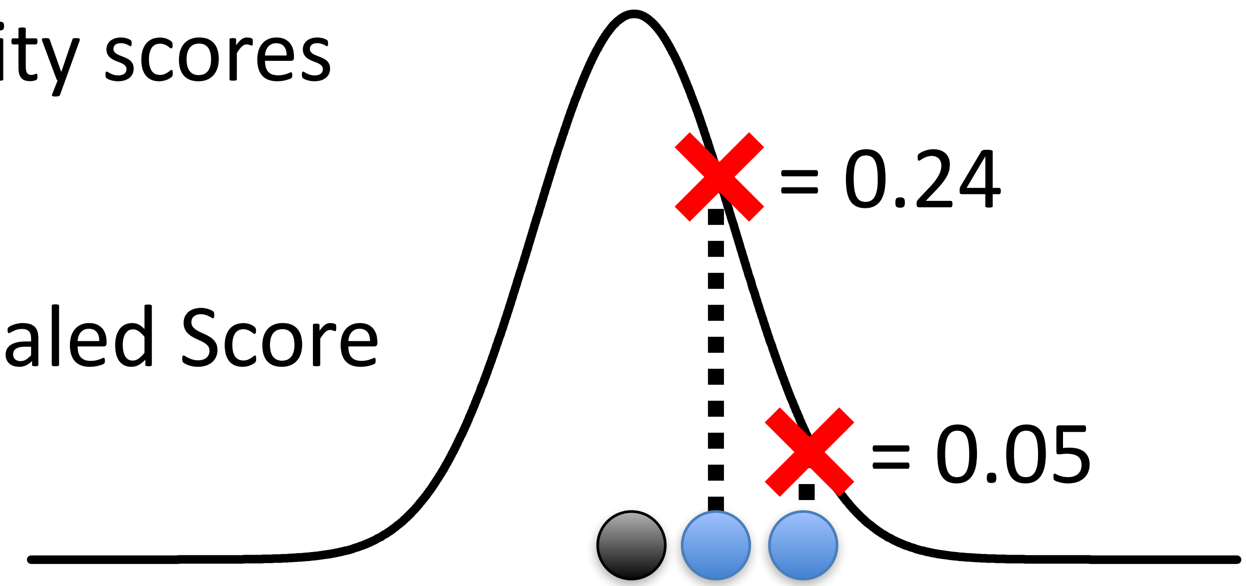
To scale the similarity scores so they sum to 1:

$$\frac{\text{Score}}{\text{Sum of all scores}} = \text{Scaled Score}$$



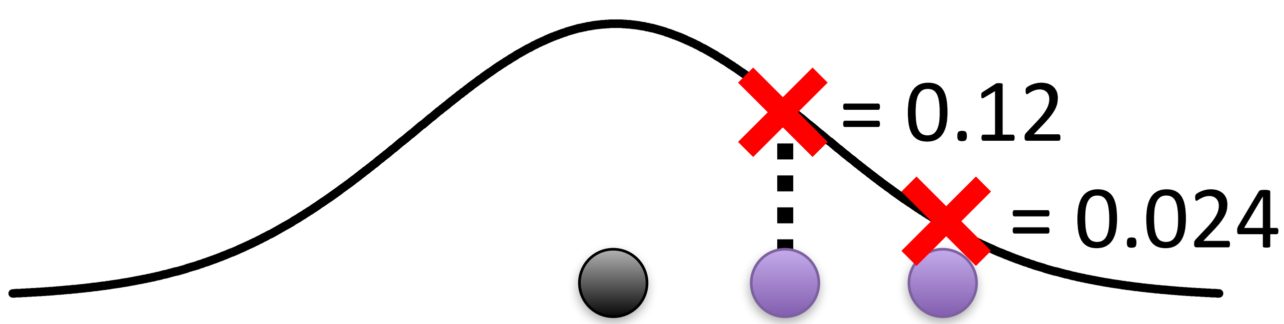
To scale the similarity scores so they sum to 1:

$$\frac{\text{Score}}{\text{Sum of all scores}} = \text{Scaled Score}$$



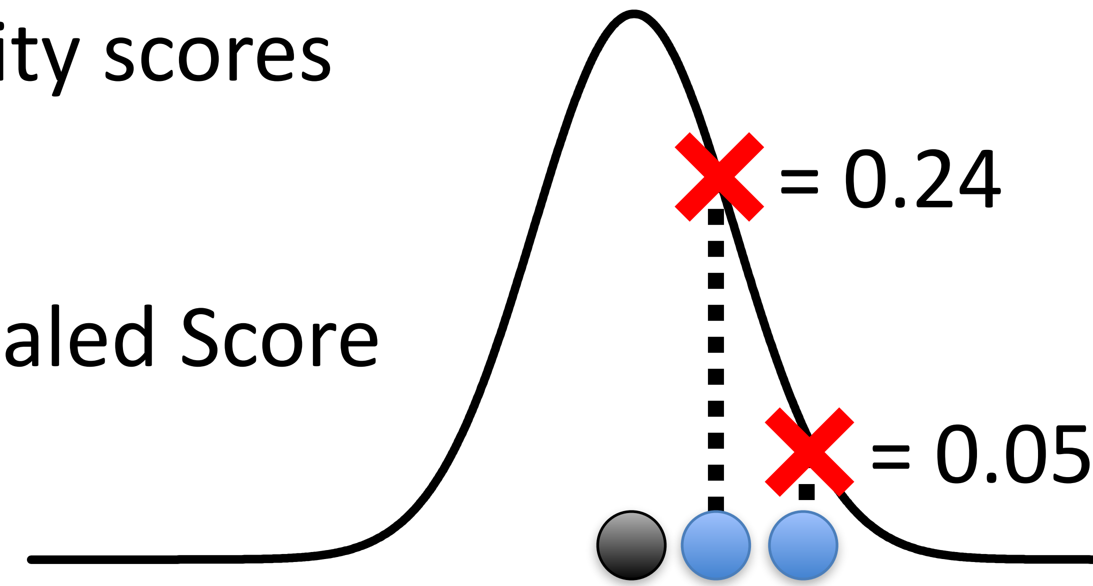

$$\frac{0.24}{0.24 + 0.05} = 0.82$$

$$\frac{0.05}{0.24 + 0.05} = 0.18$$



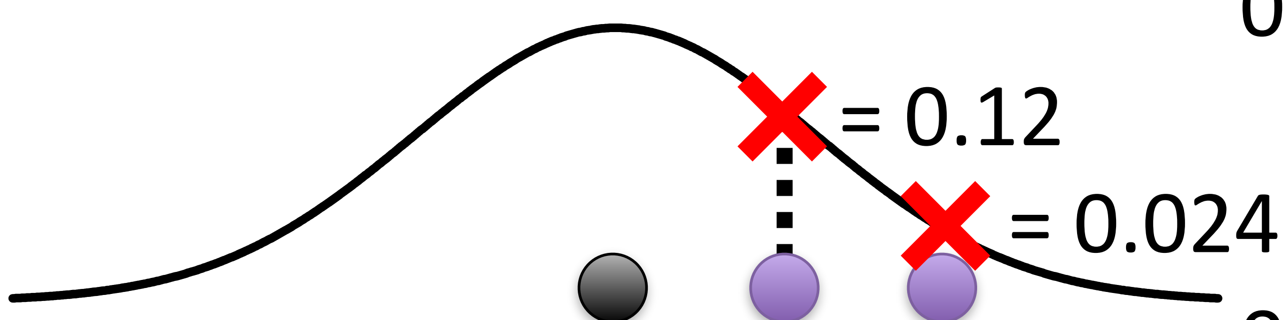
To scale the similarity scores so they sum to 1:

$$\frac{\text{Score}}{\text{Sum of all scores}} = \text{Scaled Score}$$




$$\frac{0.24}{0.24 + 0.05} = 0.82$$

$$\frac{0.05}{0.24 + 0.05} = 0.18$$

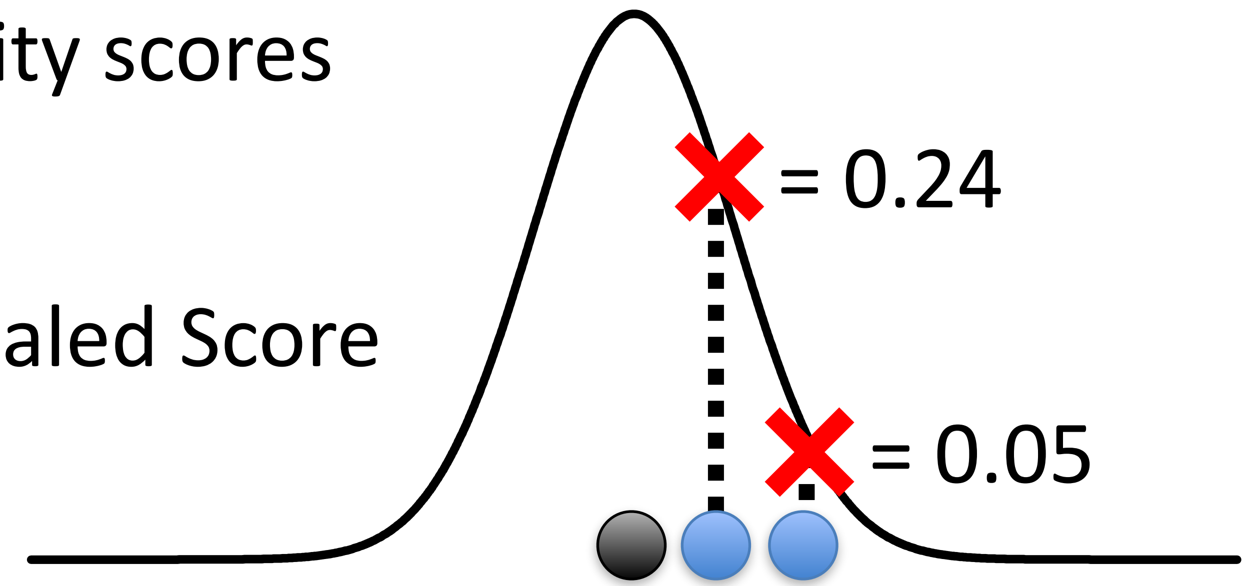



$$\frac{0.12}{0.12 + 0.024} = 0.82$$

$$\frac{0.024}{0.12 + 0.024} = 0.18$$

To scale the similarity scores so they sum to 1:

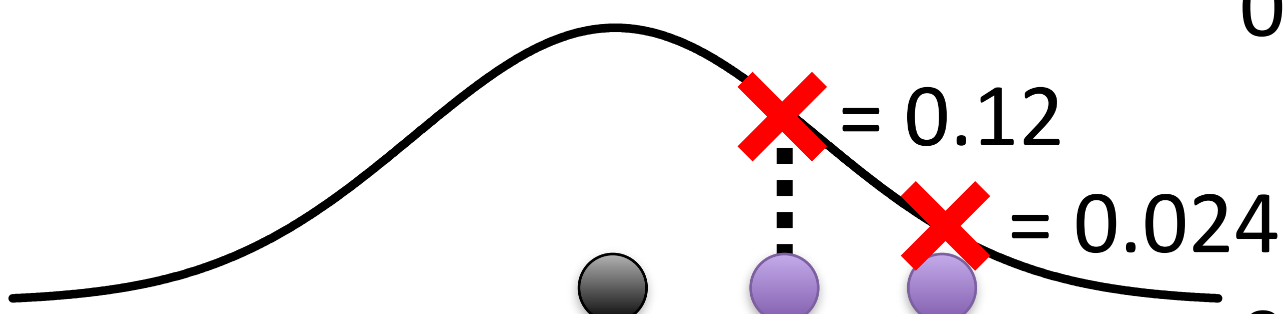
$$\frac{\text{Score}}{\text{Sum of all scores}} = \text{Scaled Score}$$




$$\frac{0.24}{0.24 + 0.05} = \boxed{0.82}$$

$$\frac{0.05}{0.24 + 0.05} = \boxed{0.18}$$

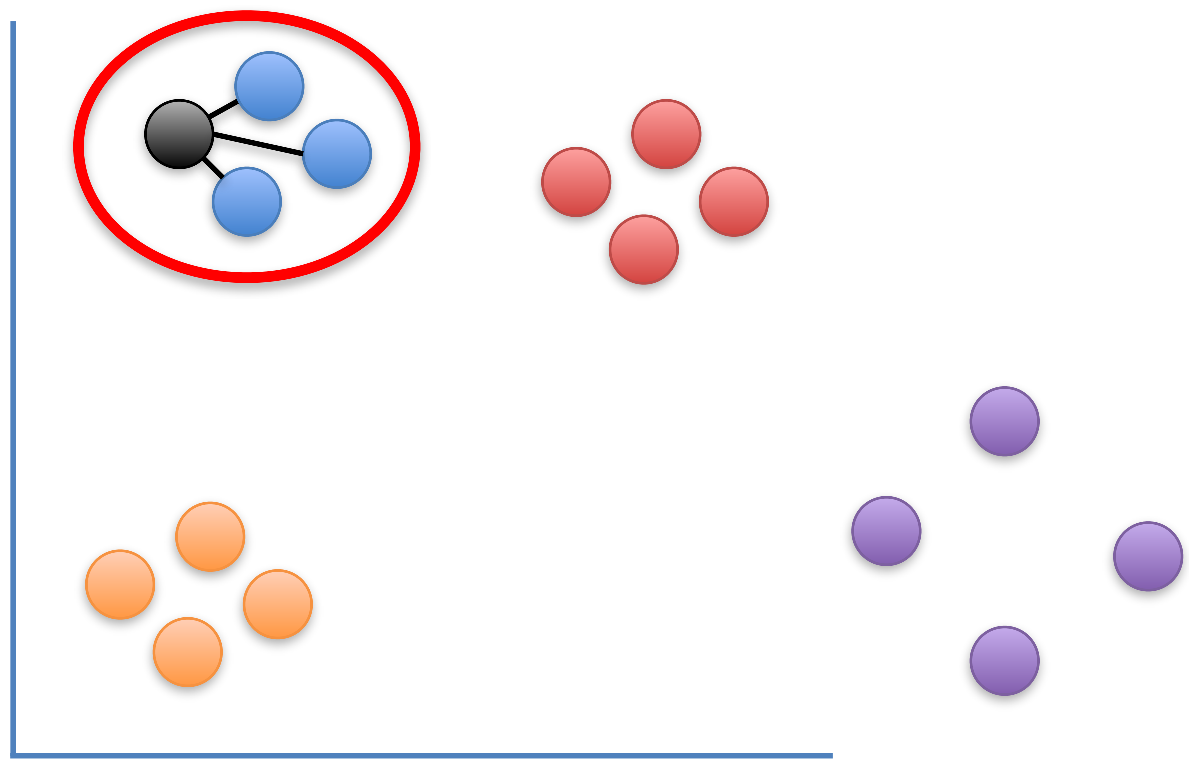
Same values!




$$\frac{0.12}{0.12 + 0.024} = \boxed{0.82}$$

$$\frac{0.024}{0.12 + 0.024} = \boxed{0.18}$$

That implies that the scaled similarity scores for this relatively tight cluster…



$$\frac{0.24}{0.24 + 0.05} = 0.82$$

$$\frac{0.05}{0.24 + 0.05} = 0.18$$

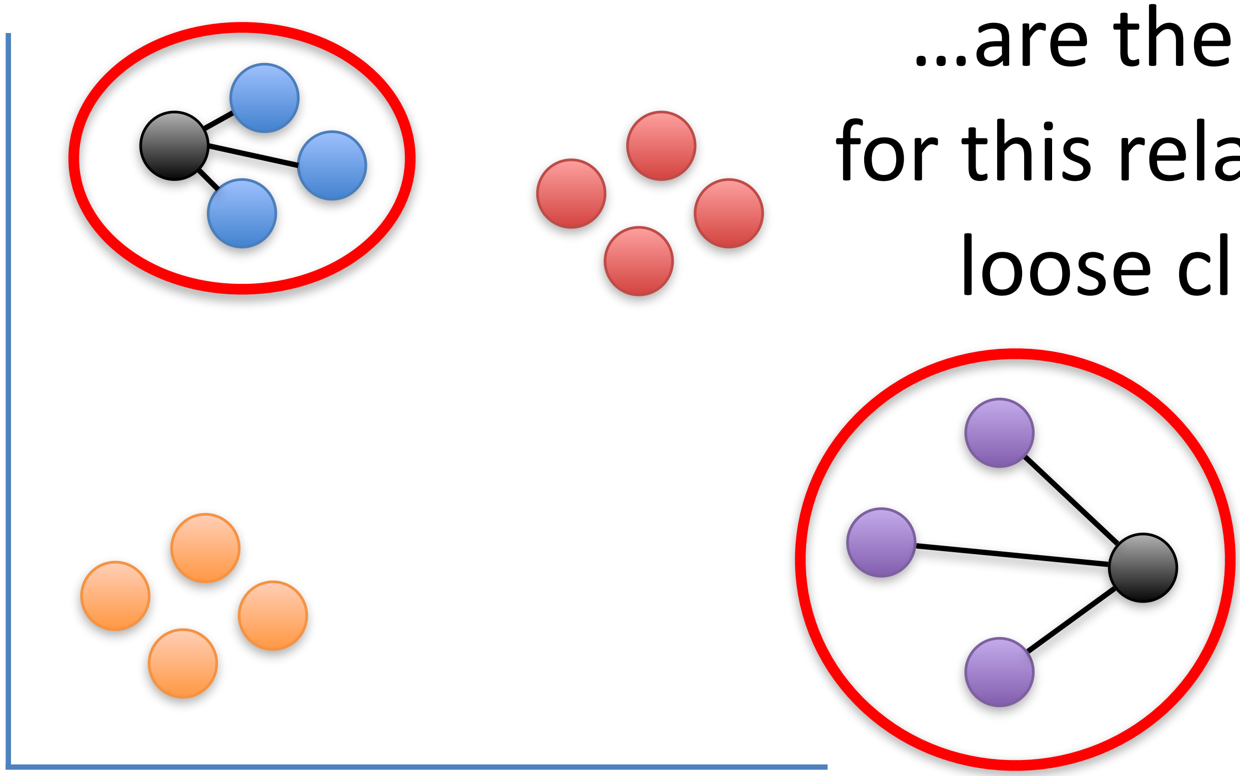
$$\frac{0.12}{0.12 + 0.024} = 0.82$$

$$\frac{0.024}{0.12 + 0.024} = 0.18$$

That implies that the scaled similarity scores for this relatively tight cluster…

$$\frac{0.24}{0.24 + 0.05} = 0.82$$

$$\frac{0.05}{0.24 + 0.05} = 0.18$$

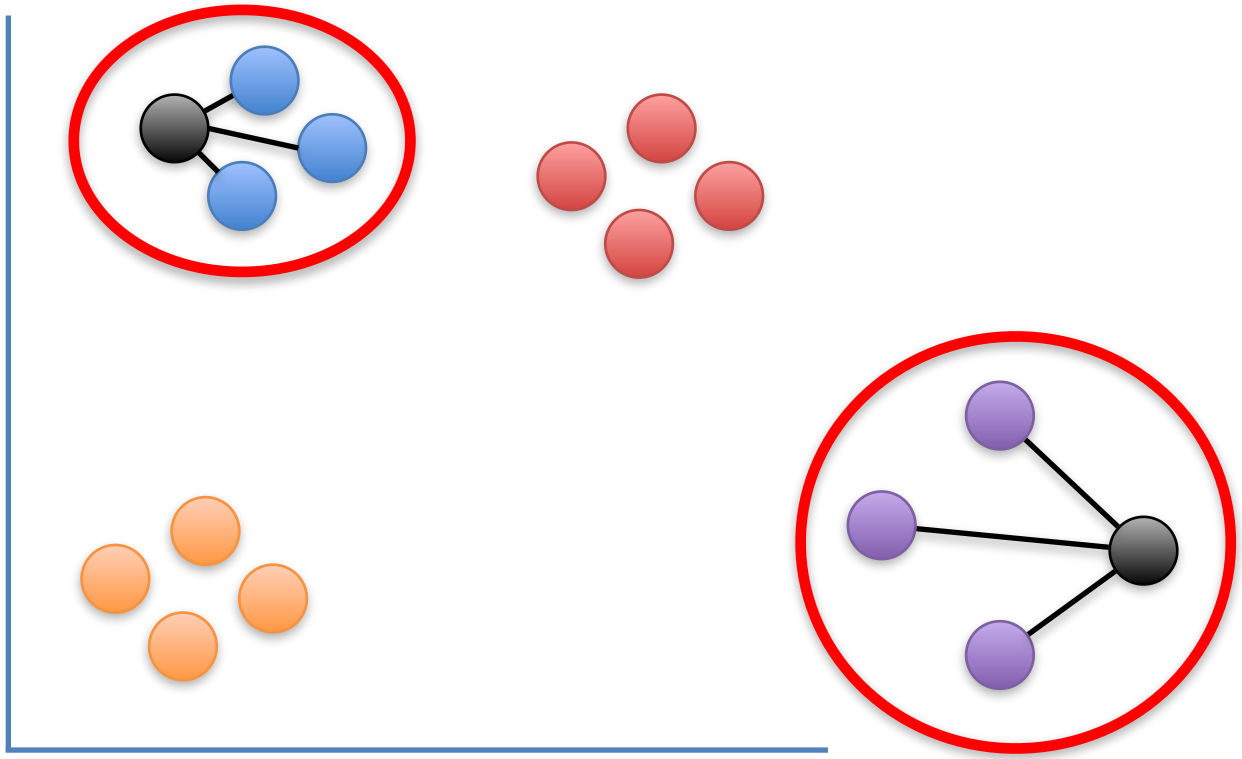


…are the same for this relatively loose cluster!

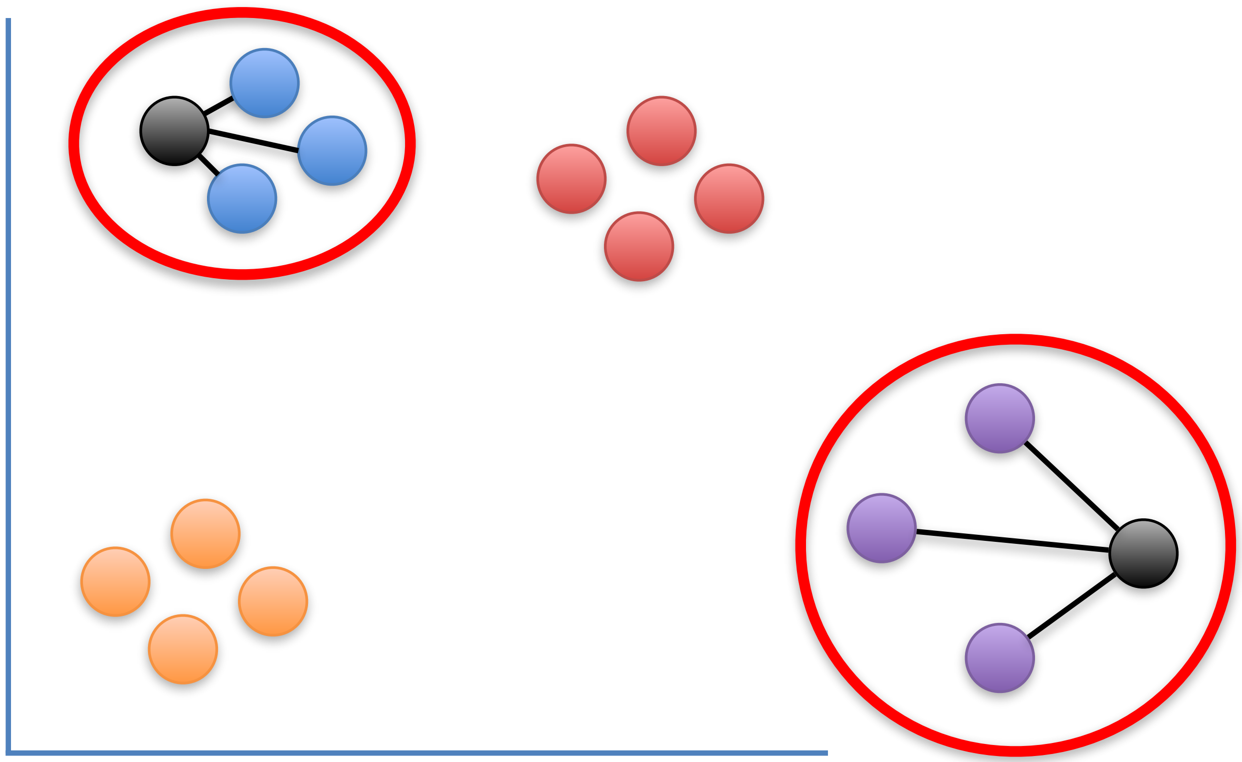
$$\frac{0.12}{0.12 + 0.024} = 0.82$$

$$\frac{0.024}{0.12 + 0.024} = 0.18$$

The reality is a little more
complicated, but only slightly.

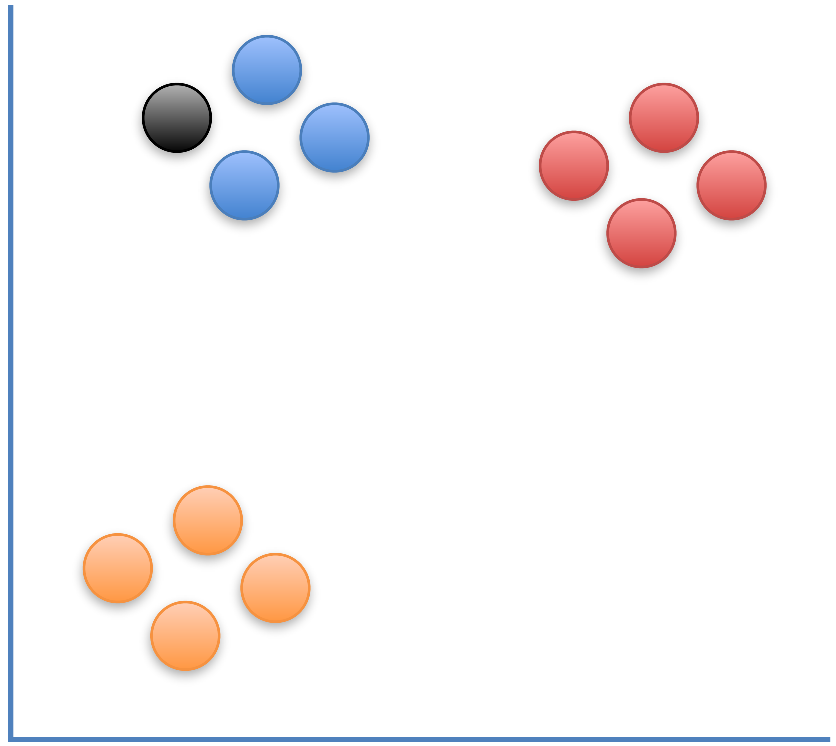


The reality is a little more complicated, but only slightly.

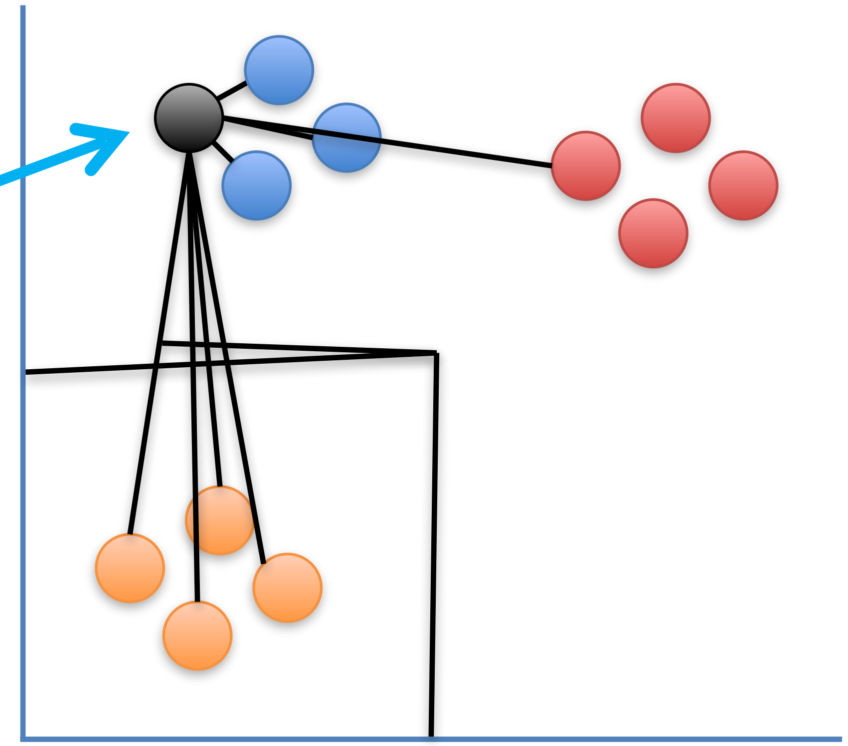


t-SNE has a “perplexity” parameter equal to the expected density, and that comes into play, but these clusters are still more “similar” than you might expect.

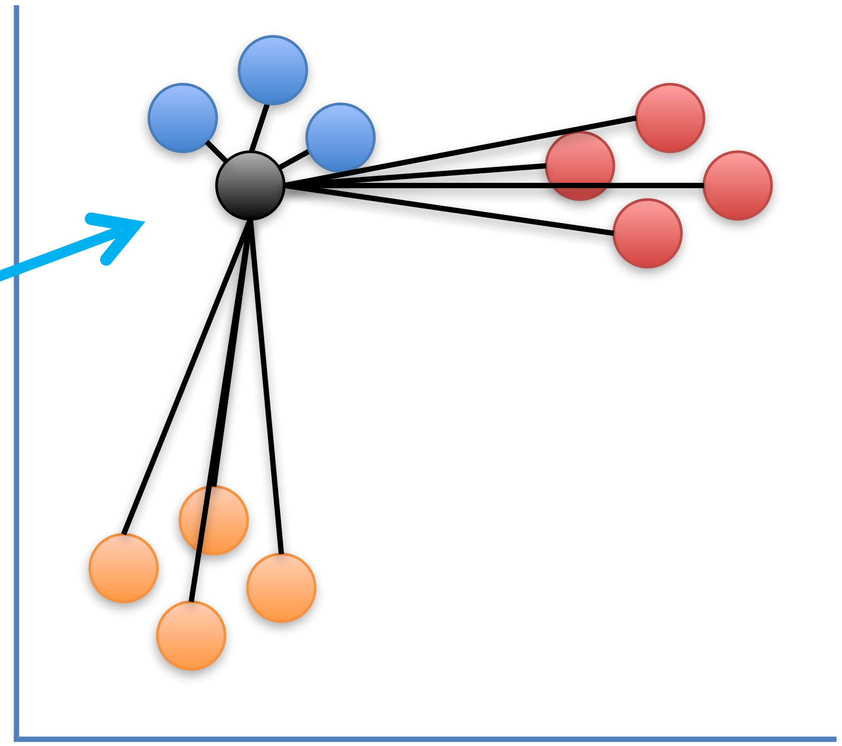
Now back to the original scatter plot…



We’ve calculated similarity scores for this point.

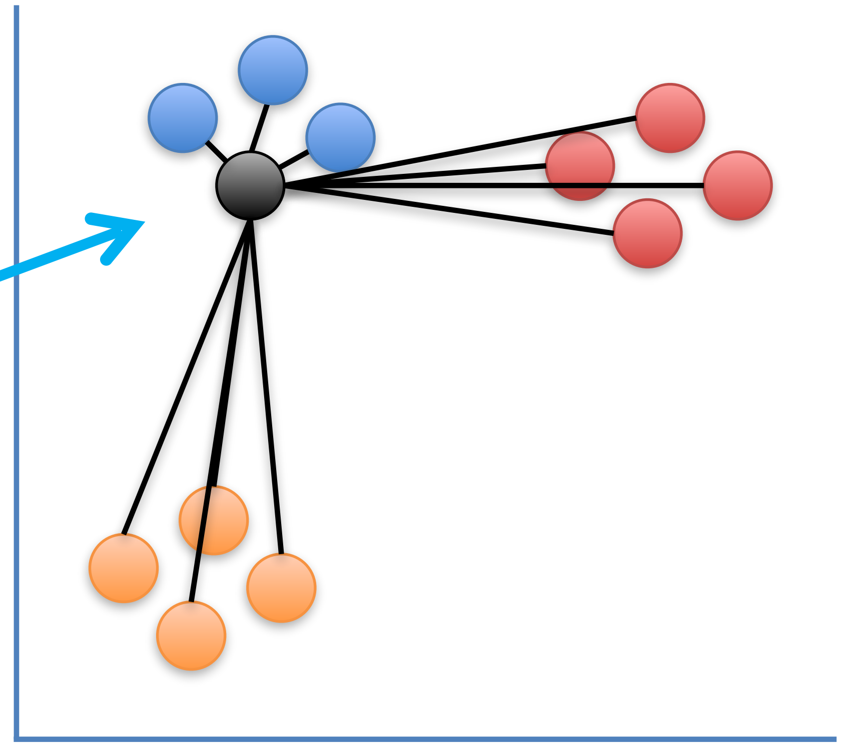


Now we do it for this point…

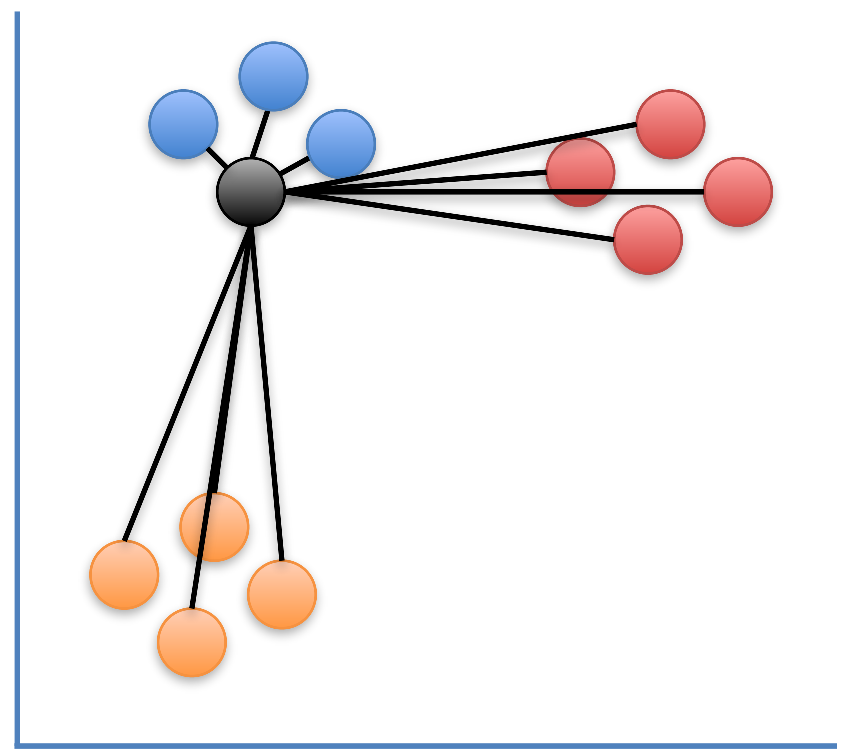


Now we do it for this point...

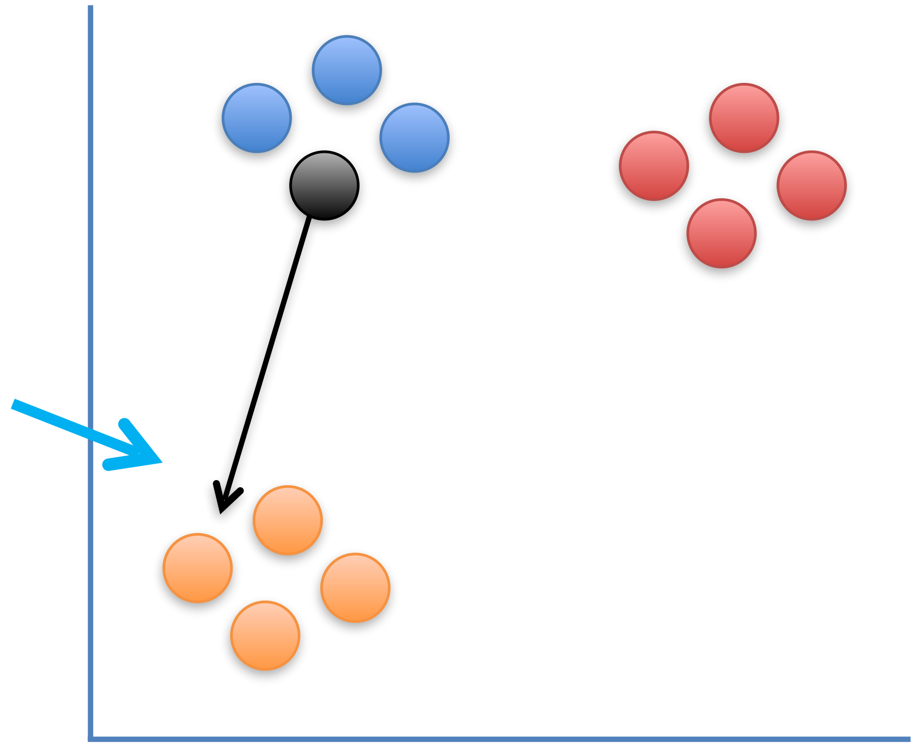
...and we do it for all the points.



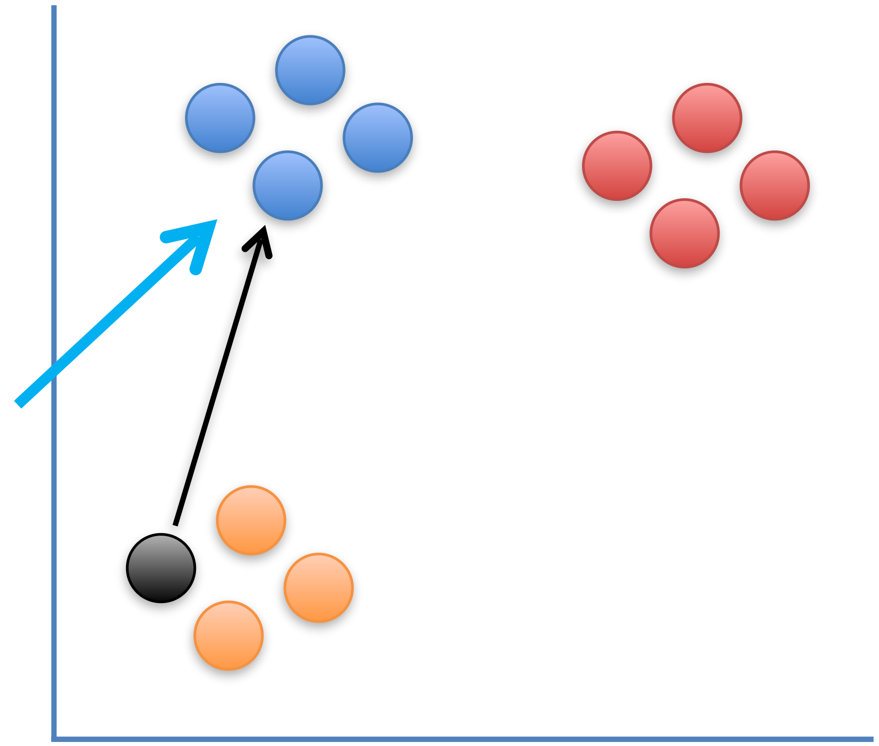
One last thing and the scatter plot will be all set with similarity scores!!!

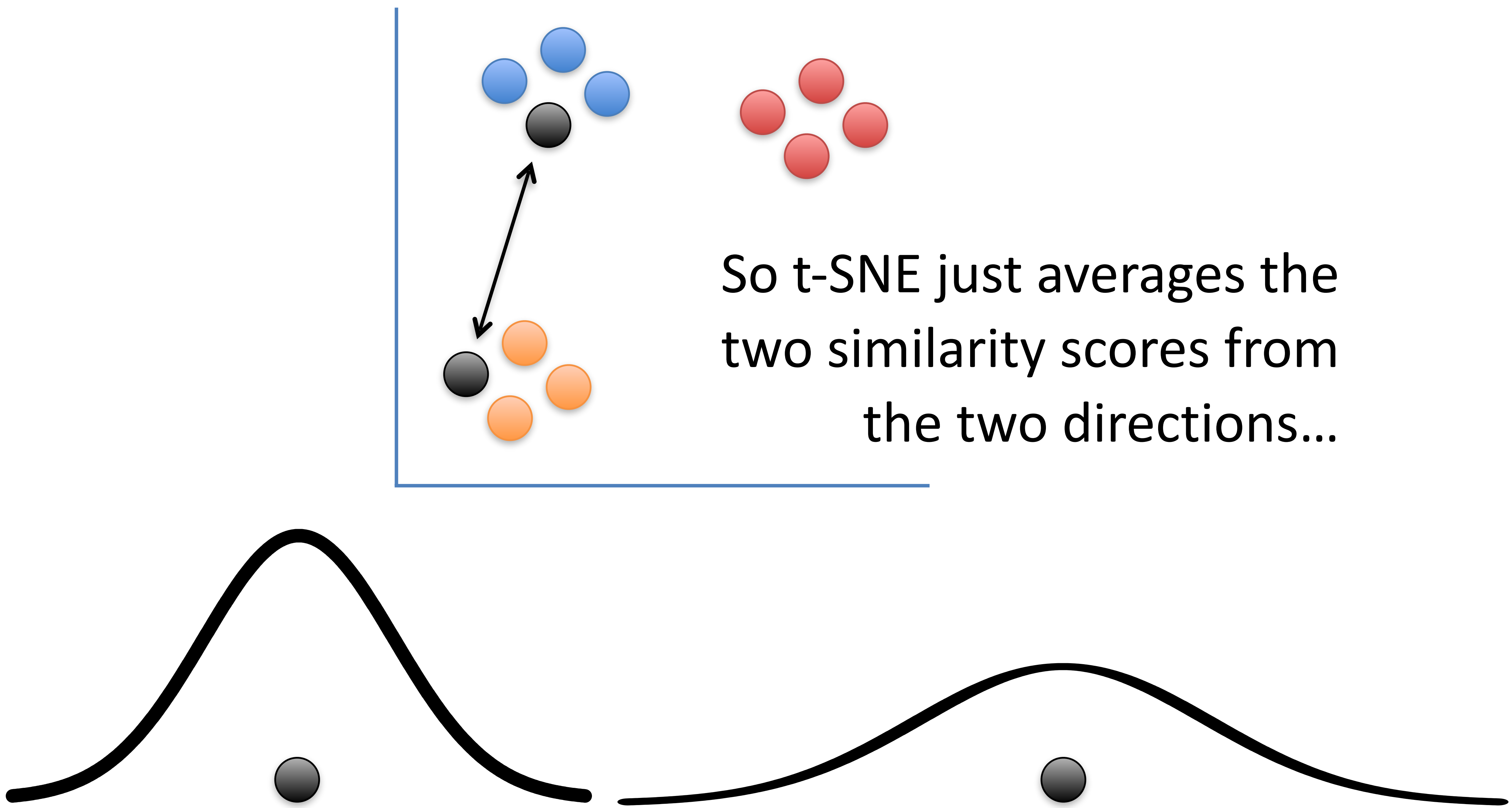


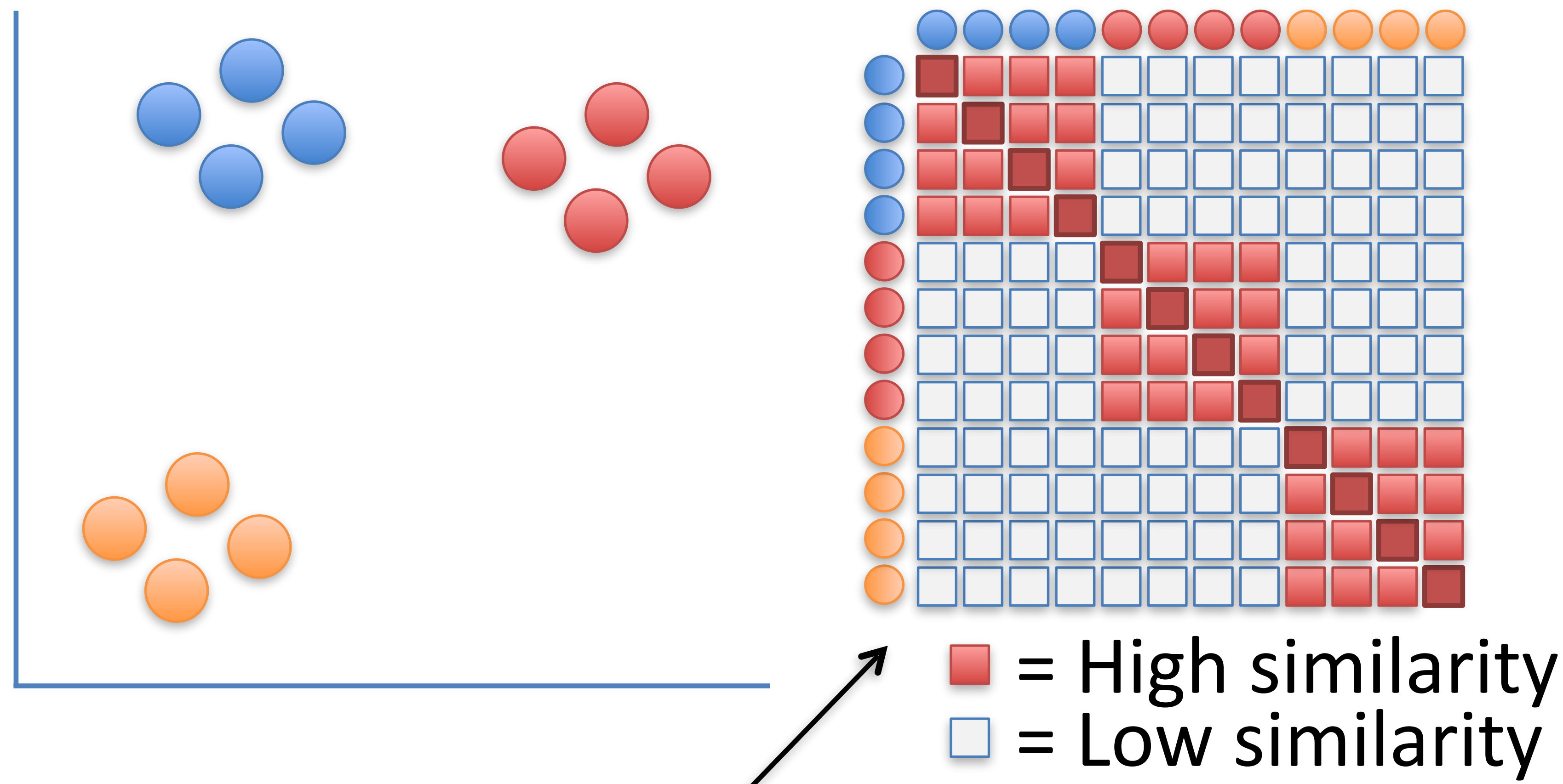
Because the width of the distribution is based on the density of the surrounding data points, the similarity score to this node...



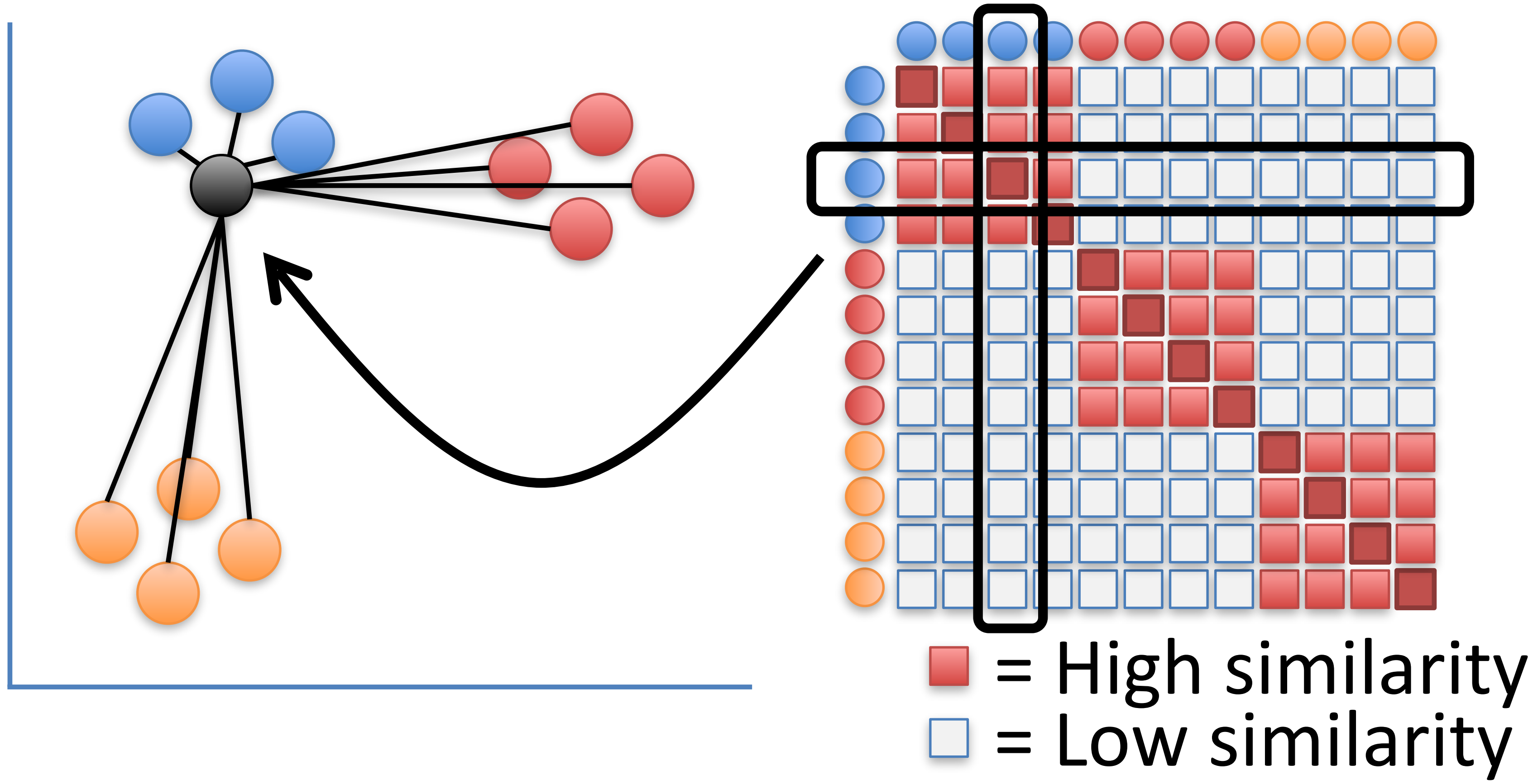
…might not be the same as the similarity to this node.



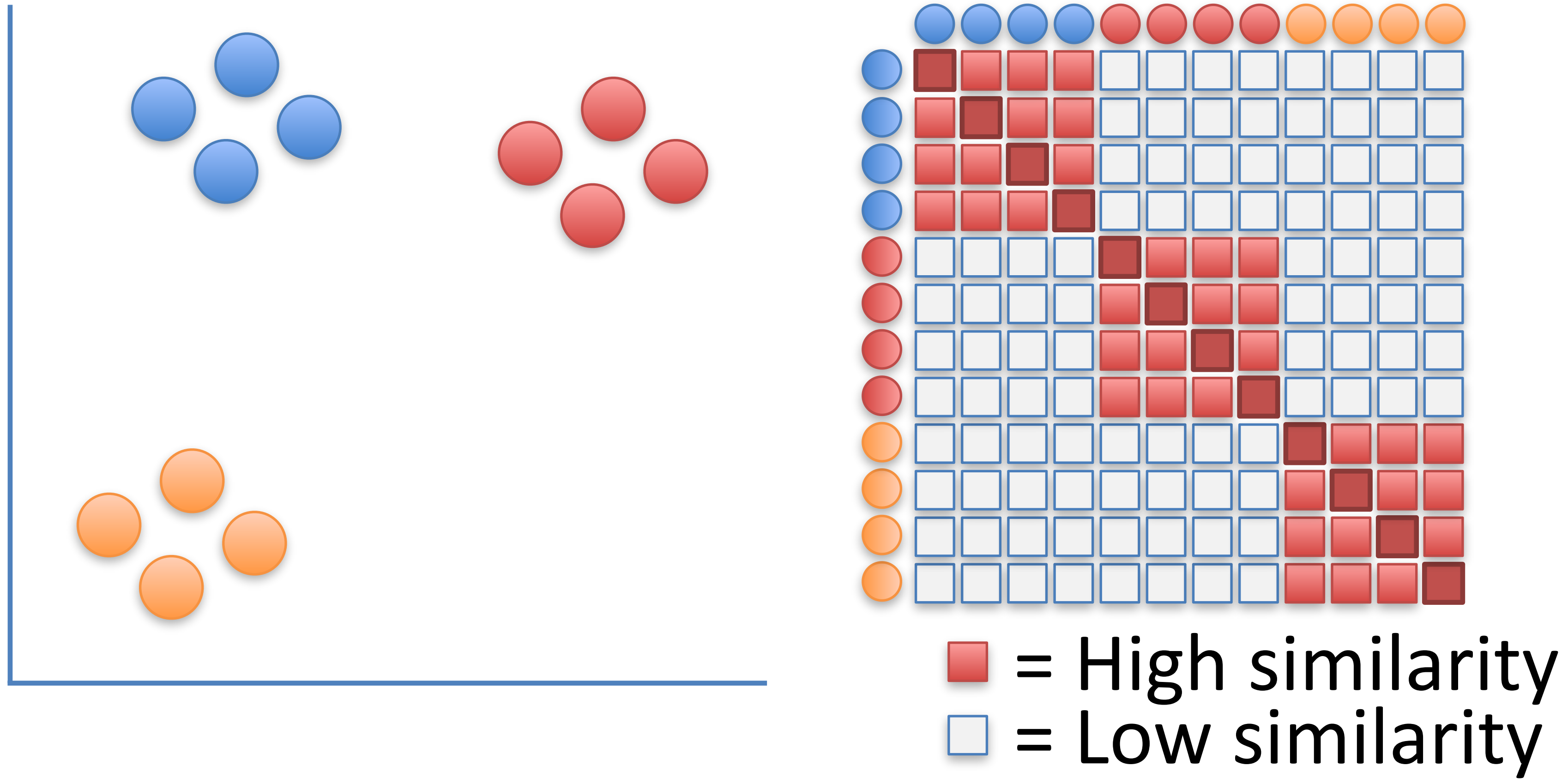

So t-SNE just averages the two similarity scores from the two directions…




Ultimately, you end up with a matrix of similarity scores.

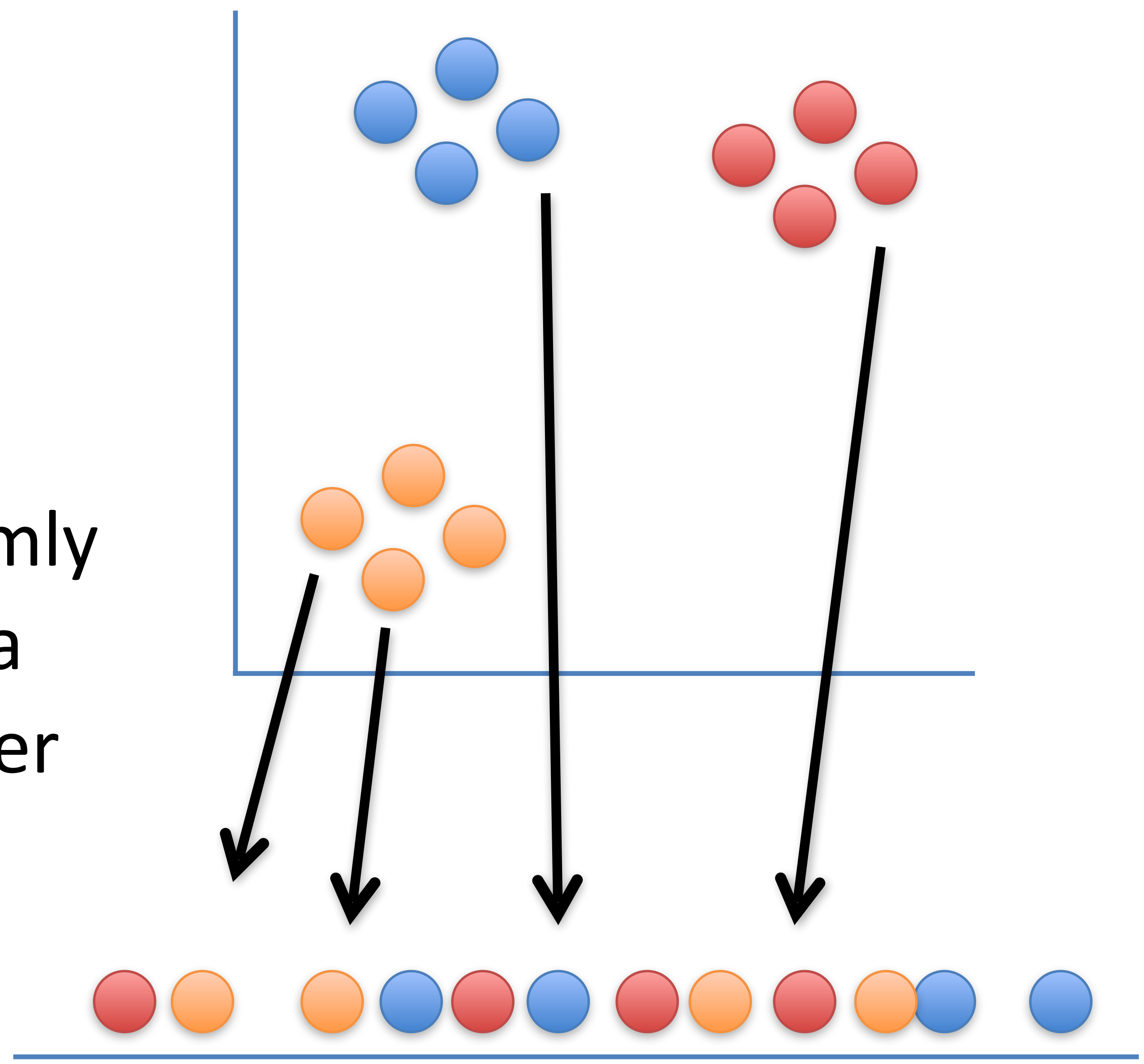


= High similarity
= Low similarity

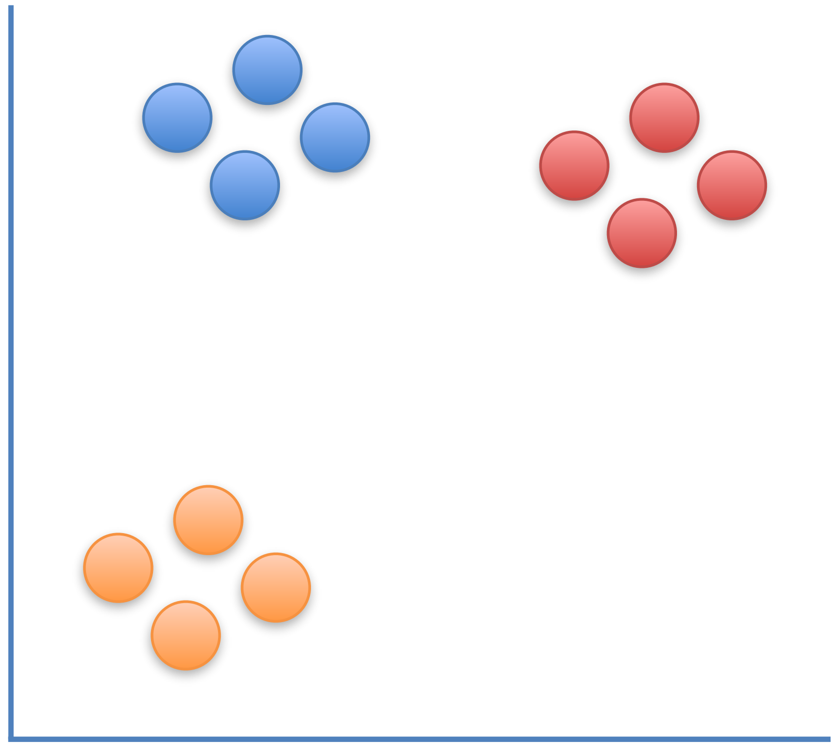



Hooray!!! We're done doing calculating similarity scores for the scatter plot!

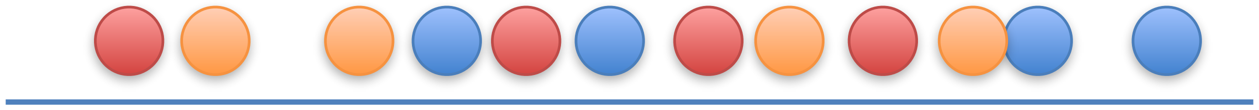
Now we randomly project the data onto the number line…

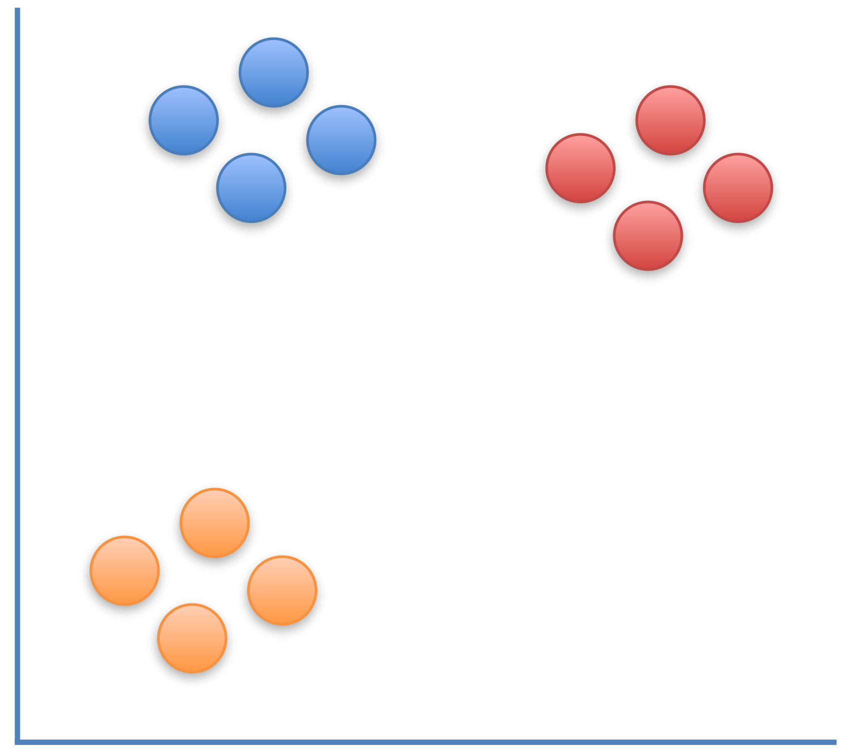


Now we randomly project the data onto the number line…

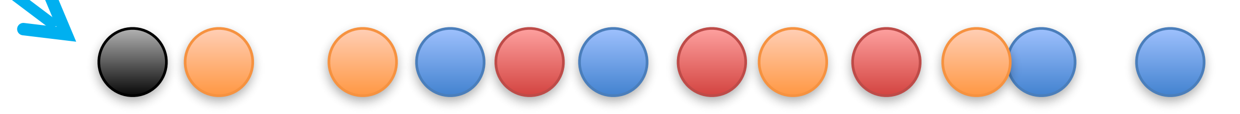


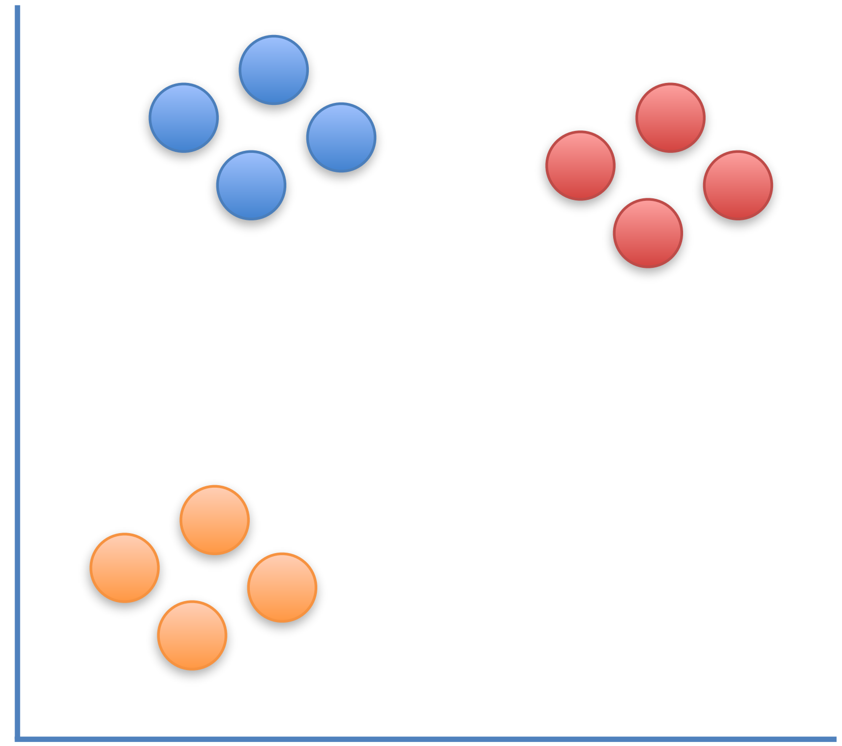
… and calculate similarity scores for the points on the number line.





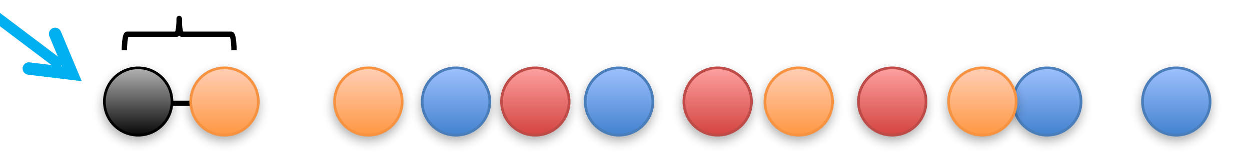
Just like before, that means picking a point...



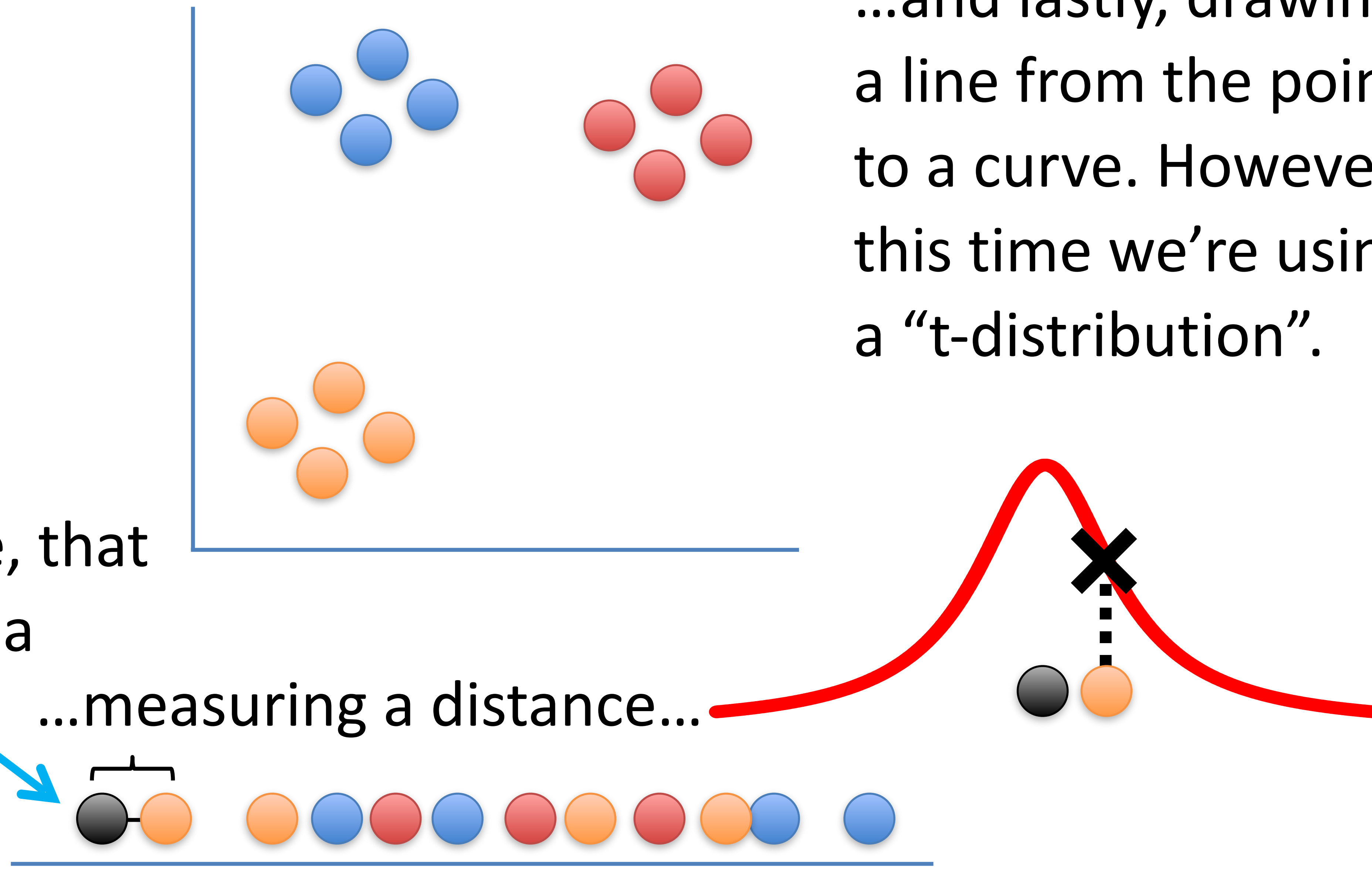


Just like before, that means picking a point…

…measuring a distance…



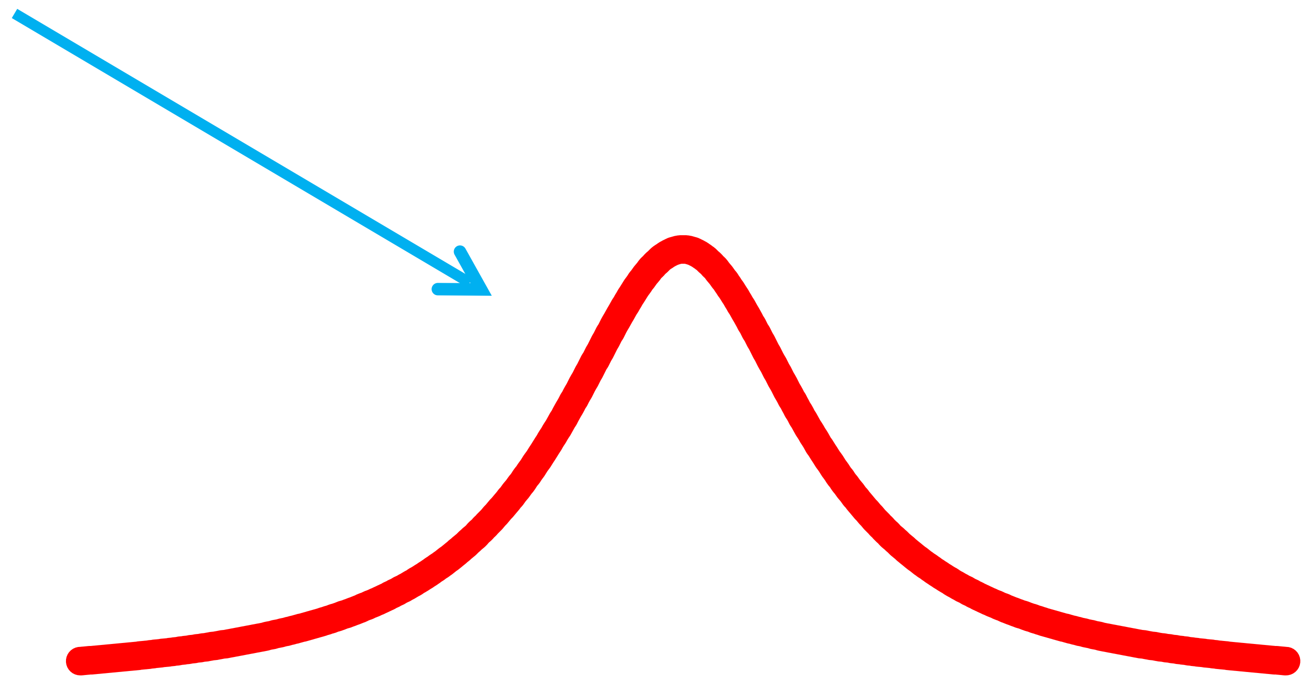
…and lastly, drawing a line from the point to a curve. However, this time we're using a "t-distribution".



Just like before, that means picking a point…

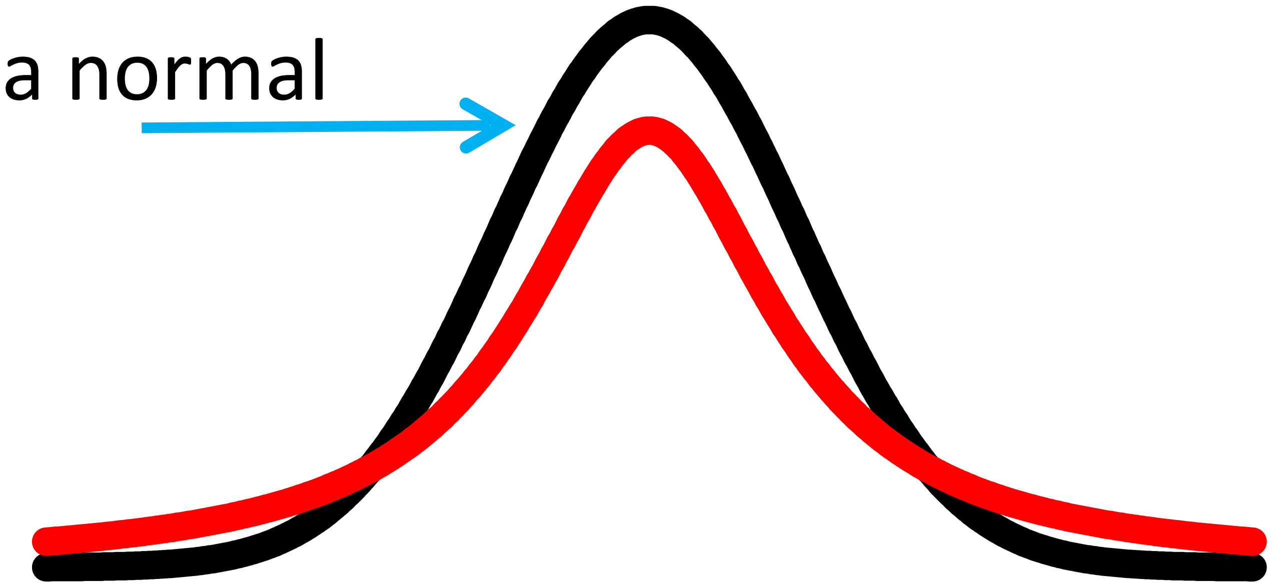
…measuring a distance…

A “t-distribution”…



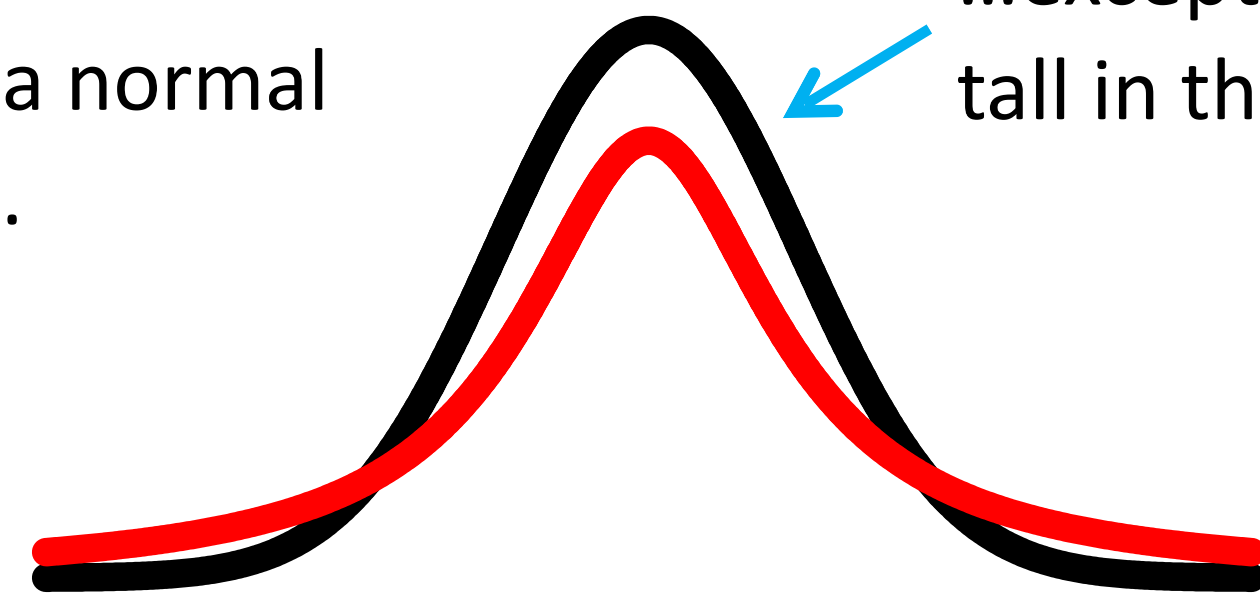
A “t-distribution”…

…is a lot like a normal distribution



A “t-distribution”…

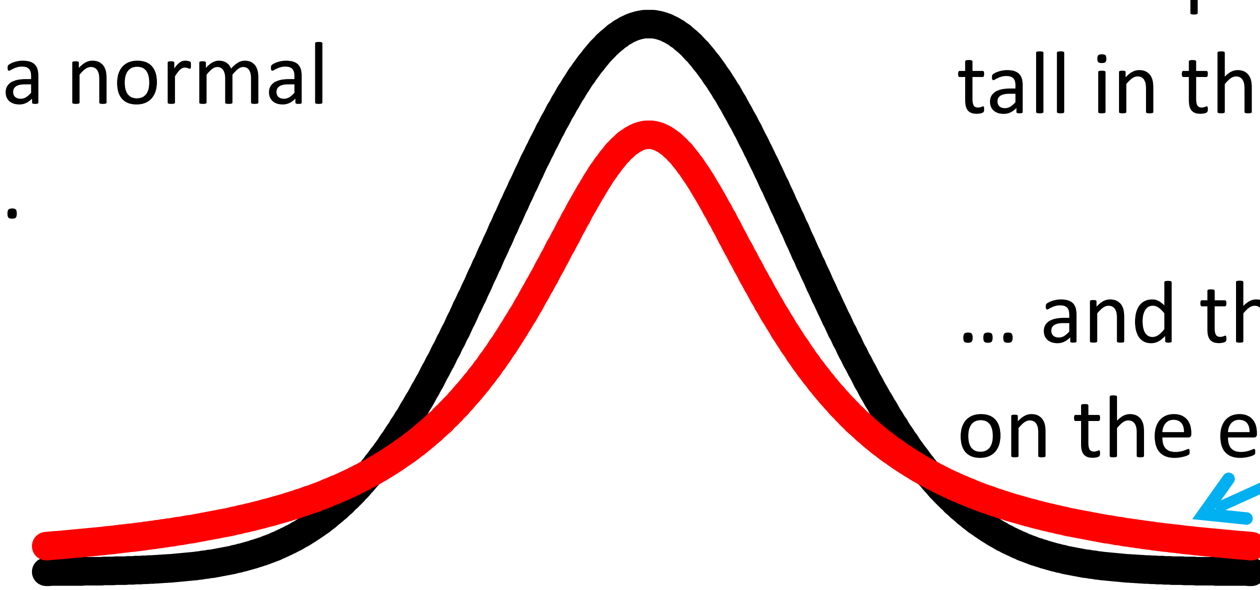
…is a lot like a normal distribution…



…except the “t” isn’t as tall in the middle…

A “t-distribution”…

…is a lot like a normal distribution…



…except the “t” isn’t as tall in the middle…

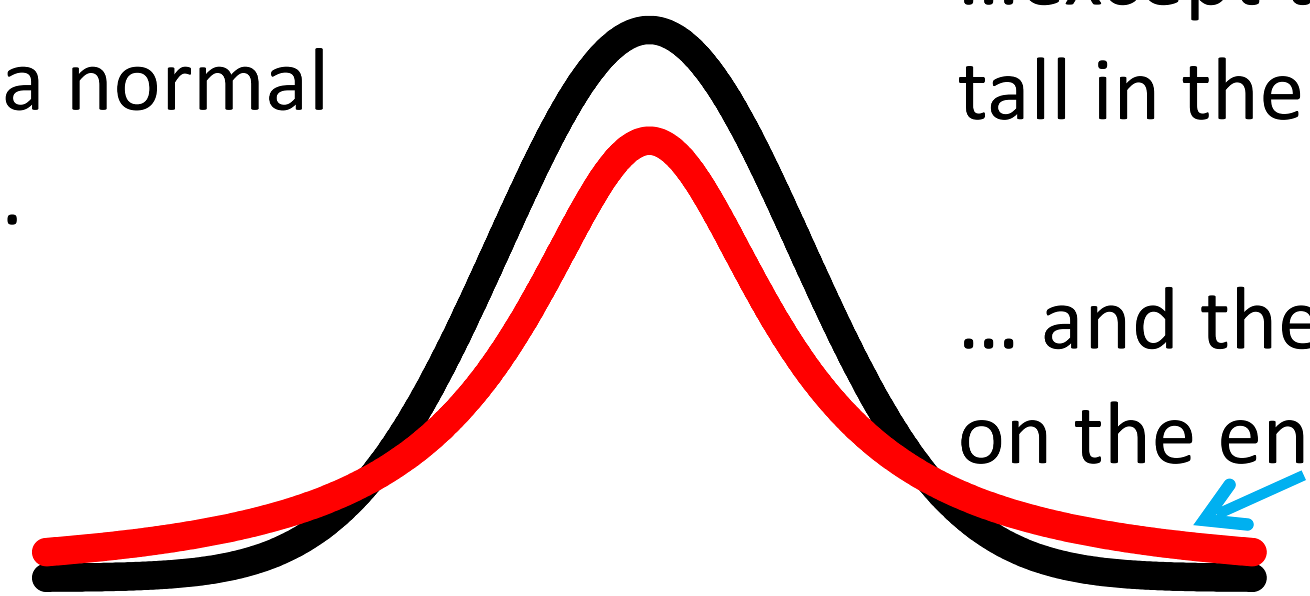
… and the tails are taller on the ends.

A “t-distribution”…

…is a lot like a normal distribution…

…except the “t” isn’t as tall in the middle…

… and the tails are taller on the ends.



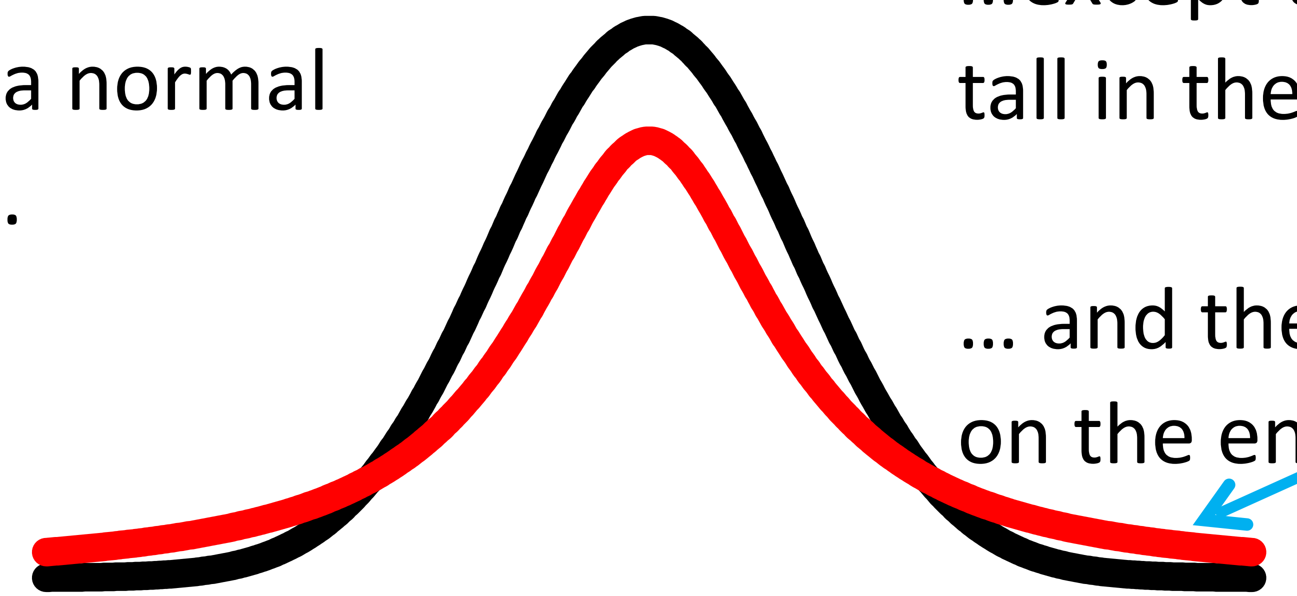
The “t-distribution” is the “t” in t-SNE.

A “t-distribution”…

…is a lot like a normal distribution…

…except the “t” isn’t as tall in the middle…

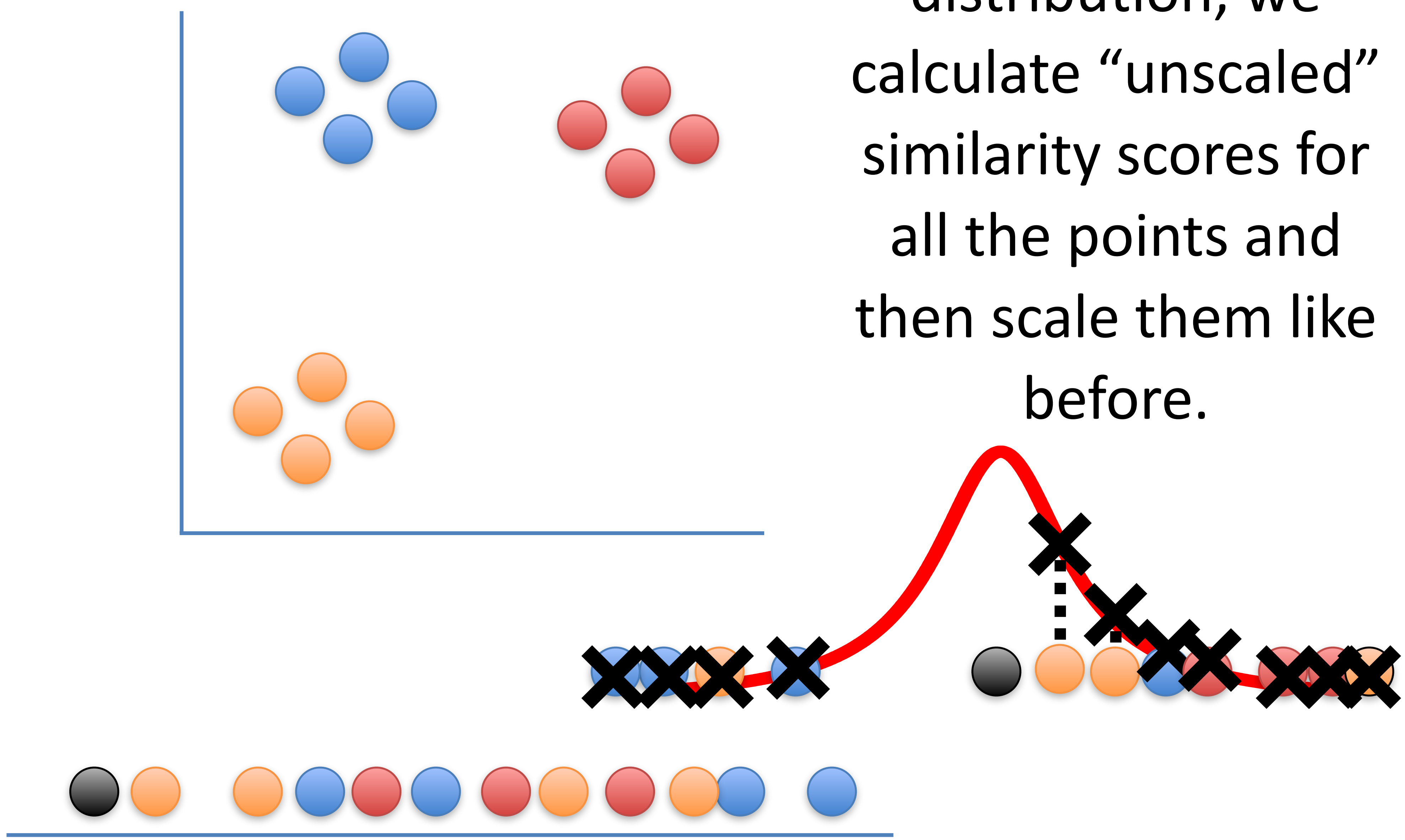
… and the tails are taller on the ends.

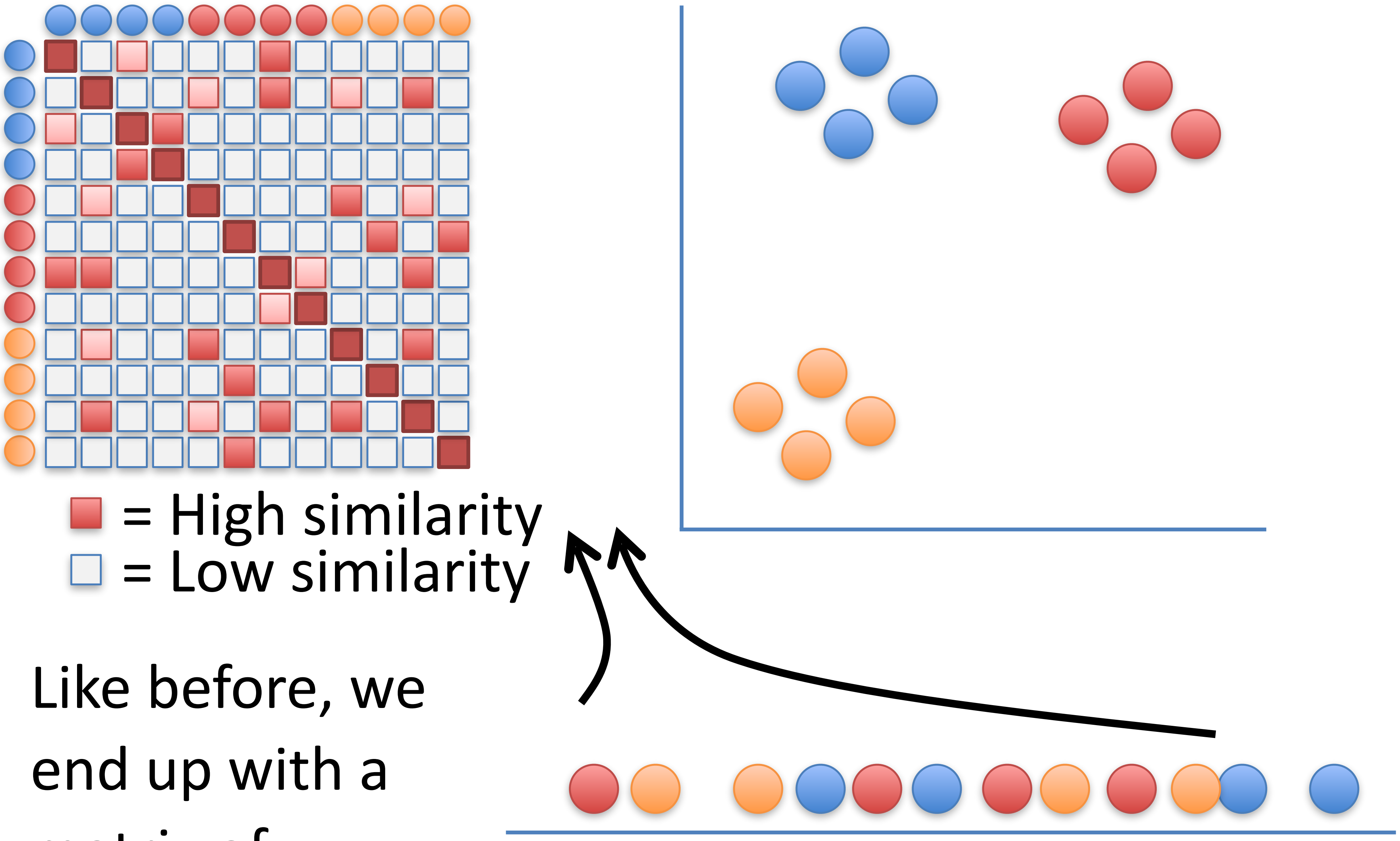


The “t-distribution” is the “t” in t-SNE.

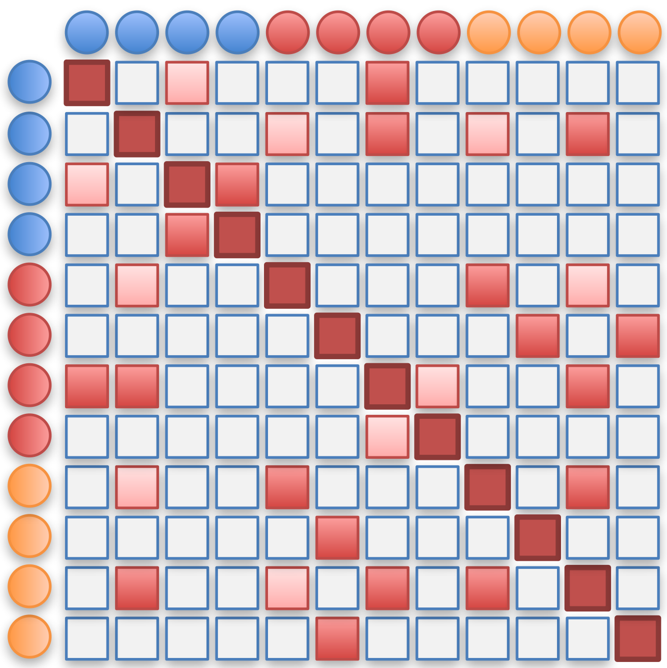
We’ll talk about why the t-distribution is used in a bit…

So, using a t-distribution, we calculate "unscaled" similarity scores for all the points and then scale them like before.

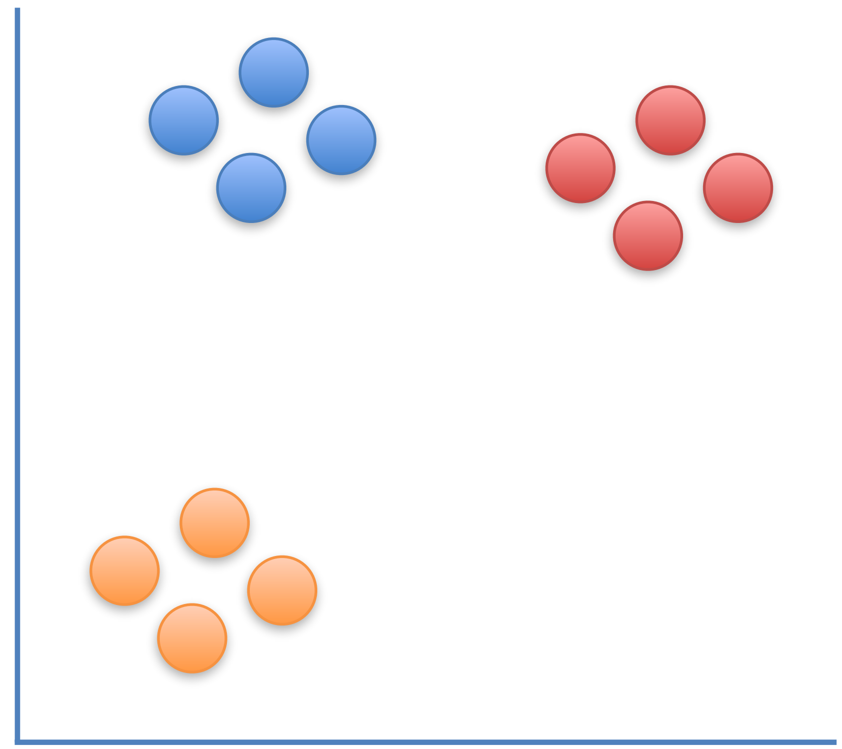


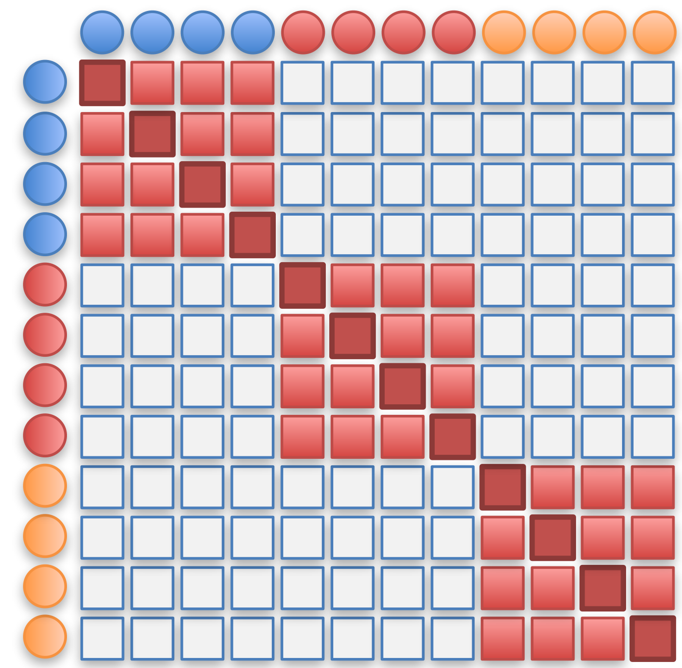



Like before, we end up with a matrix of similarity scores, but this matrix is a mess...



= High similarity
= Low similarity

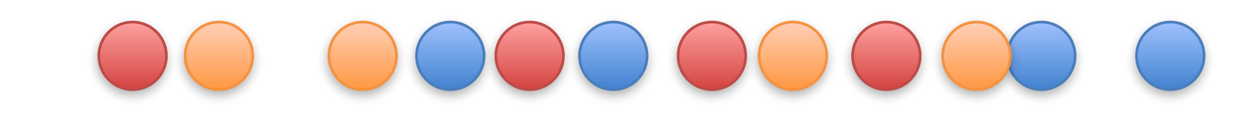


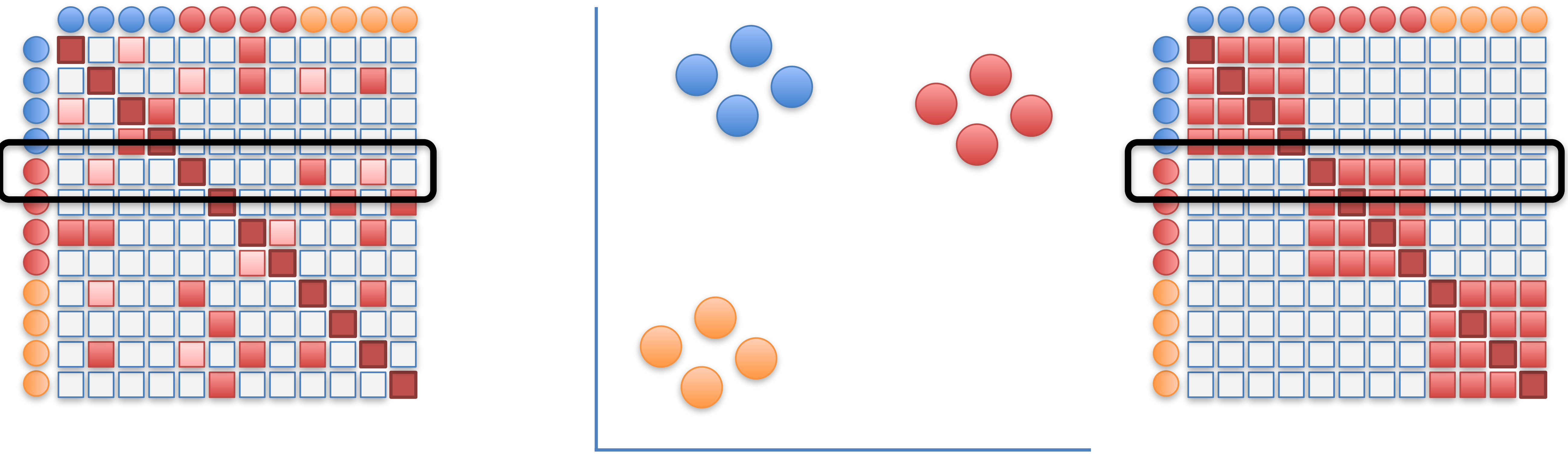


= High similarity
= Low similarity

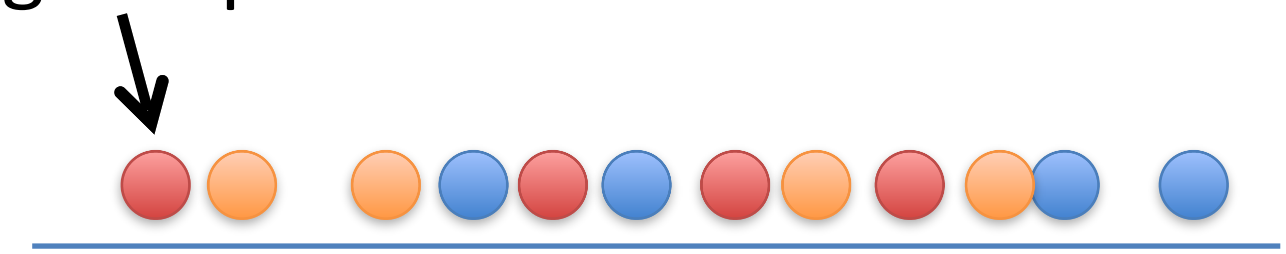
...compared to the original matrix.

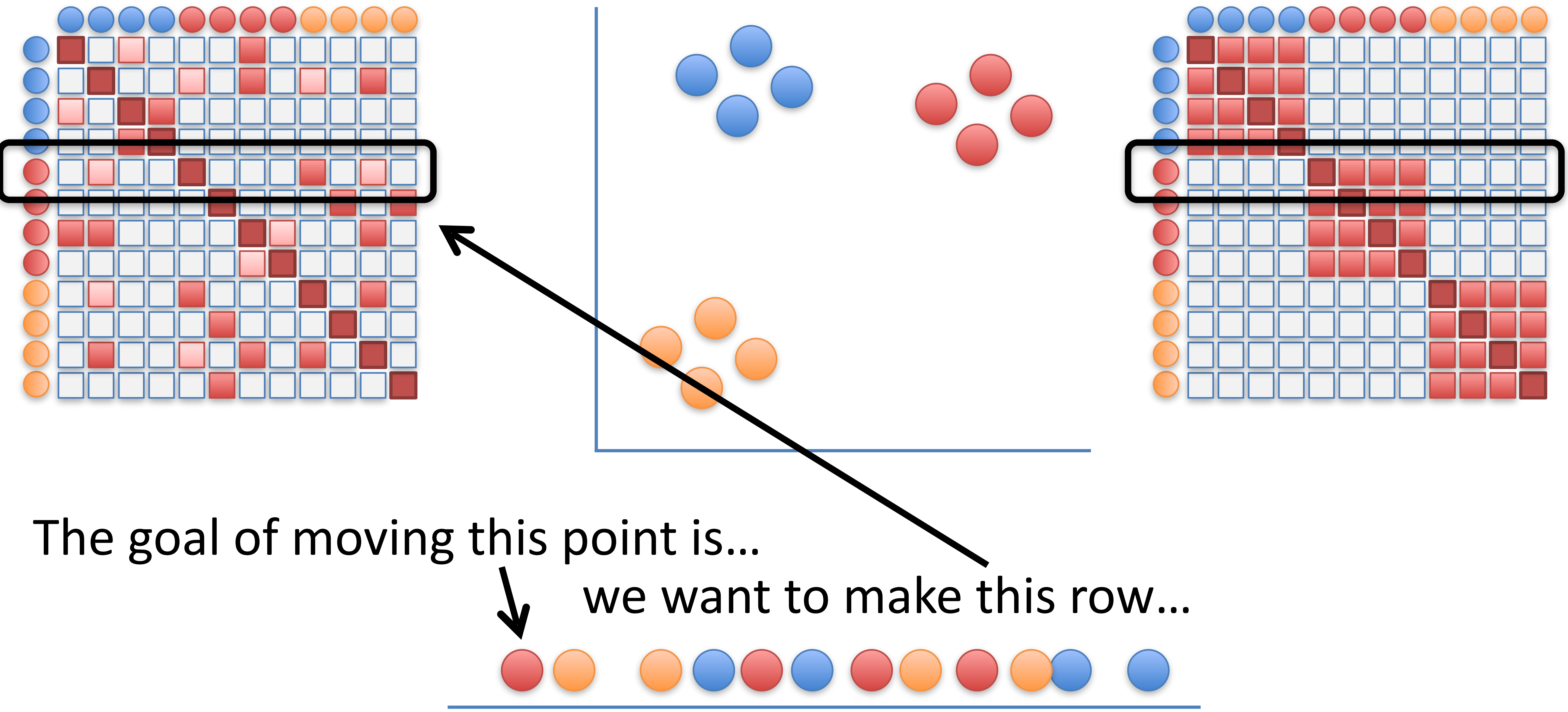
Like before, we end up with a matrix of similarity scores, but this matrix is a mess...

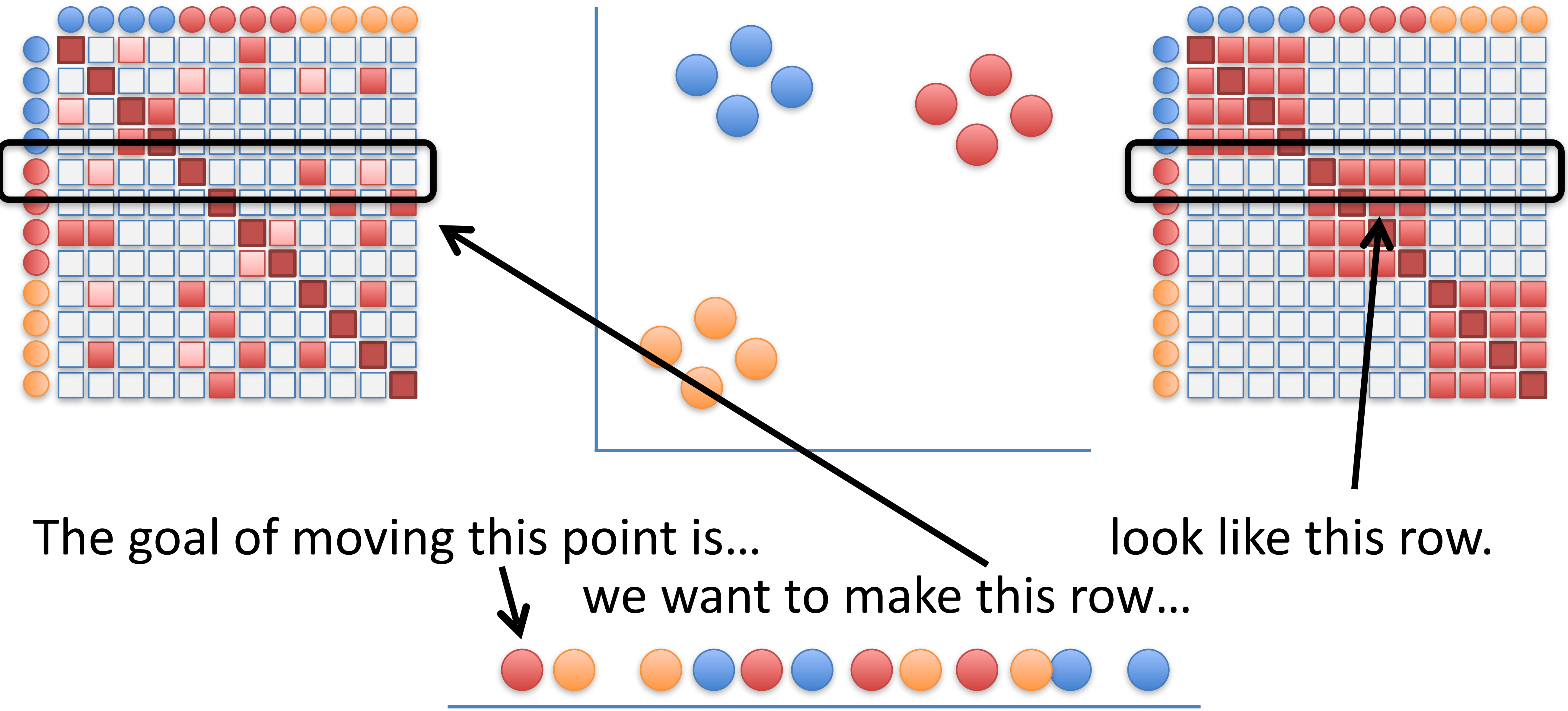


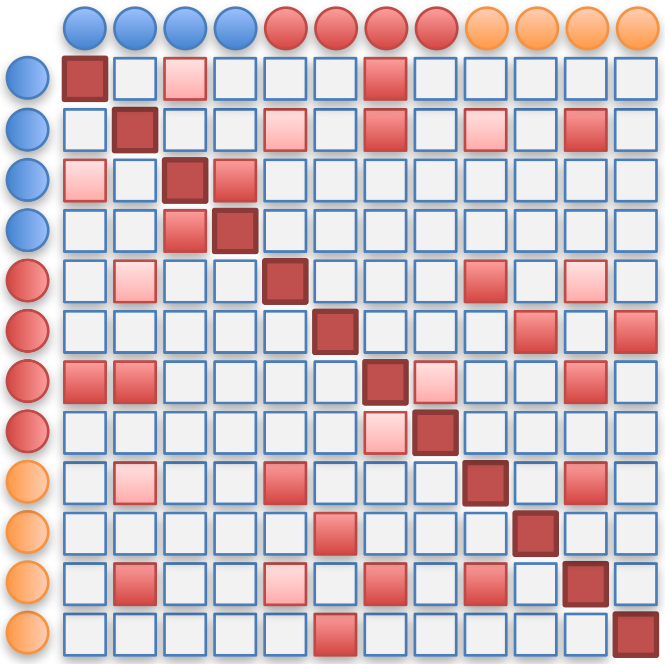


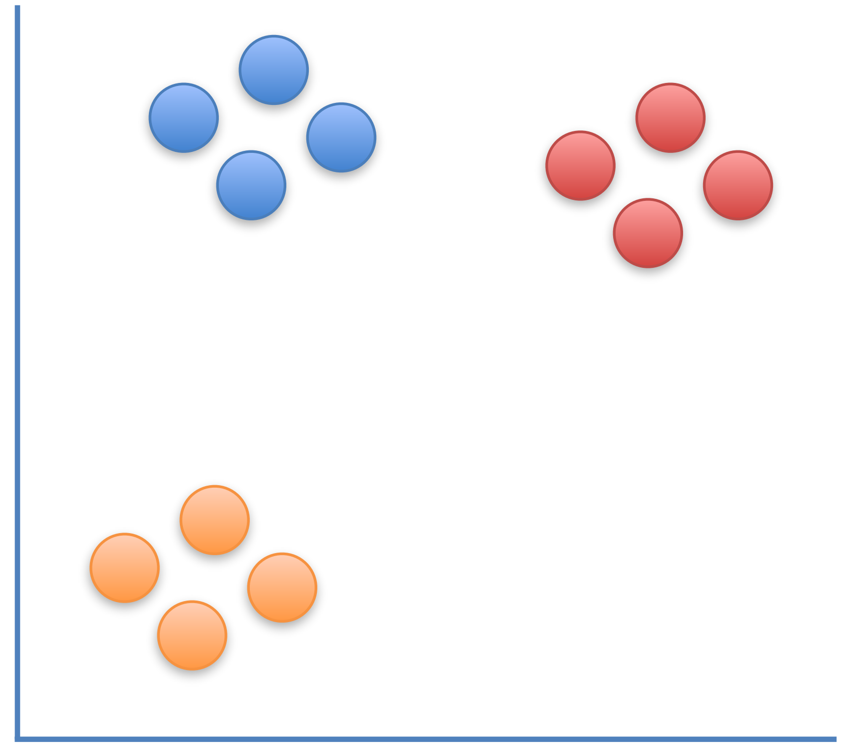
The goal of moving this point is...

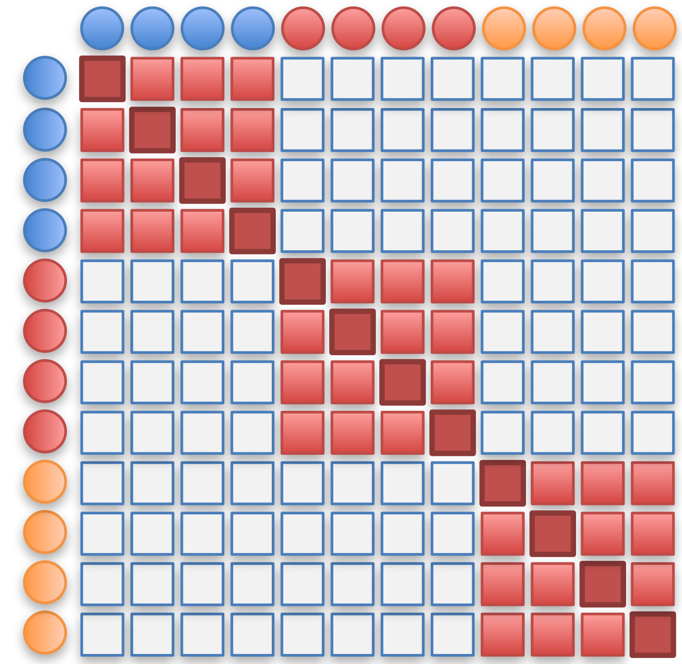



The goal of moving this point is...
we want to make this row...

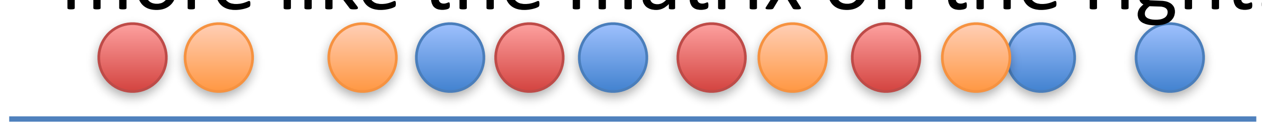

The goal of moving this point is...
we want to make this row...
look like this row.

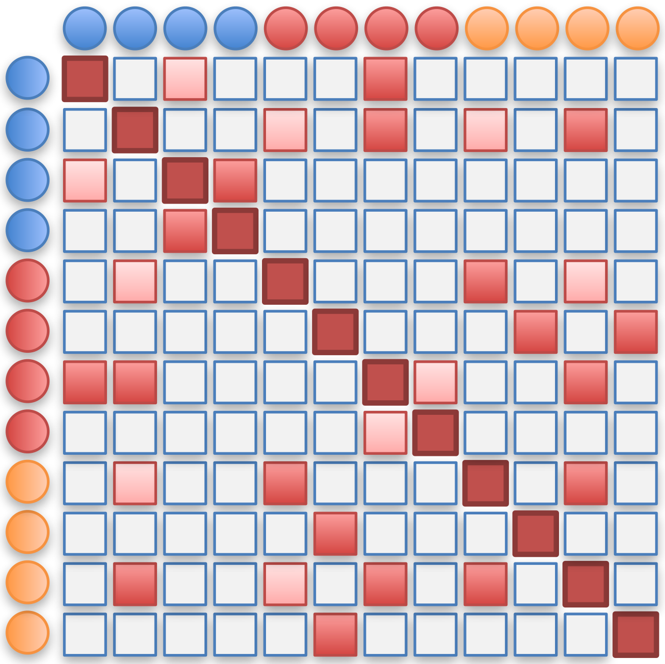


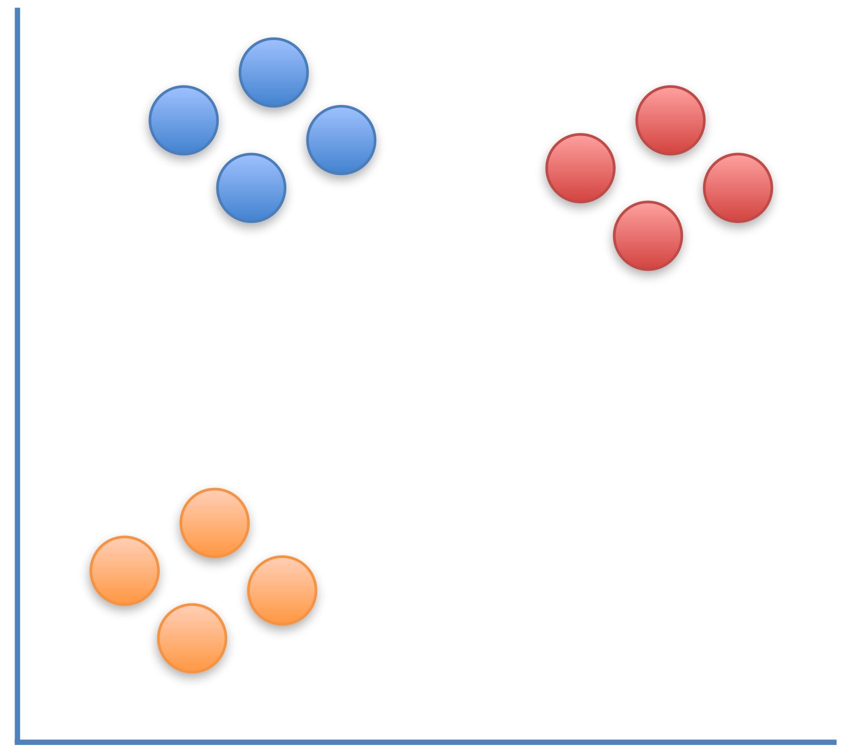


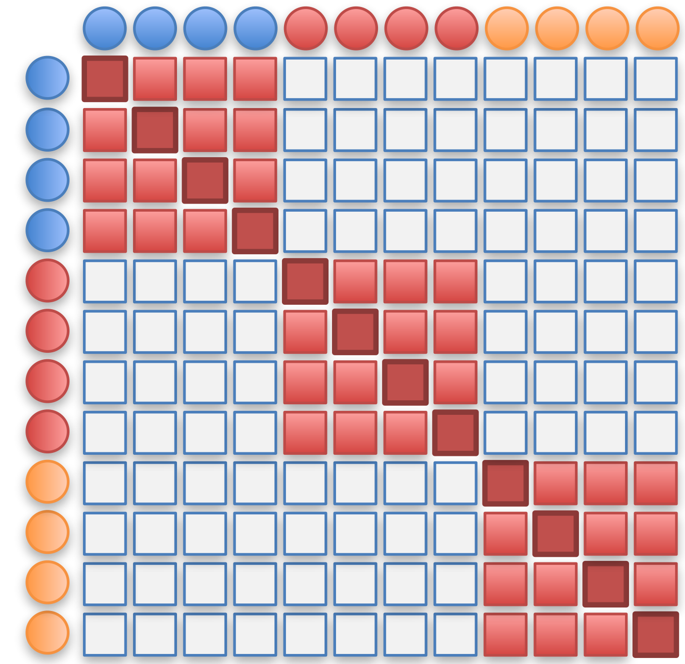


t-SNE moves the points a little bit at a time, and each step it chooses a direction that makes the matrix on the left more like the matrix on the right.

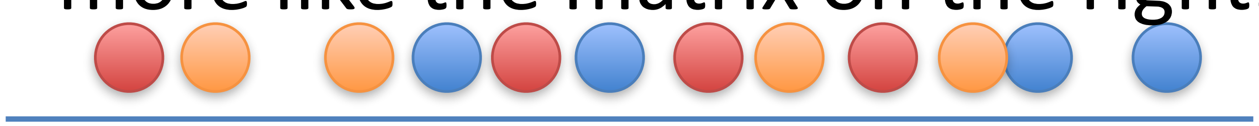




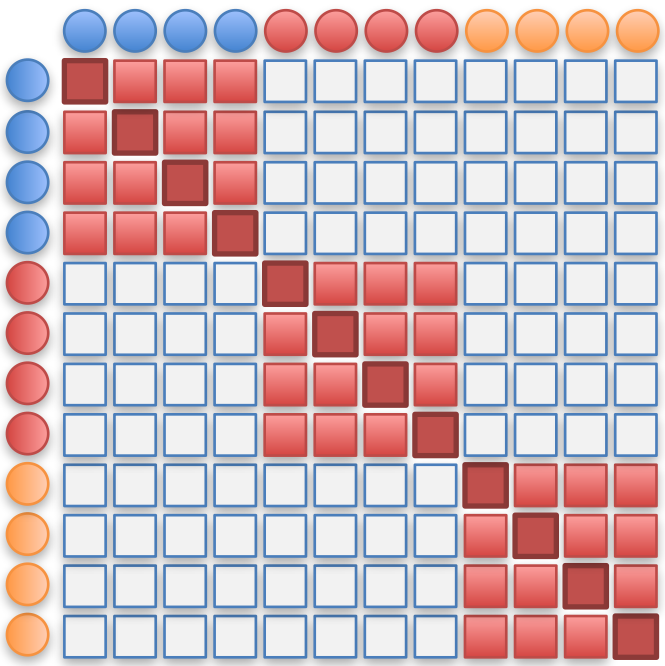


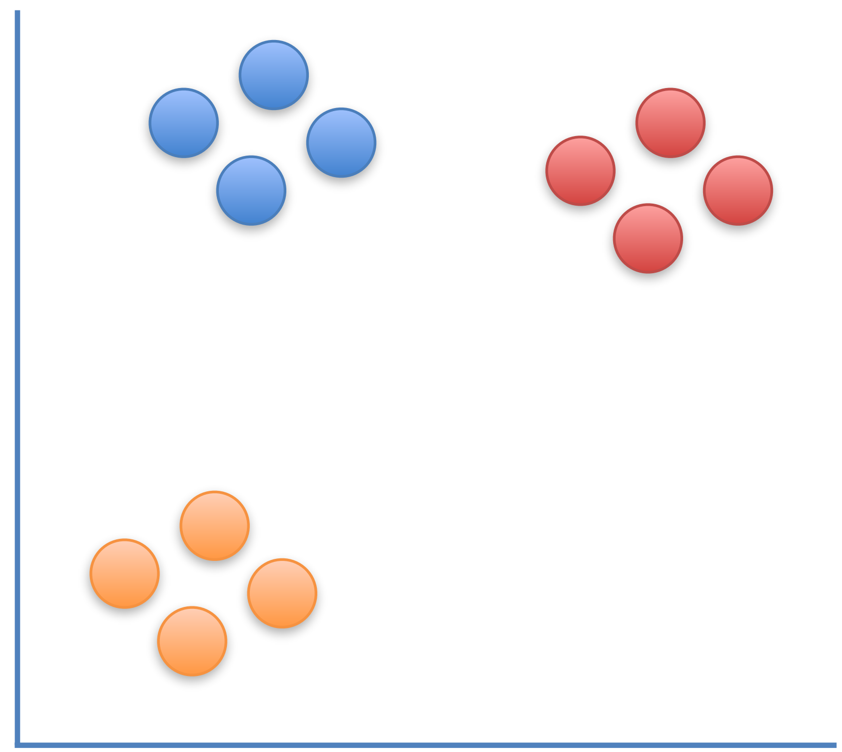


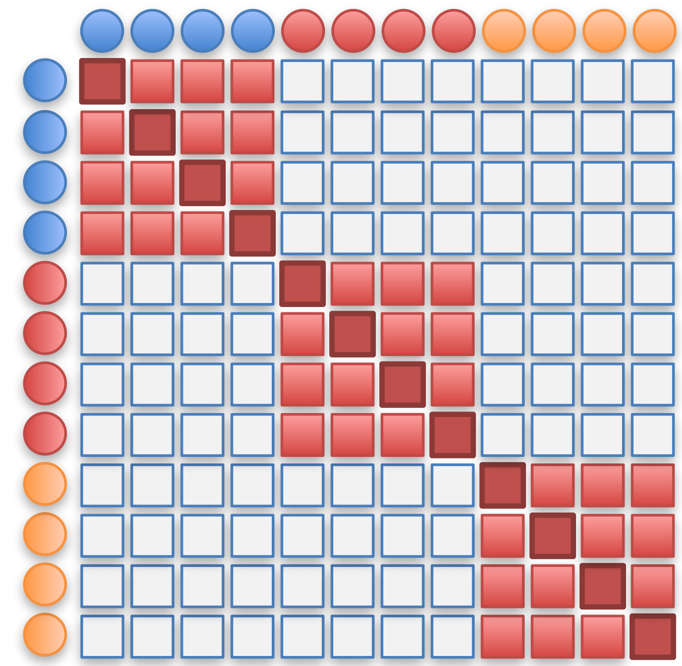
t-SNE moves the points a little bit at a time, and each step it chooses a direction that makes the matrix on the left more like the matrix on the right.



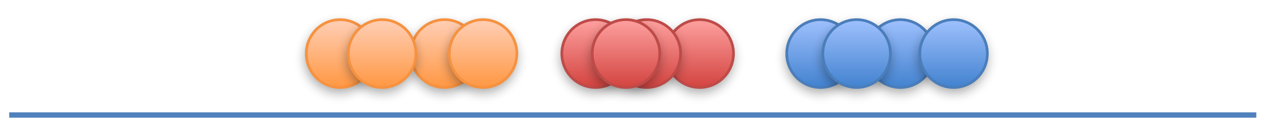
It uses small steps, because it's a little bit like a chess game and can't be solved all at once. Instead, it goes one move at at time.

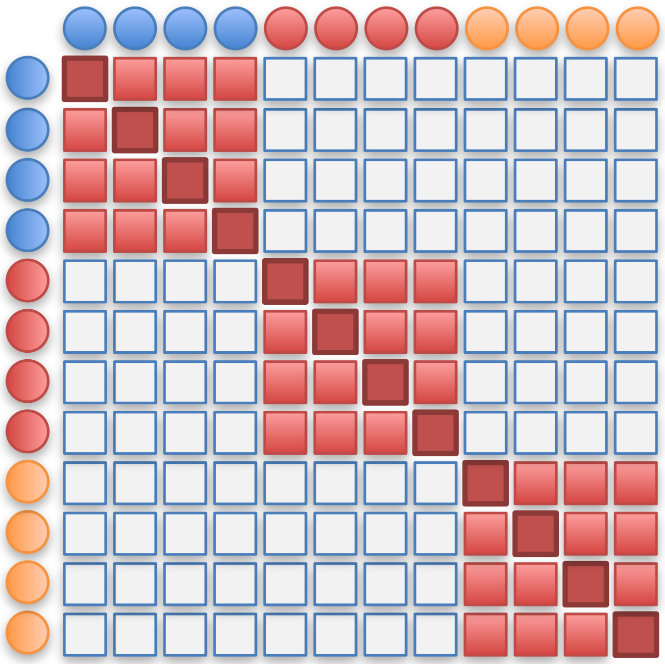


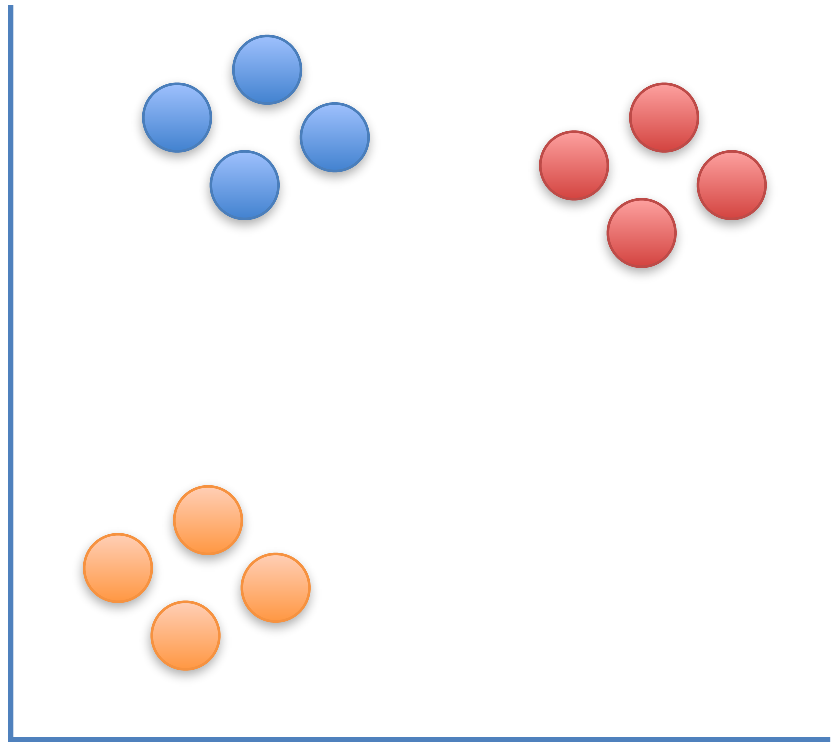


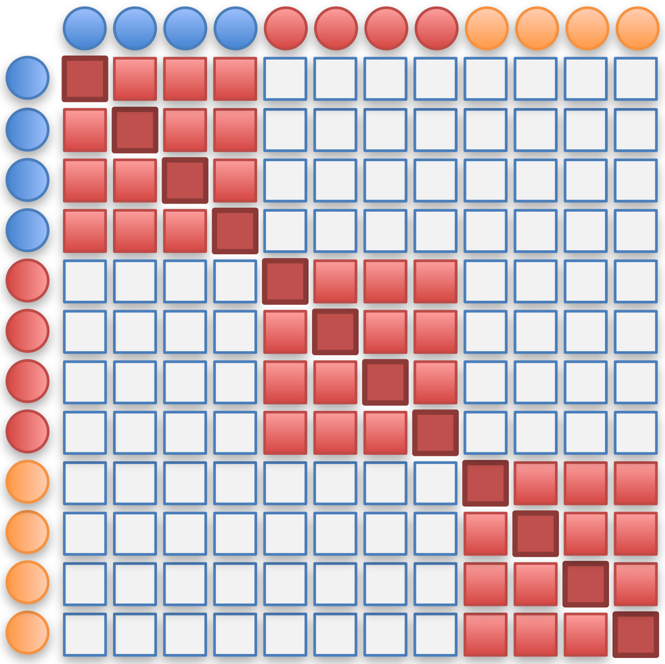


BAM!!!

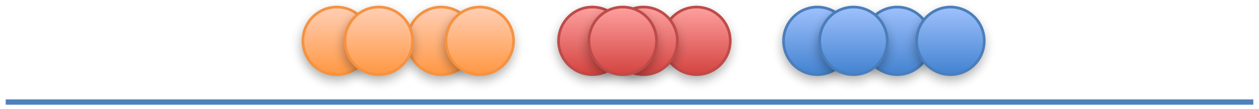


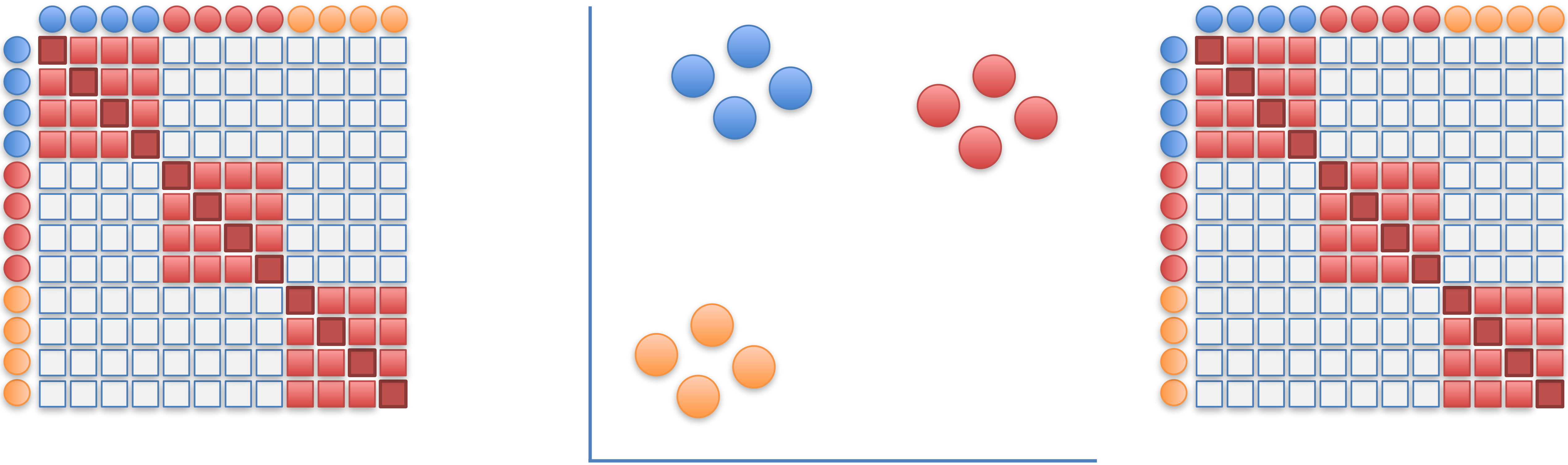




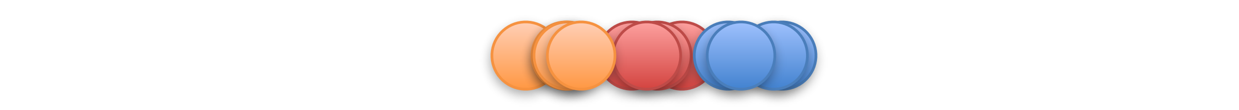


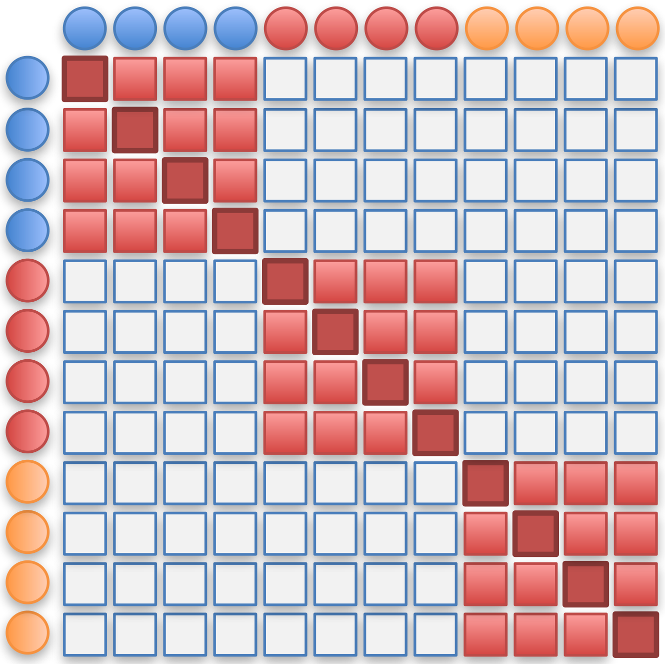
Now to finally tell you why the “t-distribution” is used…

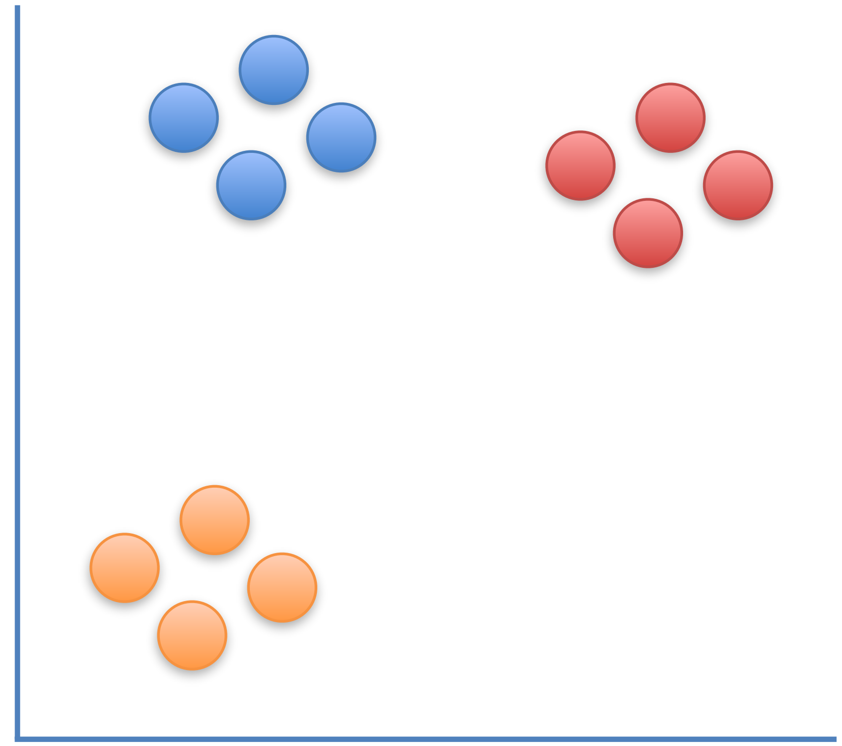


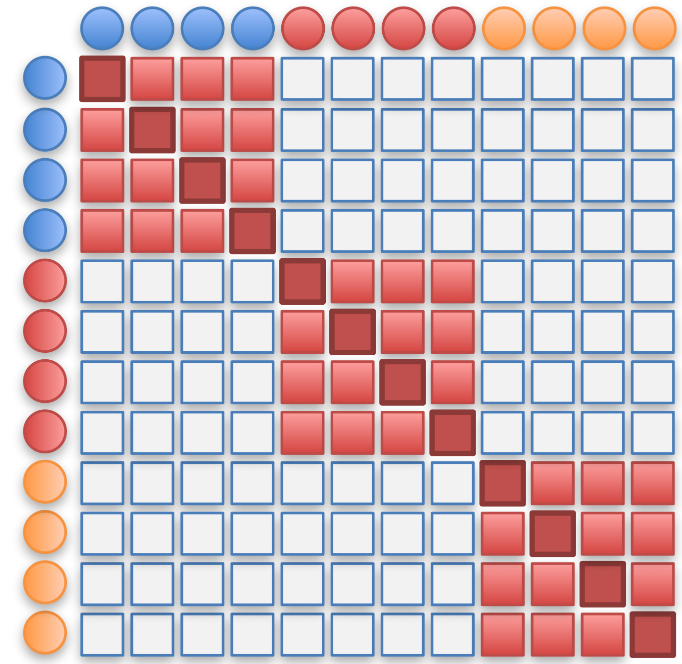


Originally, the "SNE" algorithm used a normal distribution throughout and the clusters clumped up in the middle and were harder to see.

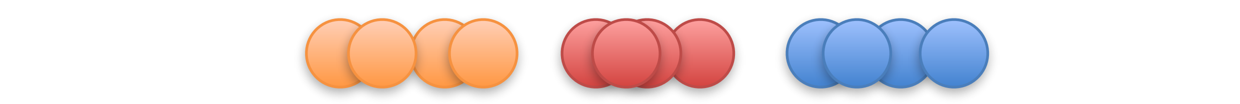


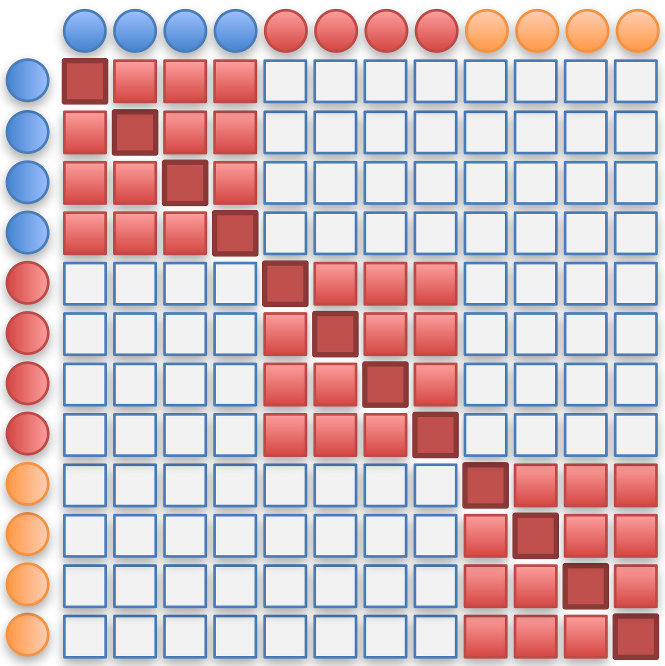


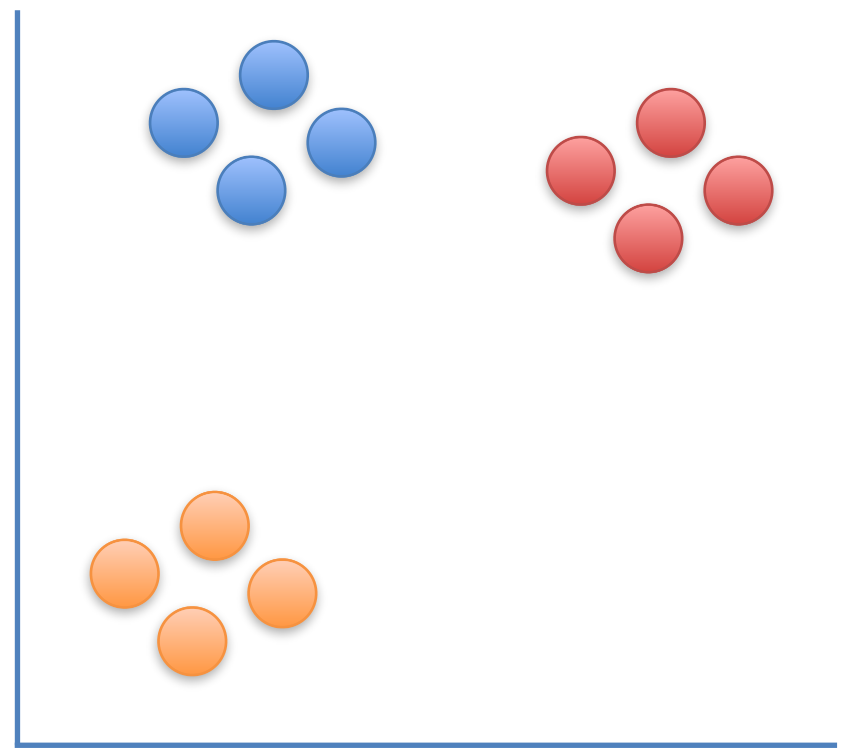


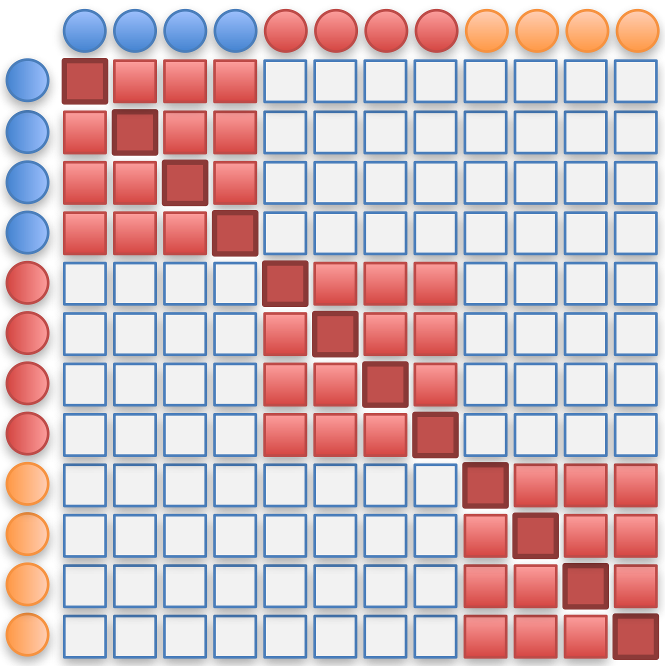


The t-distribution forces some space between the points.









**Triple Bam!!!**