CSE 445
Lecture 15

Random Forest - 2 & T-SNE

Handling missing values

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	???	???	No

Here's our dataset…

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	???	???	No

We've got data for 4 patients…

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	???	???	No

Random Forests consider 2 types of missing data…

1) Missing data in the original dataset used to create the random forest.

2) Missing data in a new sample that you want to categorize.

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	???	???	No

New Sample

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	???	

Random Forests consider 2 types of missing data…

1) Missing data in the original dataset used to create the random forest.

We'll start with this one…

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	???	???	No

So we want to create a random forest from this data…

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**???**	**???**	No

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	???	???	No

However, we don't know if this patient has blocked arteries or their weight.

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	???	???	No

The general idea for dealing with missing data in this context is to make an initial guess that could be bad, then gradually refine the guess until it is (hopefully) a good guess.

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	???	???	No

So the initial (possibly bad) guess for the blocked arteries value is just the most common value for "Blocked Arteries".

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	???	???	No

"**No**" is the most common value for Blocked arteries - it occurs in 2 out of 3 samples.

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	???	No

← So "**No**" is our initial guess.

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	No	???	No

Since weight is numeric, our initial guess will be the median value.

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	No	180	No

← In this case, the median value is 180

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

Here's our new dataset with the filled in missing values…

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

Now we want to refine these guesses.

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

Now we want to refine these guesses.

We do this by first determining which samples are similar to the one with missing data.

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

Now we want to refine these guesses.

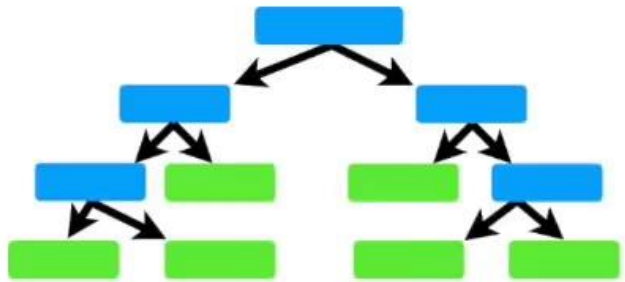
We do this by first determining which samples are similar to the one with missing data.

So let's talk about how to determine similarity…

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

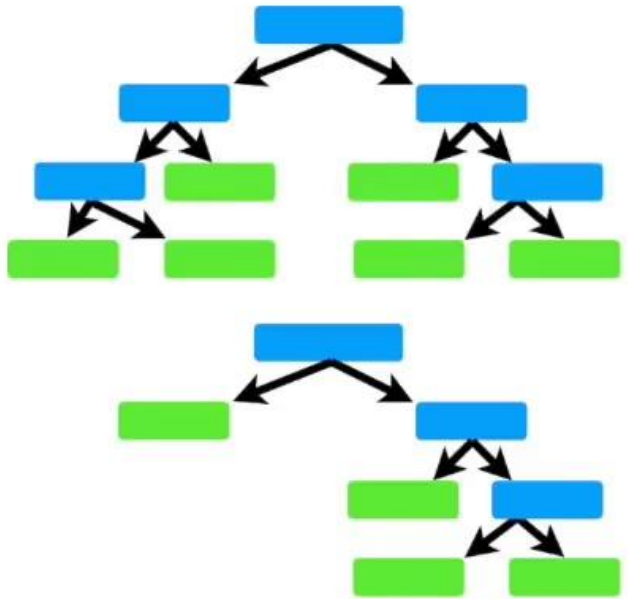
Step 1: Build a random forest…



Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

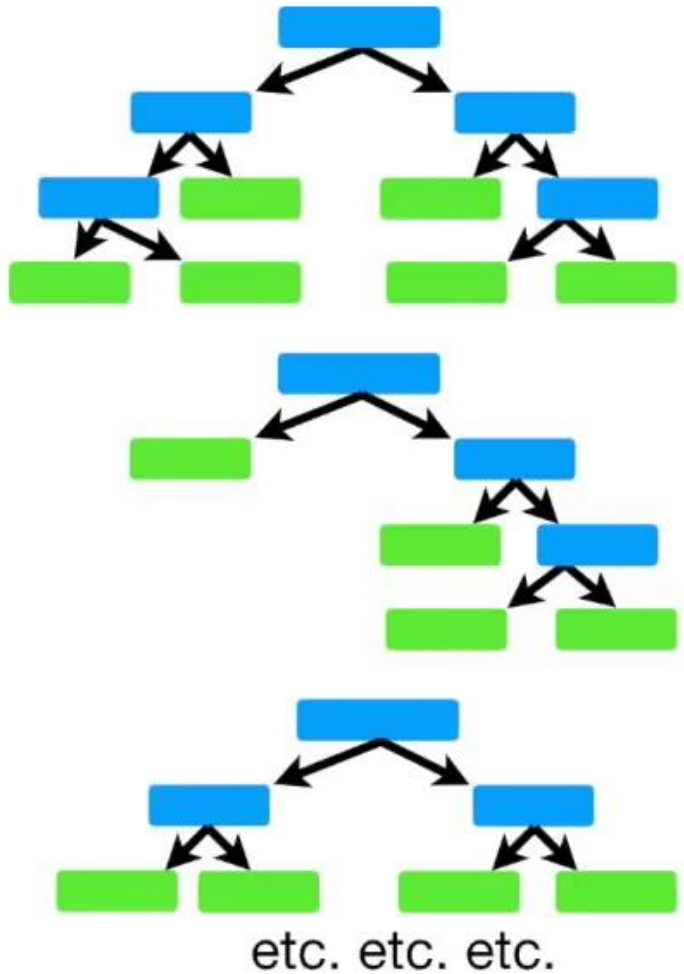
Step 1: Build a random forest…



Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

Step 1: Build a random forest…

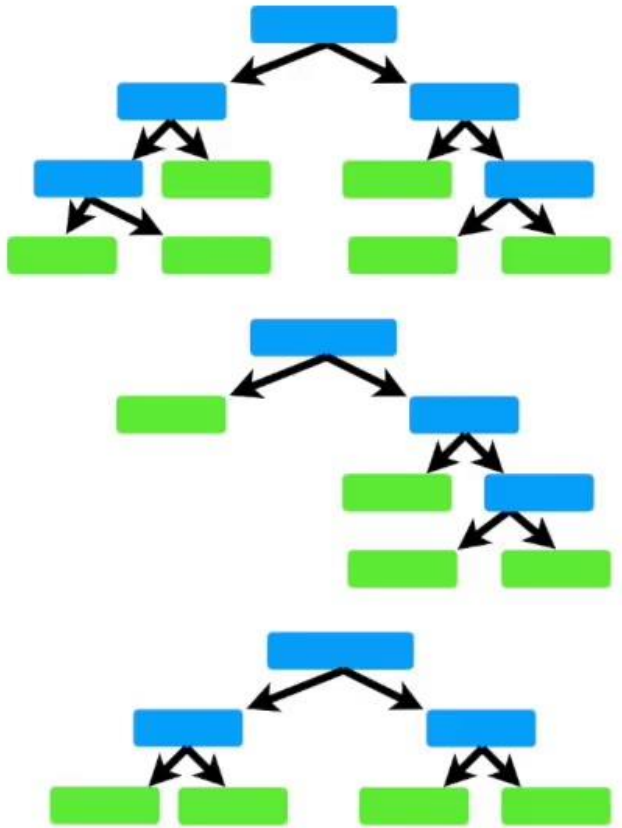


etc. etc. etc.

Filled-in Missing Values

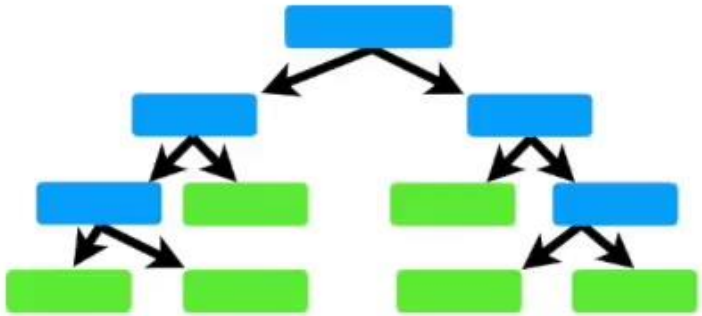
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

Step 2: Run all of the data down all of the trees.



Filled-in Missing Values

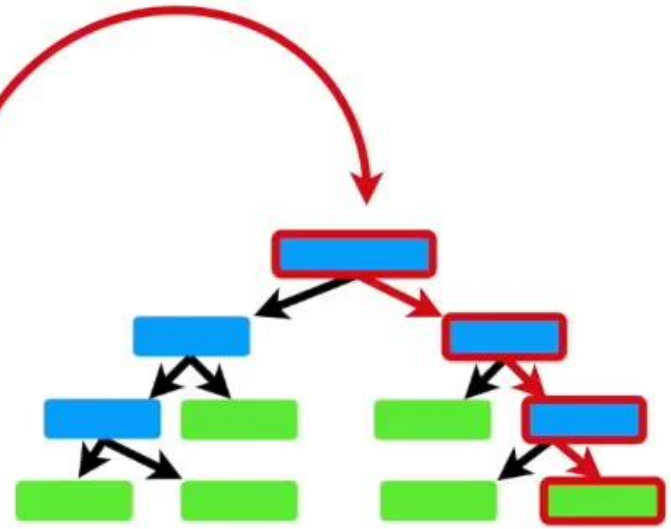
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No



We'll start by running all of the data down the first tree.

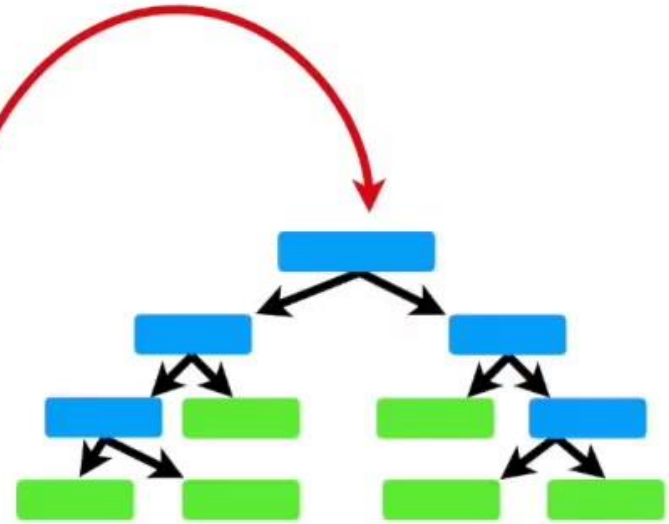
Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No



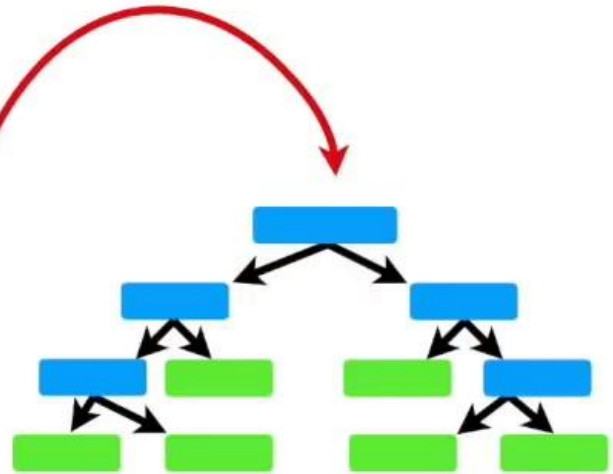
Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No



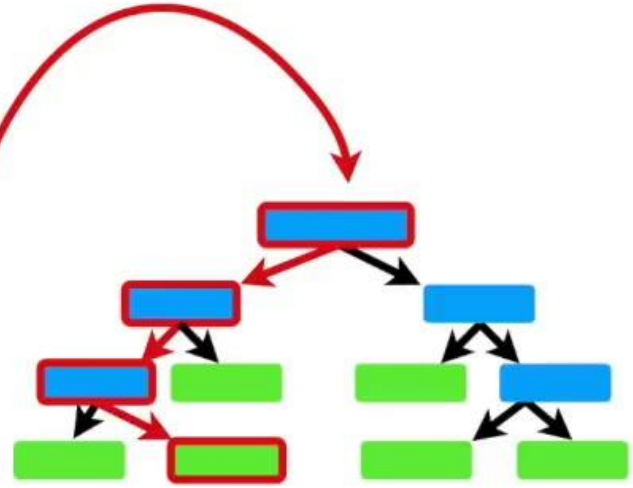
Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No



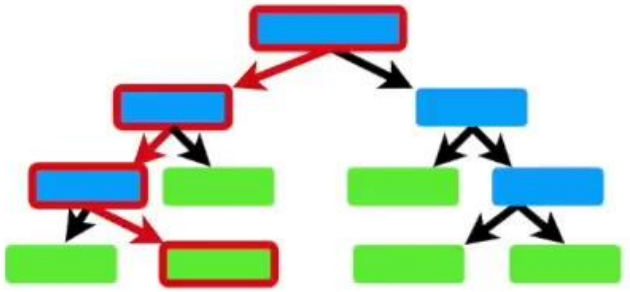
Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No



Filled-in Missing Values

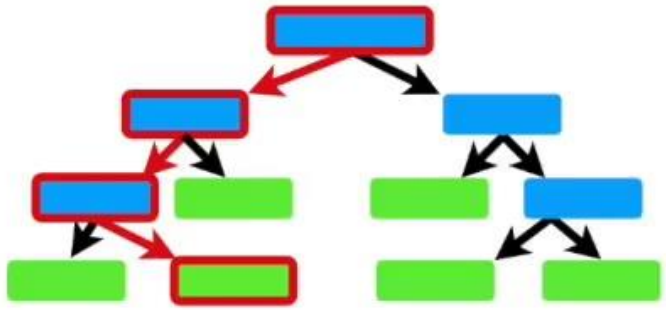
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No



Notice that Sample 3 and Sample 4 both ended up at the same leaf node.

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No



That means they are similar.

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

We keep track of similar samples using a "Proximity Matrix"

	1	2	3	4
1				
2				
3				
4				

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

The proximity matrix has a row for each sample..

	1	2	3	4
1				
2				
3				
4				

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

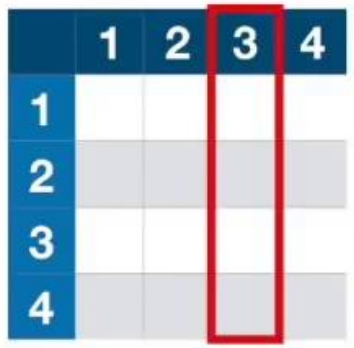
…and it has a column for each sample.

	1	2	3	4
1				
2				
3				
4				

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

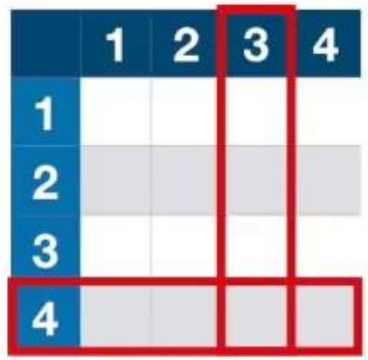
Because sample 3…



Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

Because sample 3…

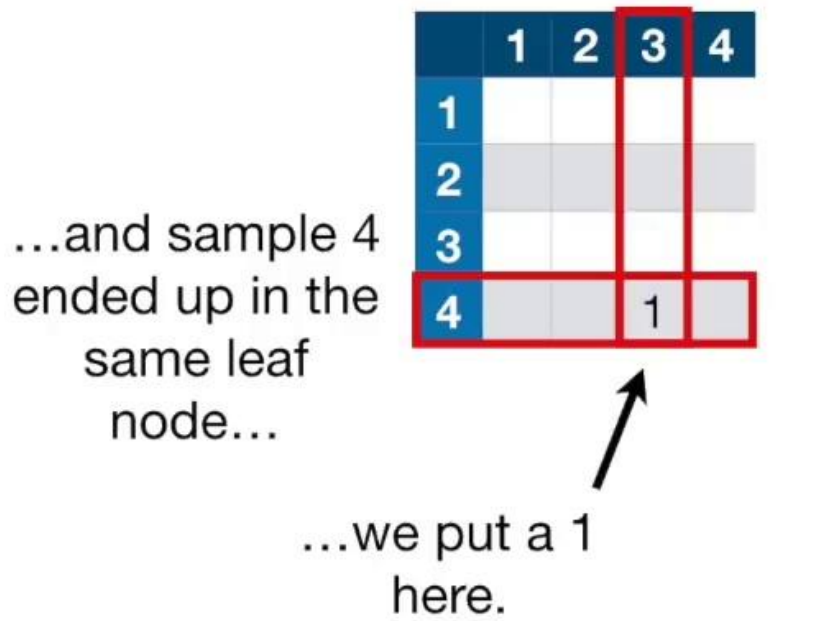


…and sample 4 ended up in the same leaf node…

Filled-in Missing Values

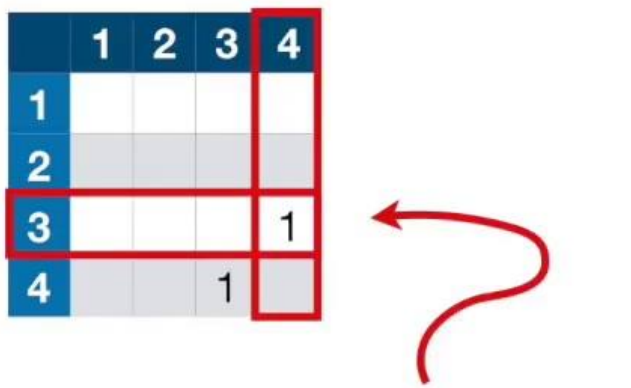
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

Because sample 3…



Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No



We also put a 1 here, since this position also represents samples 3 and 4.

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

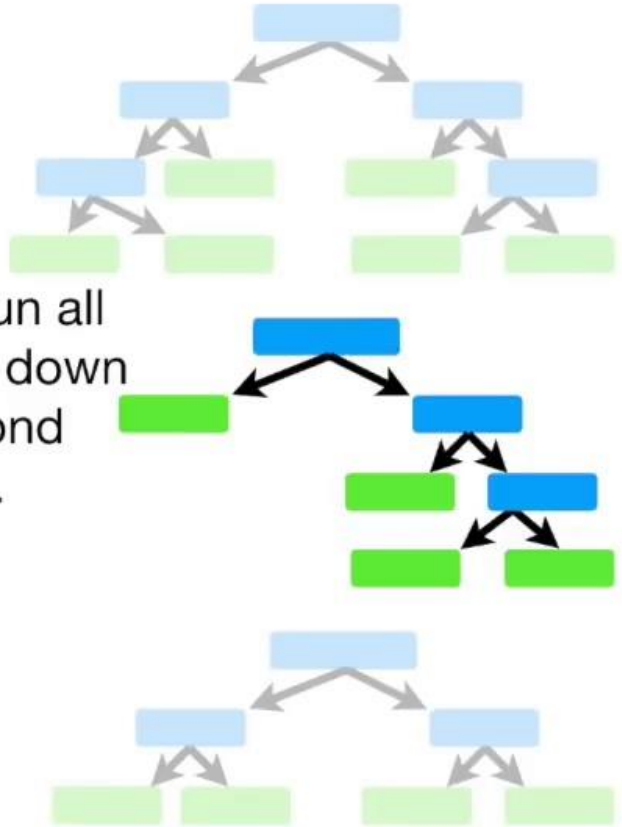
Because no other pair of samples ended in the same leaf node, our proximity matrix looks like this after running the samples down the first tree.

	1	2	3	4
1				
2				
3				1
4			1	

Filled-in Missing Values

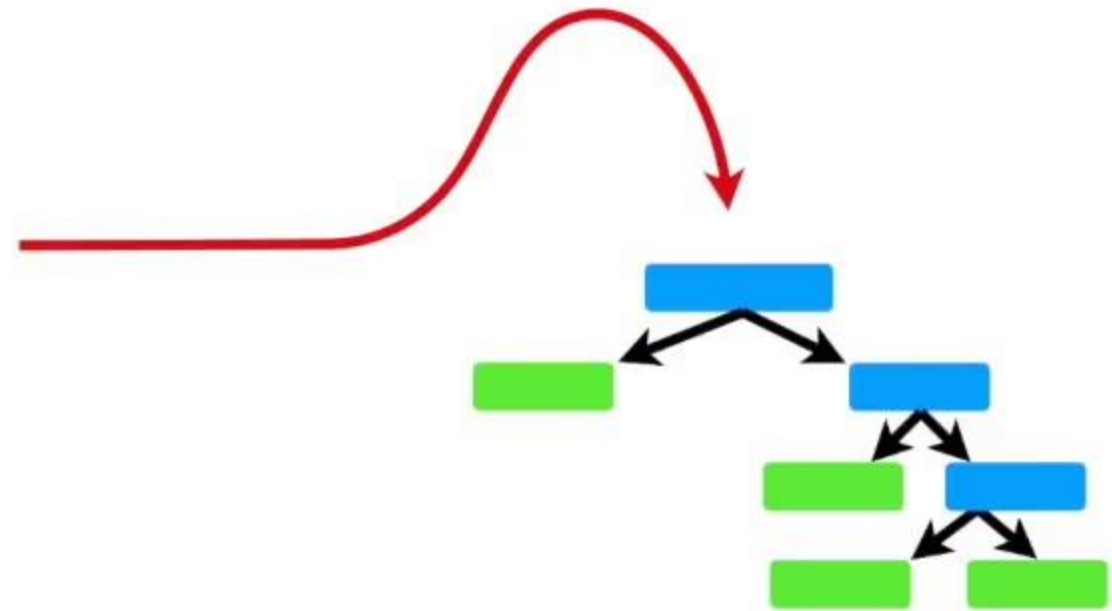
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

Now we run all of the data down the second tree…



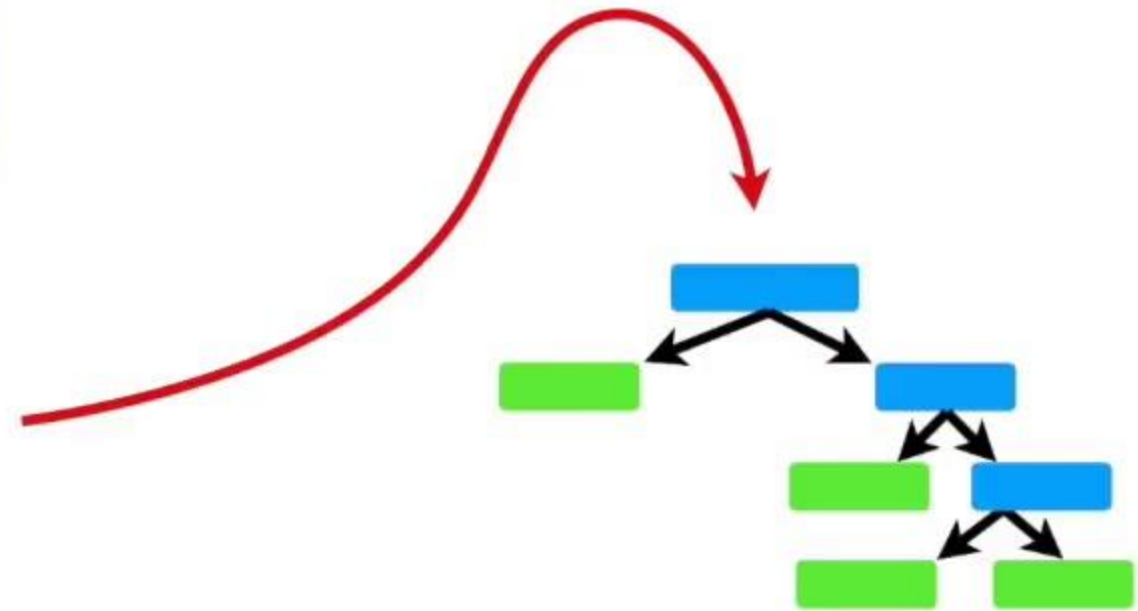
Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No



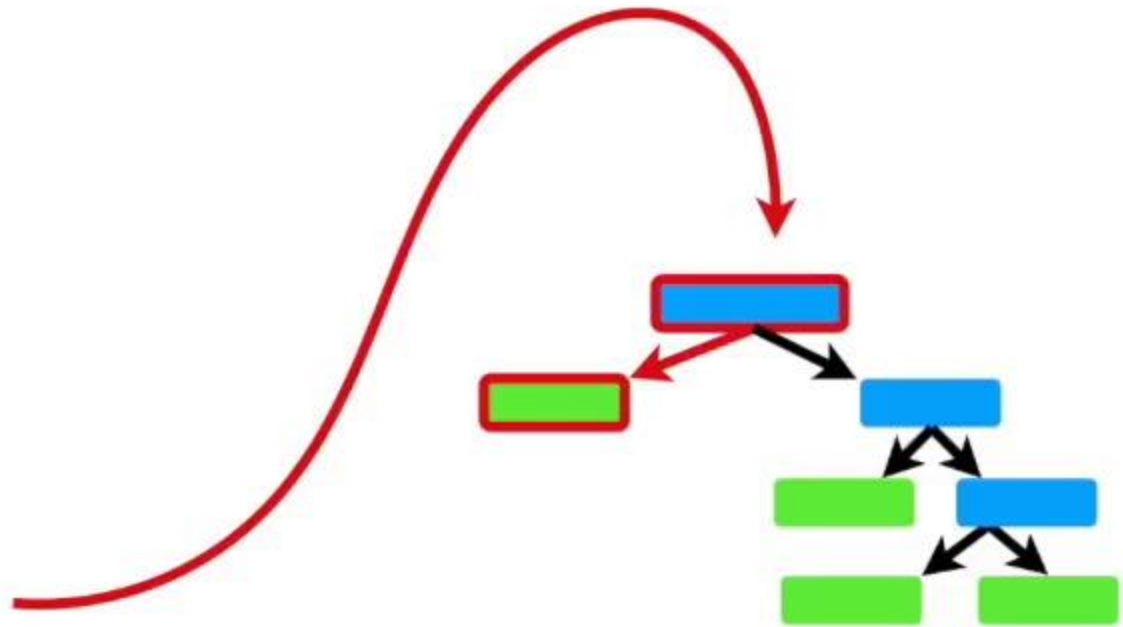
Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No



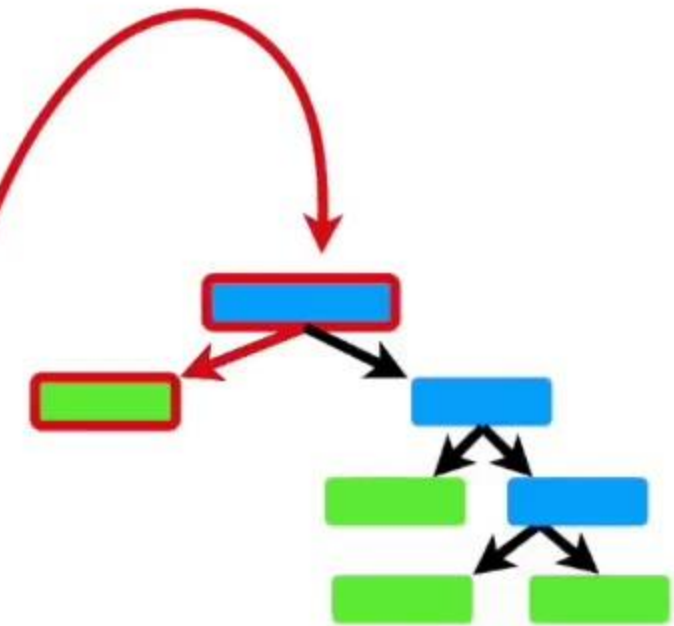
Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No



Filled-in Missing Values

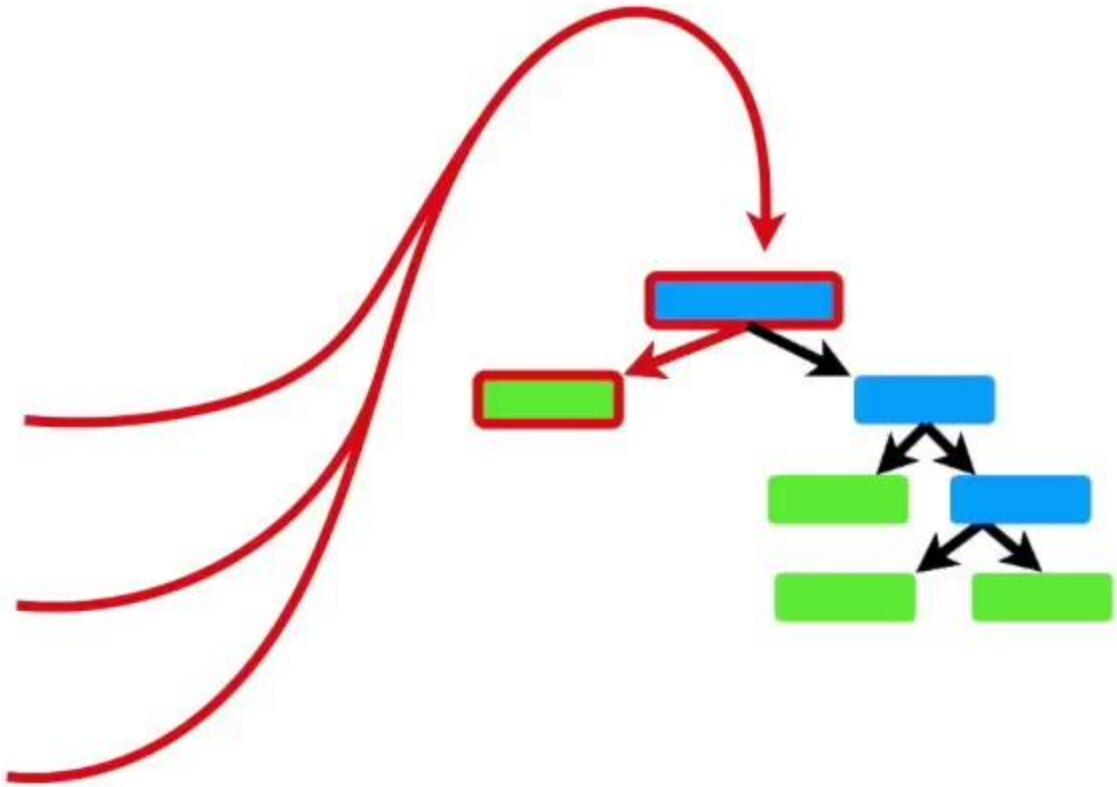
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No



Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

NOTE: Samples 2, 3 and 4 all ended up in the same leaf node.



Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

This is what the proximity matrix looked like after running the data down the first tree…

	1	2	3	4
1				
2				
3				1
4			1	

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

…after the second tree, we add 1 to any pair of samples that ended up in the same leaf node.

	1	2	3	4
1				
2			1	1
3		1		2
4		1	2	

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

…after the second tree, we add 1 to any pair of samples that ended up in the same leaf node.

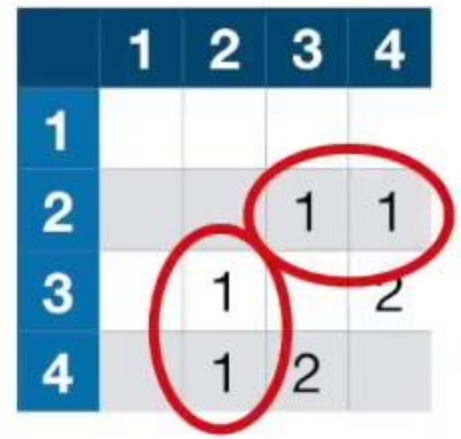
	1	2	3	4
1				
2			1	1
3		1		2
4		1	2	

Samples 3 and 4 ended up in the same node together again…

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

…after the second tree, we add 1 to any pair of samples that ended up in the same leaf node.

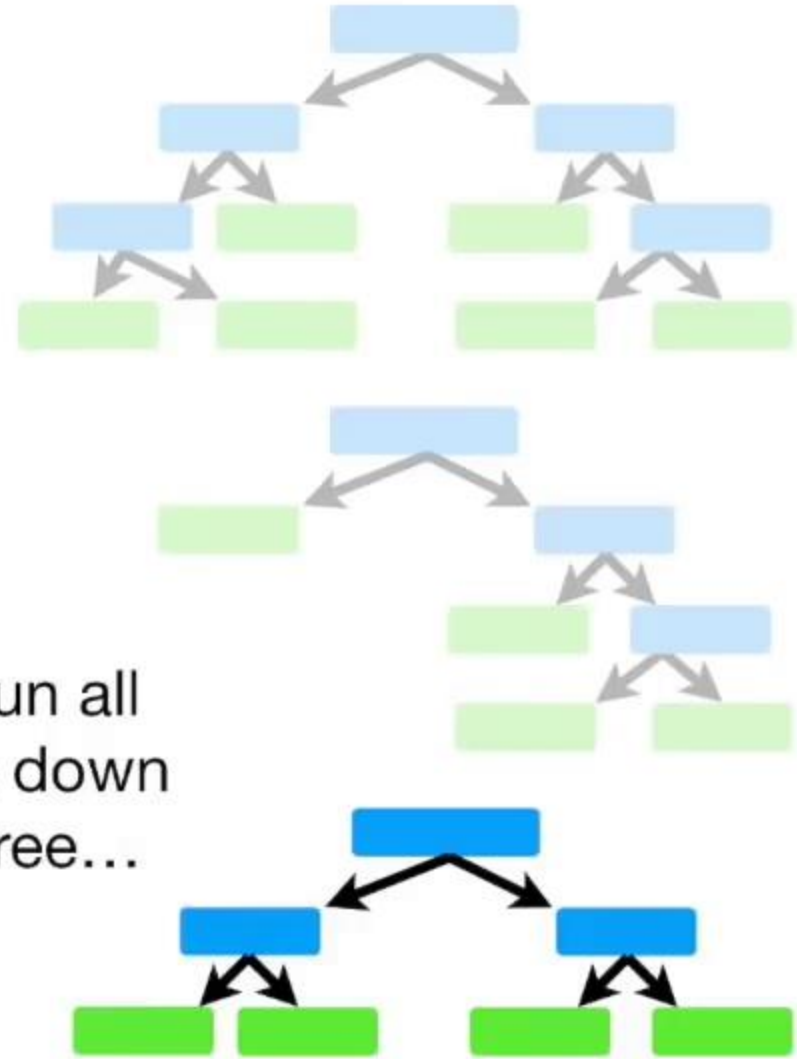


Sample 2 also ended up in that same node.

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

Now we run all of the data down the third tree…



Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

…and here's the updated proximity matrix.

	1	2	3	4
1				
2			1	1
3		1		3
4		1	3	

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

…and here's the updated proximity matrix.

	1	2	3	4
1				
2			1	1
3		1		3
4		1	3	

Only samples 3 and 4 ended up in the same leaf node.

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

Ultimately, we run the data down all the trees and the proximity matrix fills in.

	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		8
4	1	1	8	

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

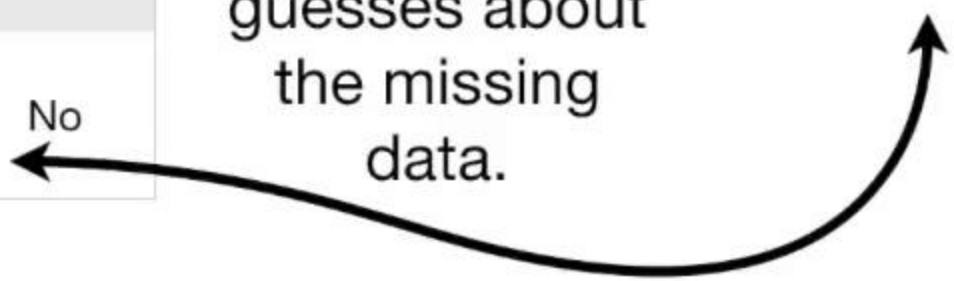
Then we divide each proximity value by the total number of trees. In this example, assume we had 10 trees.

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	???	???	No

Now we use the proximity values for sample 4 to make better guesses about the missing data.

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	



Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	???	???	No

For Blocked Arteries, we calculate the weighted frequency of “Yes” and “No, using proximity values as the weights.

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	[illegible]	[illegible]
Yes	Yes	No	210	No
Yes	Yes	???	???	No

Yes = 1/3

The frequency of “Yes”.

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	???	???	No

Yes = 1/3

No = 2/3

The frequency of “No”.

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**???**	???	No

The weighted frequency for **“Yes”** is…

Yes =

Yes = 1/3

No = 2/3

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	???	???	No

The weighted frequency for **"Yes"** is…

$$\text{Yes} = \frac{1}{3}$$

Yes = 1/3

No = 2/3

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**???**	???	No

The weighted frequency for **"Yes"** is…

$$\text{Yes} = \frac{1}{3} \times \text{The weight for "Yes"}$$

Yes = 1/3

No = 2/3

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**???**	???	No

The weighted frequency for **"Yes"** is…

$$\text{Yes} = \frac{1}{3} \times \text{The weight for "Yes"}$$

Yes = 1/3

No = 2/3

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

The weight for "Yes" =

$$\frac{\text{Proximity of "Yes"}}{\text{All Proximities}}$$

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**???**	???	No

The weighted frequency for **"Yes"** is…

Yes = $\frac{1}{3}$ x The weight for **"Yes"**

Yes = 1/3

No = 2/3

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

The weight for "Yes" = 0.1

The proximity value for Sample 2 (the only one with "Yes")

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**???**	???	No

The weighted frequency for **"Yes"** is…

$$\text{Yes} = \frac{1}{3} \times \text{The weight for "Yes"}$$

Yes = 1/3

No = 2/3

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

$$\text{The weight for "Yes"} = \frac{0.1}{0.1 + 0.1 + 0.8}$$

Divided by the sum of the proximities for Sample 4.

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**???**	???	No

The weighted frequency for **"Yes"** is…

$$\text{Yes} = \frac{1}{3} \times 0.1$$

Yes = 1/3

No = 2/3

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

The weight for "Yes" =

$$\frac{0.1}{0.1 + 0.1 + 0.8} = \frac{0.1}{1} = 0.1$$

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**???**	???	No

The weighted frequency for **"Yes"** is…

$$\text{Yes} = \frac{1}{3} \times 0.1 = 0.03$$

The weighted frequency for "**Yes**".

Yes = 1/3

No = 2/3

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**???**	???	No

The weighted frequency for **"No"** is…

Yes = 1/3

No = 2/3

$$\text{Yes} = \frac{1}{3} \times 0.1 = 0.03$$

$$\text{No} = \frac{2}{3} \times \text{The weight for "No"}$$

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**???**	???	No

The weighted frequency for **"No"** is…

$$\text{Yes} = \frac{1}{3} \times 0.1 = 0.03$$

$$\text{No} = \frac{2}{3} \times \text{The weight for "No"}$$

Yes = 1/3

No = 2/3

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Samples 1 and 3 both have "No"…

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**???**	???	No

The weighted frequency for **"No"** is…

$$\text{Yes} = \frac{1}{3} \times 0.1 = 0.03$$

$$\text{No} = \frac{2}{3} \times 0.9 =$$

Yes = 1/3

No = 2/3

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

The weight for "No" = $\frac{0.1 + 0.8}{0.1 + 0.1 + 0.8} = \frac{0.9}{1} =$ 0.9

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**NO**	???	No

The weighted frequency for **"No"** is…

Yes = $\frac{1}{3}$ x 0.1 = 0.03

No = $\frac{2}{3}$ x 0.9 = 0.6

Yes = 1/3

No = 2/3

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

"No" has a way higher weighted frequency, so we'll go with it.

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	???	No

For weight, we use the proximities to calculate a weighted average.

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Weighted average =

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	???	No

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Weighted average = 125

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	???	No

Sample 1's weight…

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Weighted average = 125 x Sample 1's weighted average weight…

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	???	No

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Weighted average = 125 x Sample 1's weighted average weight…

0.1

The proximity for Sample 1

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	???	No

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Weighted average = 125 x $\frac{\text{Sample 1's weighted average weight...}}{0.1 + 0.1 + 0.8}$

Divided by the sum of the proximities.

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	???	No

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Weighted average = 125 x 0.1

$$\frac{0.1}{0.1 + 0.1 + 0.8} = \frac{0.1}{1} = 0.1$$

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	???	No

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Weighted average = (125 x 0.1) + (180 x 0.1)

…the weighted value for 180…

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	???	No

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Weighted average = (125 x 0.1) + (180 x 0.1) + (210 x 0.8)

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	???	No

…the weighted value for 210…

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Weighted average = (125 x 0.1) + (180 x 0.1) + (210 x 0.8)

= 198.5

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	???	No

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Weighted average = (125 x 0.1) + (180 x 0.1) + (210 x 0.8)

= 198.5

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	**198.5**	No

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

The weighted average weight!

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**NO**	**198.5**	No

Now that we've revised our guesses a little bit, we do the whole thing over again…

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**NO**	**198.5**	No

Now that we've revised our guesses a little bit, we do the whole thing over again…

We build a random forest, run the data through the trees, recalculate the proximities and recalculate the missing values.

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**NO**	**198.5**	No

Now that we've revised our guesses a little bit, we do the whole thing over again…

We build a random forest, run the data through the trees, recalculate the proximities and recalculate the missing values.

We do this 6 or 7 times until the missing values converge (i.e. no longer change each time we recalculate).

Let me show you something super cool
we can do with the proximity matrix!!!

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

This is the proximity matrix before we divided each value by 10, the number of trees in the pretend random forest.

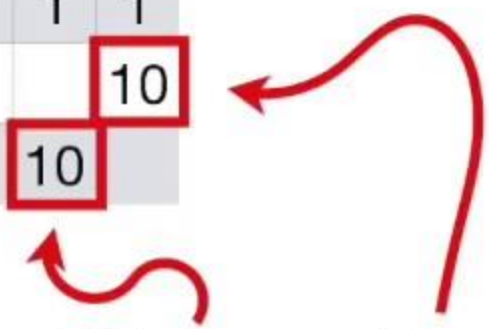
	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		8
4	1	1	8	

Just for the sake of easy math, imagine if Samples 3 and 4 ended up in the same leaf node in all 10 trees.

	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		10
4	1	1	10	

Just for the sake of easy math, imagine if Samples 3 and 4 ended up in the same leaf node in all 10 trees.

	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		10
4	1	1	10	



Now we have 10 here and here…

After dividing by 10 (the number of trees in the forest), we see that the largest number in the proximity matrix is 1.

	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		10
4	1	1	10	

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		1
4	0.1	0.1	1	

1 in the proximity matrix means the samples are as close as close can be.

	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		10
4	1	1	10	

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		1
4	0.1	0.1	1	

	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		10
4	1	1	10	

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		1
4	0.1	0.1	1	

That means…
1 - the proximity values
= ???

	1	2	3	4
1		0.8	0.9	0.9
2	0.8		0.9	0.9
3	0.9	0.9		0
4	0.9	0.9	0	

	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		10
4	1	1	10	

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		1
4	0.1	0.1	1	

That means…
1 - the proximity values
= distance

	1	2	3	4
1		0.8	0.9	0.9
2	0.8		0.9	0.9
3	0.9	0.9		0
4	0.9	0.9	0	

That means…
1 - the proximity values
= distance

	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		10
4	1	1	10	

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		1
4	0.1	0.1	1	

	1	2	3	4
1		0.8	0.9	0.9
2	0.8		0.9	0.9
3	0.9	0.9		0
4	0.9	0.9	0	

Close as can be

	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		10
4	1	1	10	

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		1
4	0.1	0.1	1	

That means…
1 - the proximity values
= distance

	1	2	3	4
1		0.8	0.9	0.9
2	0.8		0.9	0.9
3	0.9	0.9		0
4	0.9	0.9	0	

Close as can be = no distance between.

That means…
1 - the proximity values
= distance

	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		10
4	1	1	10	

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		1
4	0.1	0.1	1	

	1	2	3	4
1		0.8	0.9	0.9
2	0.8		0.9	0.9
3	0.9	0.9		0
4	0.9	0.9	0	

Not close

That means…
1 - the proximity values
= distance

	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		10
4	1	1	10	

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		1
4	0.1	0.1	1	

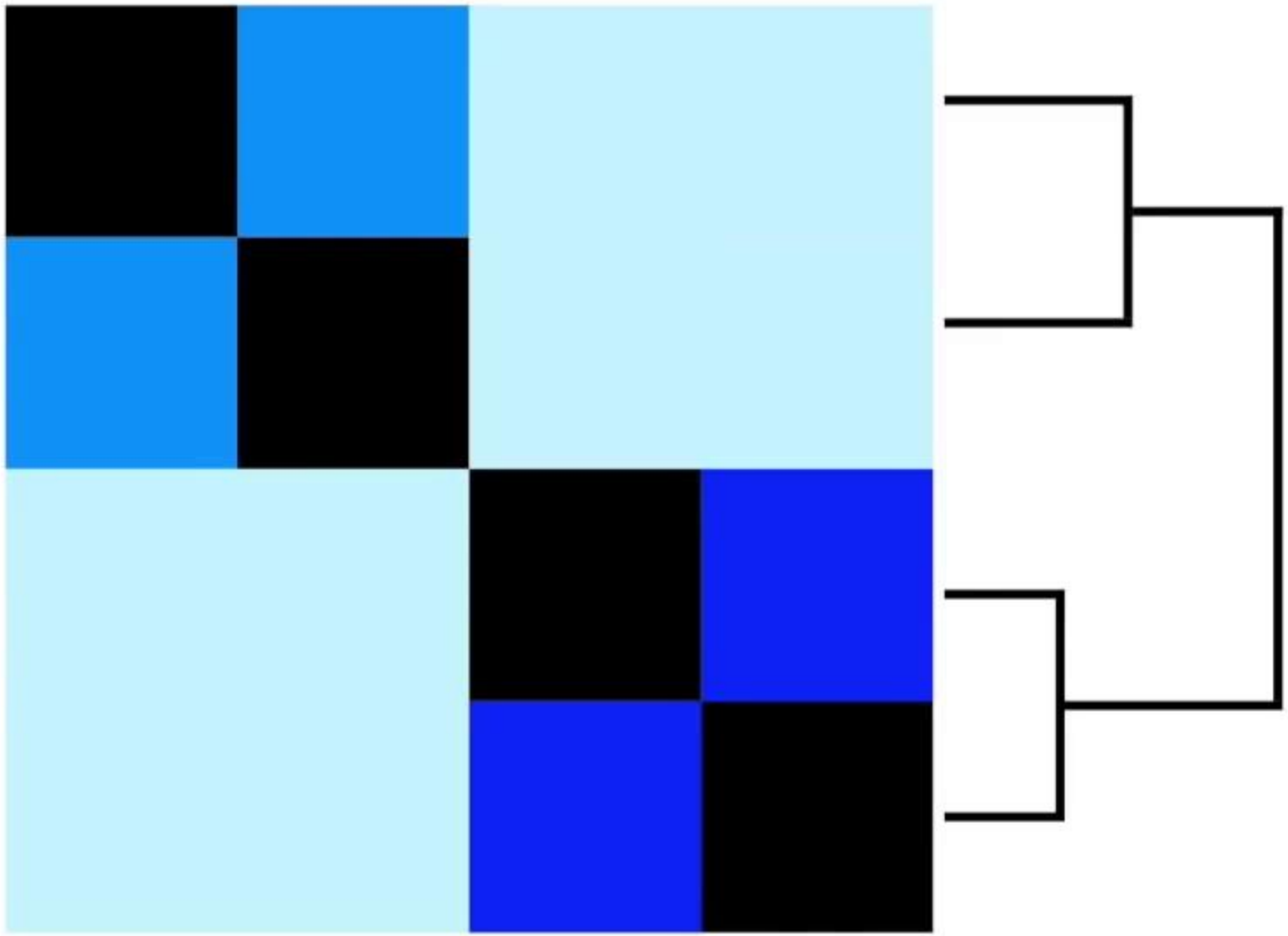
	1	2	3	4
1		0.8	0.9	0.9
2	0.8		0.9	0.9
3	0.9	0.9		0
4	0.9	0.9	0	

Not close = far away

This is a distance matrix…

	1	2	3	4
1		0.8	0.9	0.9
2	0.8		0.9	0.9
3	0.9	0.9		0
4	0.9	0.9	0	

Sample 1 Sample 2 Sample 3 Sample 4

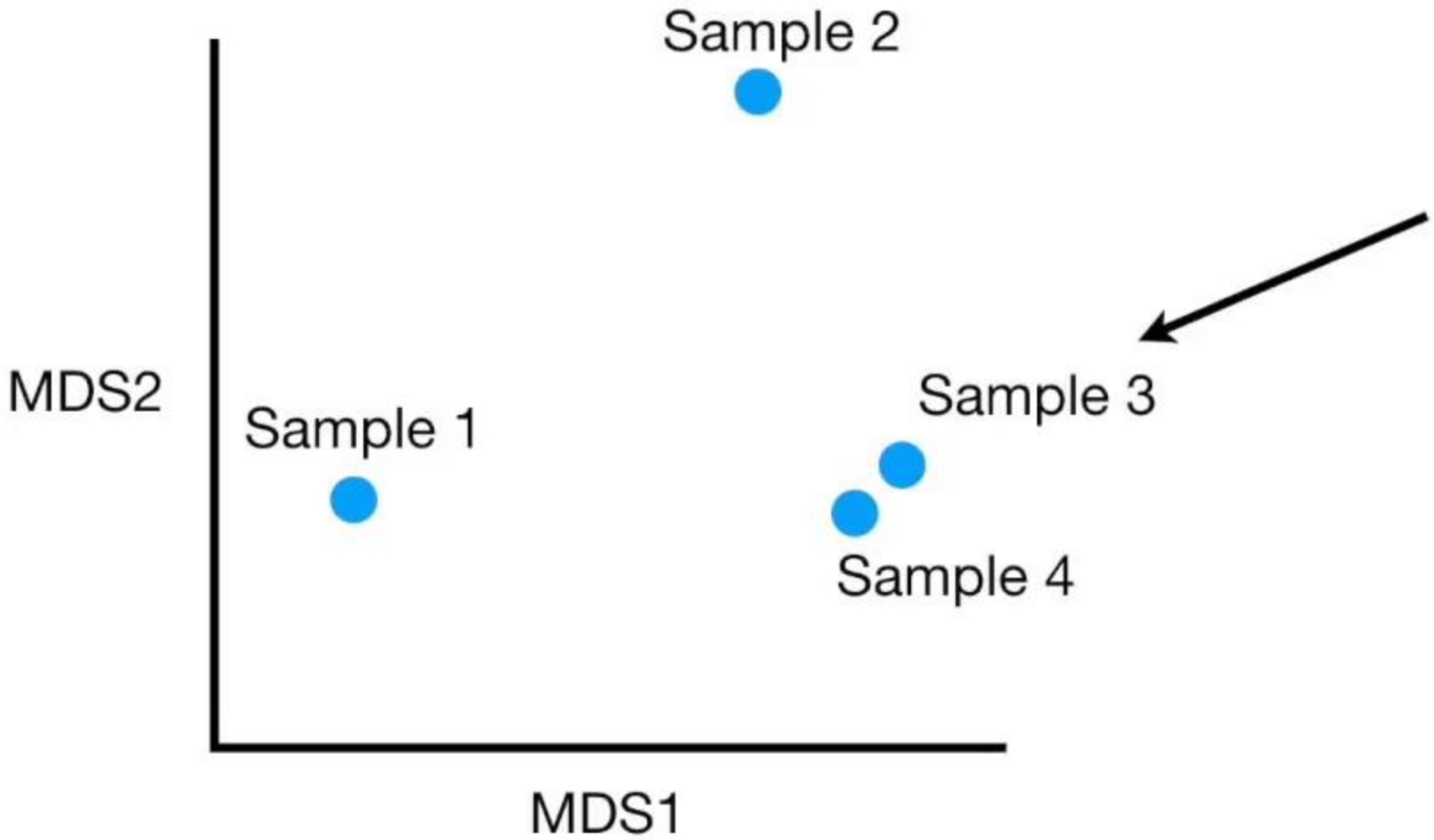


This is a distance matrix…

…and that means we can draw a heatmap with it!!!

	1	2	3	4
1		0.8	0.9	0.9
2	0.8		0.9	0.9
3	0.9	0.9		0
4	0.9	0.9	0	

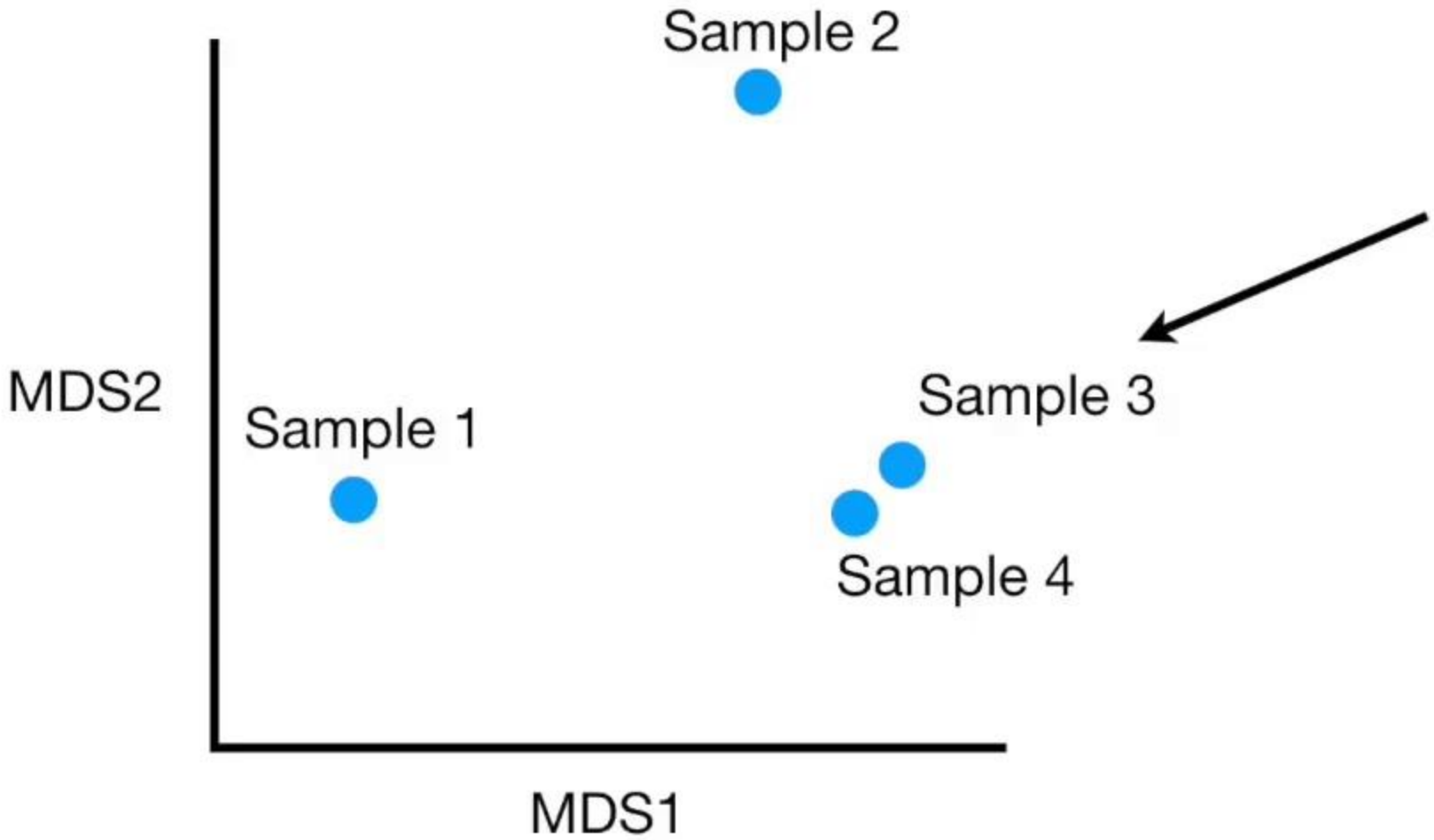
This is a distance matrix…



…and we can also draw an MDS plot with it!

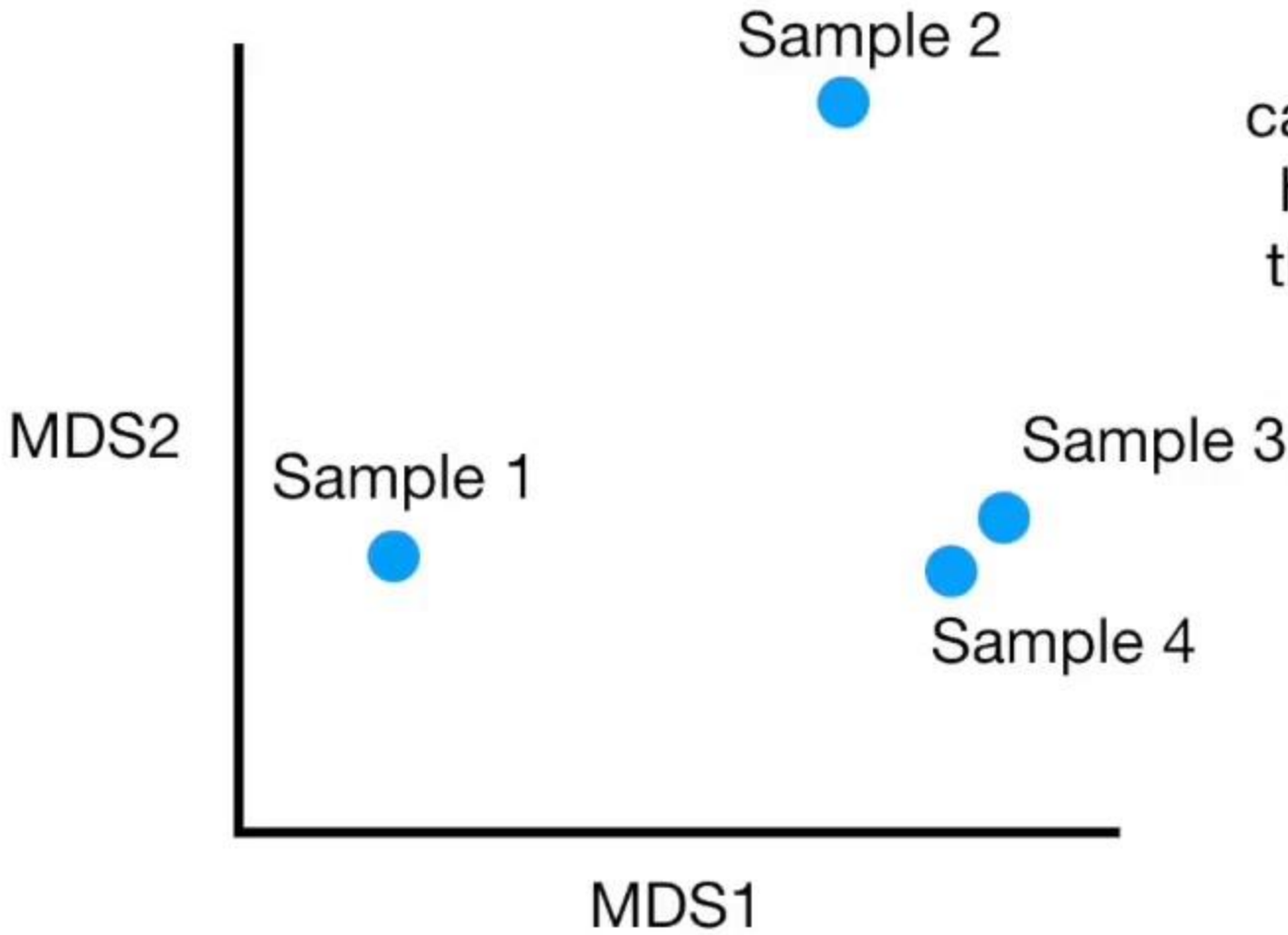
	1	2	3	4
1		0.8	0.9	0.9
2	0.8		0.9	0.9
3	0.9	0.9		0
4	0.9	0.9	0	

This is a distance matrix…



…and we can also draw an MDS plot with it!

	1	2	3	4
1		0.8	0.9	0.9
2	0.8		0.9	0.9
3	0.9	0.9		0
4	0.9	0.9	0	



This is super cool because it means that no matter what the data are (ranks, multiple choice, numeric, etc)… if we can use it to make a tree, we can draw a heatmap or an MDS plot to show how the samples are related to each other!!!

	1	2	3	4
1		0.8	0.9	0.9
2	0.8		0.9	0.9
3	0.9	0.9		0
4	0.9	0.9	0	

Random Forests consider 2 types of missing data…

1) Missing data in the original dataset used to create the random forest.

2) Missing data in a new sample that you want to categorize.

At long last, we'll talk about the second method!

New Sample

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	???	

Imagine we had already built a Random Forest with existing data and wanted to classify this new patient.

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	**???**	168	

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	???	168	

So we want to know if they have heart disease or not…

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	???	168	

…but we don't know if they have blocked arteries…

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	???	168	

…so we need to make a guess about Blocked Arteries so we can run the patient down all the trees in the forest.

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	???	168	YES

The first thing we do is create two copies of the data, one that has heart disease…

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	???	168	NO

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	???	168	YES

…and one that doesn't have heart disease.

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	???	168	NO

Then we use the iterative method we just talked about to make a good guess about the missing values.

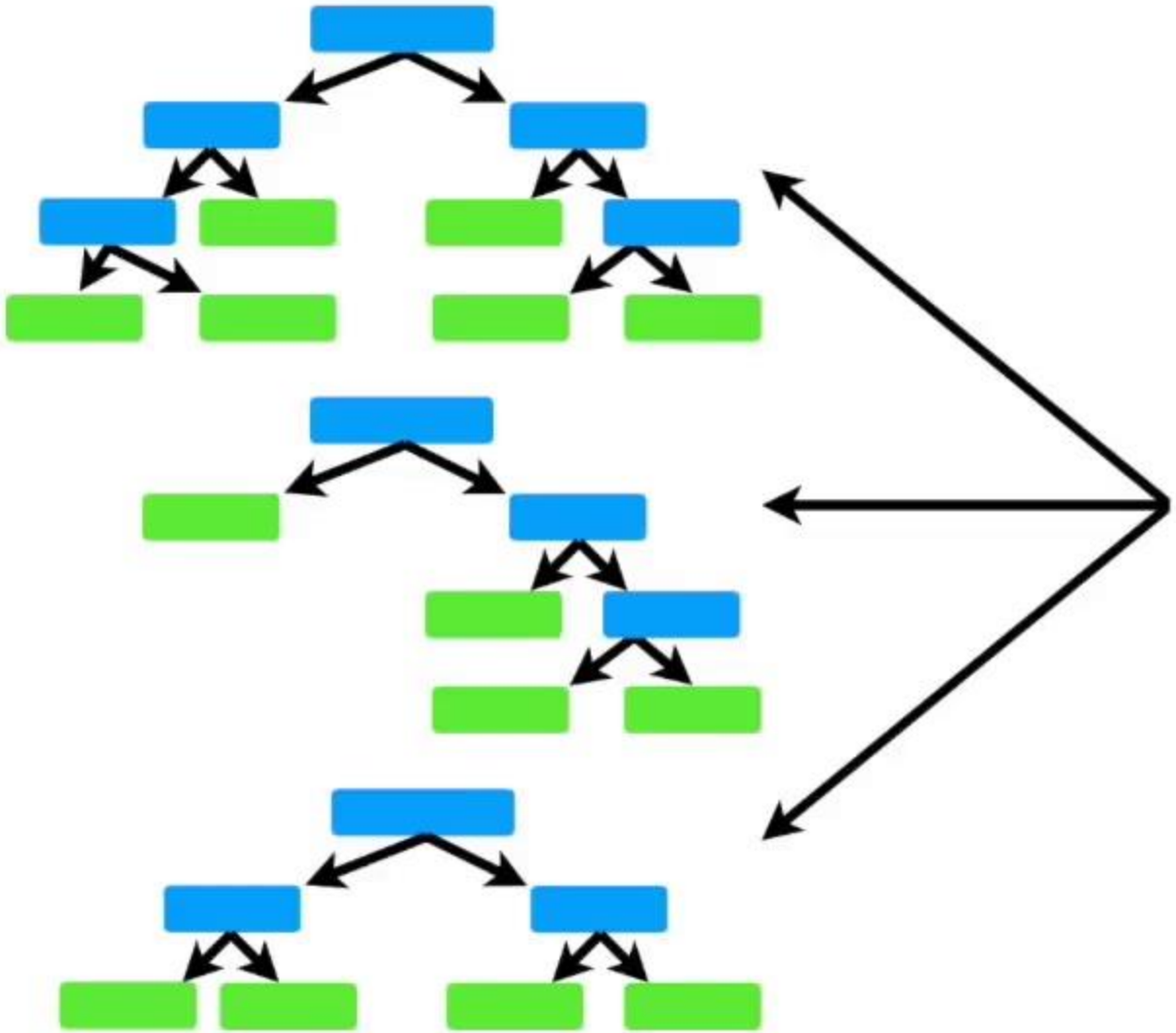
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	???	168	YES

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	???	168	NO

Then we use the iterative method we just talked about to make a good guess about the missing values.

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	**YES**	168	**YES**

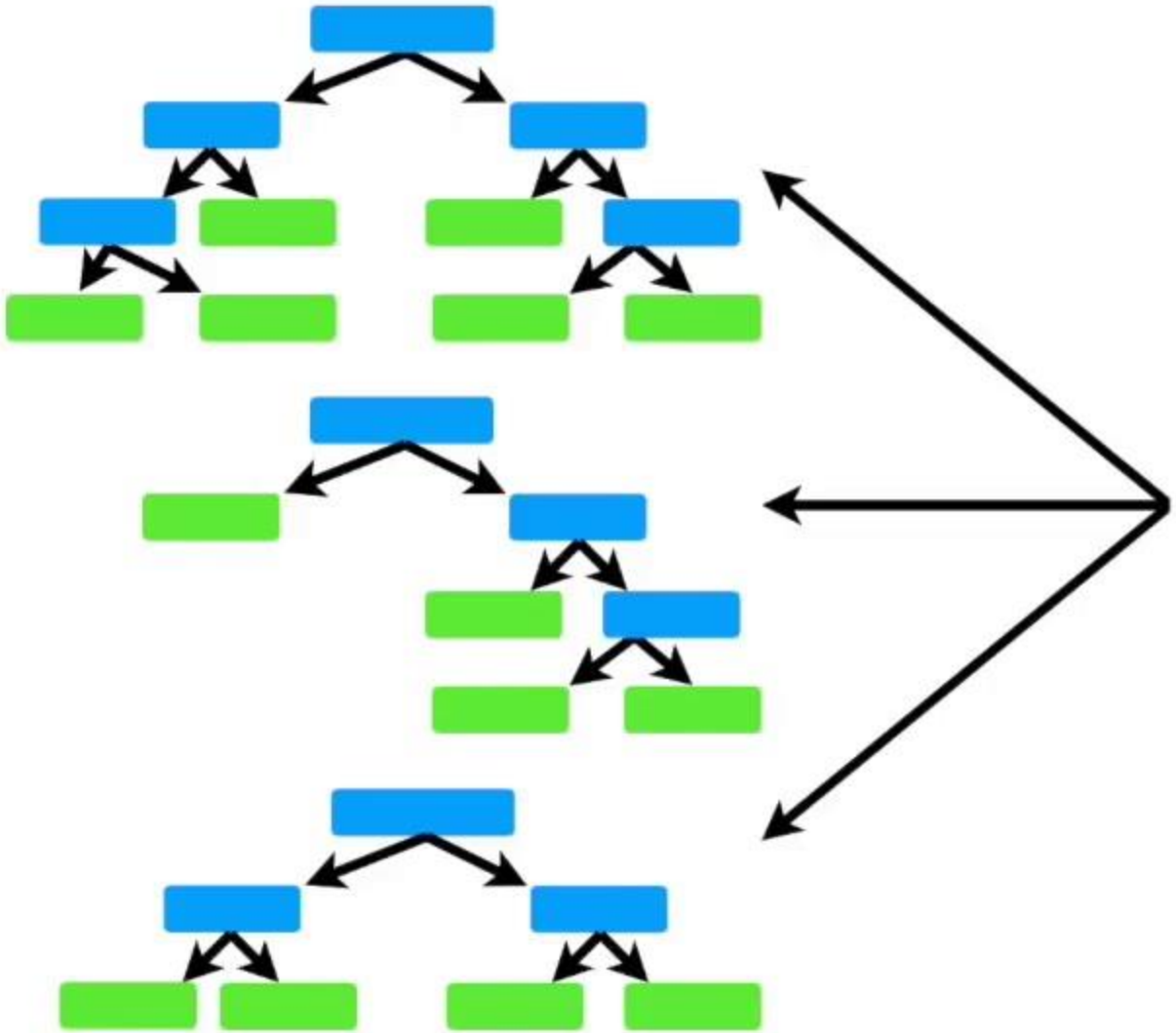
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	**NO**	168	**NO**



Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	YES	168	YES

Then we run the two samples down the trees in the forest…

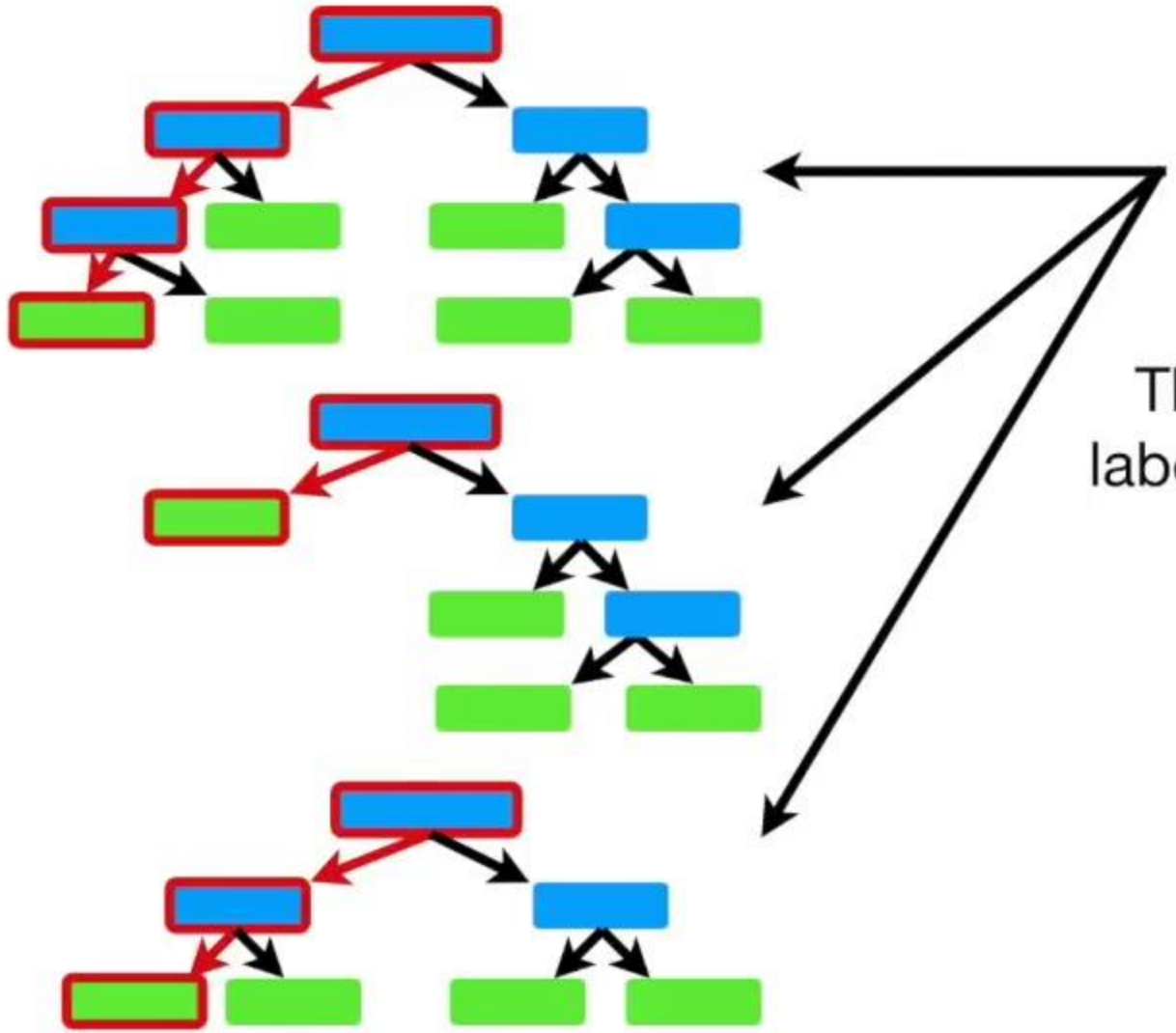
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	NO	168	NO



Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	**YES**	168	**YES**

…and we see which of the two is correctly labeled by the random forest the most times.

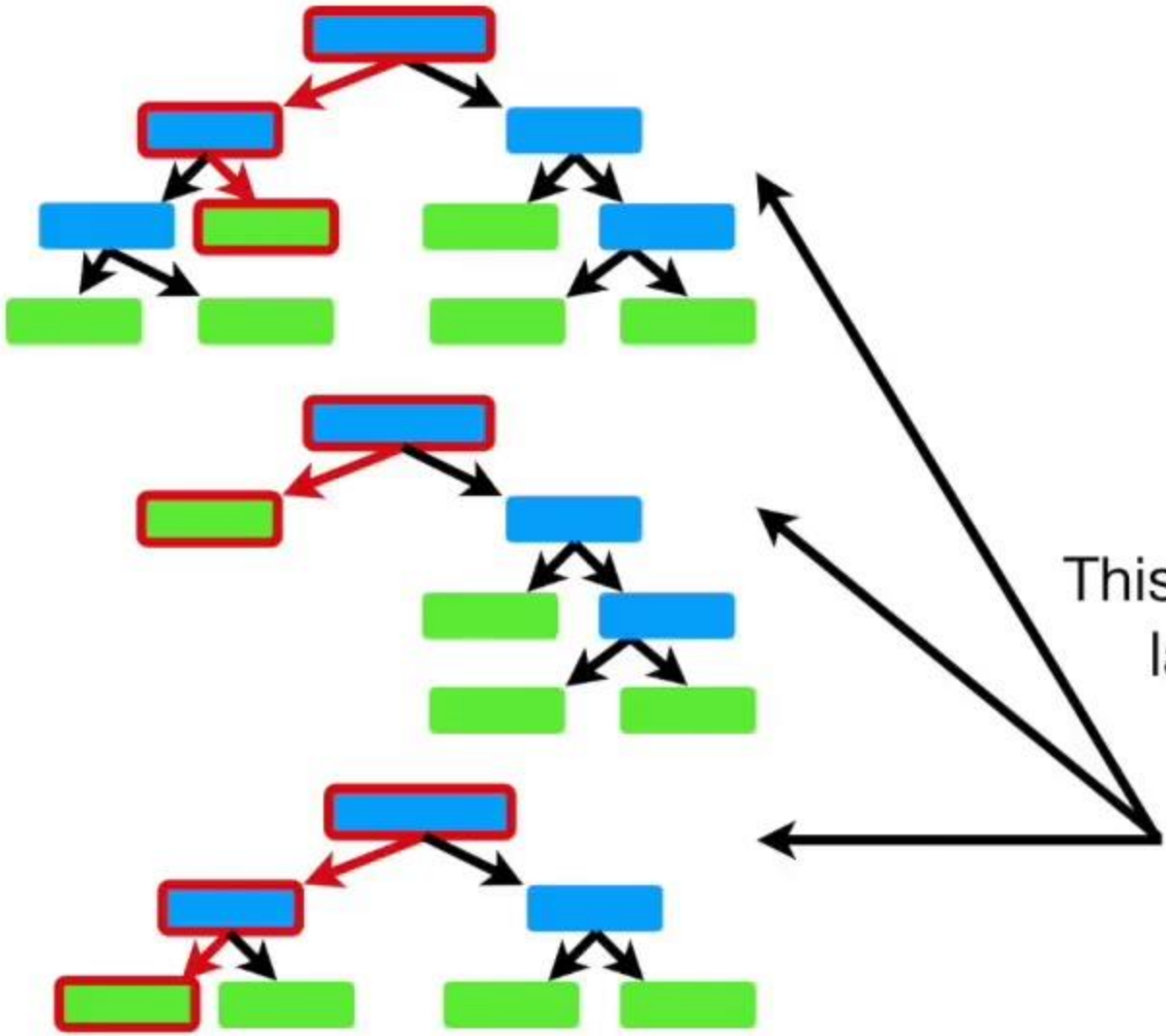
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	**NO**	168	**NO**



Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	**YES**	168	**YES**

This option was correctly labeled "**Yes**" in all 3 trees…

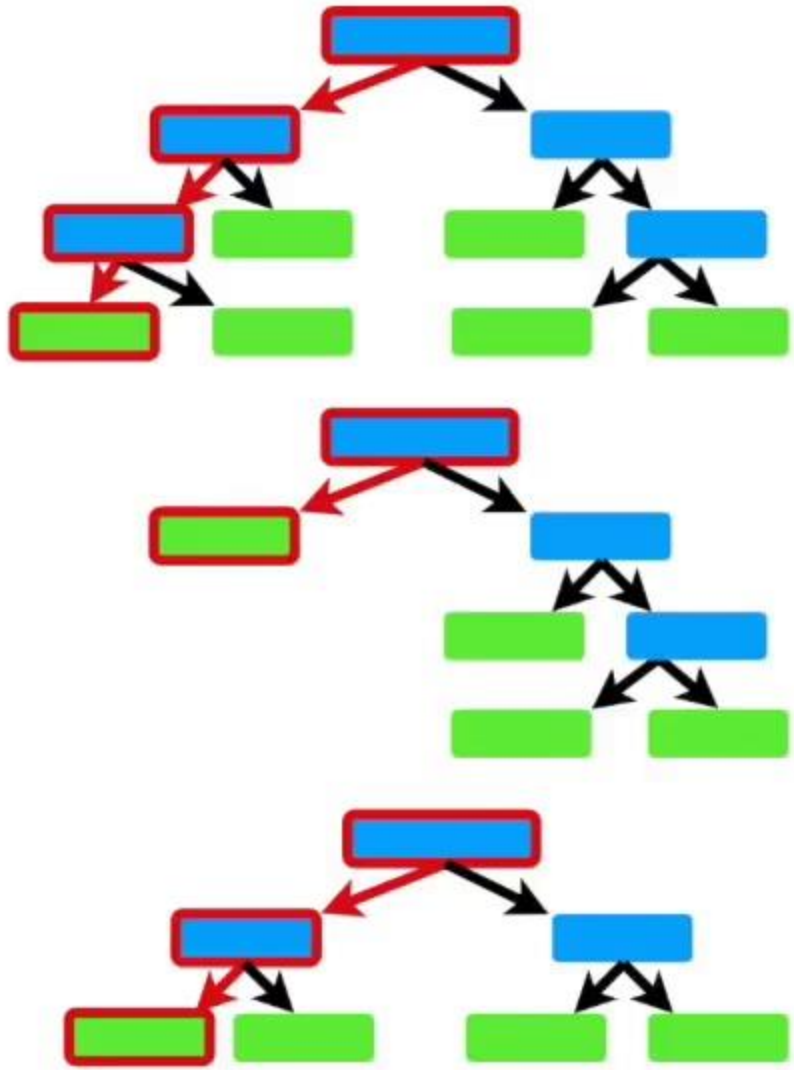
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	**NO**	168	**NO**



Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	**YES**	168	**YES**

This option was only correctly labeled "**No**" in 1 tree…

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	**NO**	168	**NO**



Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	**YES**	168	**YES**

This option wins because it was correctly labeled more than the other option.

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	**NO**	168	**NO**

T-SNE

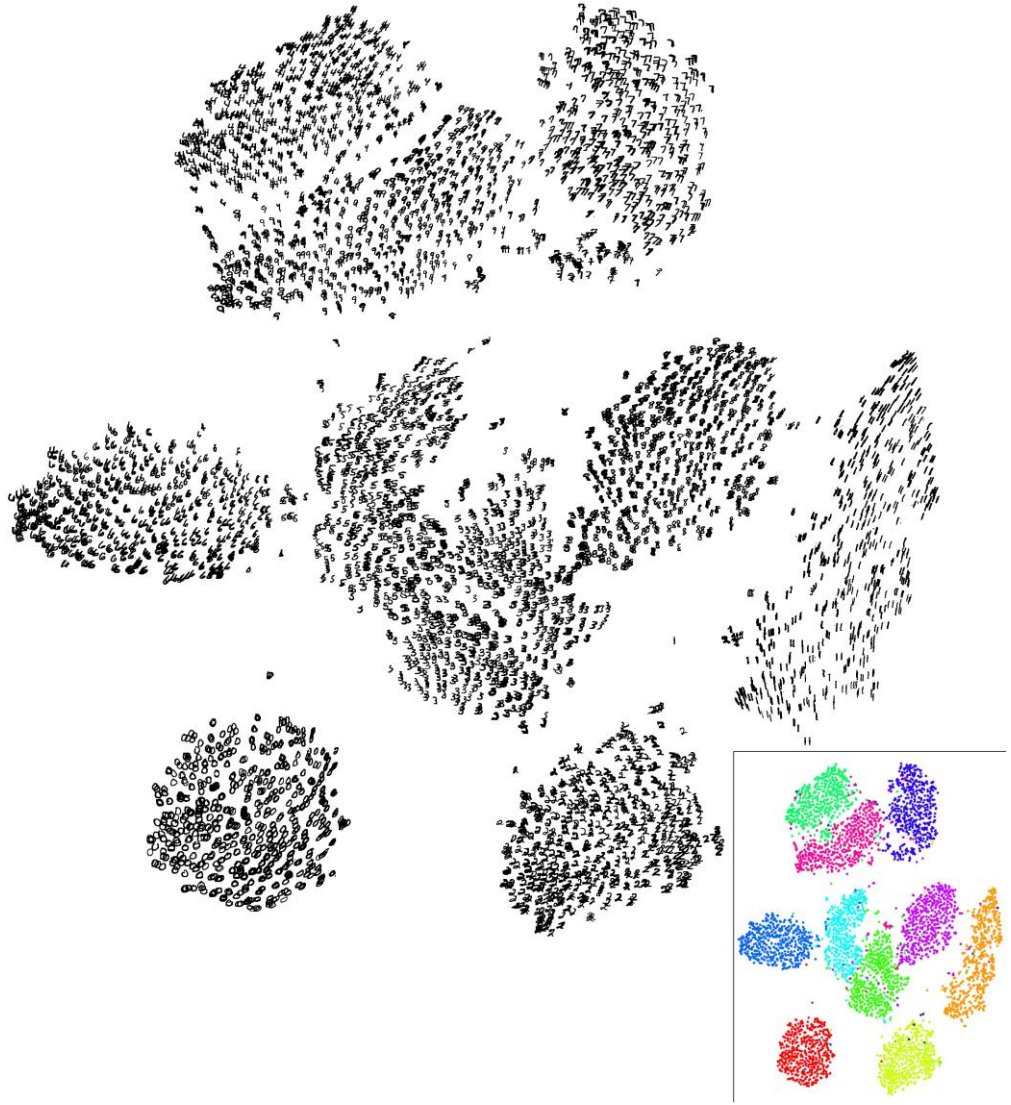
T-SNE

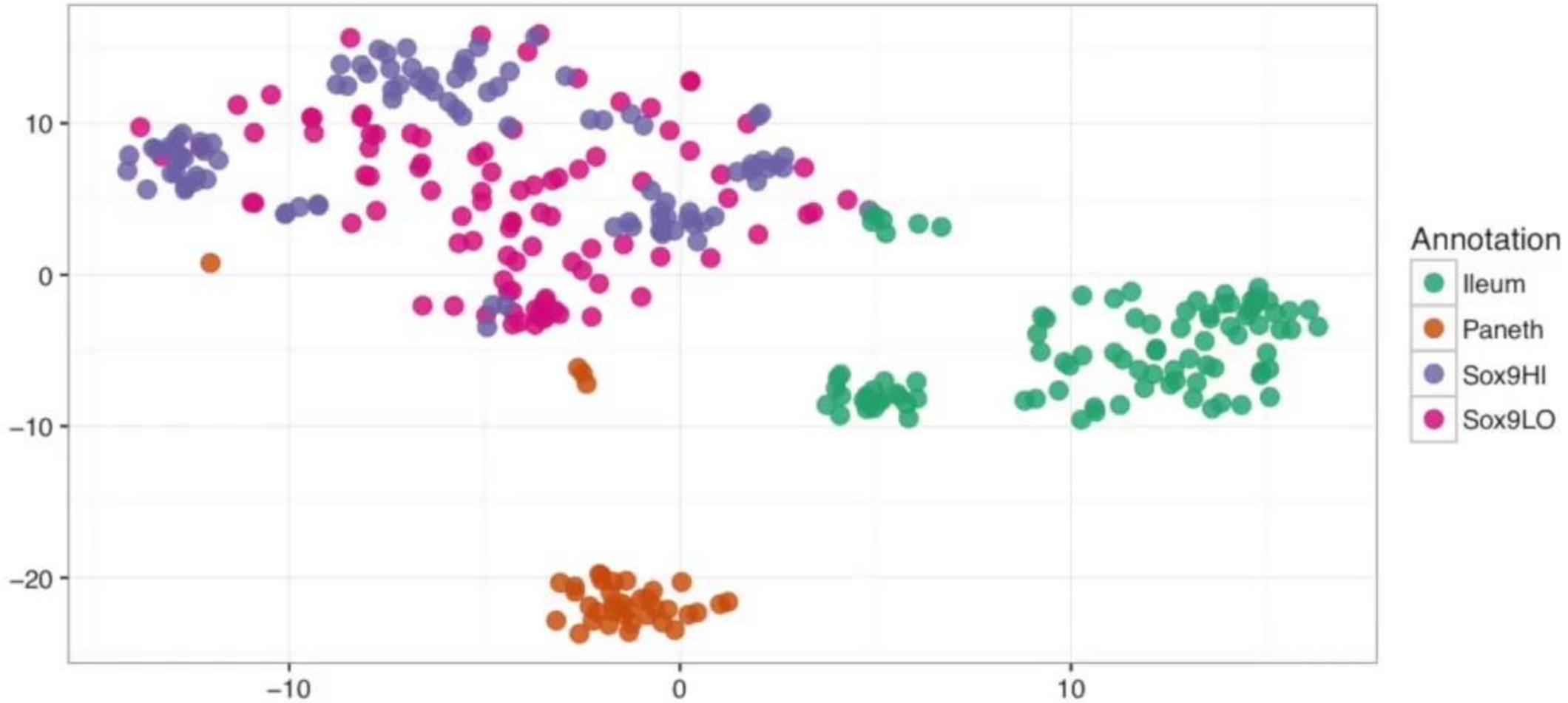
- T-SNE is a dimensionality reduction method
- PCA tries to find global structure
  - Low dimensional subspace
  - Can lead to local inconsistencies
    - Far away points can become nearest neighbors
- T-SNE tries to preserve local structure
  - Low dimensional neighborhood should be the same as original neighborhood
- Unlike PCA almost only used for visualization

Basic idea

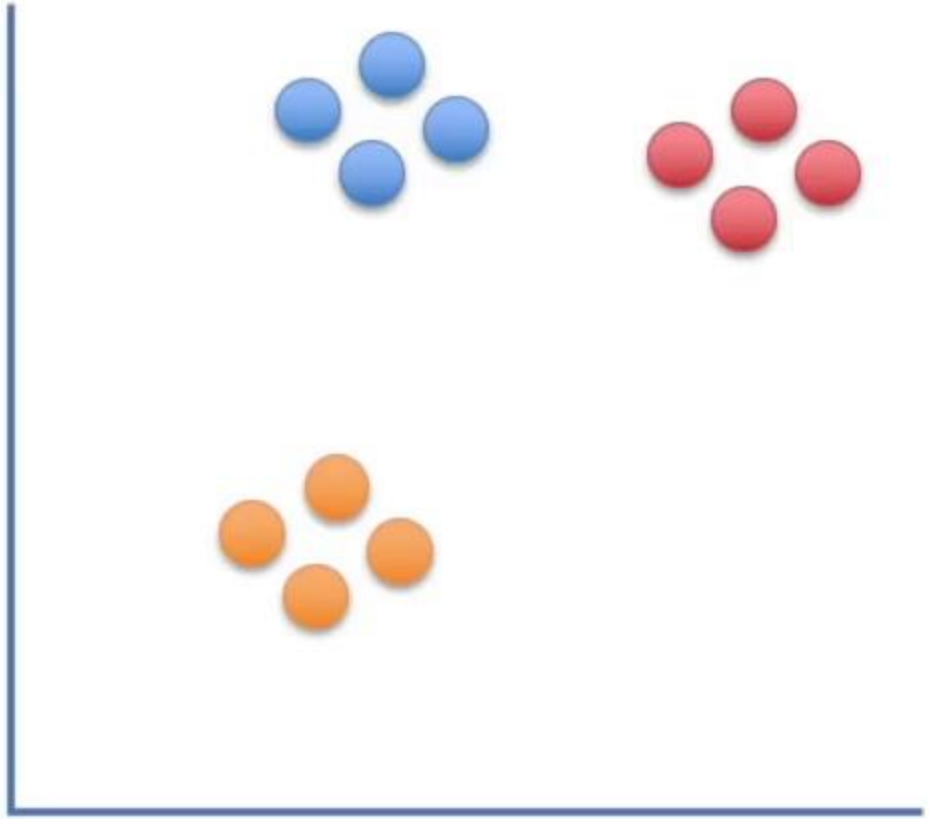
- Encode high dimensional neighborhood information as a distribution
- Random walk between data points
- Find low dimensional points such that their neighborhood distribution is similar
- The technique can be implemented via Barnes-Hut approximations, allowing it to be applied on large real-world datasets

MNIST dataset visualization using t-SNE



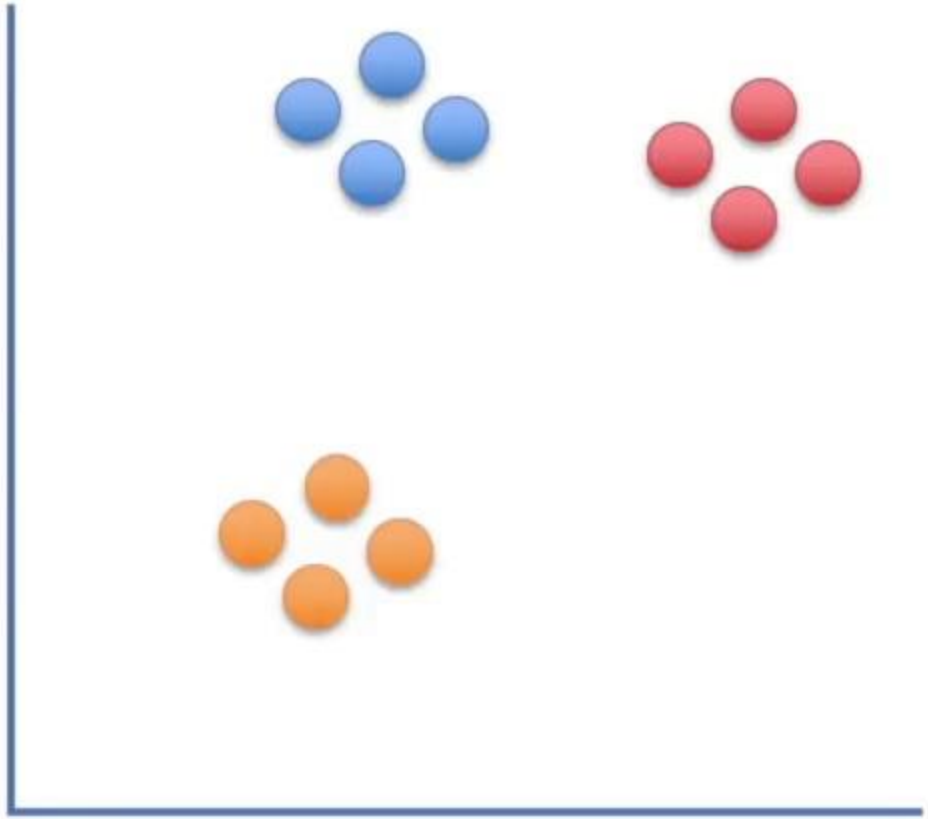

10
0
−10
−20
−10
0
10
Annotation
Ileum
Paneth
Sox9HI
Sox9LO

Here's a basic 2-D scatter plot.



Here's a basic 2-D scatter plot.

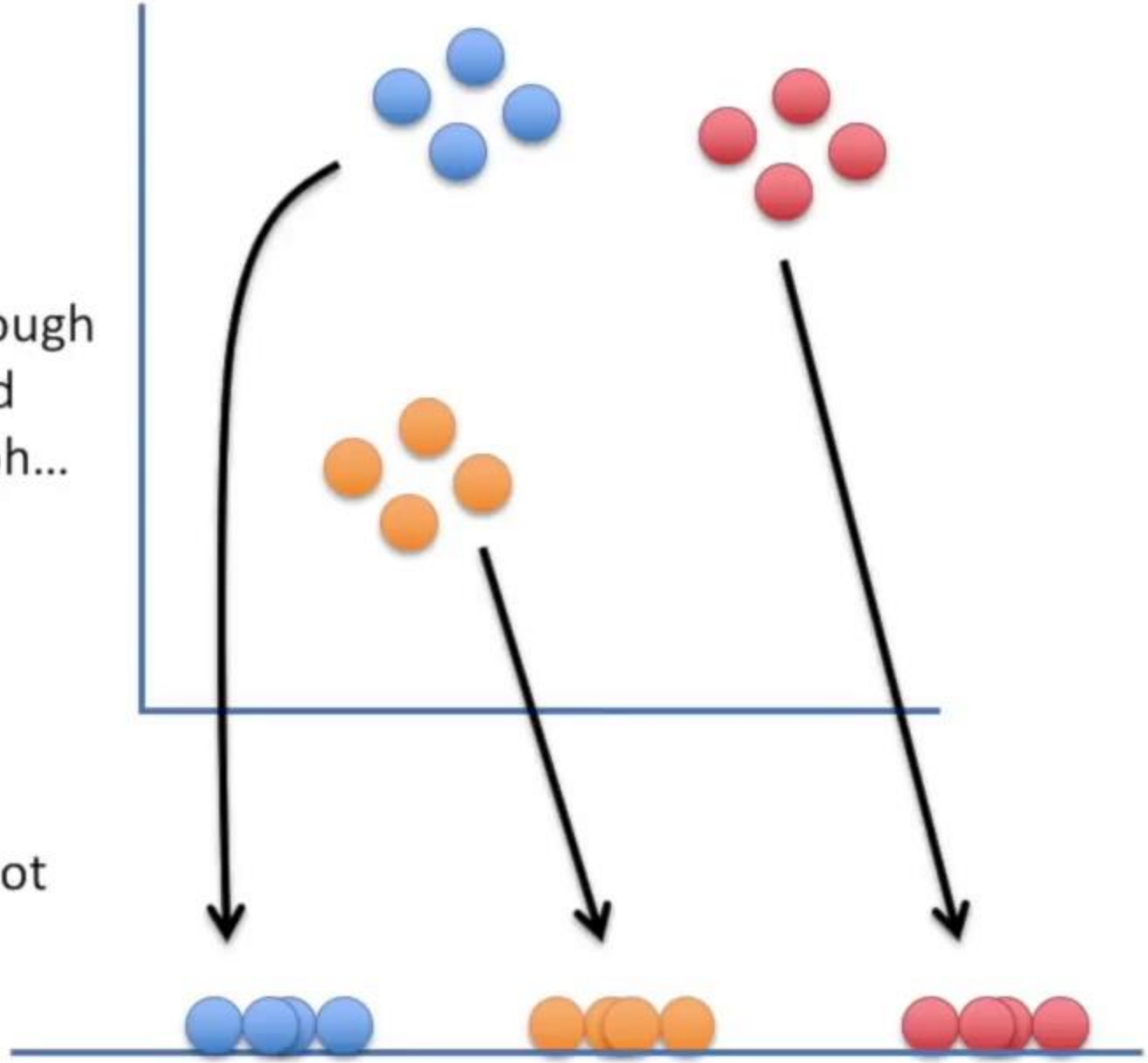
Let's do a walk through of how t-SNE would transform this graph...

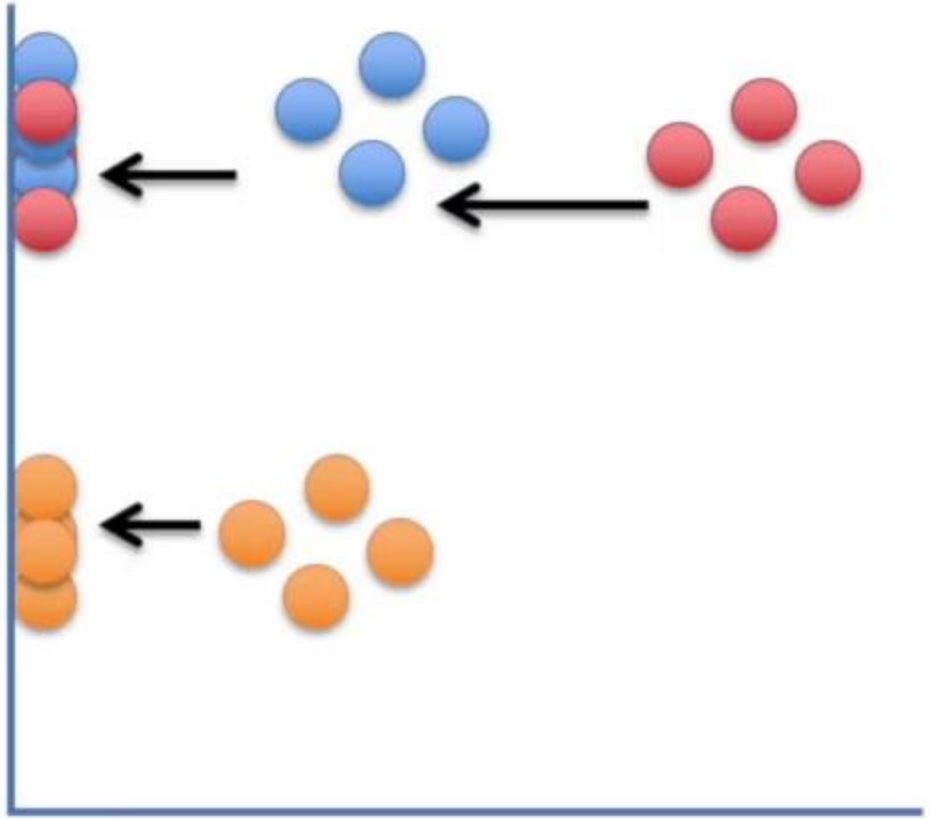


Here's a basic 2-D scatter plot.

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

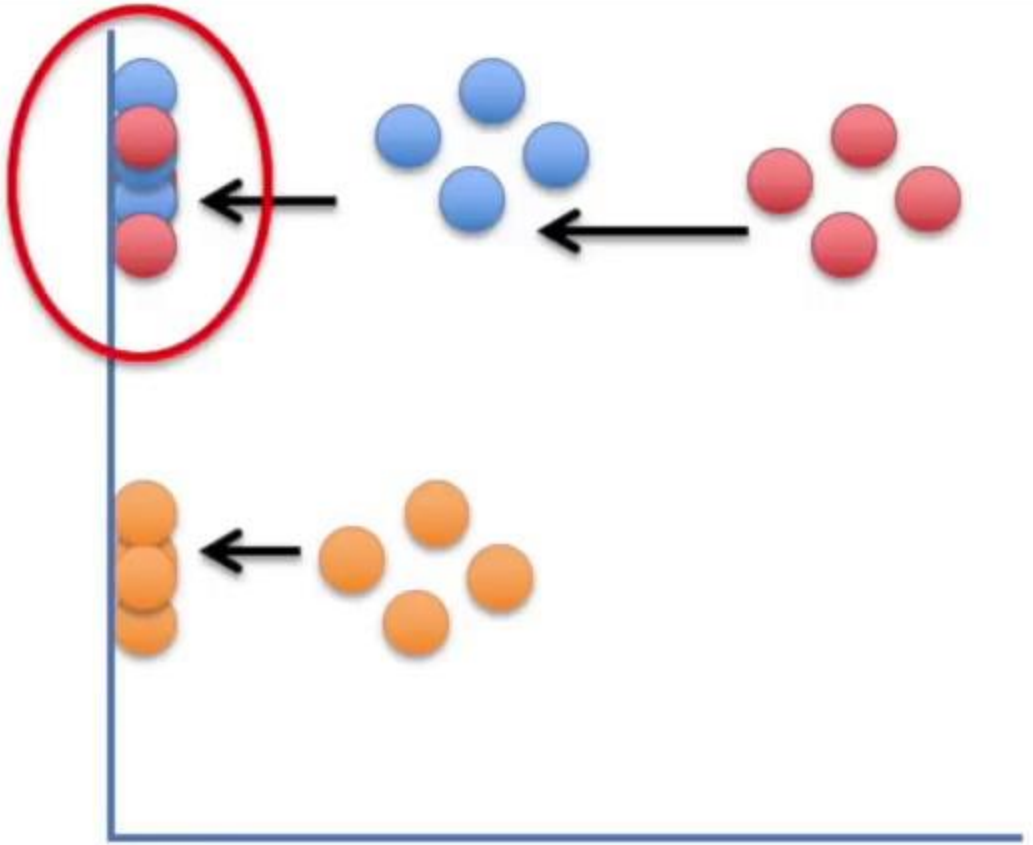
…into a flat, 1-D plot on a number line.

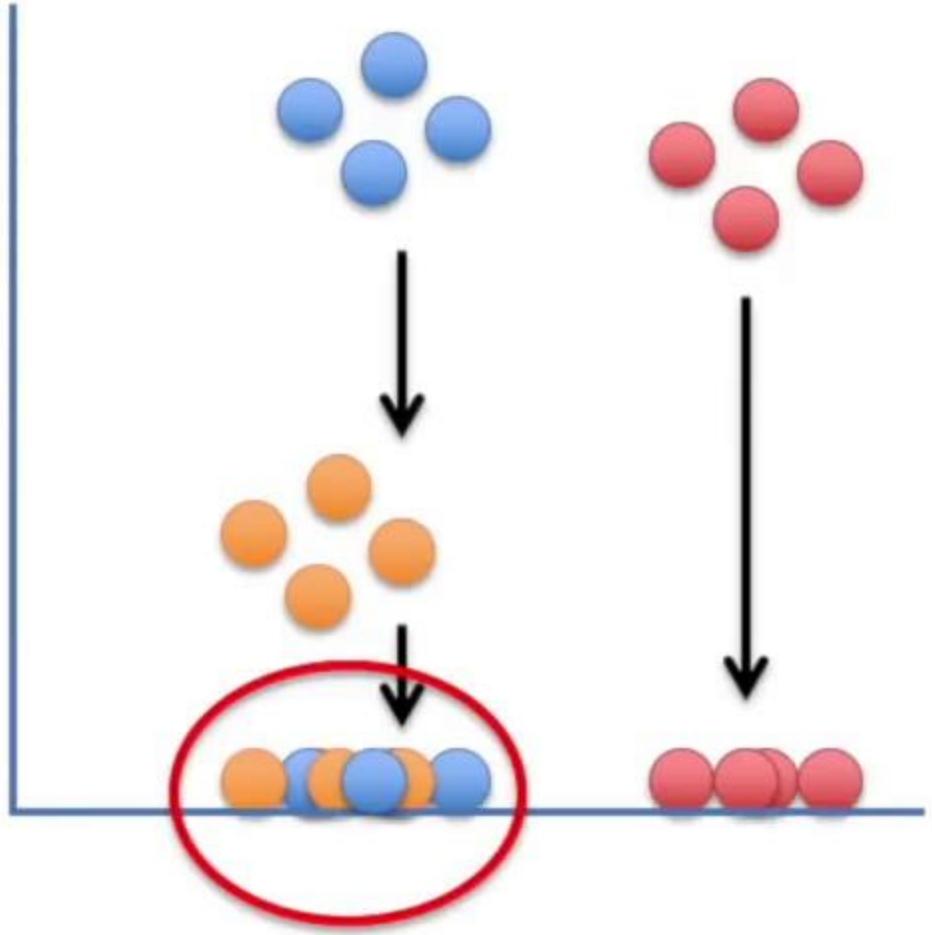




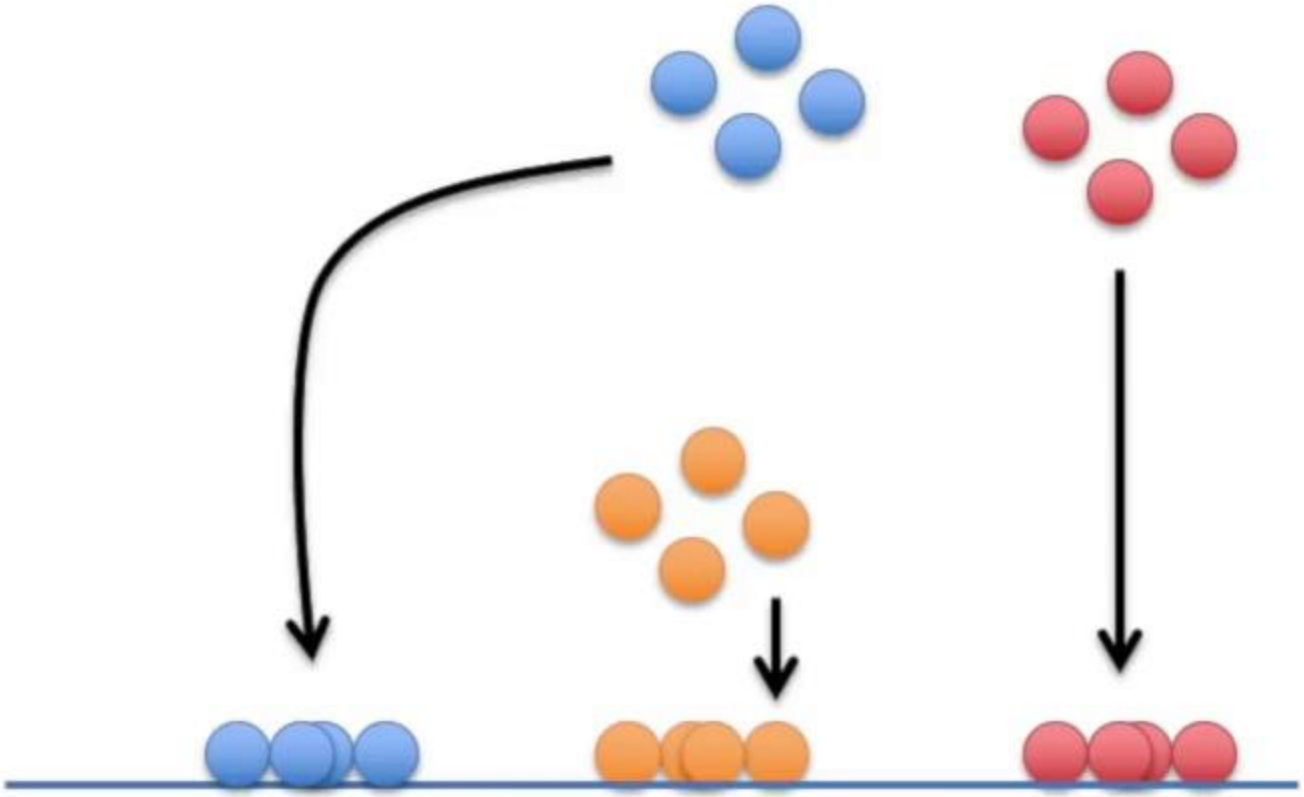
NOTE: If we just projected the data onto one of the axes, we'd just get a big mess that doesn't preserve the original clustering.

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

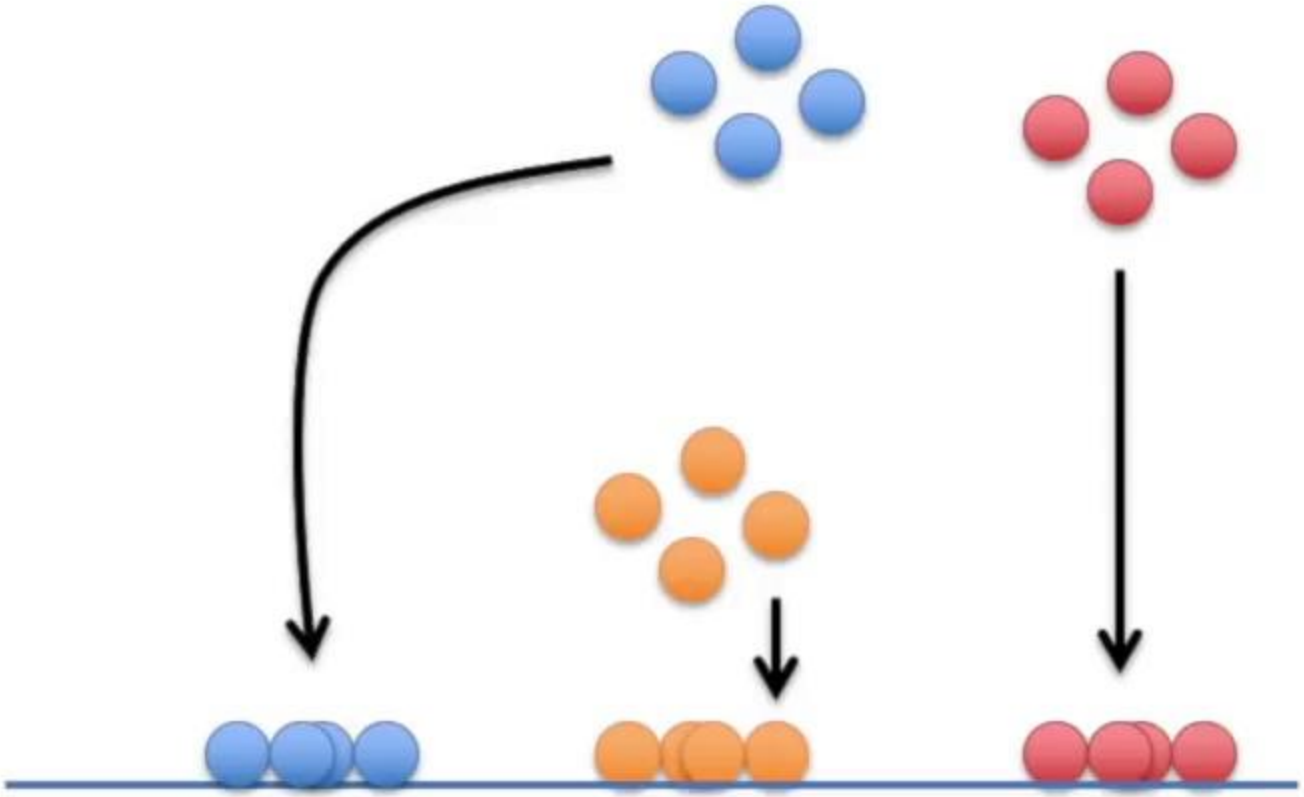




Same here...

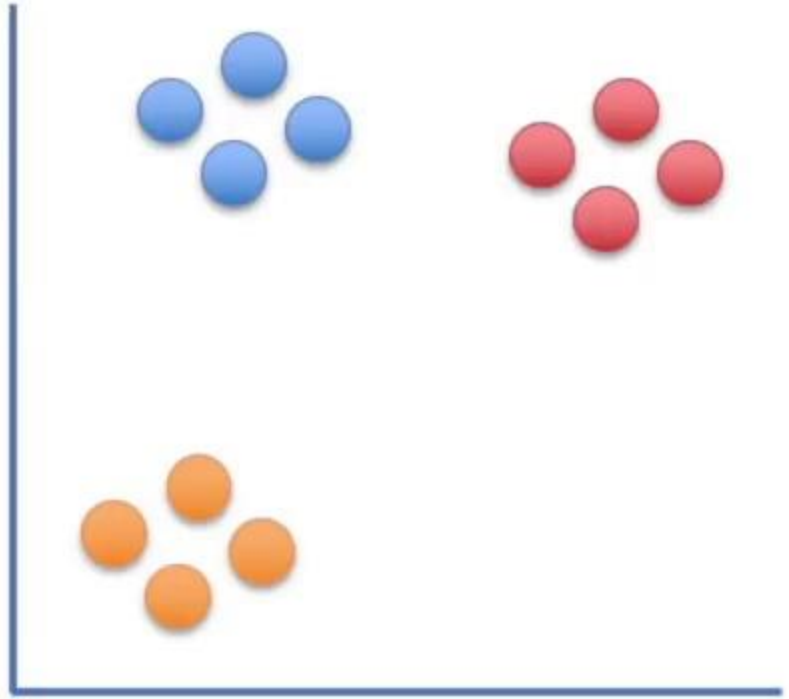


What t-SNE does is find a way to project data into a low dimensional space (in this case, the 1-D number line) so that the clustering in the high dimensional space (in this case, the 2-D scatter plot) is preserved.



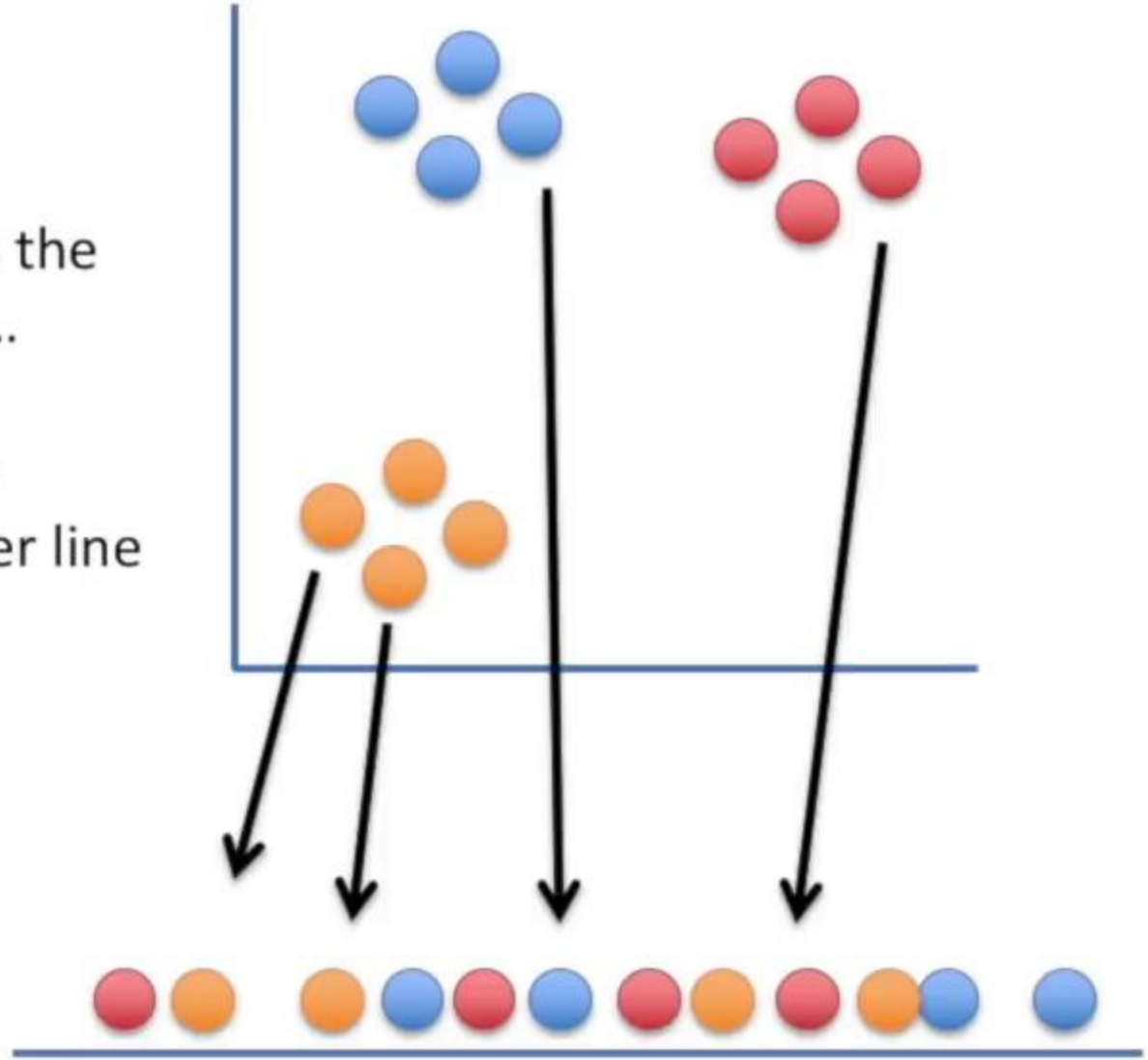
So let’s step through the basic ideas of how t-SNE does this.

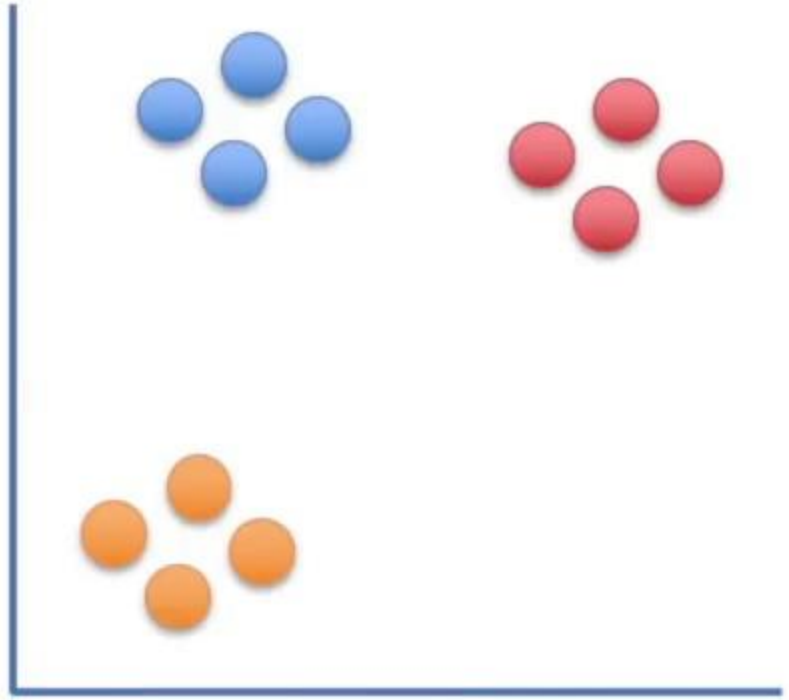
We'll start start with the original scatter plot…



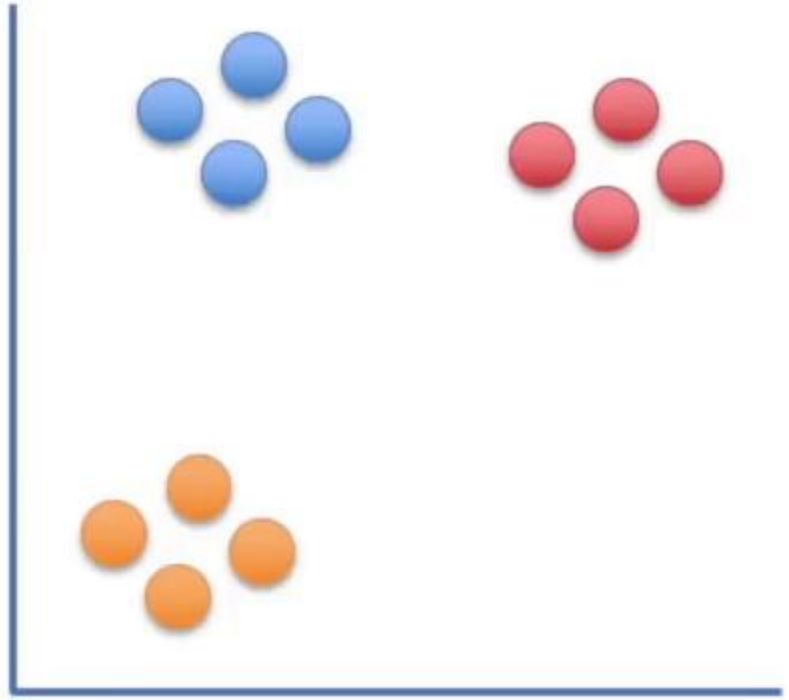
We'll start start with the original scatter plot…

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

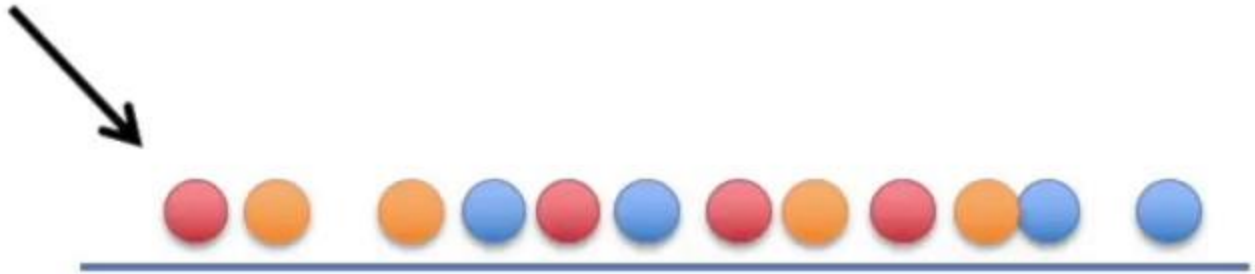


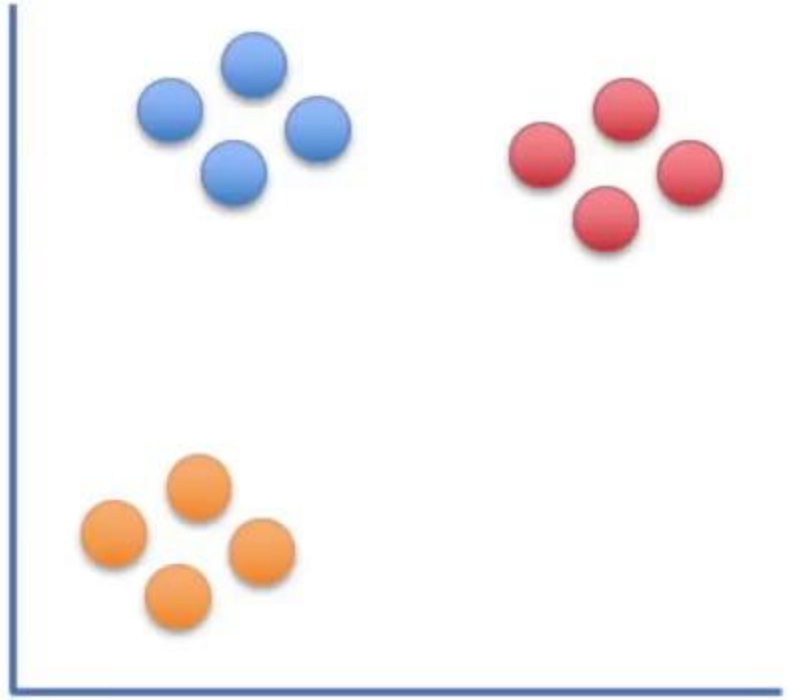


From here on out, t-SNE moves these points, a little bit at a time, until it has clustered them.



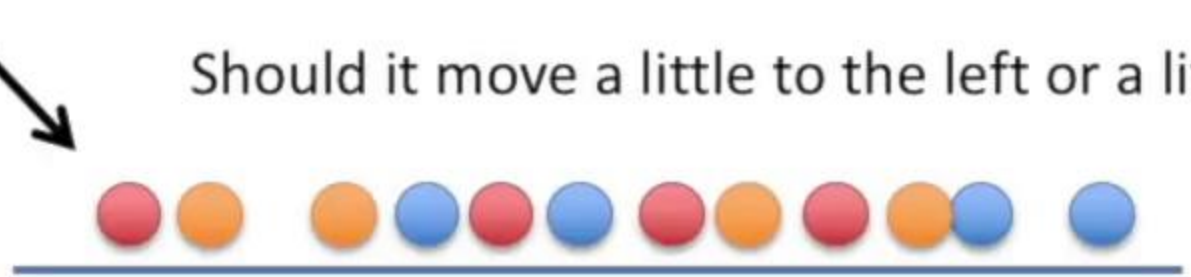
Let’s figure out where to move this first point…

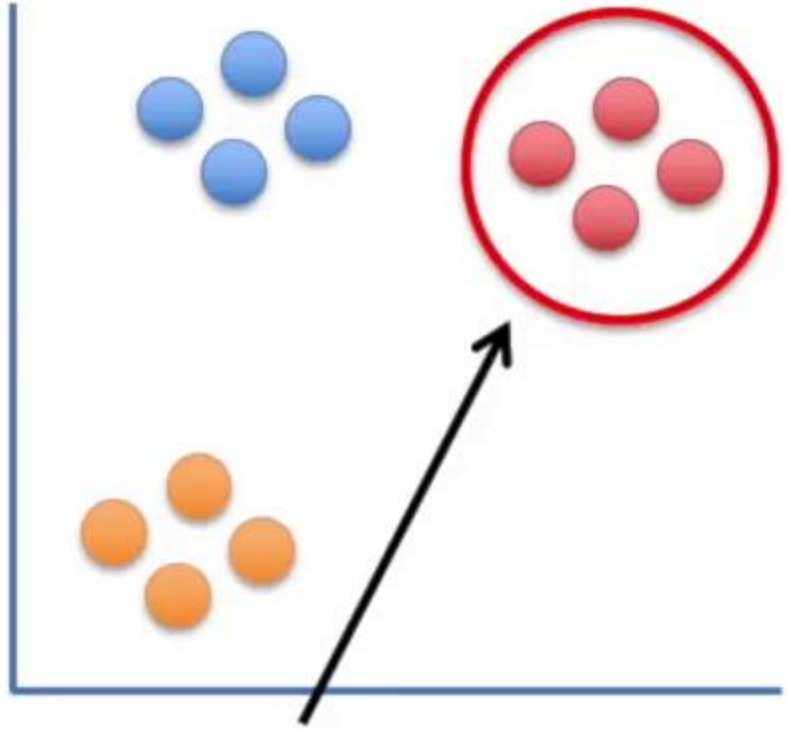




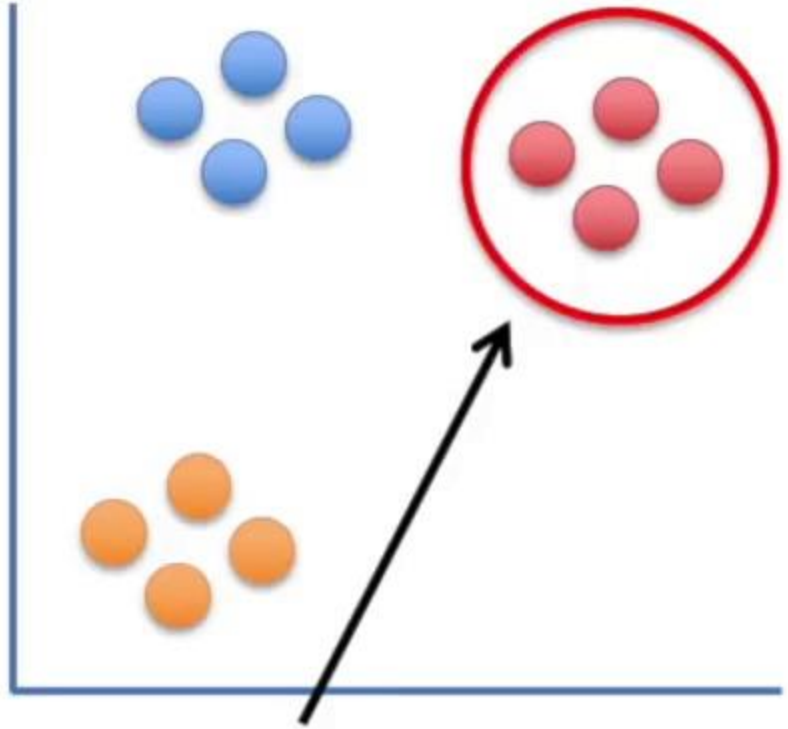
Let’s figure out where to move this first point…

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



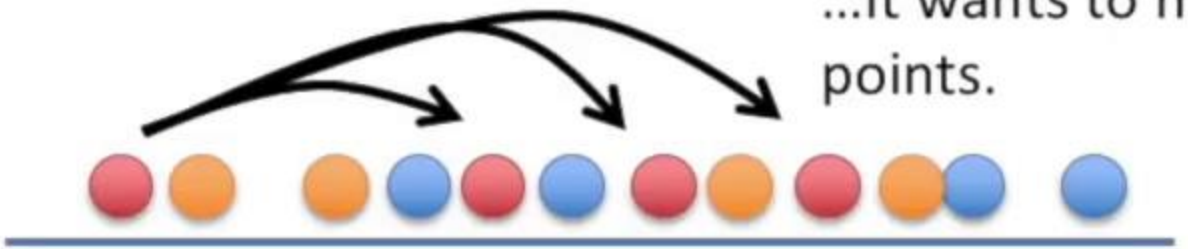


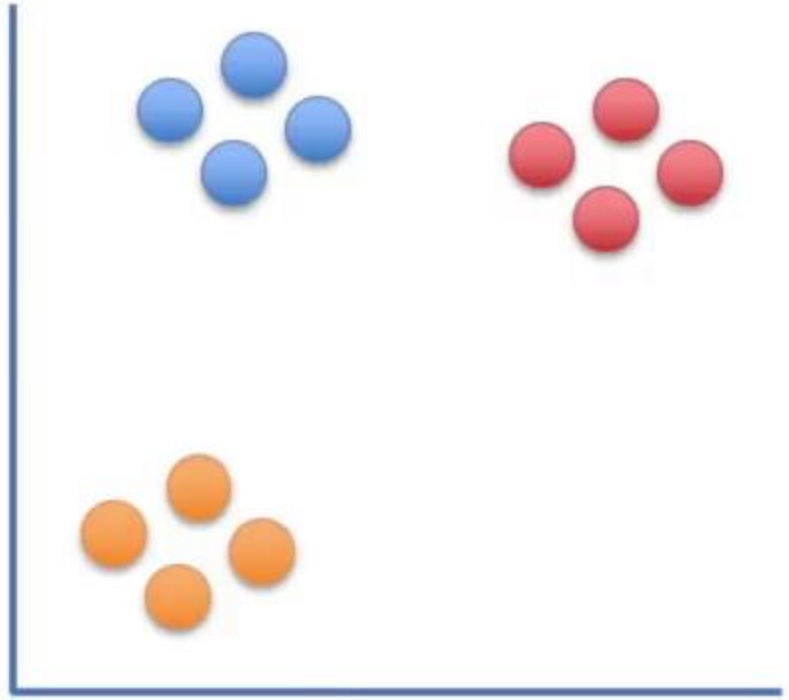
Because it is part of this cluster…



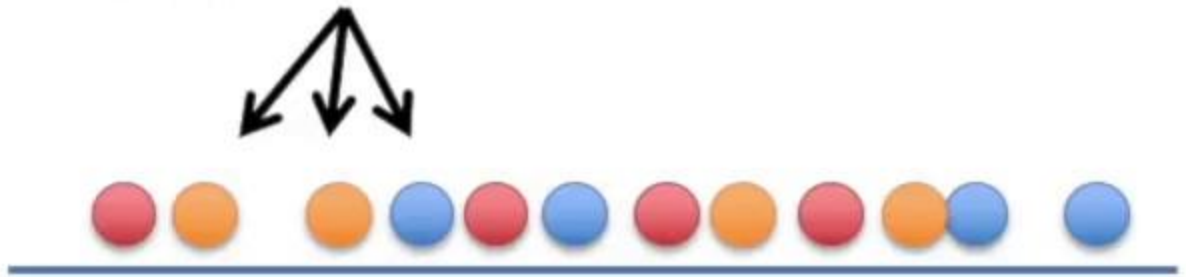
Because it is part of this cluster…

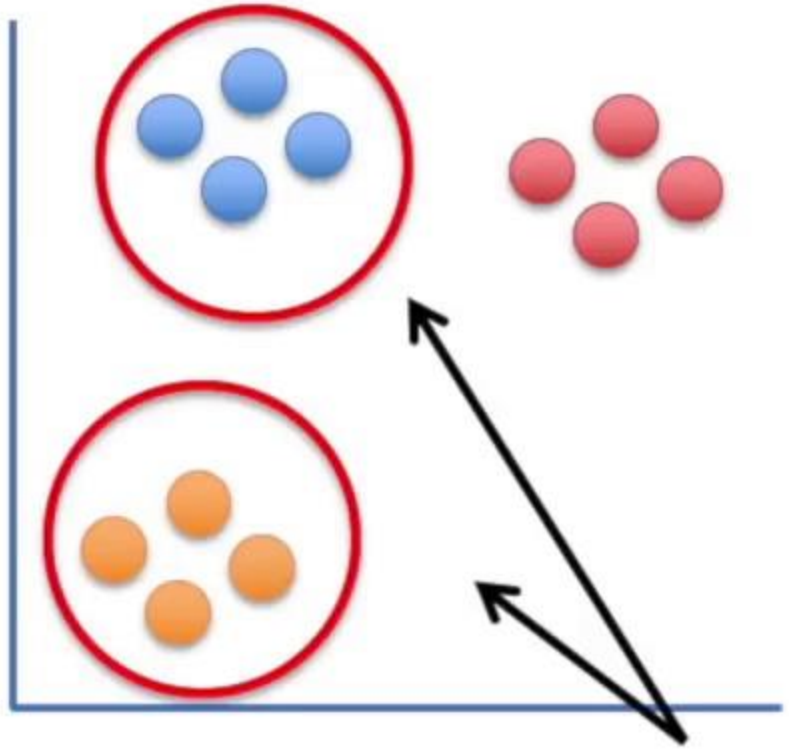
…it wants to move closer to these points.



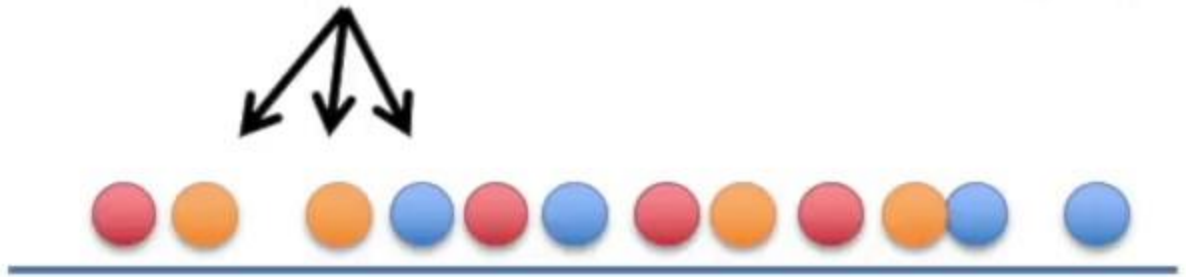


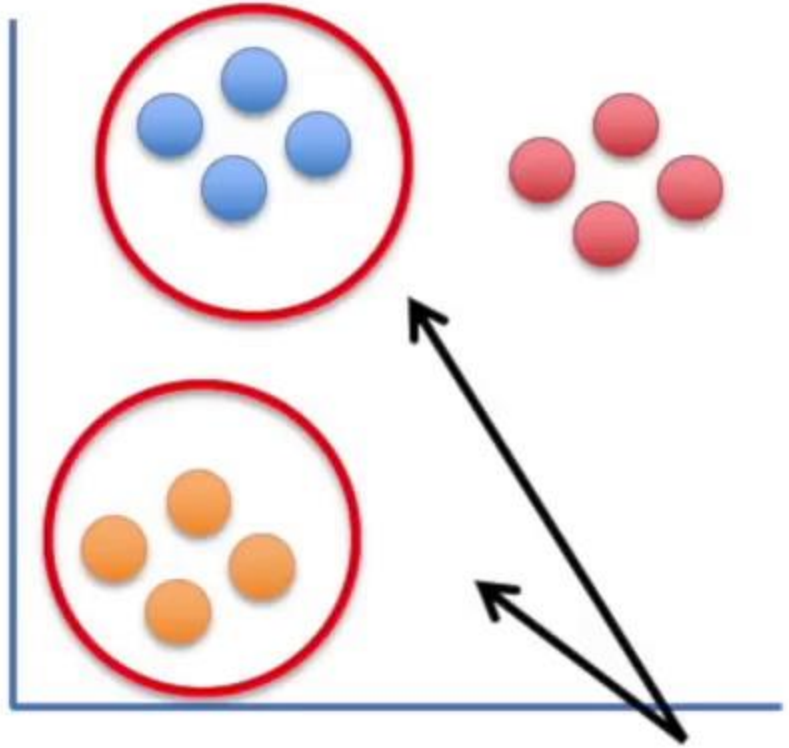
But at the same time, these points…



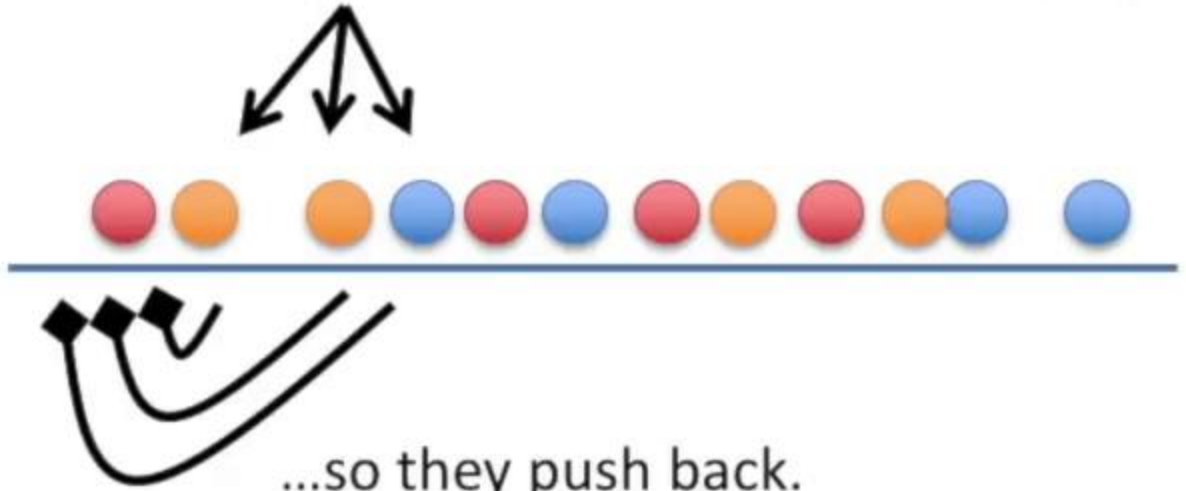


But at the same time, these points… …are far away in the scatter plot…

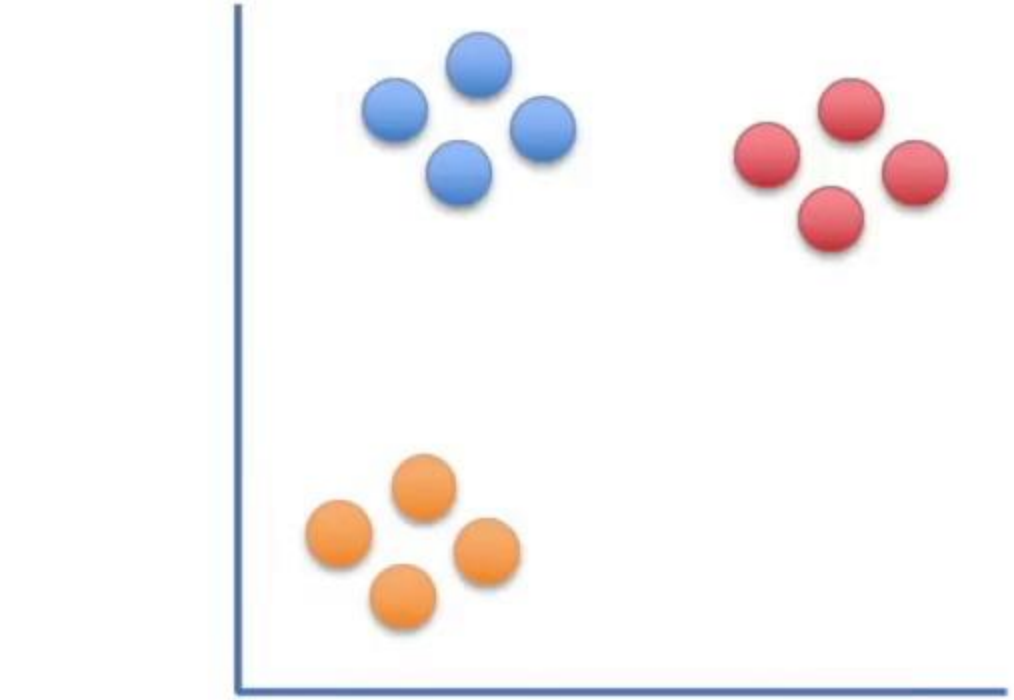


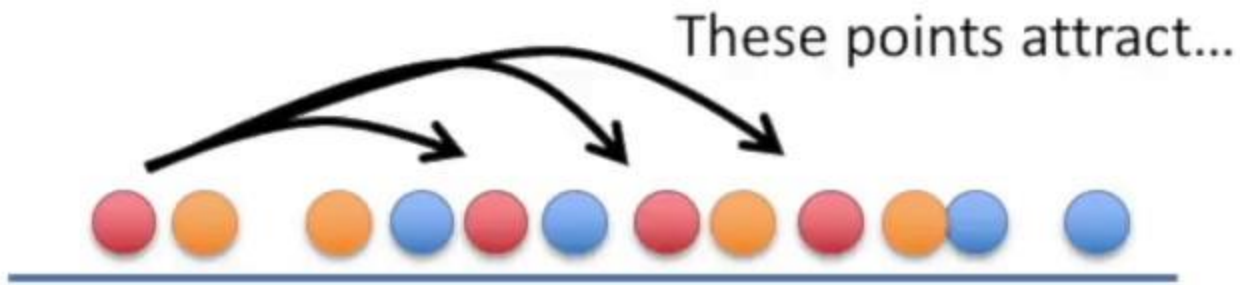


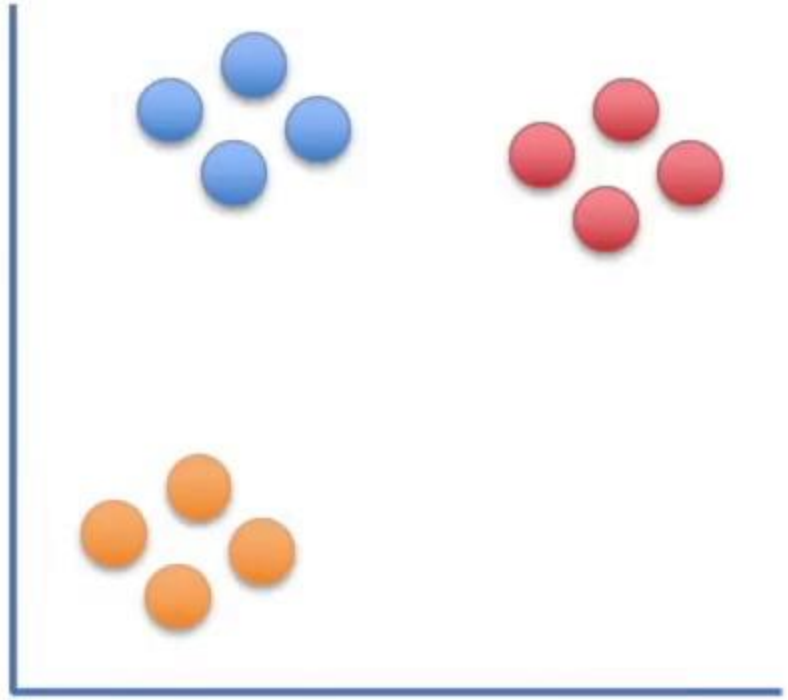
But at the same time, these points… …are far away in the scatter plot…

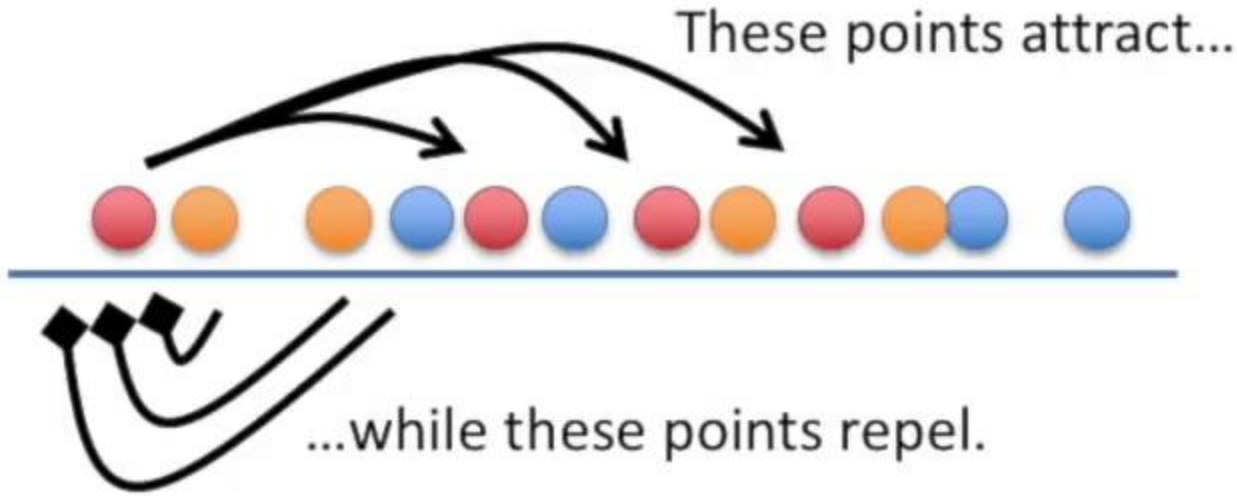


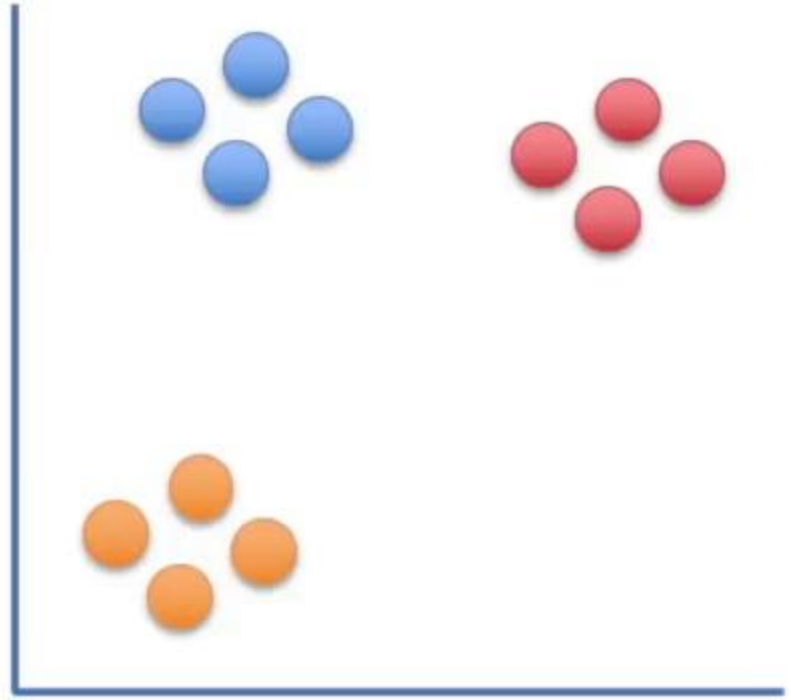
…so they push back.



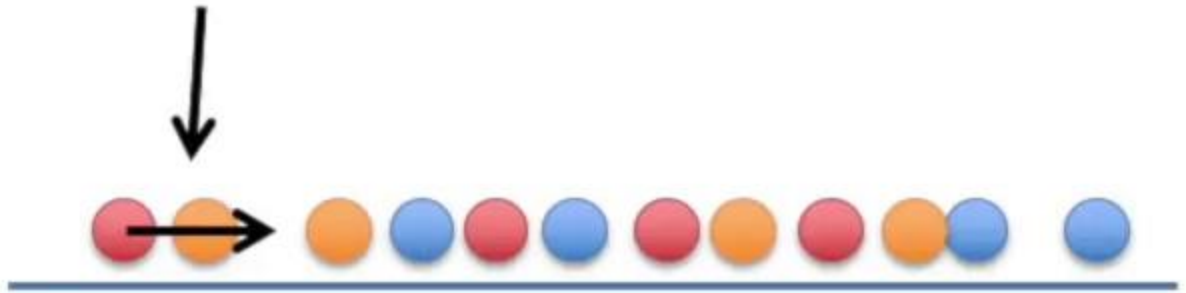

These points attract…

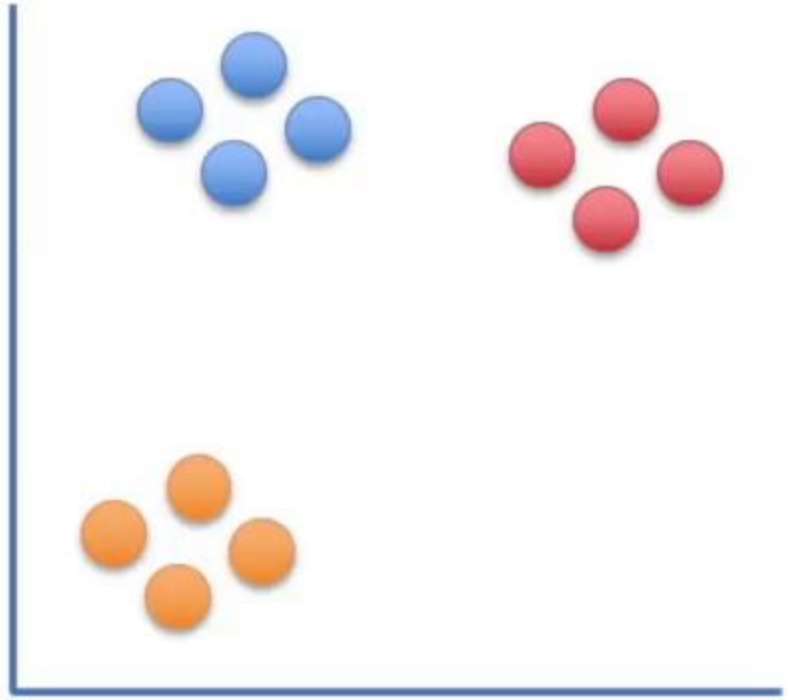



These points attract…
…while these points repel.

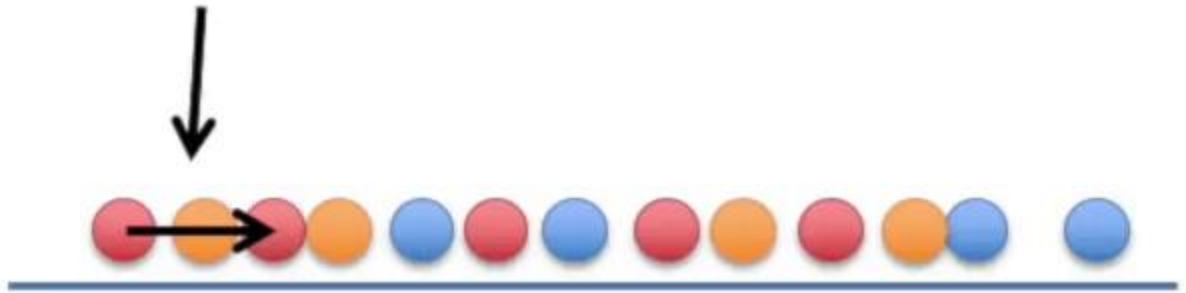


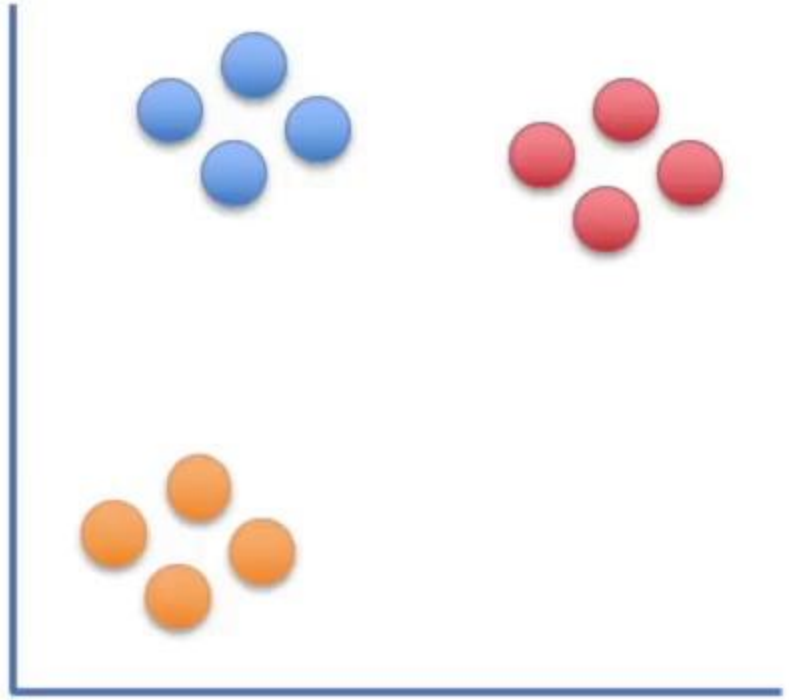
In this case, the attraction is strongest, so the point moves a little to the right.



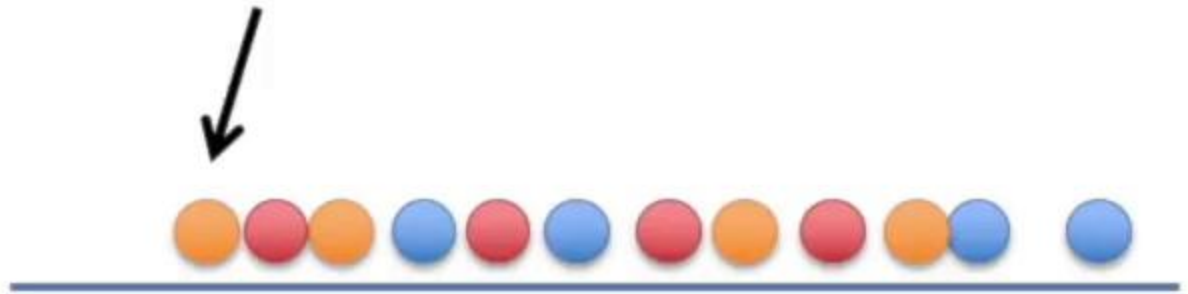


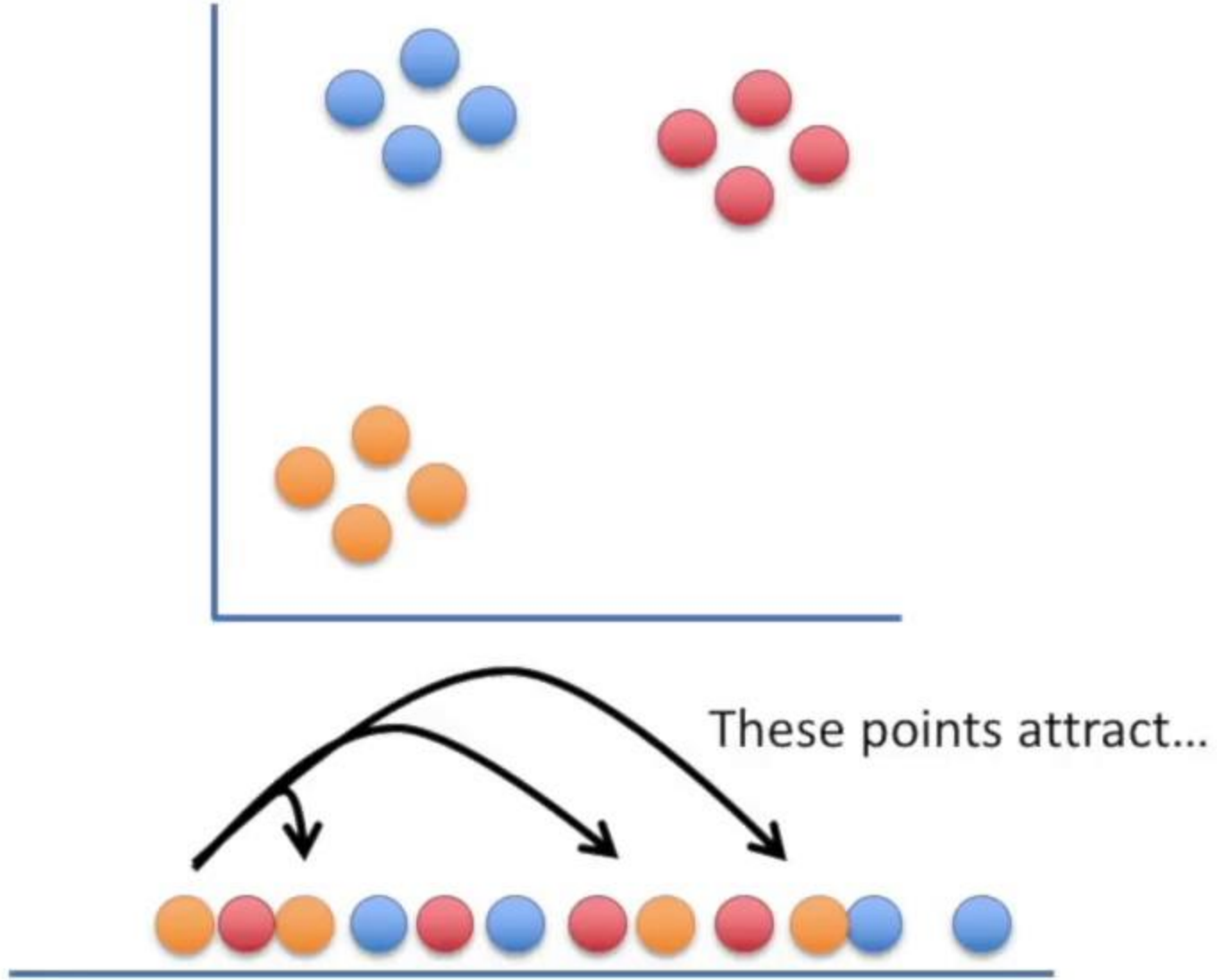
In this case, the attraction is strongest, so the point moves a little to the right.

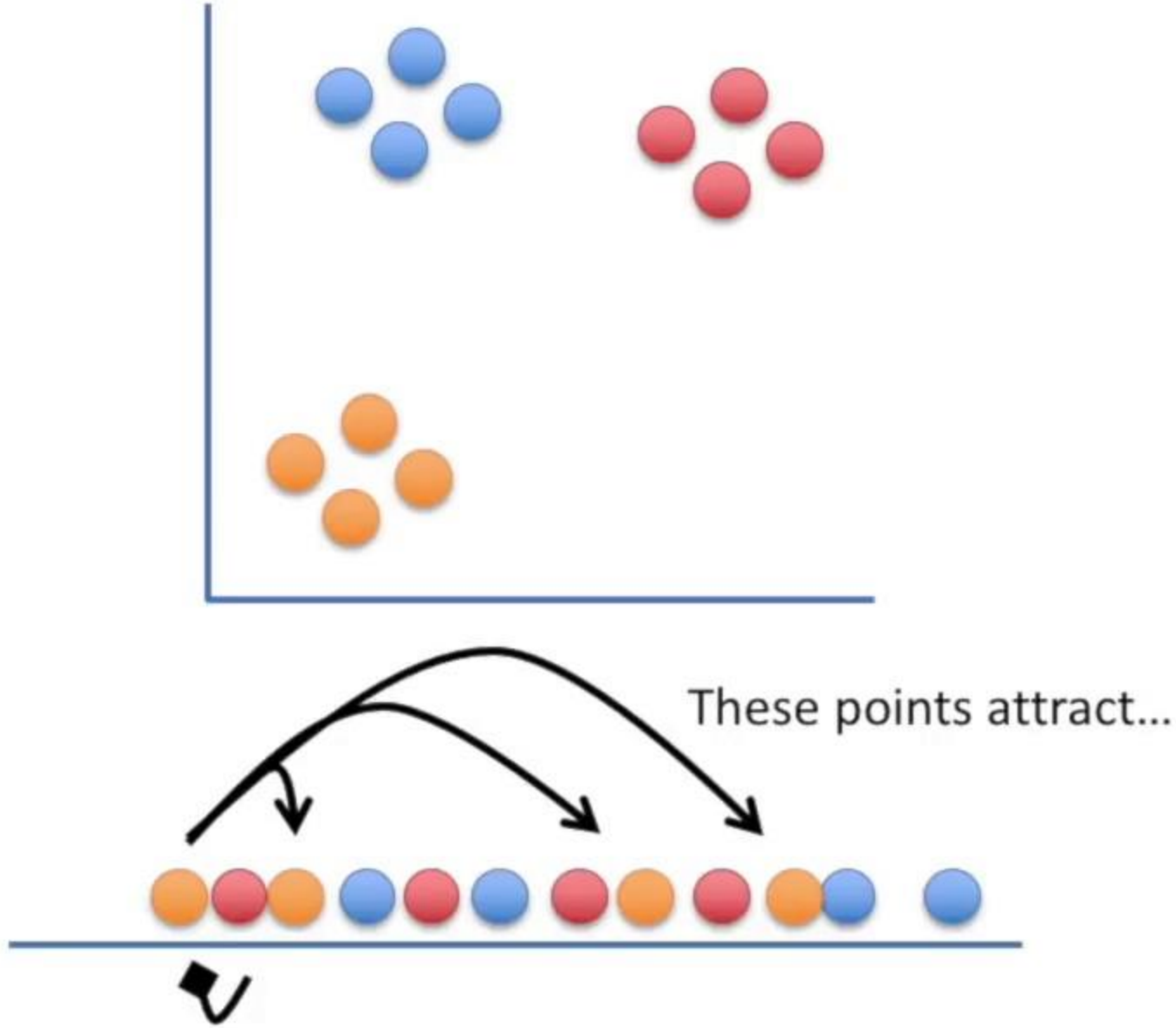


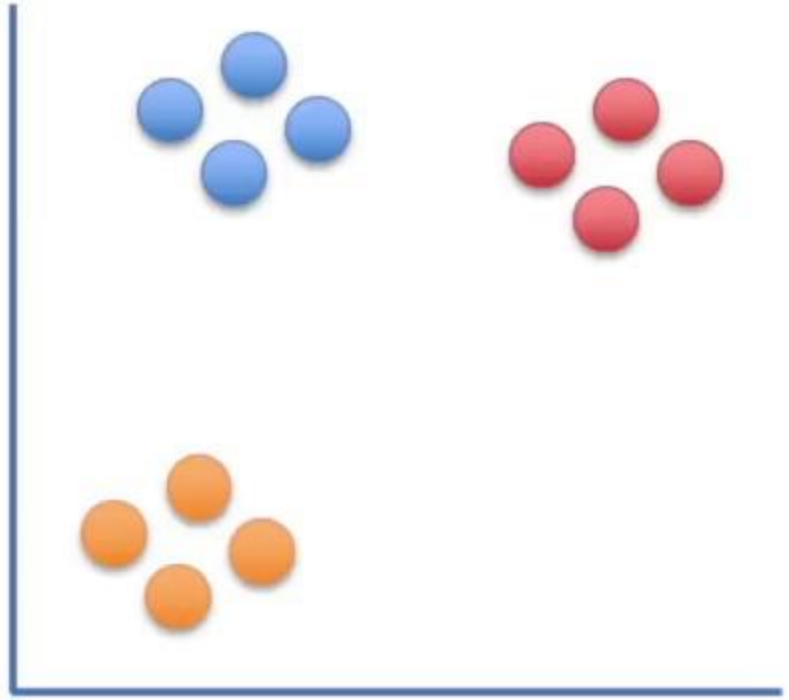


Now let's move this point a little bit…

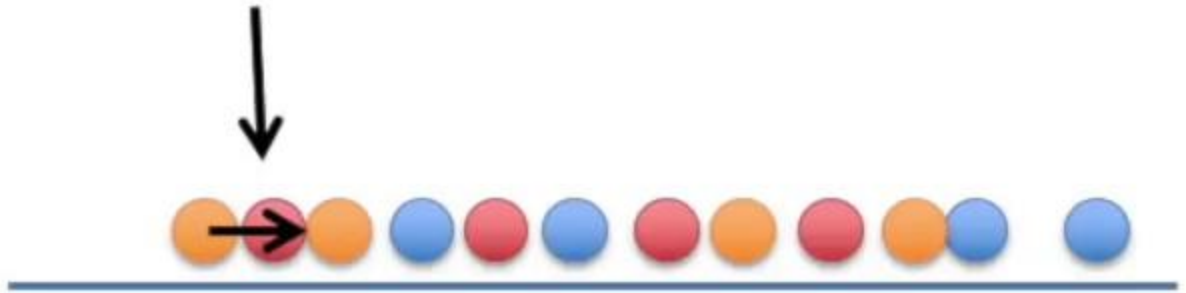


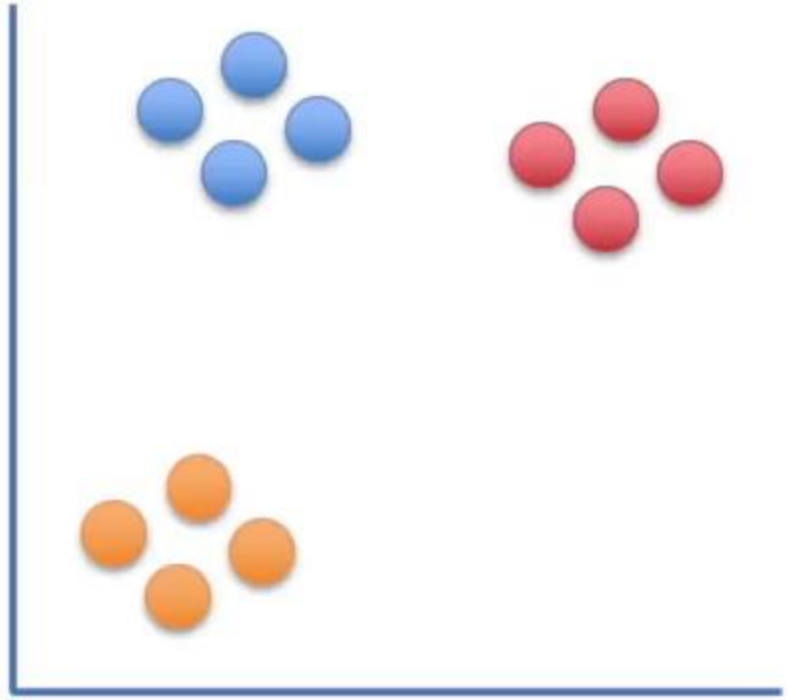

These points attract…


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

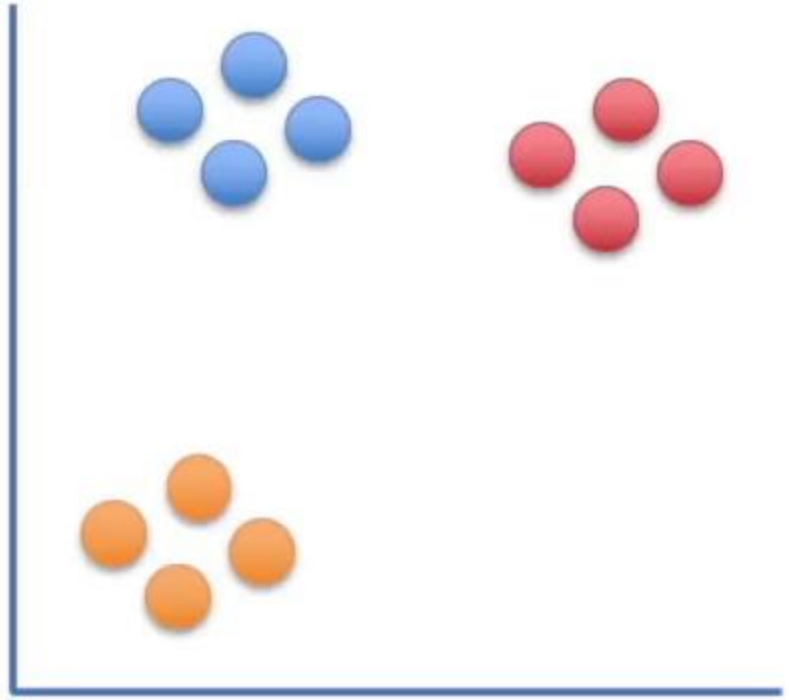


So it moves a little to closer to the other orange points.

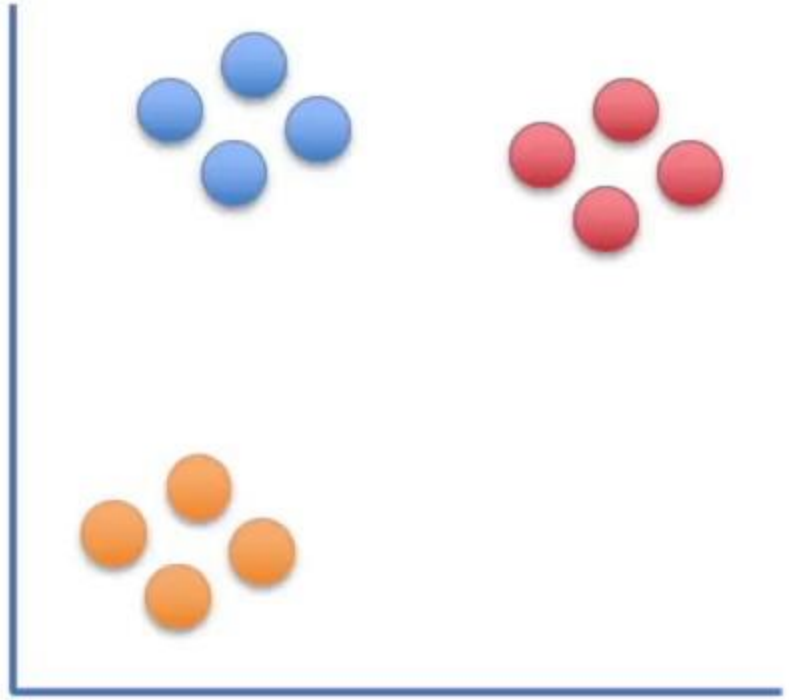




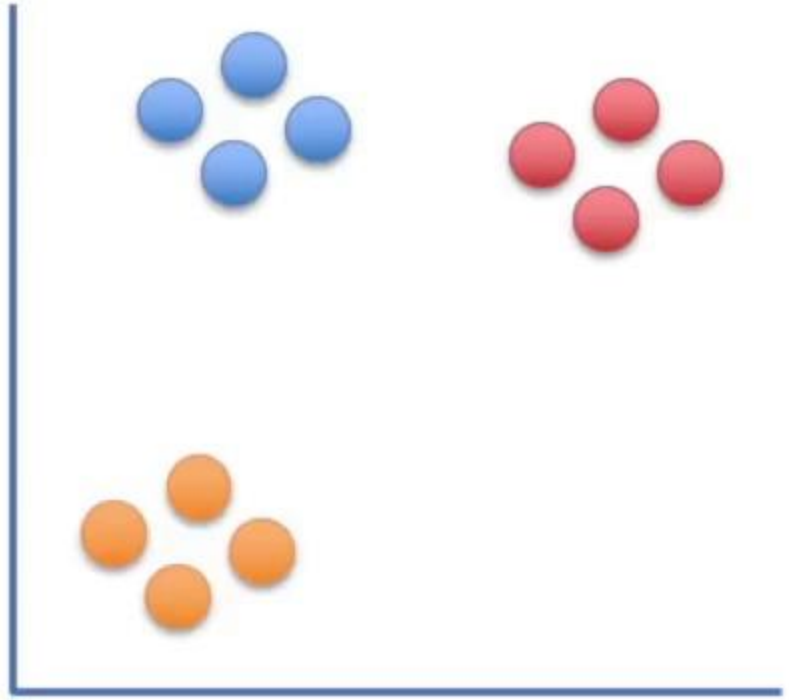
At each step, a point on the line is attracted to points it is near in the scatter plot, and repelled by points it is far from…



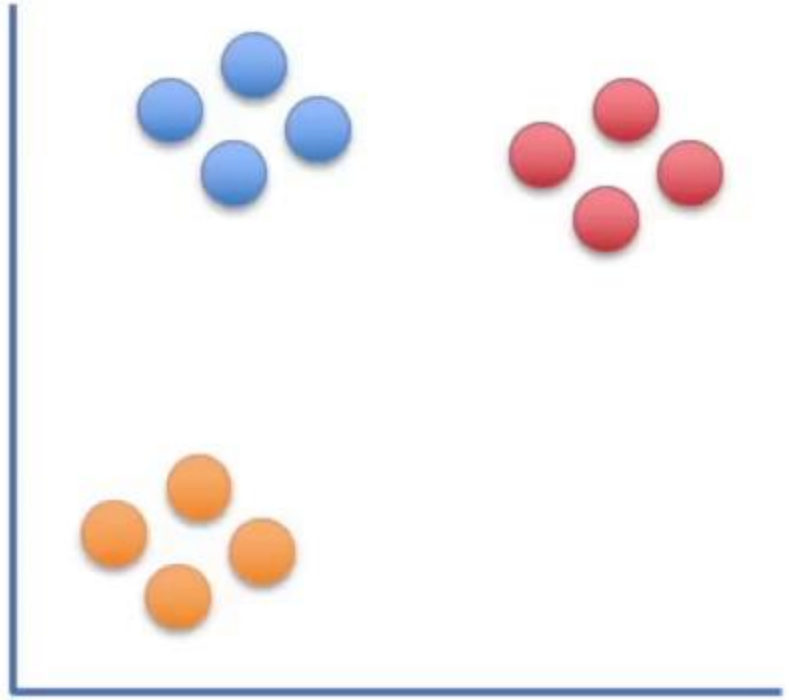
At each step, a point on the line is attracted to points it is near in the scatter plot, and repelled by points it is far from…



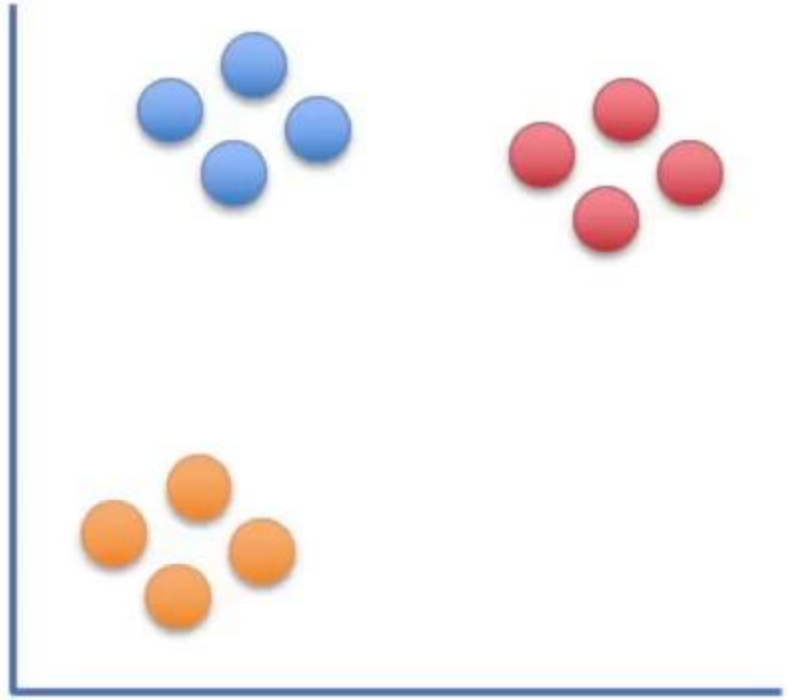
At each step, a point on the line is attracted to points it is near in the scatter plot, and repelled by points it is far from…



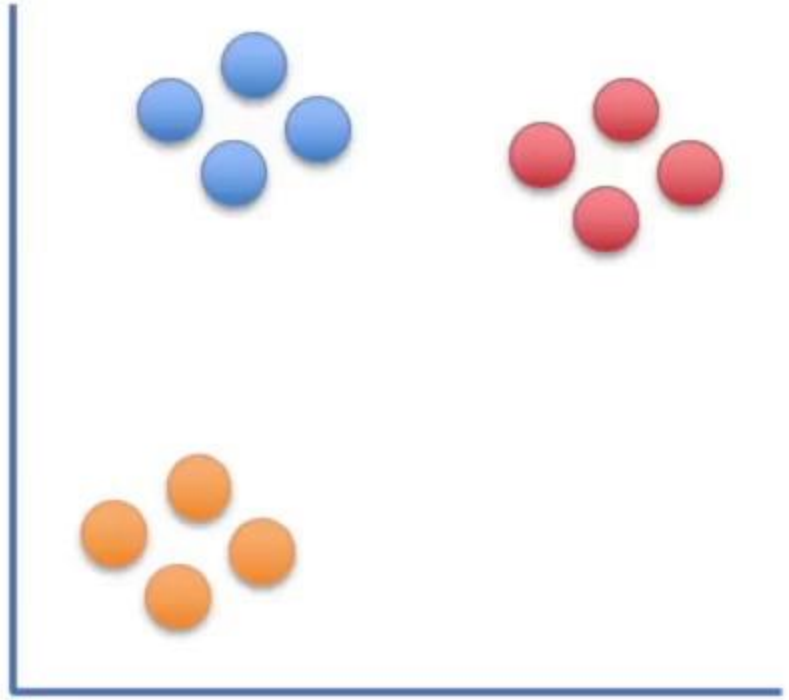
At each step, a point on the line is attracted to points it is near in the scatter plot, and repelled by points it is far from…



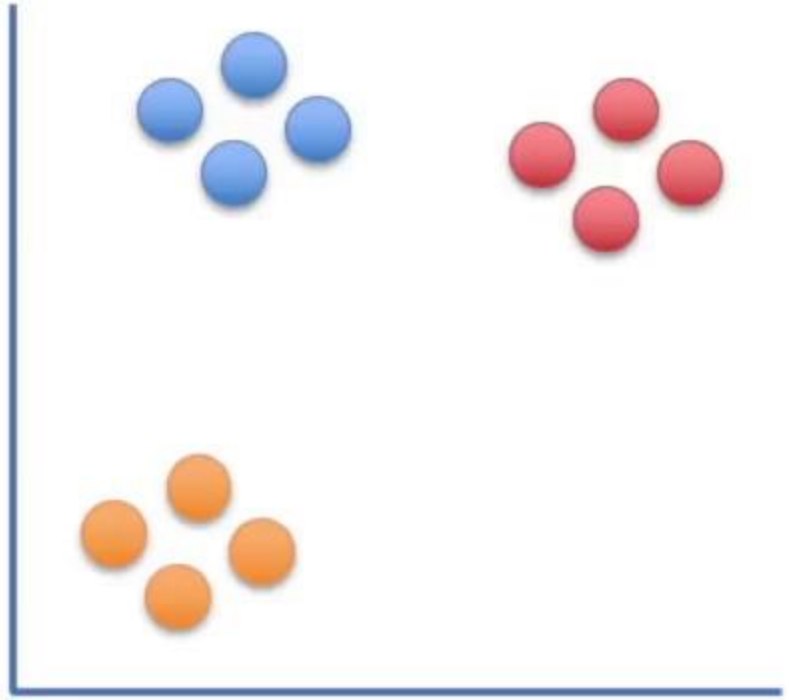
At each step, a point on the line is attracted to points it is near in the scatter plot, and repelled by points it is far from…



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…

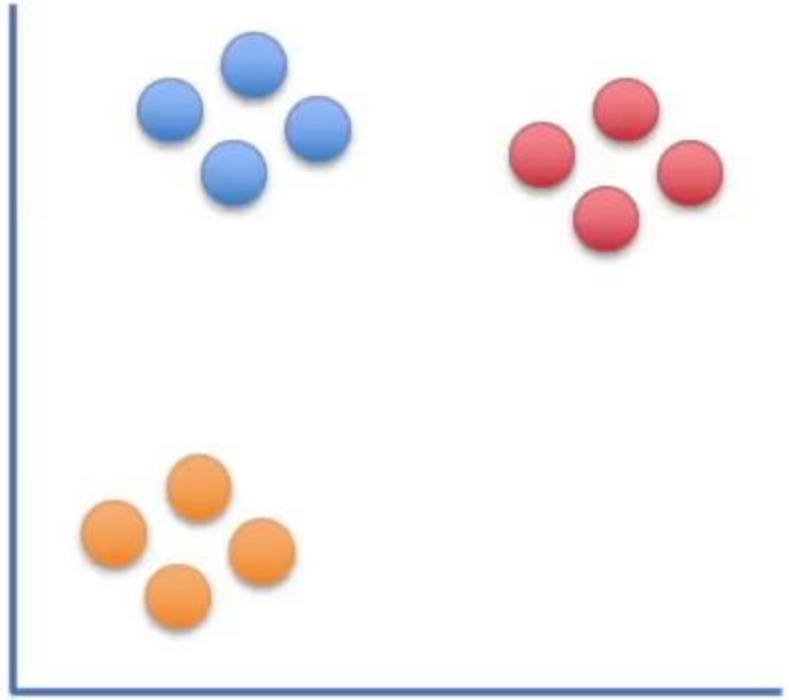


At each step, a point on the line is attracted to points it is near in the scatter plot, and repelled by points it is far from…

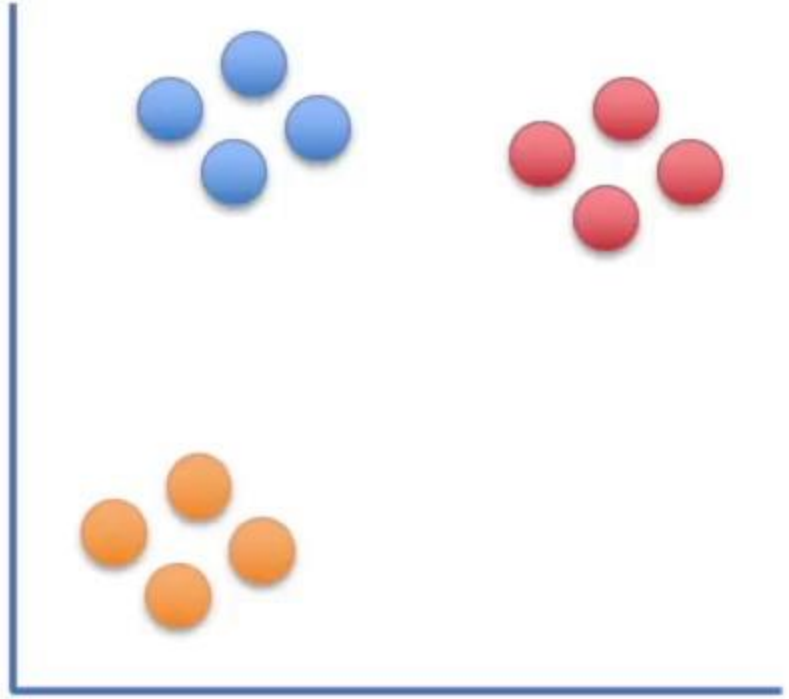


At each step, a point on the line is attracted to points it is near in the scatter plot, and repelled by points it is far from…

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.

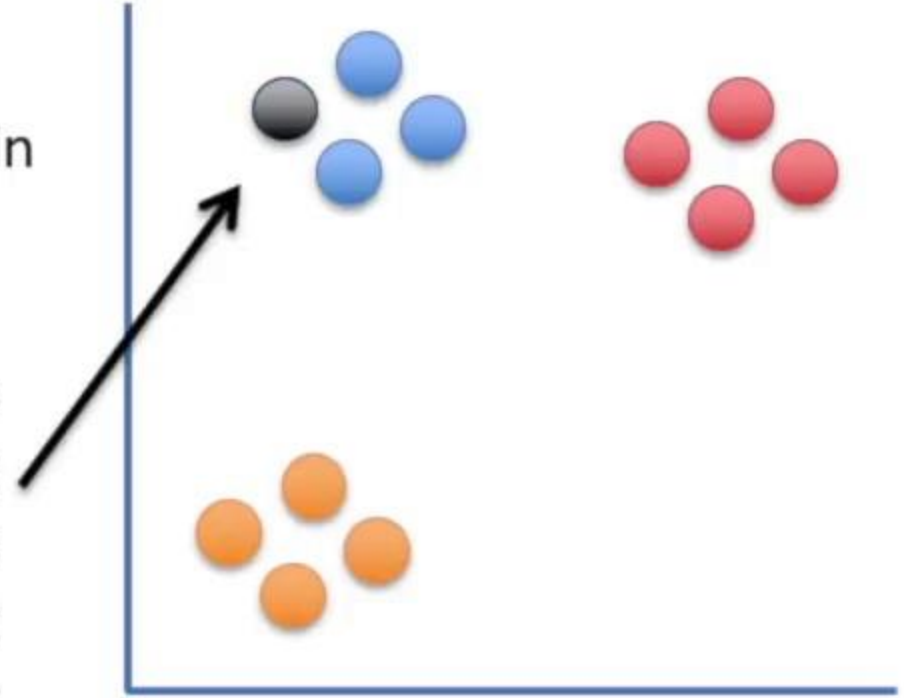


Step 1: Determine the "similarity" of all the points in the scatter plot.

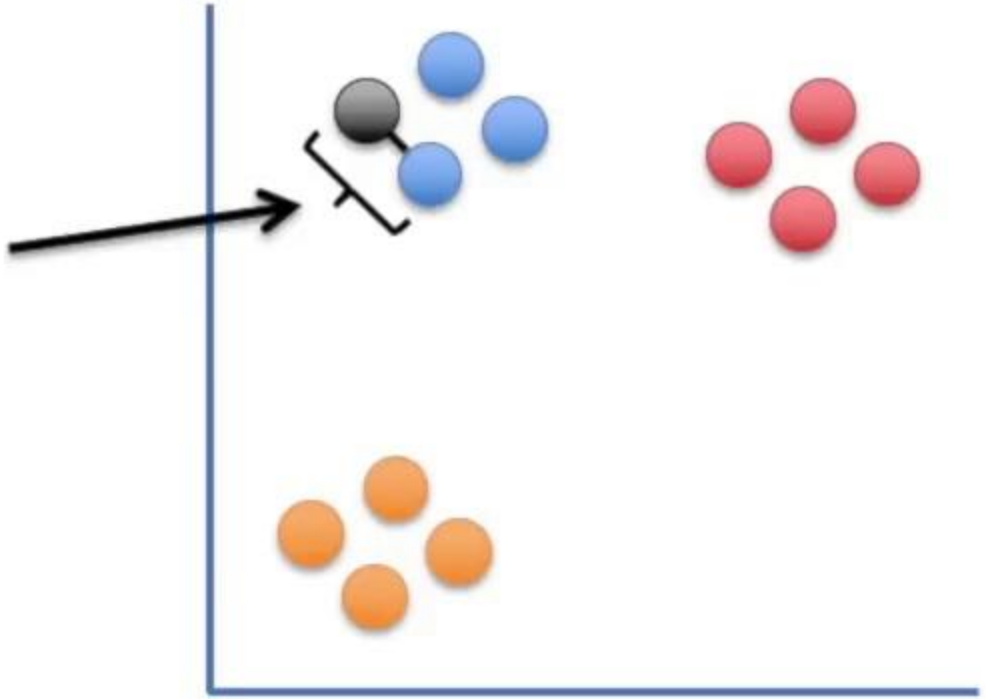


Step 1: Determine the "similarity" of all the points in the scatter plot.

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

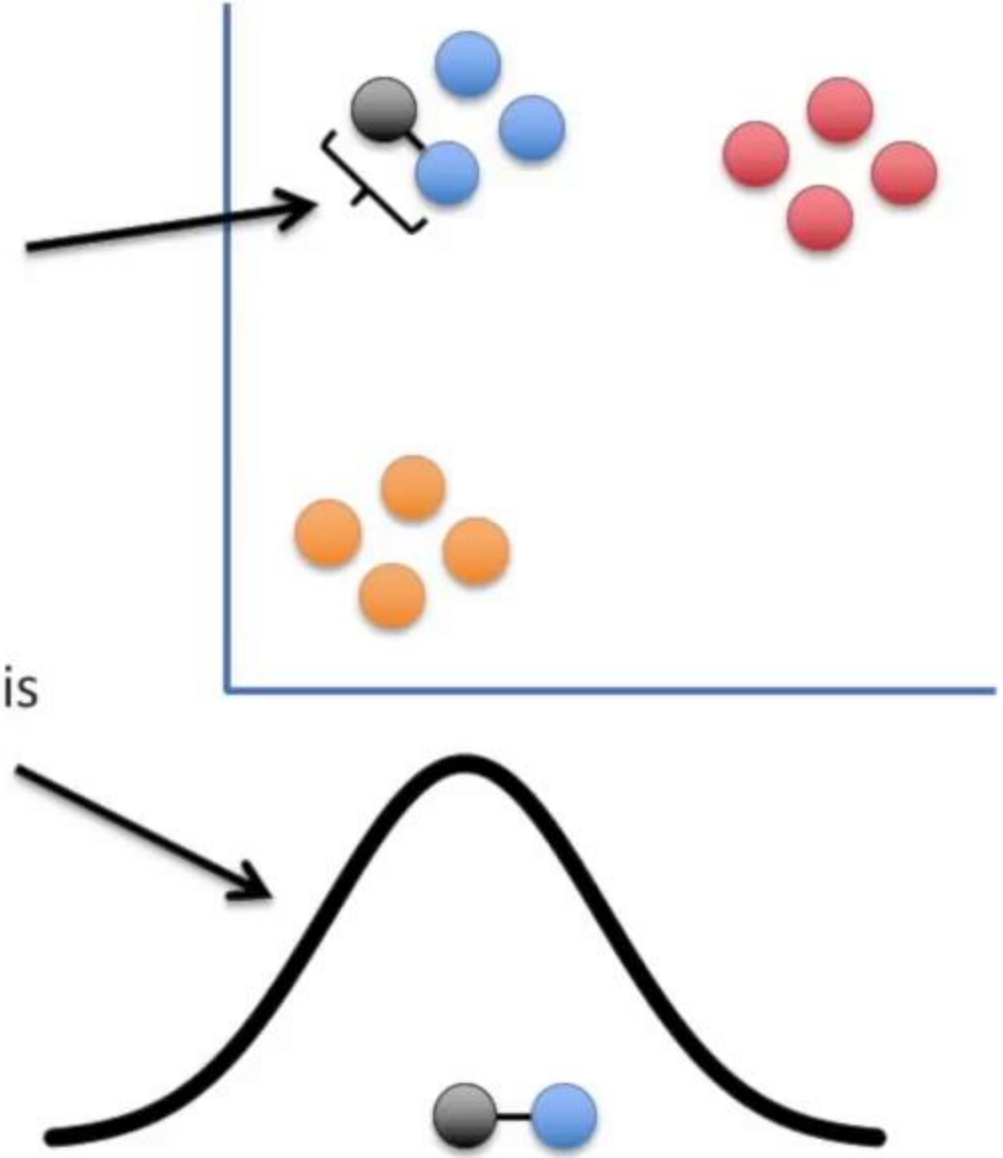


First, measure the distance between two points…

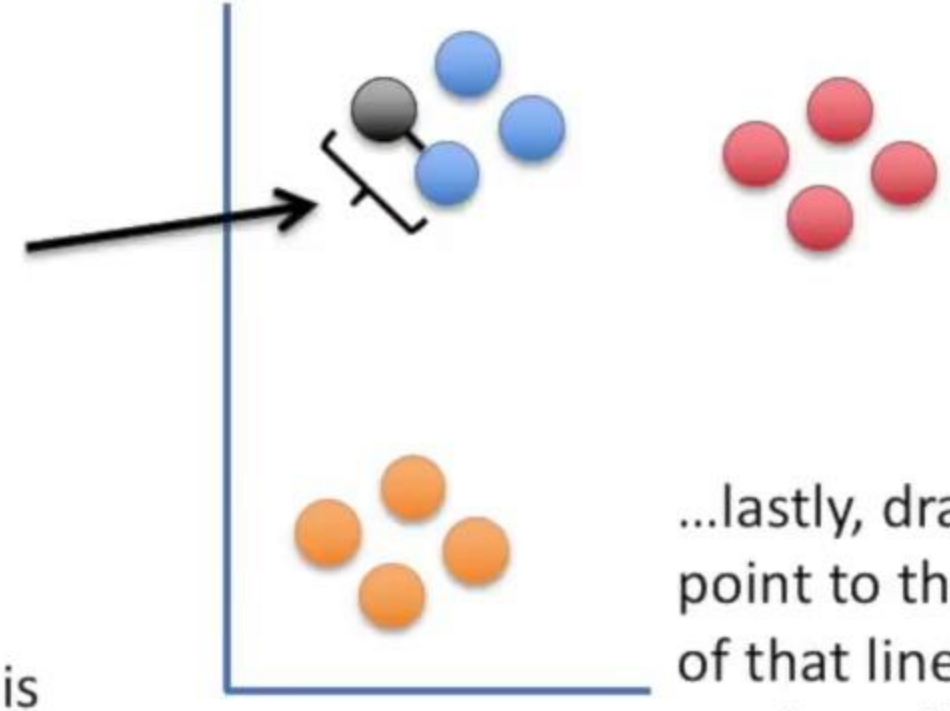


First, measure the distance between two points…

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

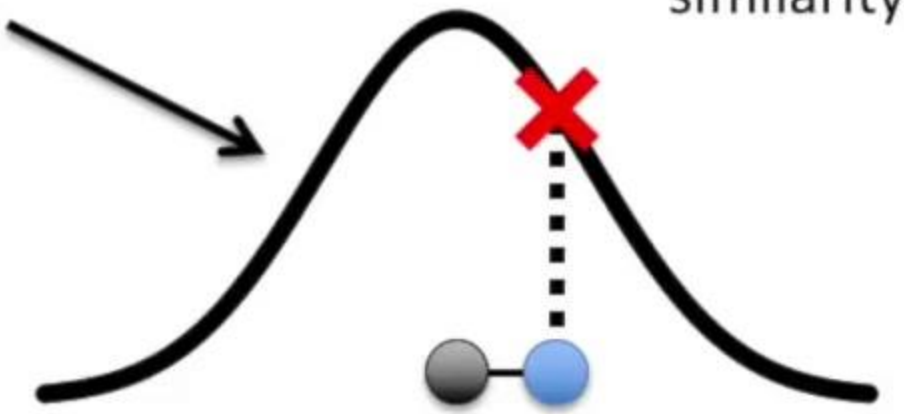


First, measure the distance between two points…

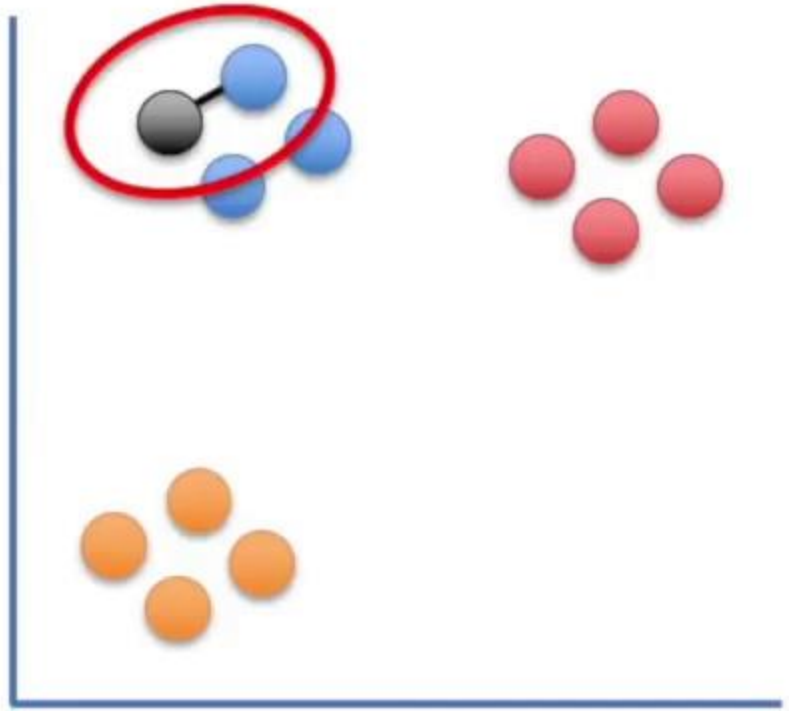


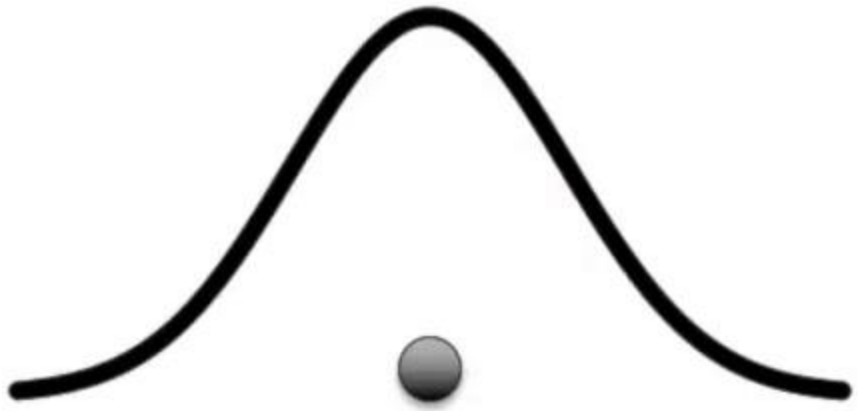
…lastly, draw a line from the point to the curve. The length of that line is the “unscaled similarity”.

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

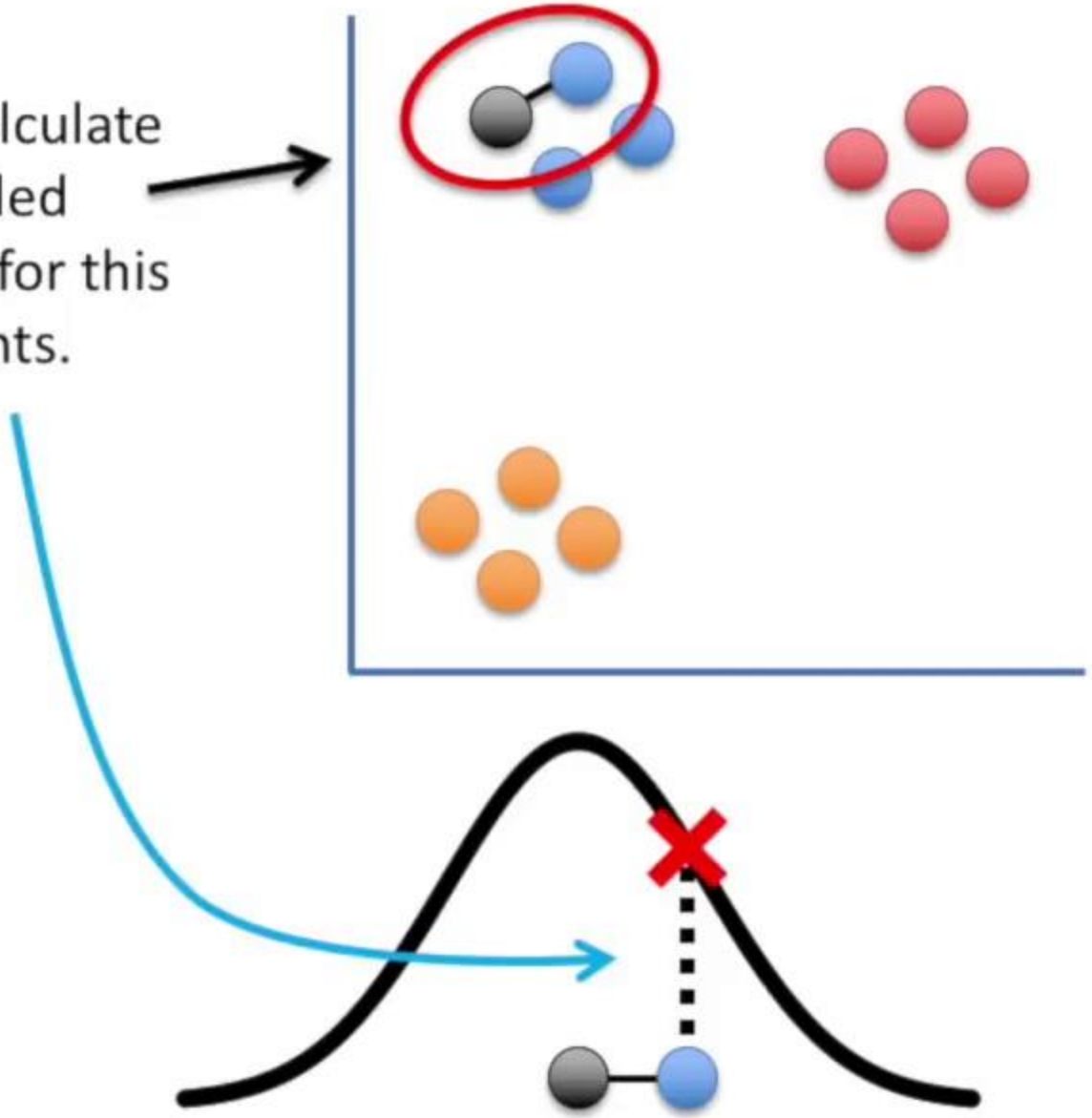


Now we calculate the “unscaled similarity” for this pair of points.

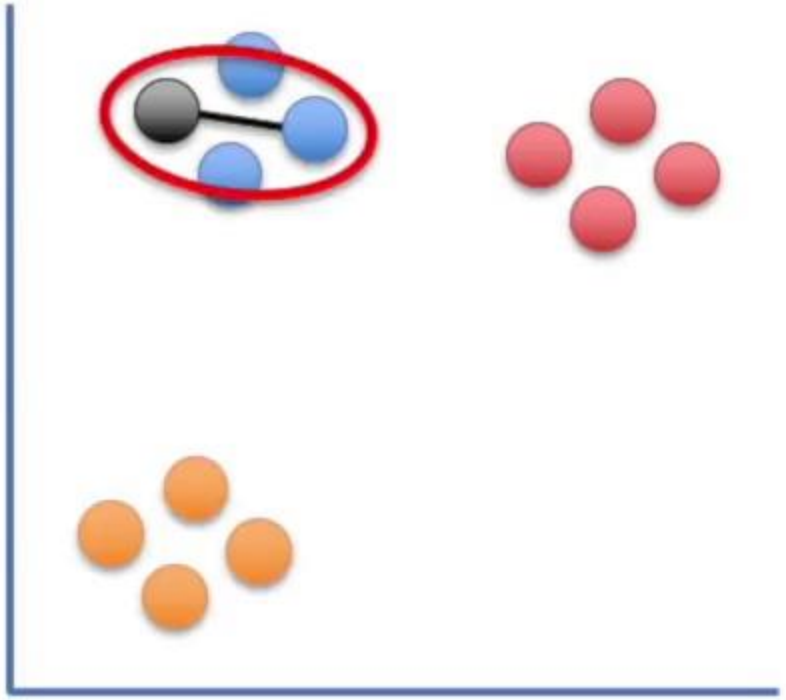


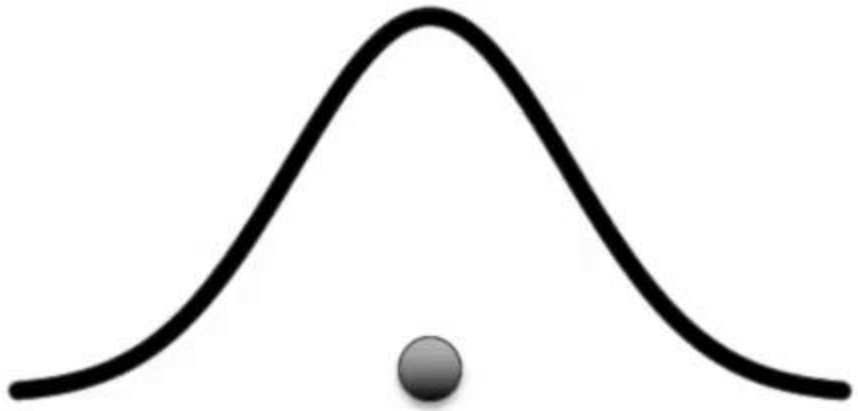


Now we calculate the “unscaled similarity” for this pair of points.

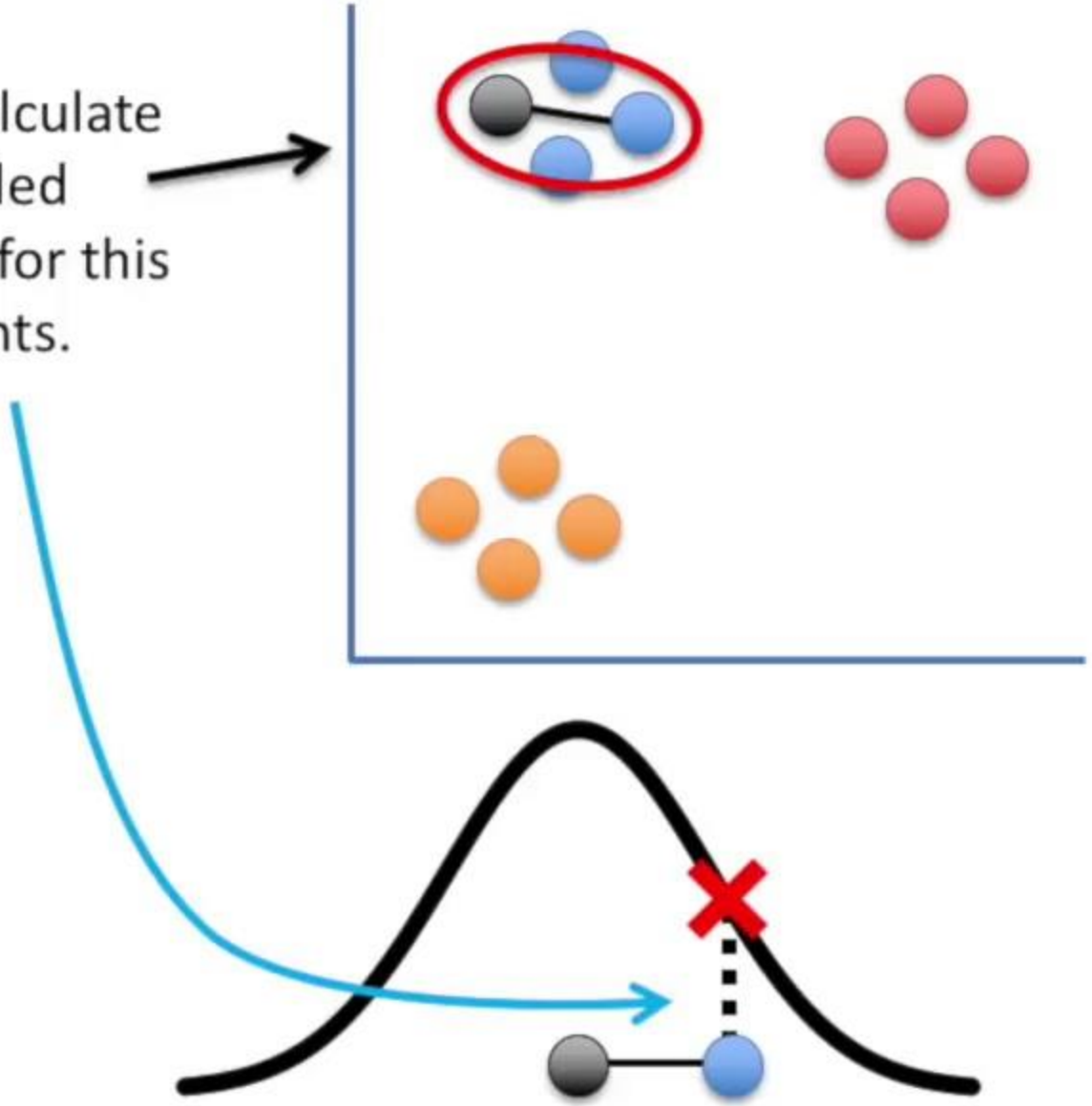


Now we calculate the “unscaled similarity” for this pair of points.

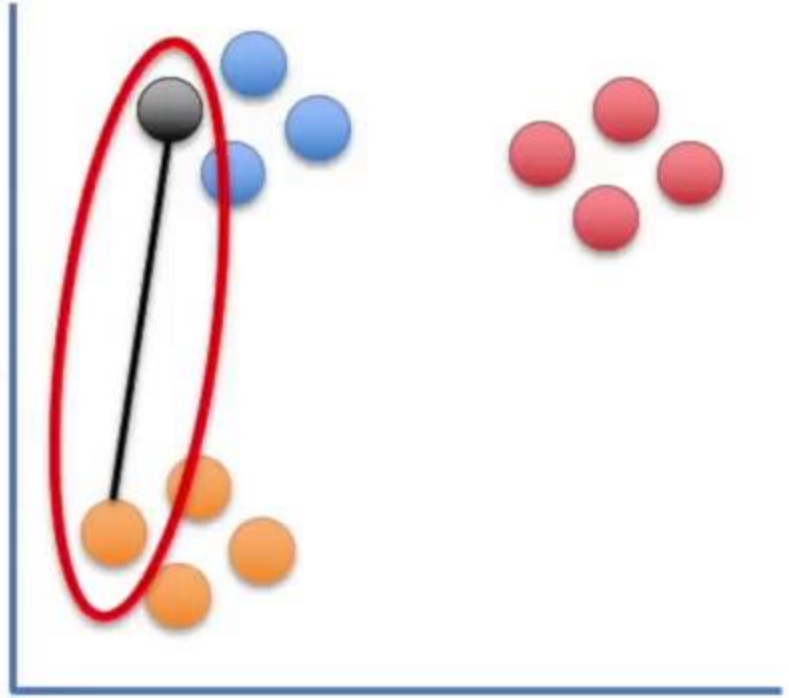


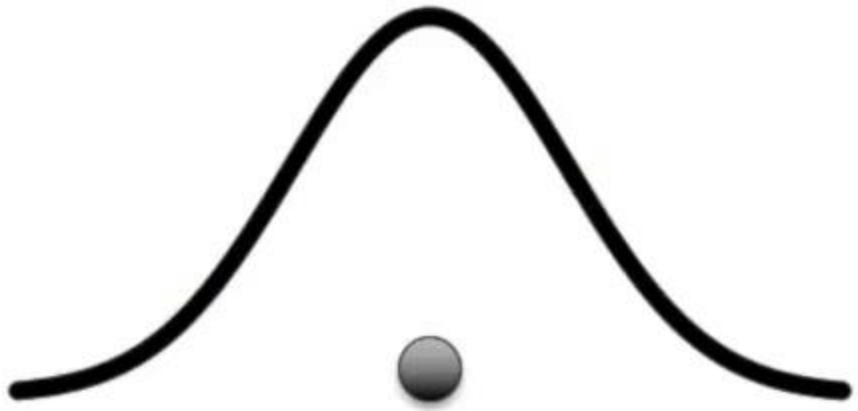


Now we calculate the “unscaled similarity” for this pair of points.

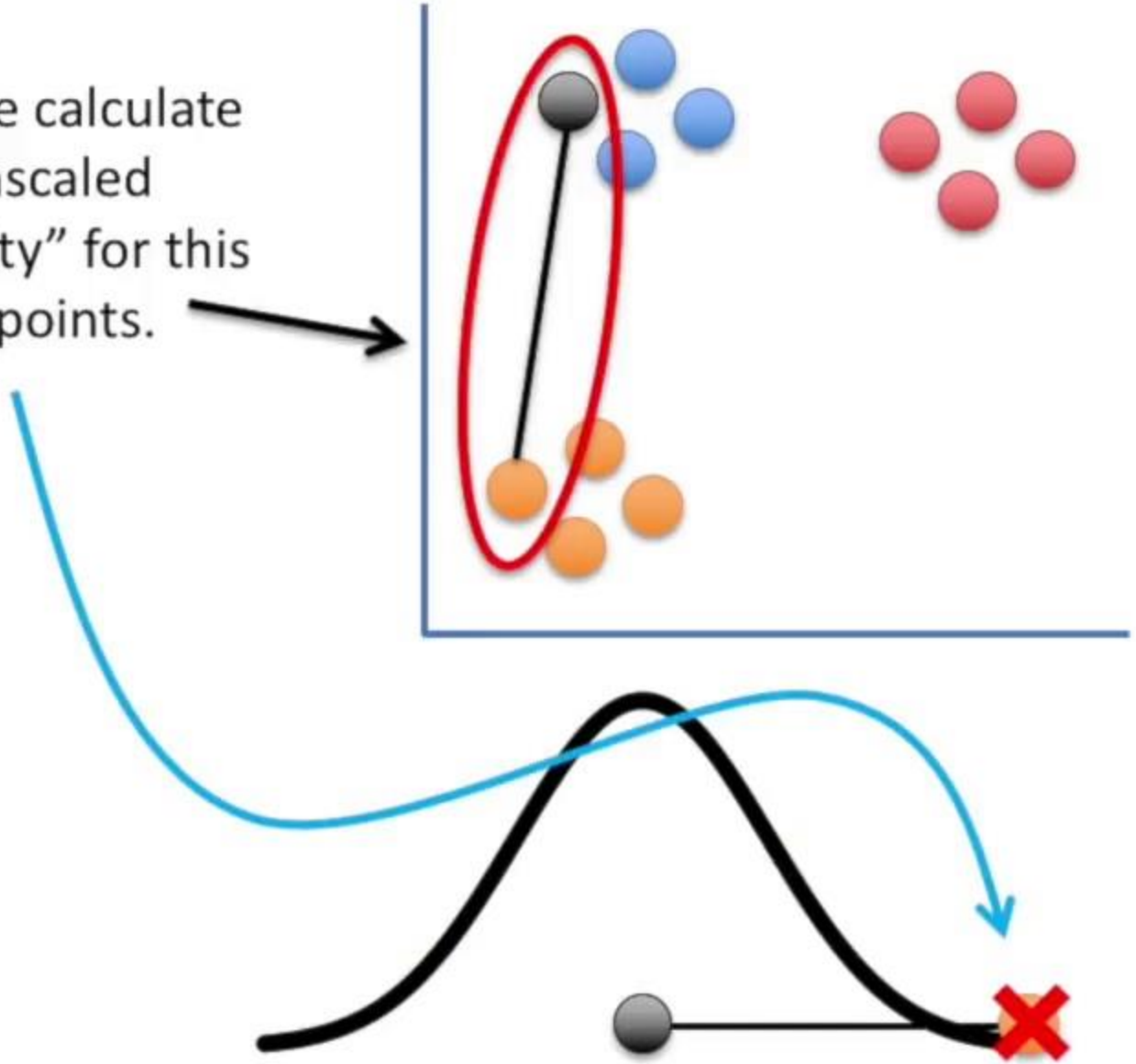


Now we calculate the “unscaled similarity” for this pair of points.

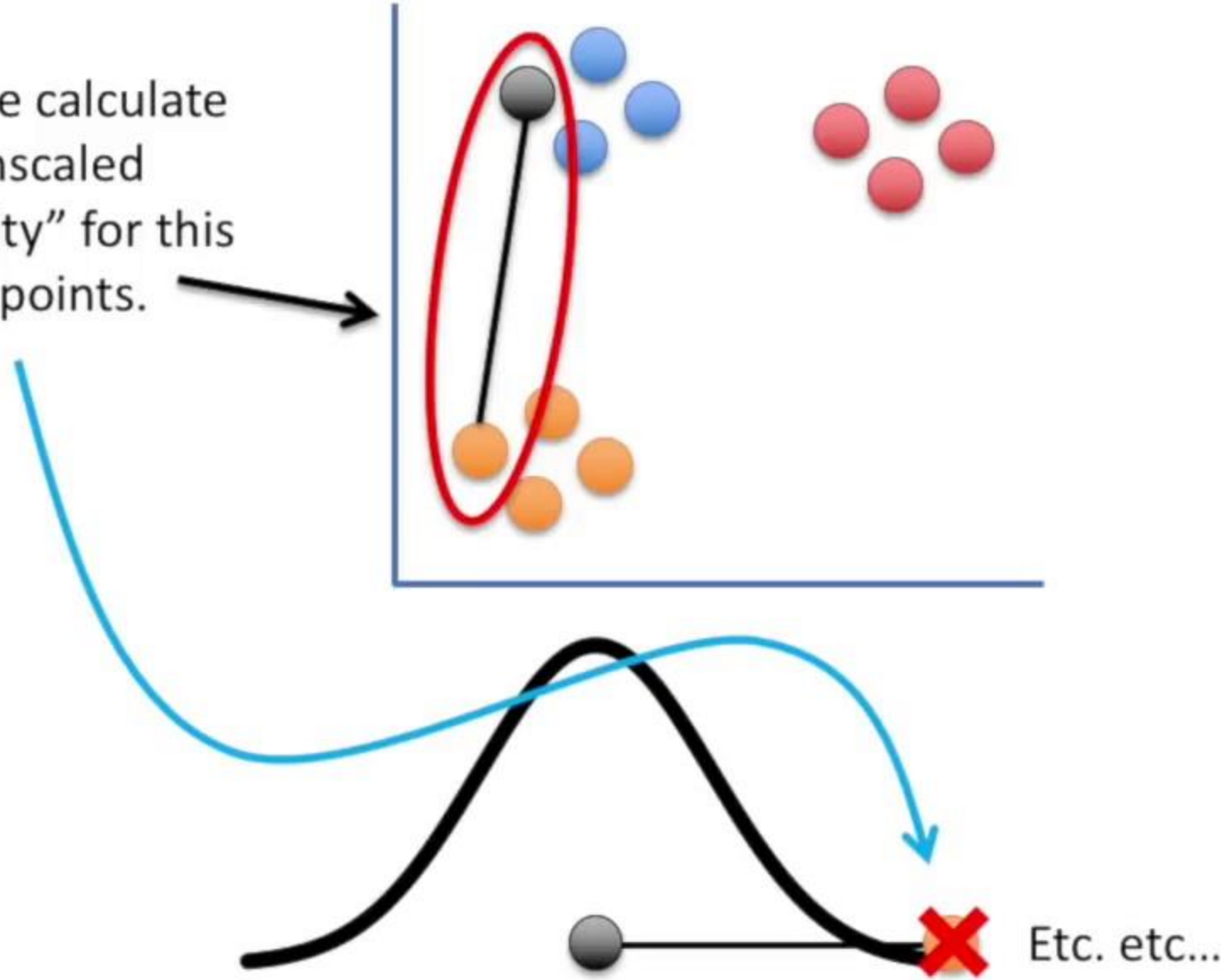




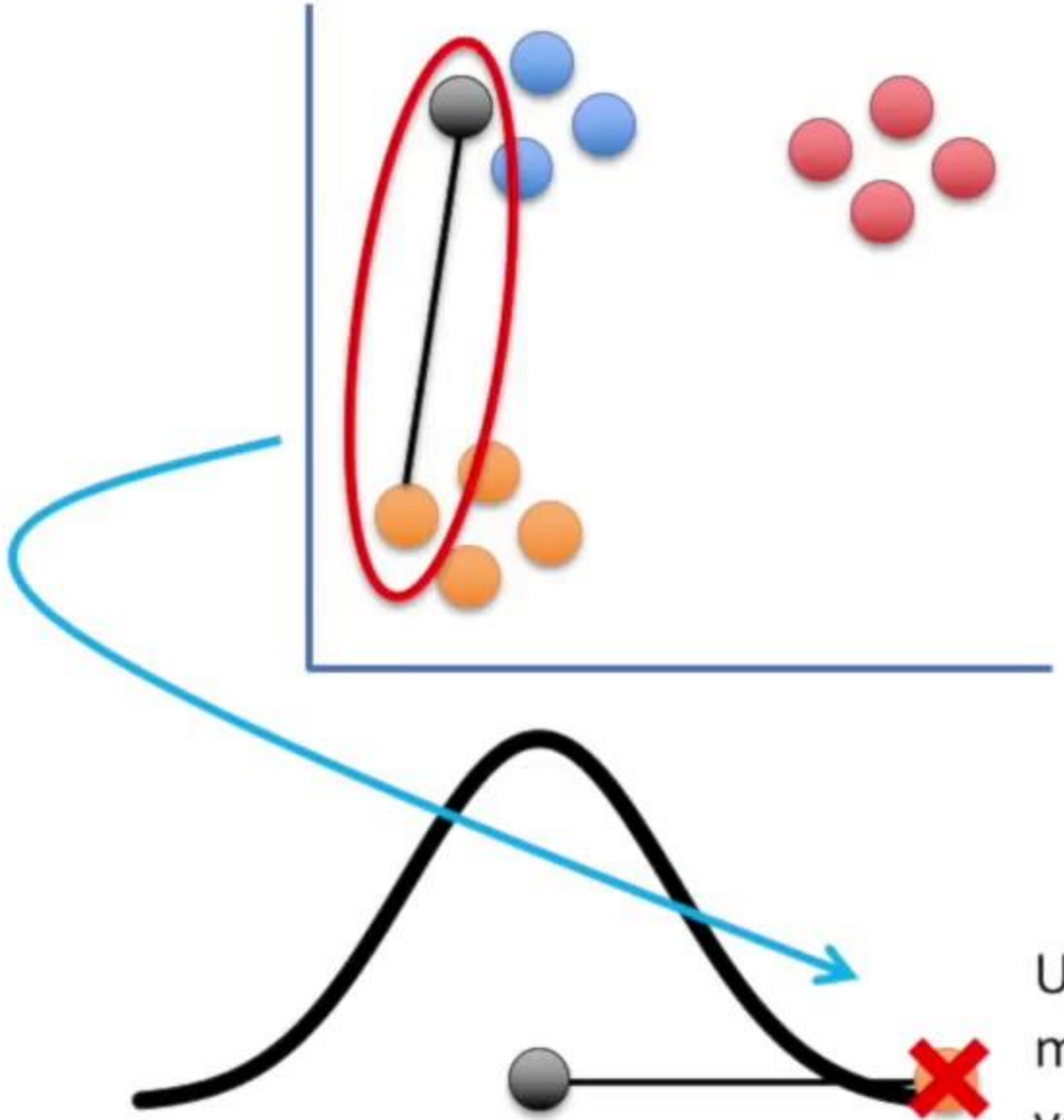
Now we calculate the “unscaled similarity” for this pair of points.



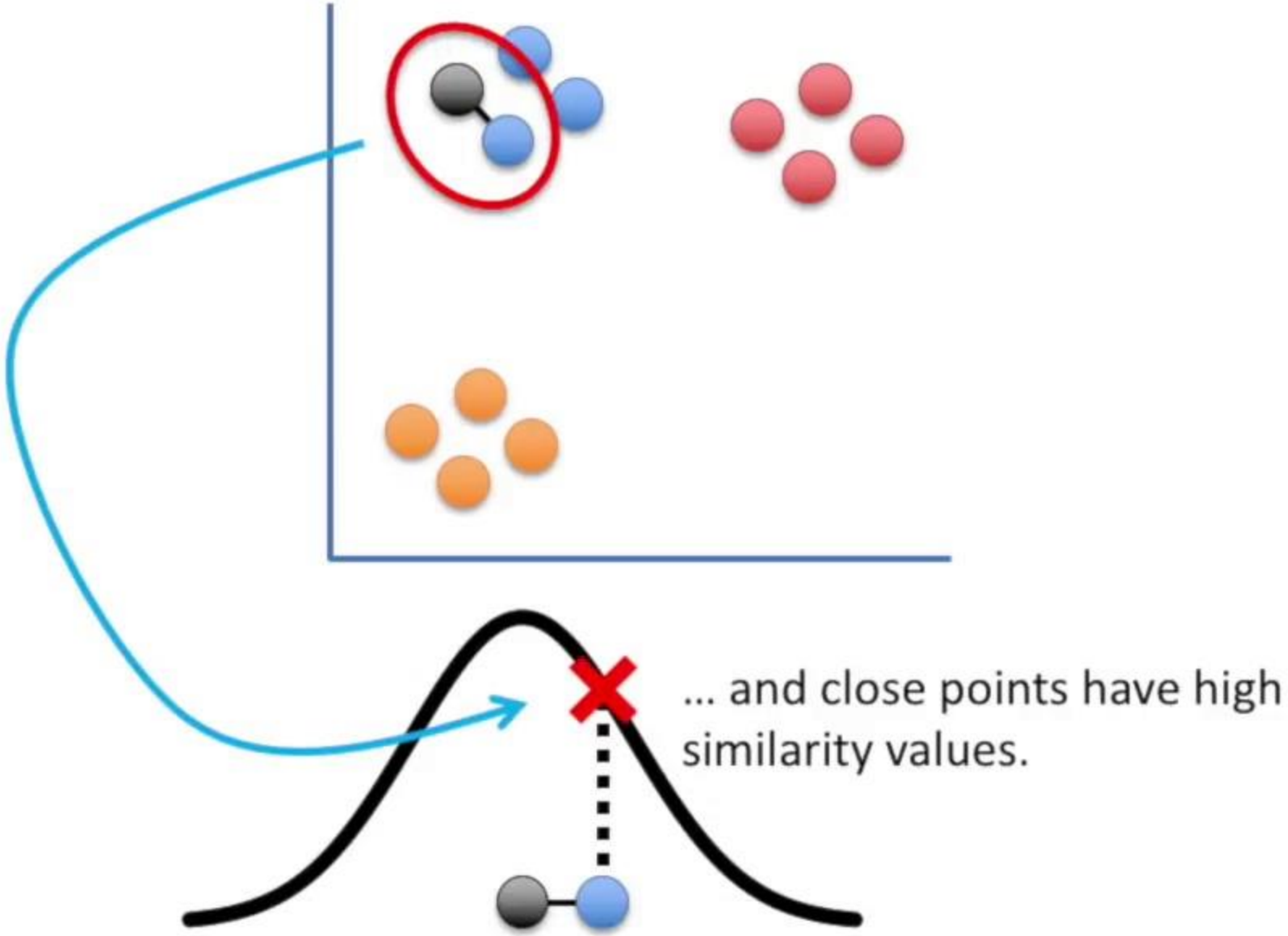
Now we calculate the “unscaled similarity” for this pair of points.



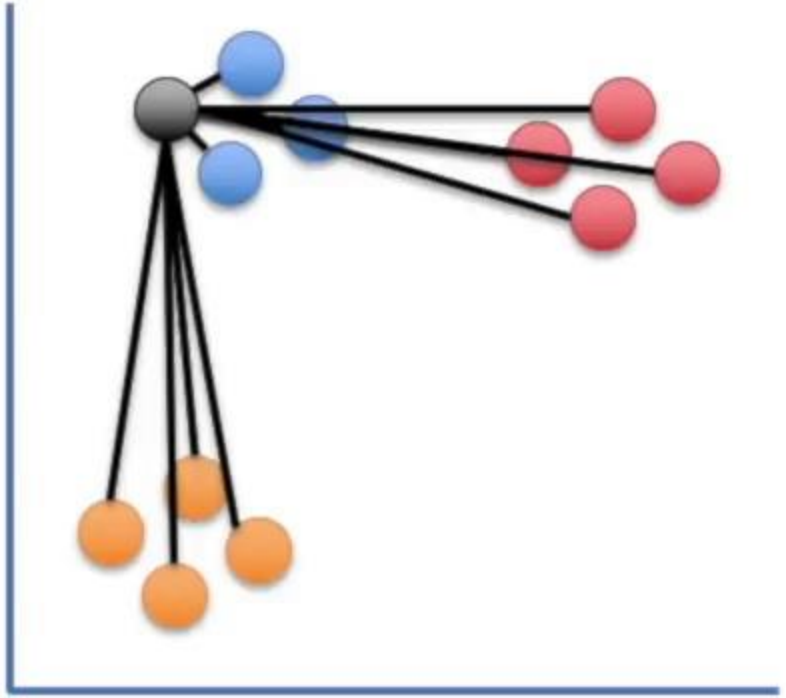
Etc. etc…



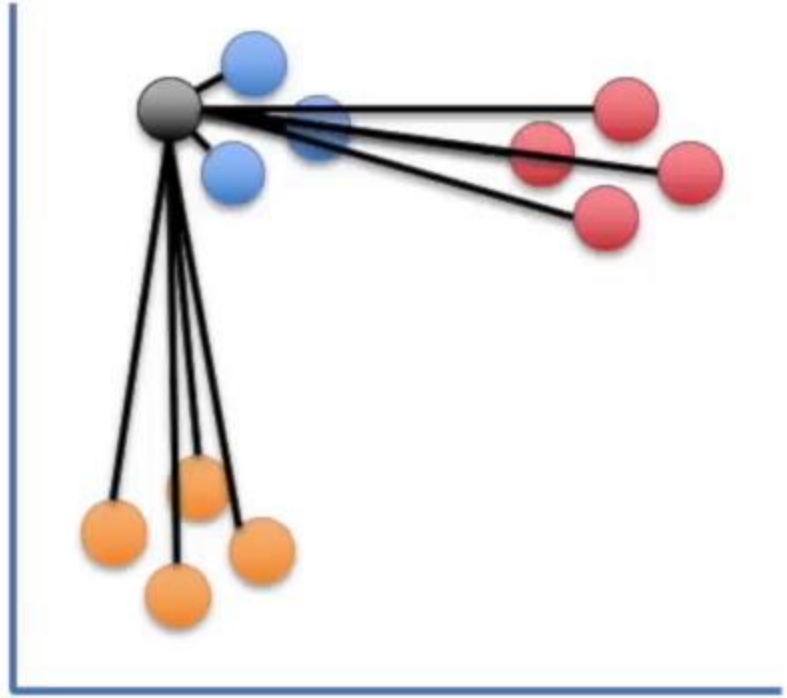
Using a normal distribution means that distant points have very low similarity values….


… and close points have high
similarity values.

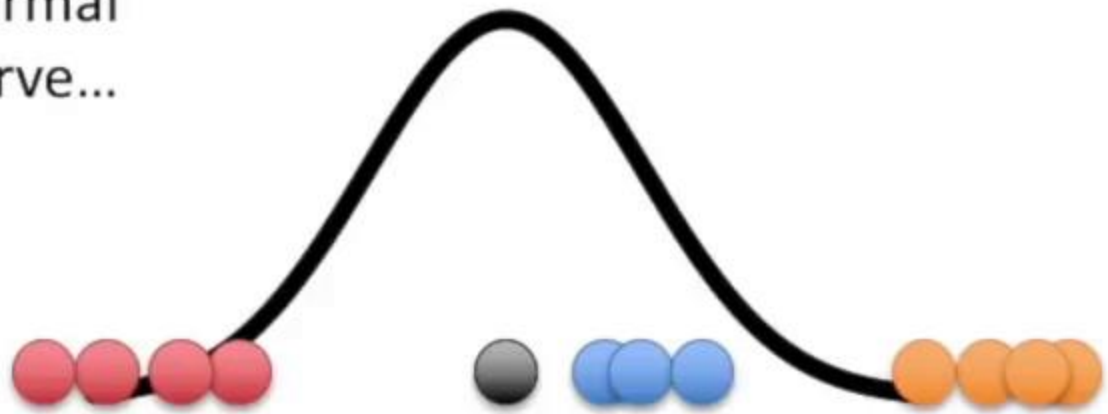
Ultimately, we measure the distances between all of the points and the point of interest…



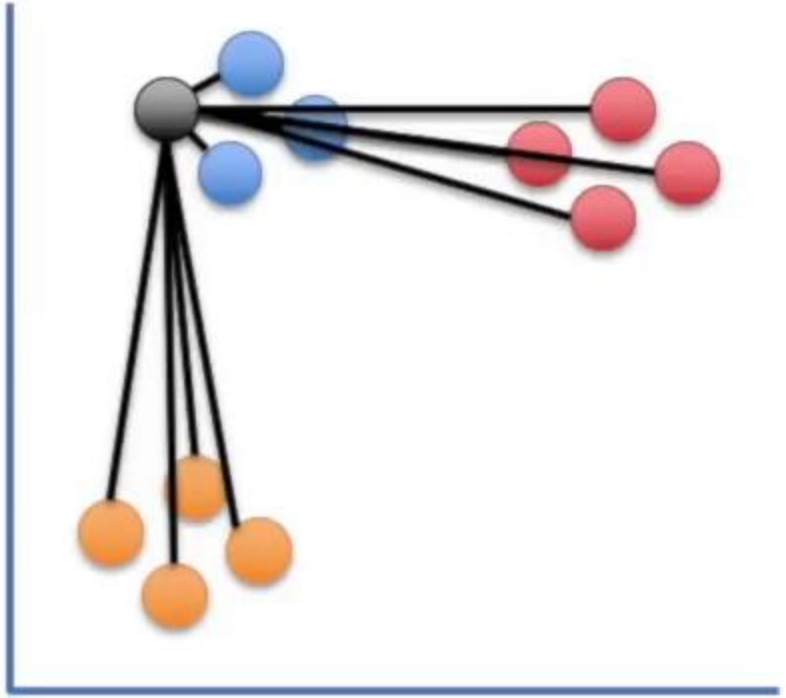
Ultimately, we measure the distances between all of the points and the point of interest…



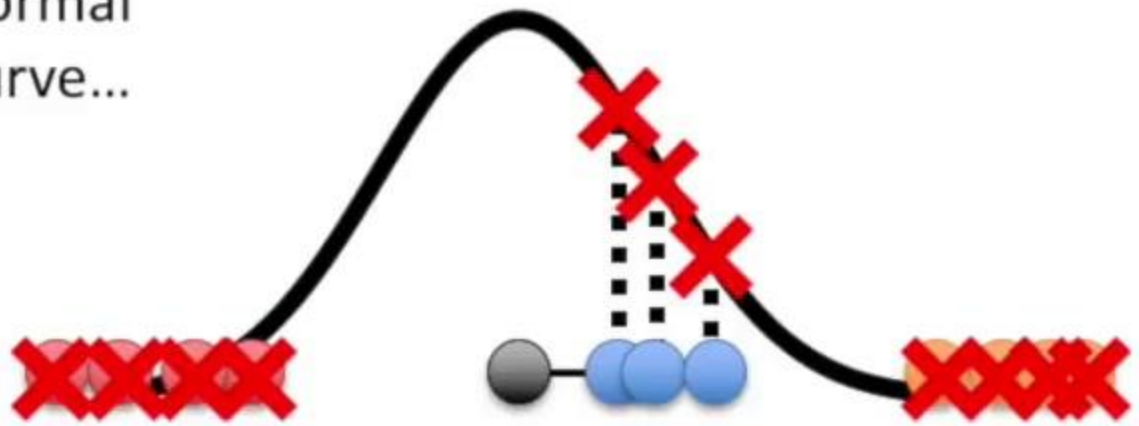
Plot them on the normal curve…



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

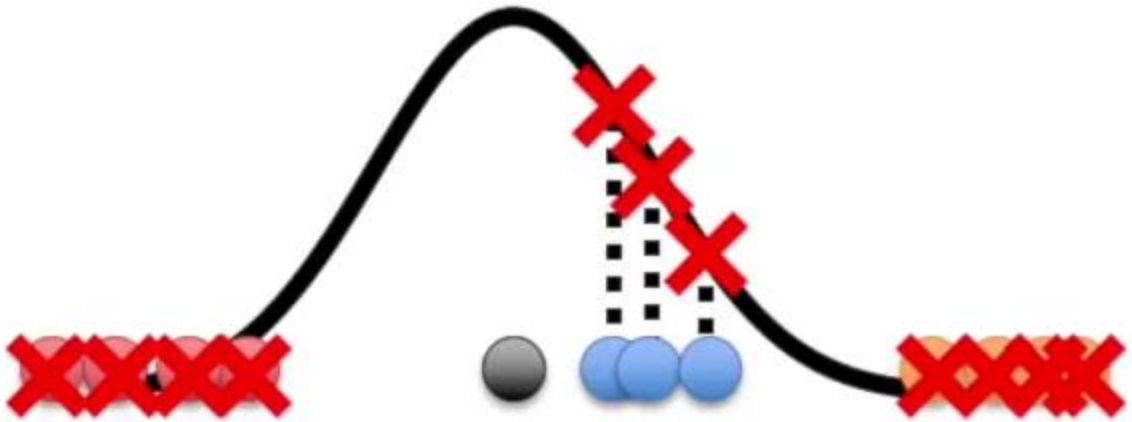


Plot them on the normal curve…



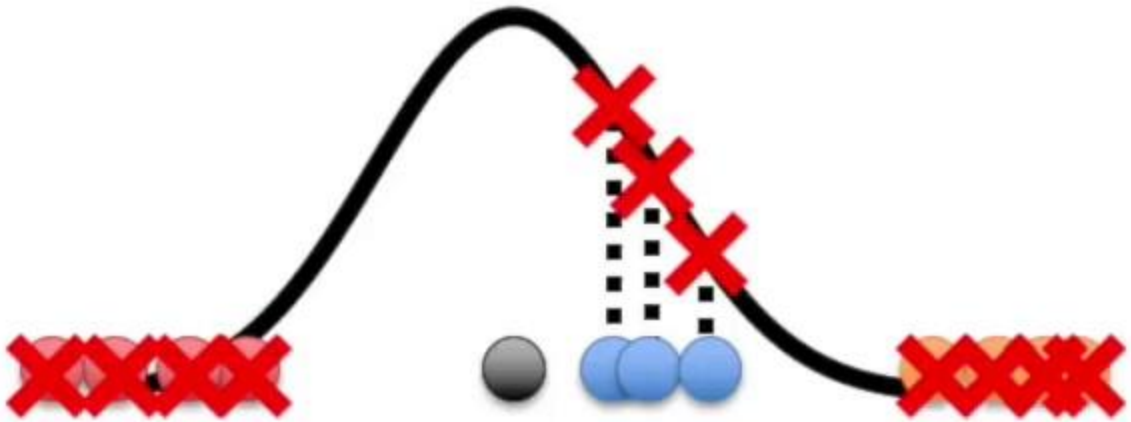
…and then measure the distances from the points to the curve to get the unscaled similarity scores with respect to the point of interest.

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

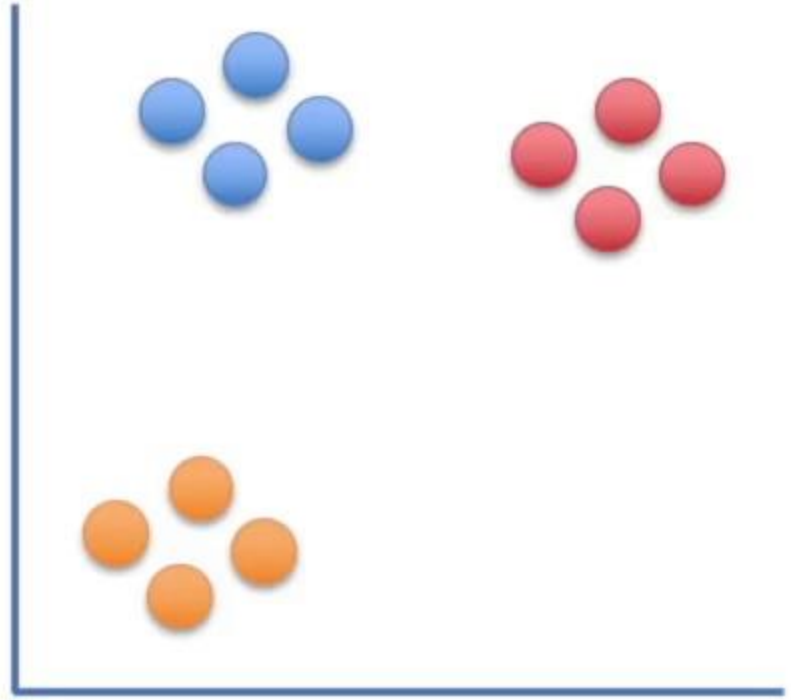


The next step is to scale the unscaled similarities so that they add up to 1.

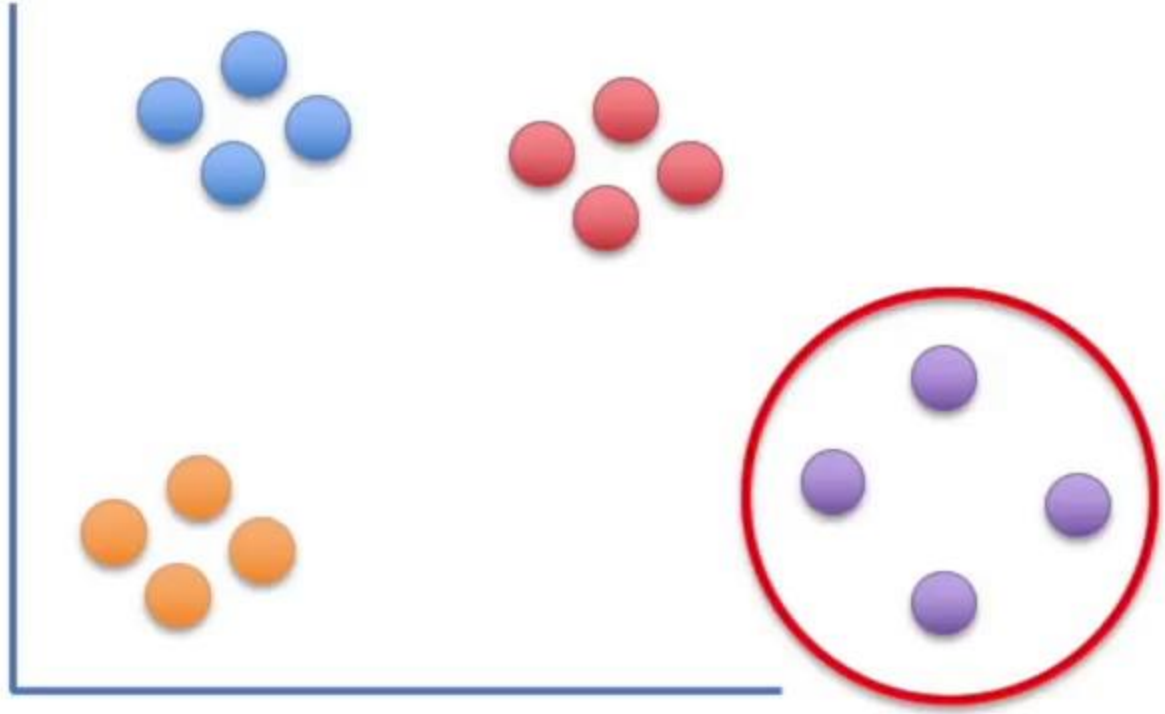
Umm… Why do the similarity scores need to add up to 1?



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

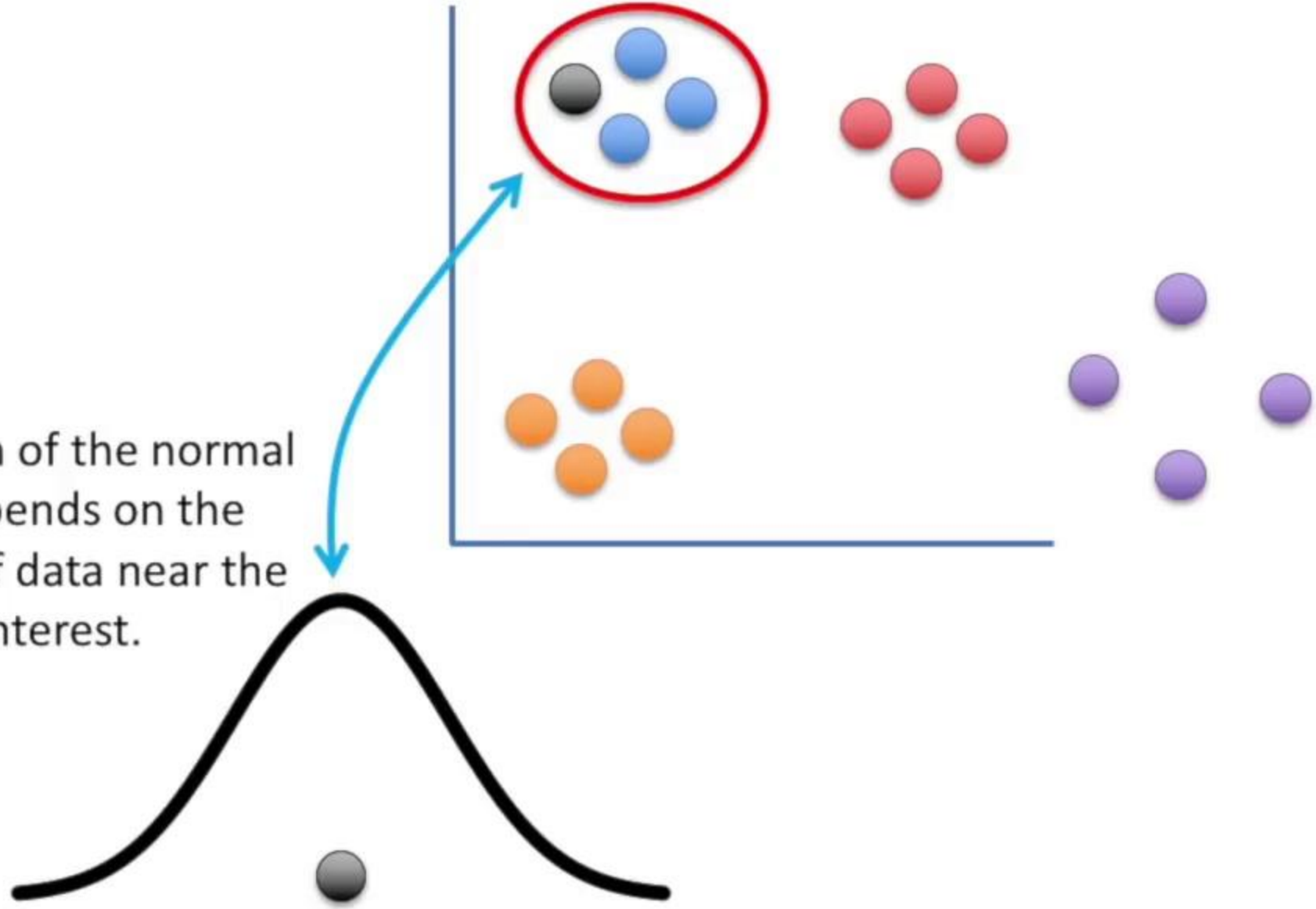


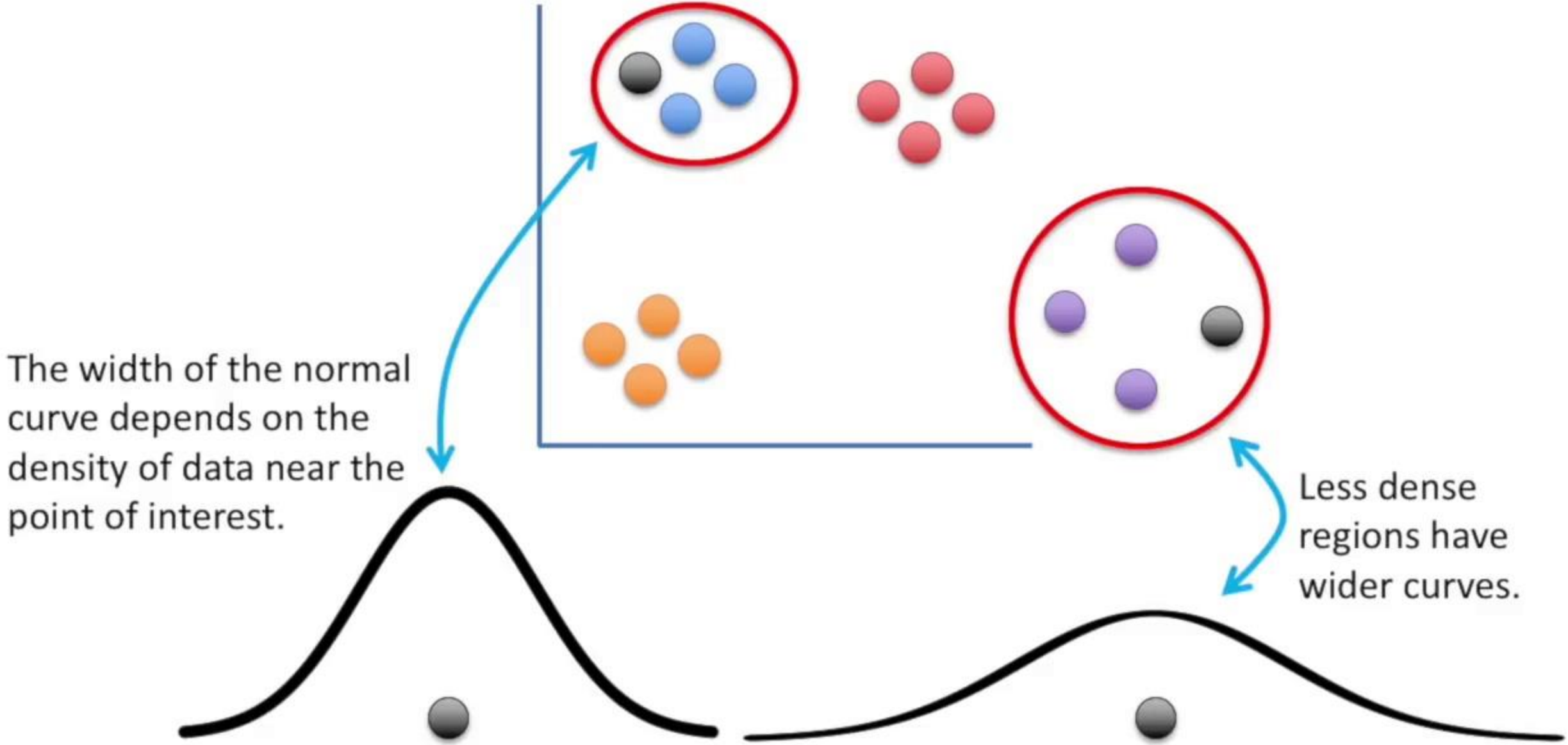
It has to do with something
I didn't tell you earlier…

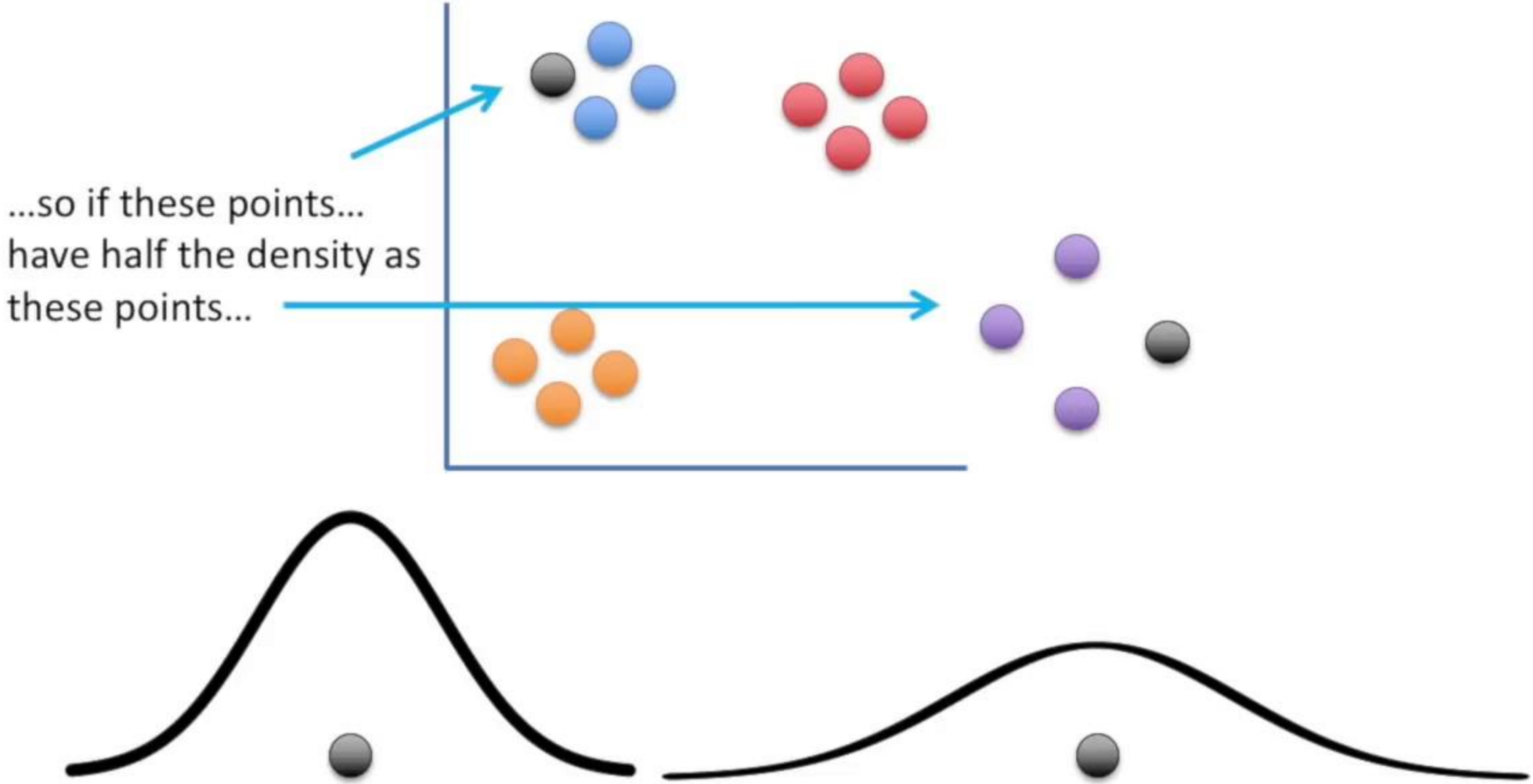


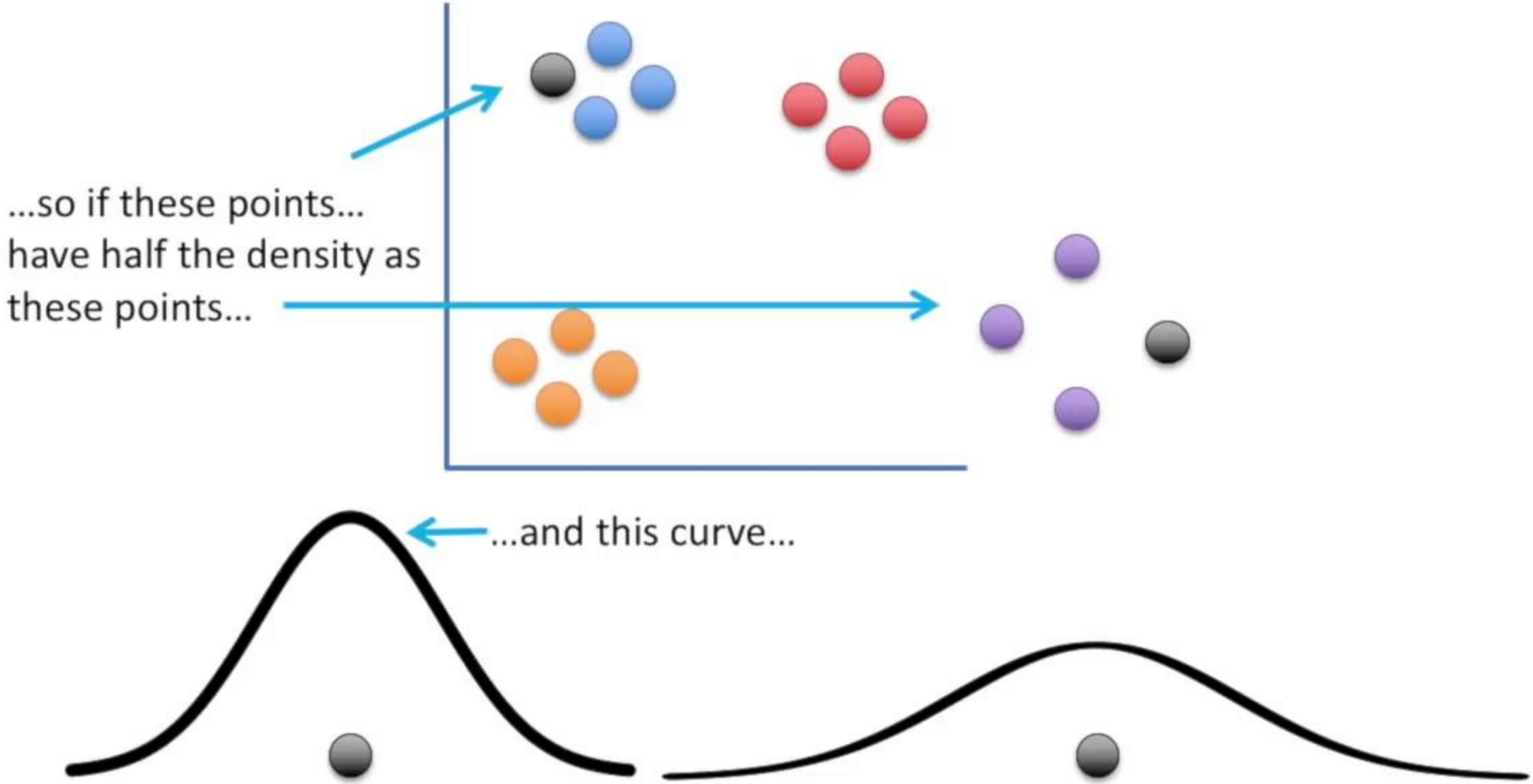
…and to illustrate the concept, I need to add a cluster that is half as dense as the others.

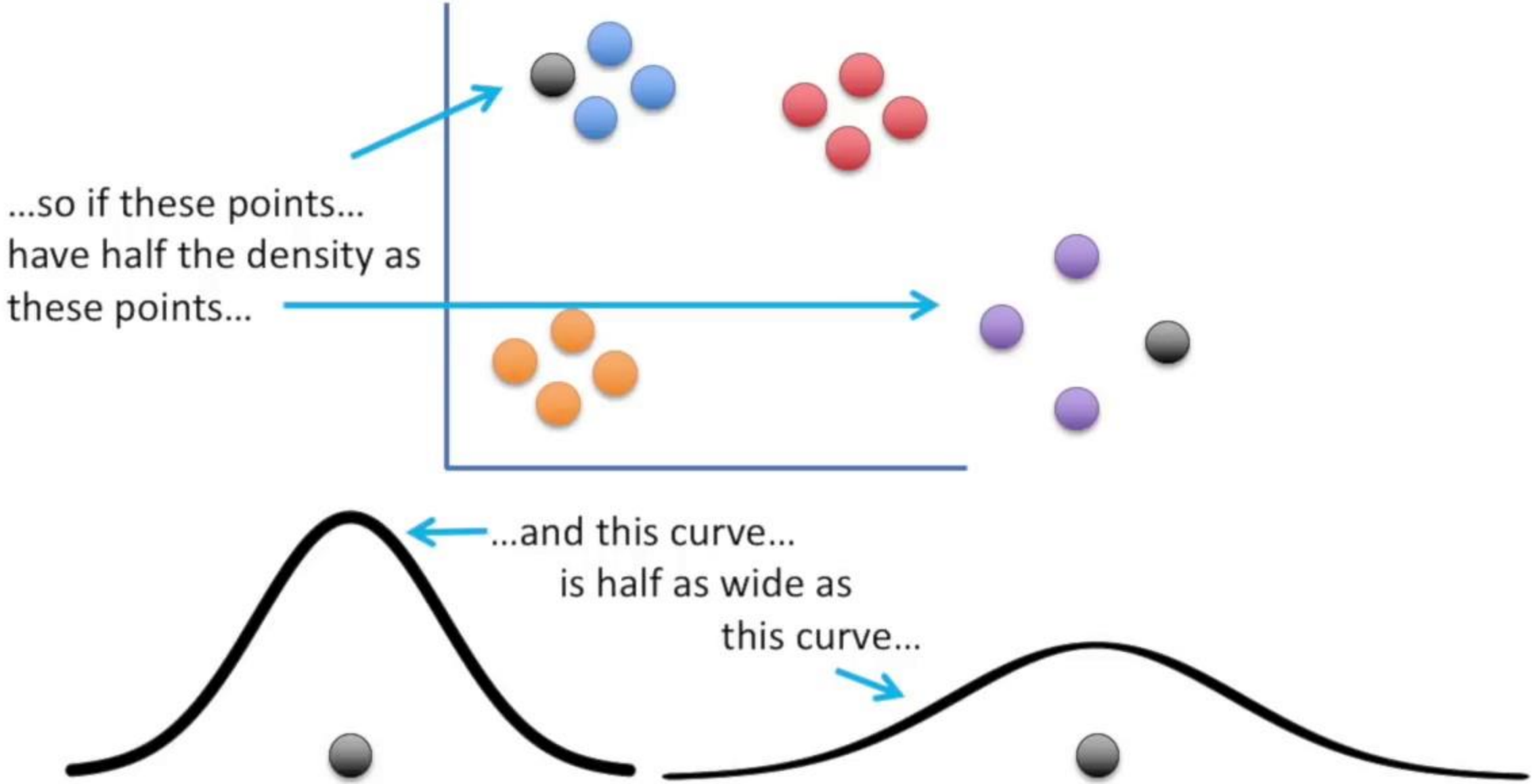
The width of the normal curve depends on the density of data near the point of interest.

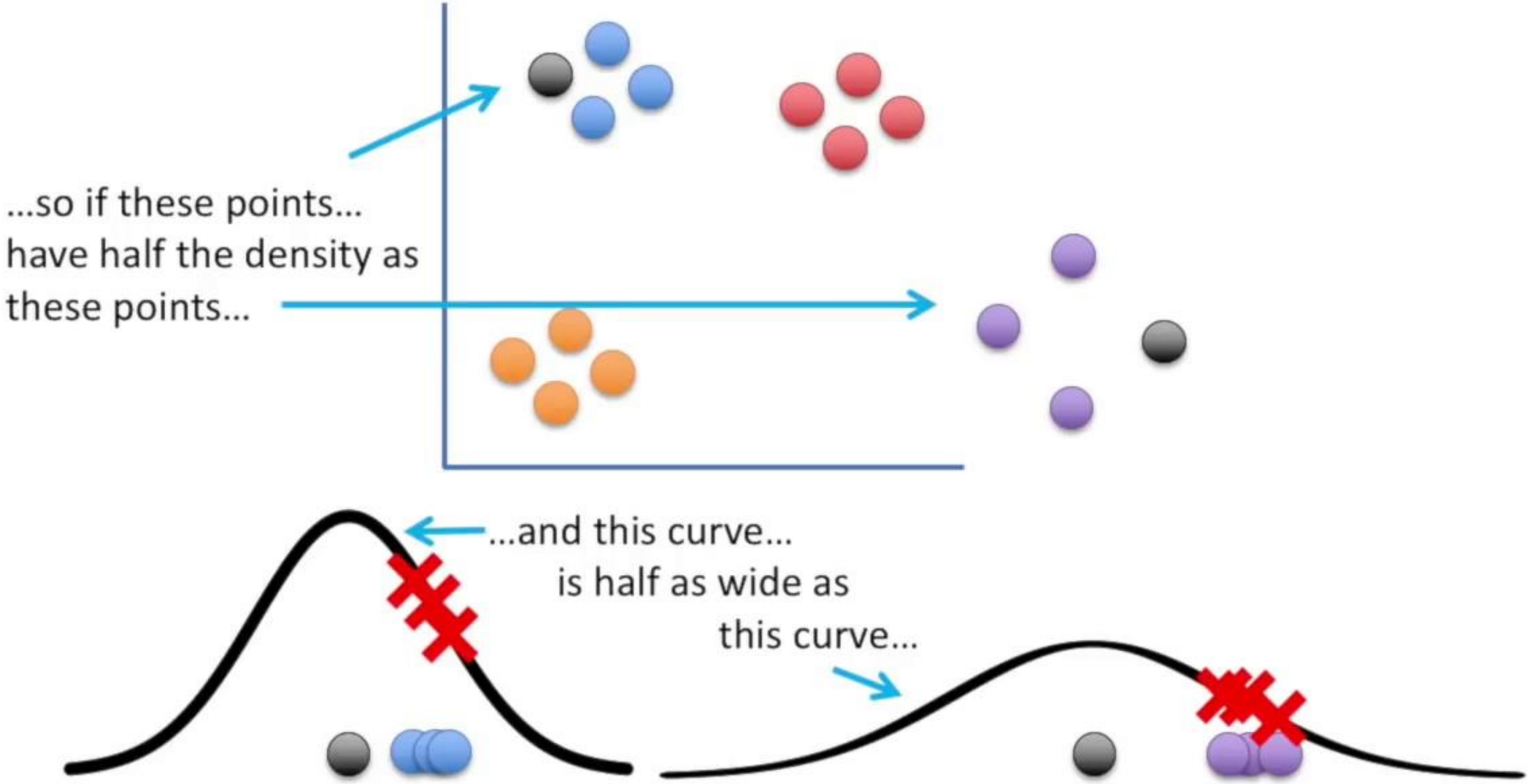



The width of the normal curve depends on the density of data near the point of interest.
Less dense regions have wider curves.


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

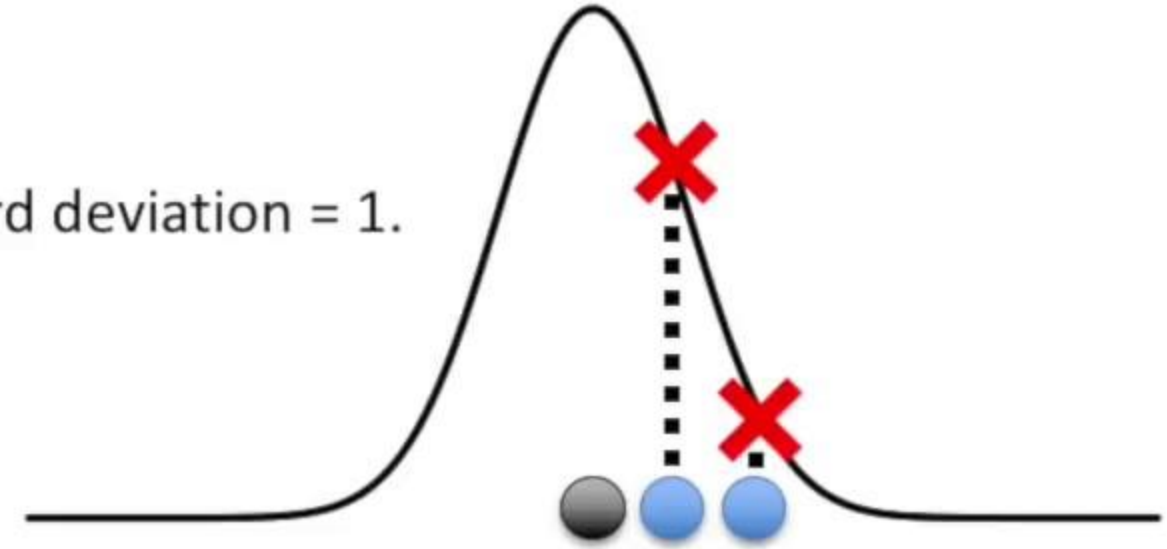

…so if these points…
have half the density as
these points…
…and this curve…


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


...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.

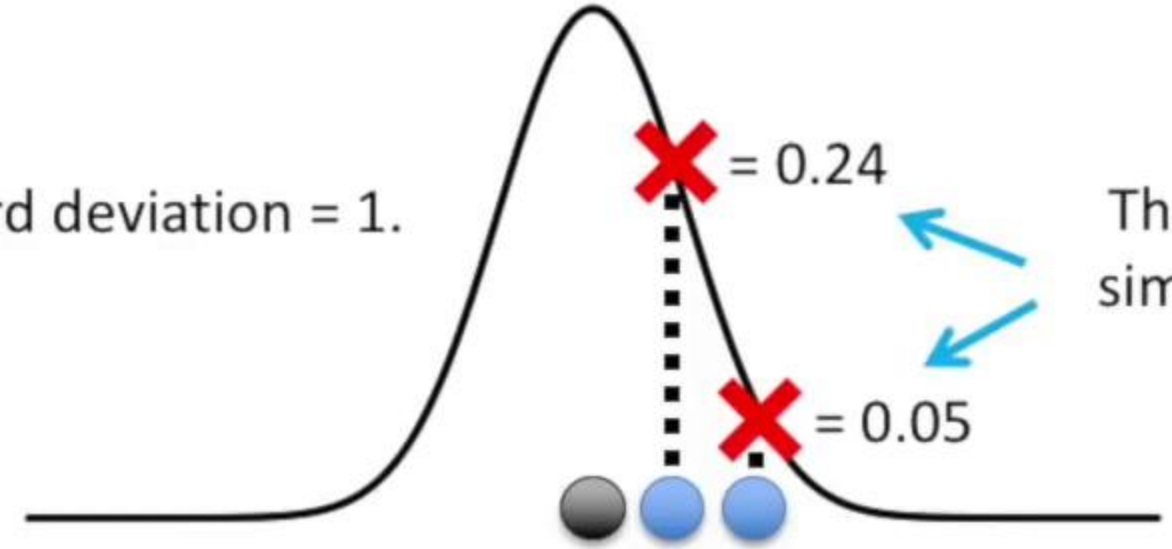
Here's an example…

This curve has a standard deviation = 1.



Here's an example…

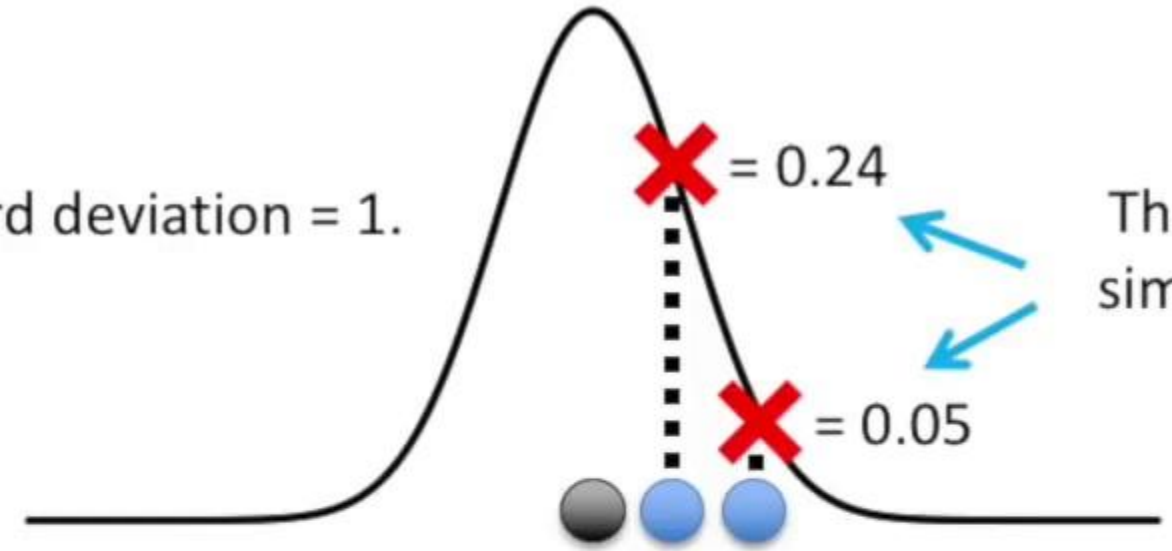
This curve has a standard deviation = 1.



The "unscaled" similarity values

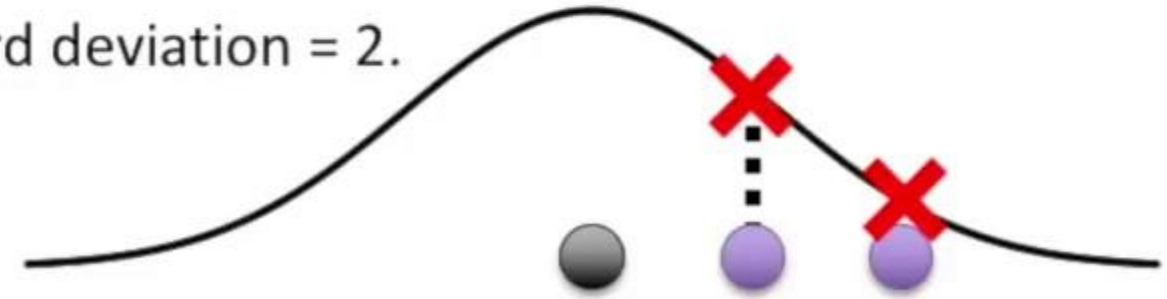
Here's an example…

This curve has a standard deviation = 1.



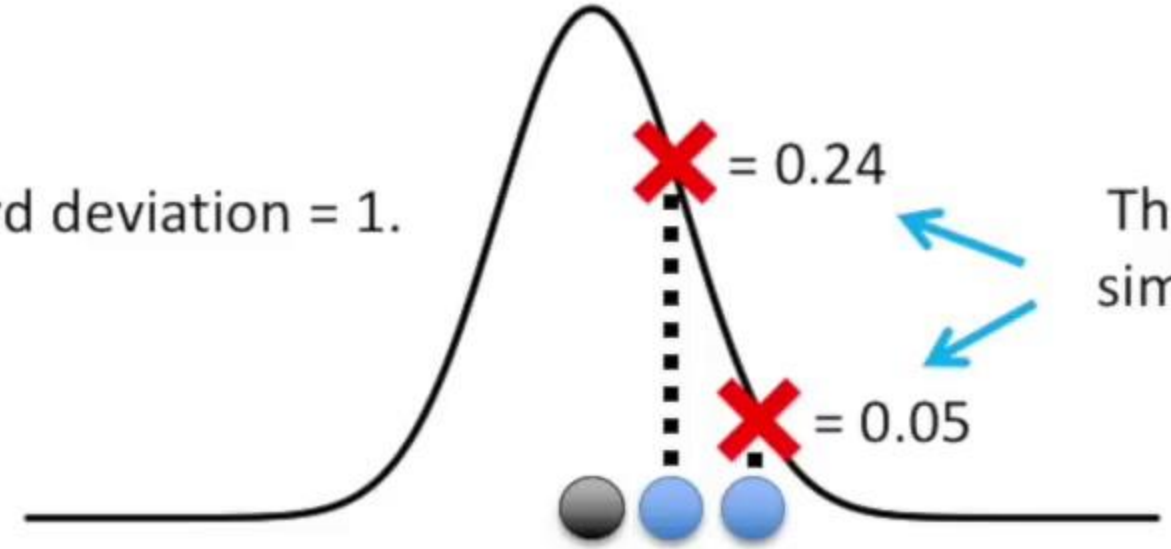
The "unscaled" similarity values

This curve has a standard deviation = 2.



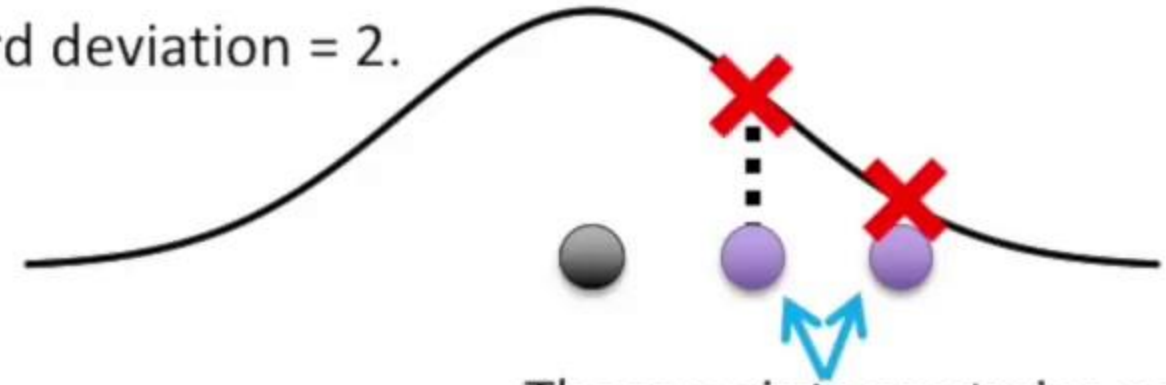
Here's an example…

This curve has a standard deviation = 1.



The "unscaled" similarity values

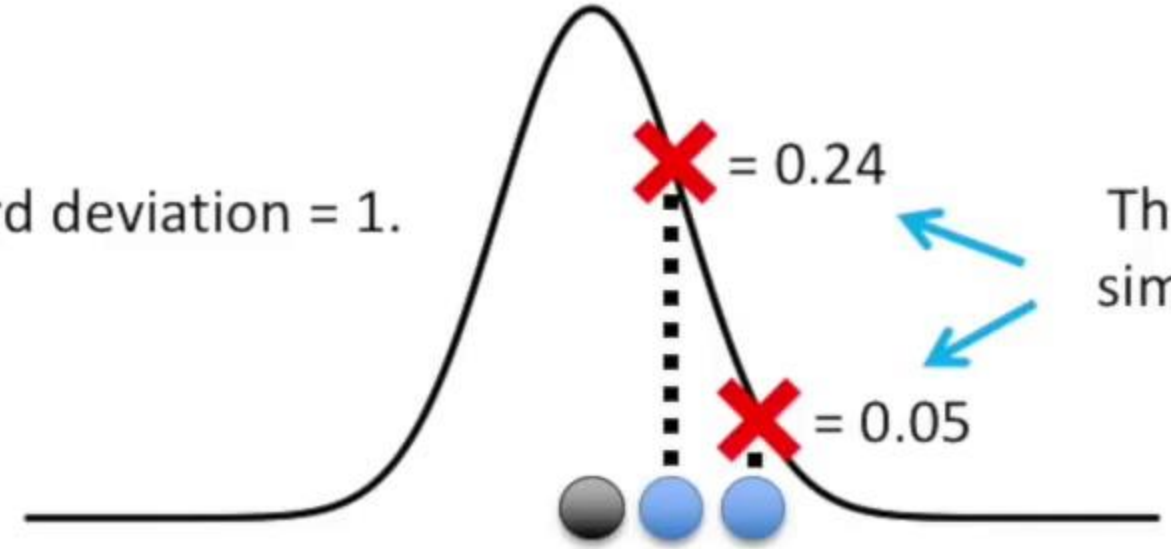
This curve has a standard deviation = 2.



These points are twice as far from the middle.

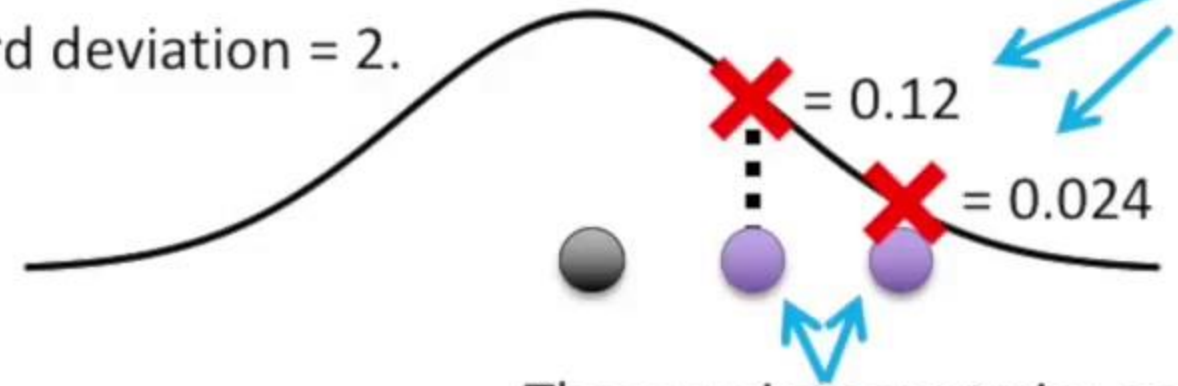
Here's an example…

This curve has a standard deviation = 1.



The "unscaled" similarity values

This curve has a standard deviation = 2.

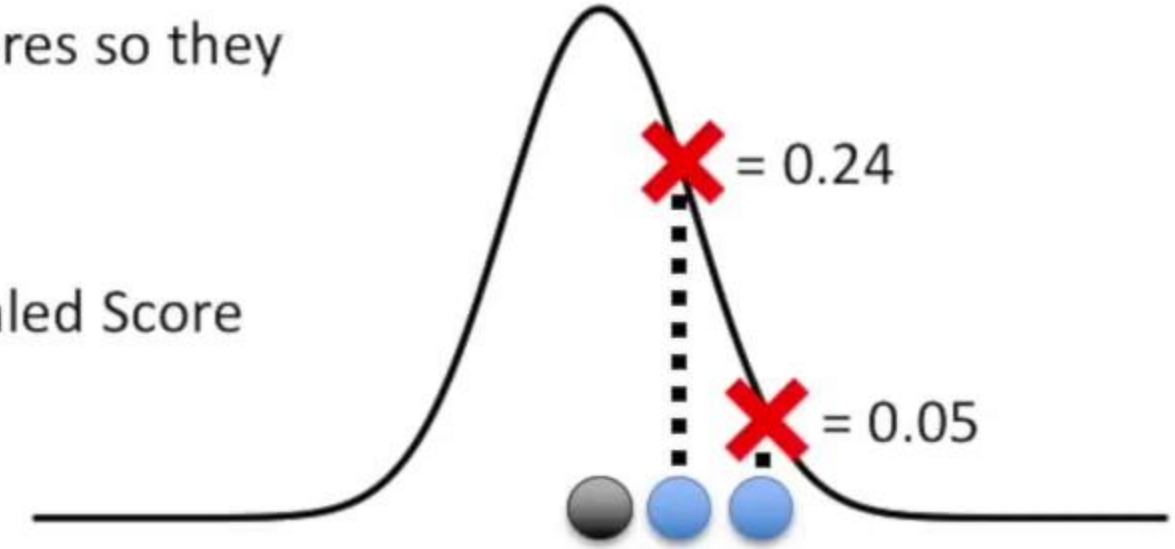


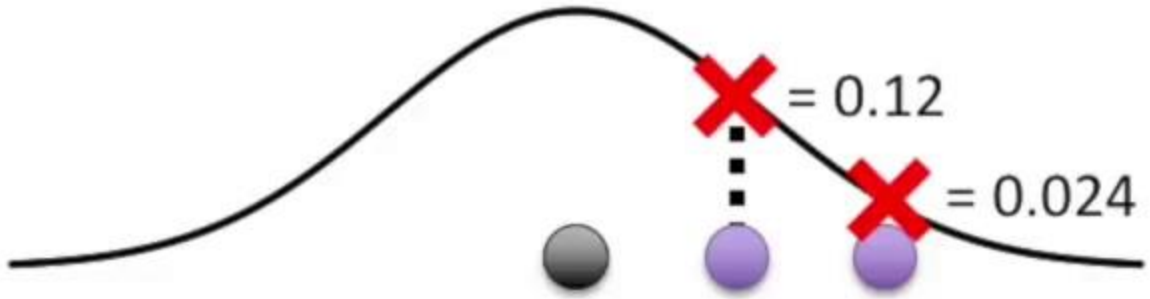
The "unscaled" similarity values are half of the other ones.

These points are twice as far from the middle.

To scale the similarity scores so they sum to 1:

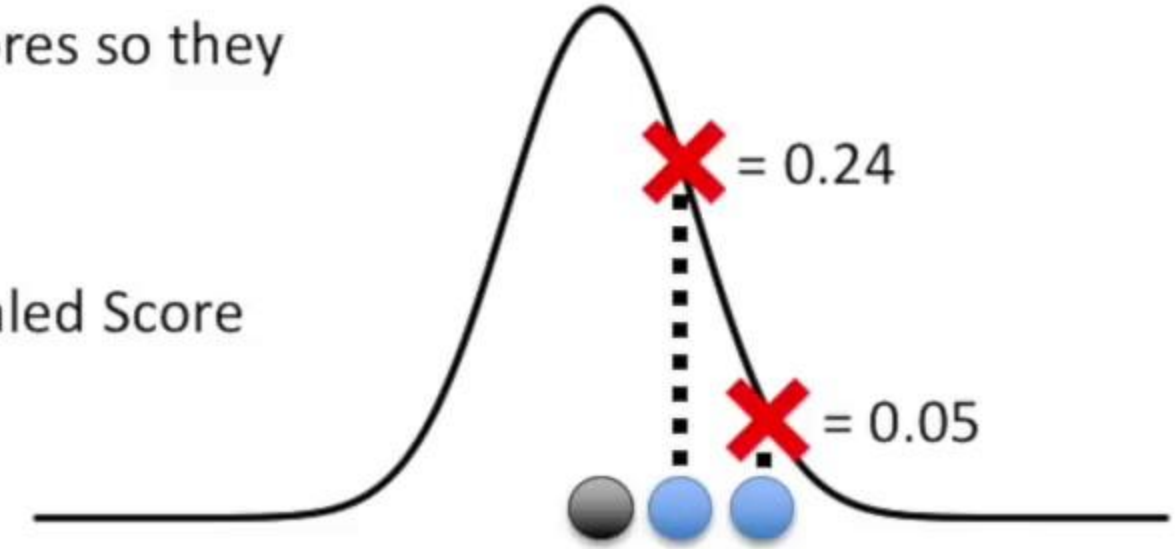
$$\frac{\text{Score}}{\text{Sum of all scores}} = \text{Scaled Score}$$





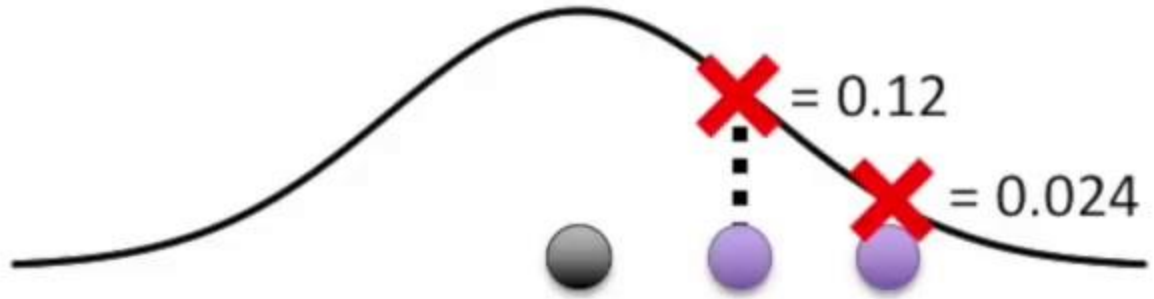
To scale the similarity scores so they sum to 1:

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

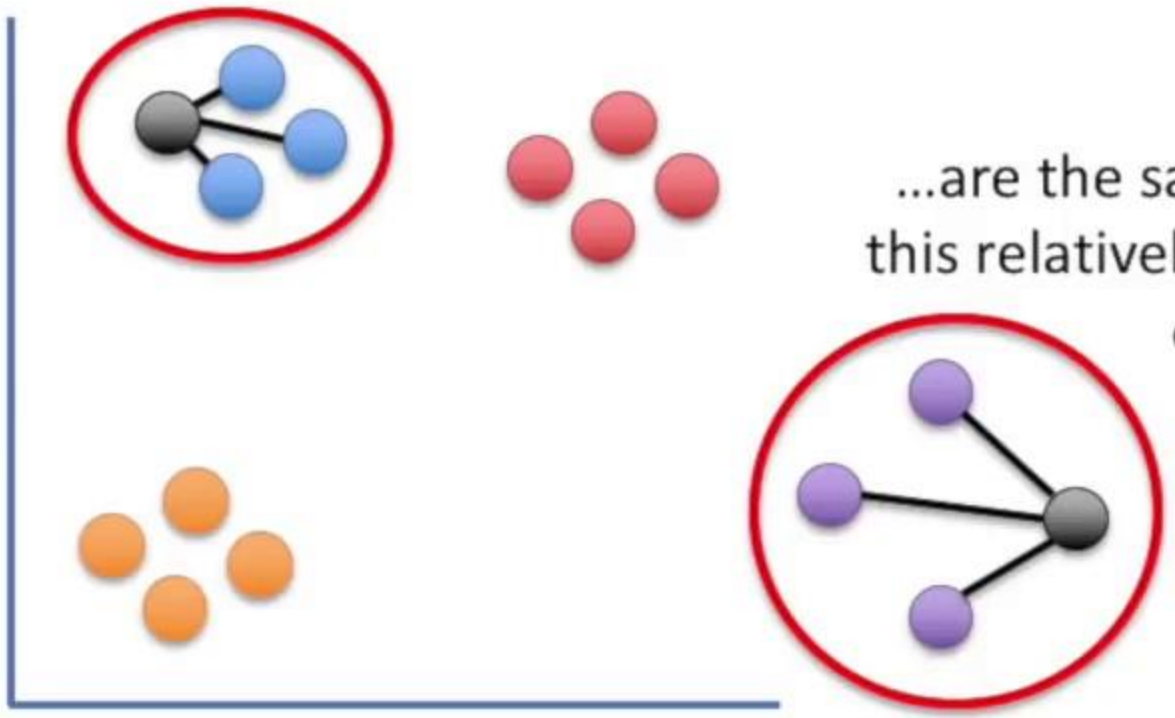


$$\frac{0.24}{0.24 + 0.5} = 0.82$$

$$\frac{0.05}{0.24 + 0.5} = 0.18$$



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



…are the same for this relatively loose cluster!

$$\frac{0.24}{0.24 + 0.5} = 0.82$$

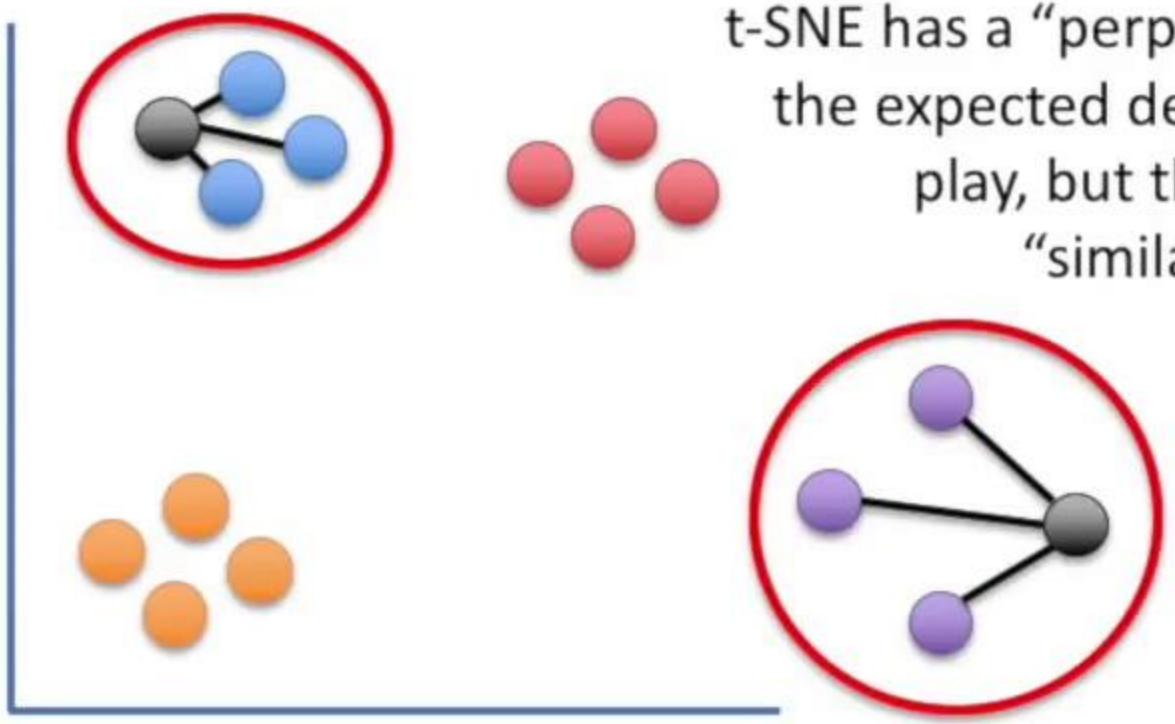
$$\frac{0.05}{0.24 + 0.5} = 0.18$$

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

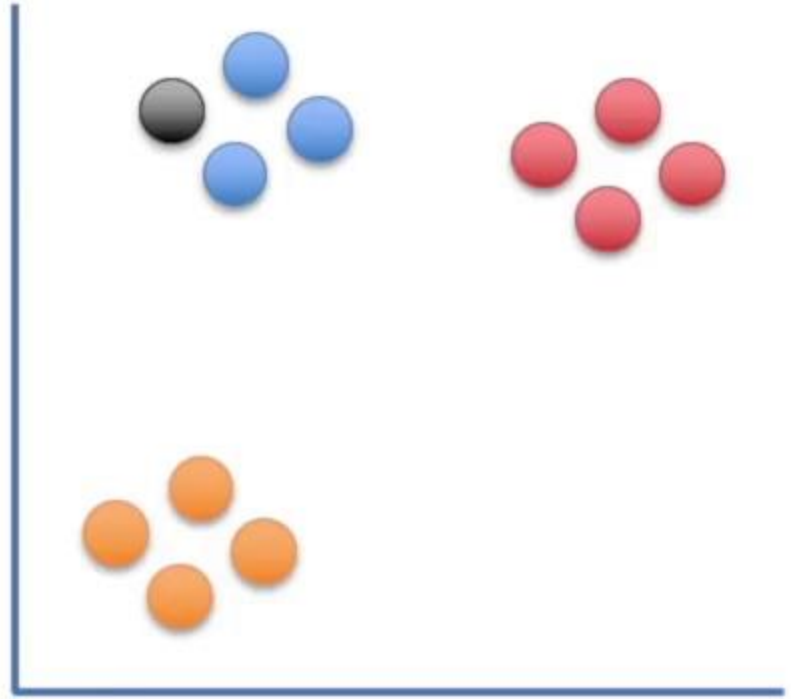
$$\frac{0.024}{0.12 + 0.024} = 0.18$$

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

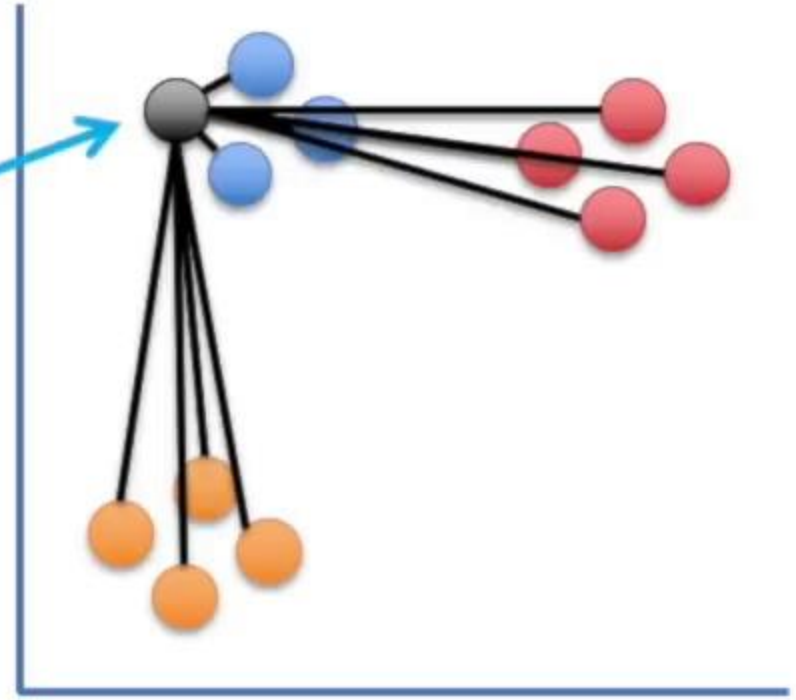
t-SNE has a “perplexity” parameter equal to the expected density, and that comes into play, but these clusters are still more “similar” than you might expect.



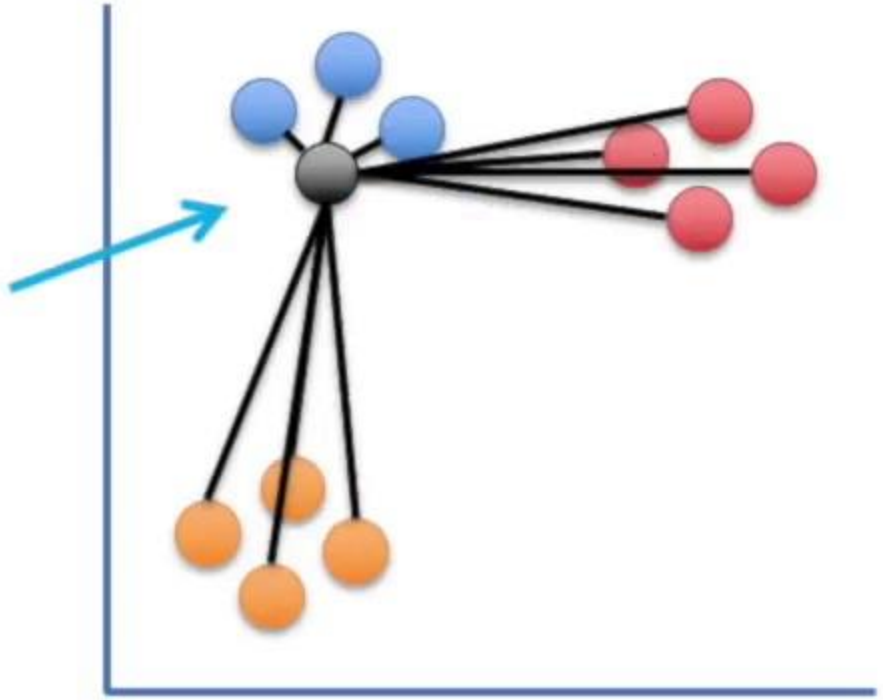
Now back to the original scatter plot…



We've calculated similarity scores for this point.

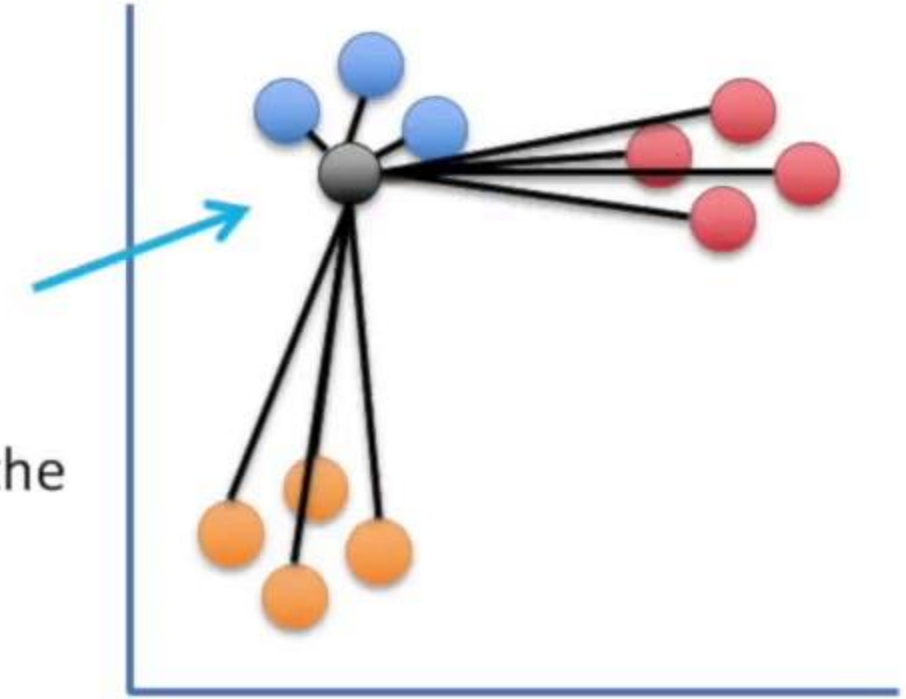


Now we do it for this point…

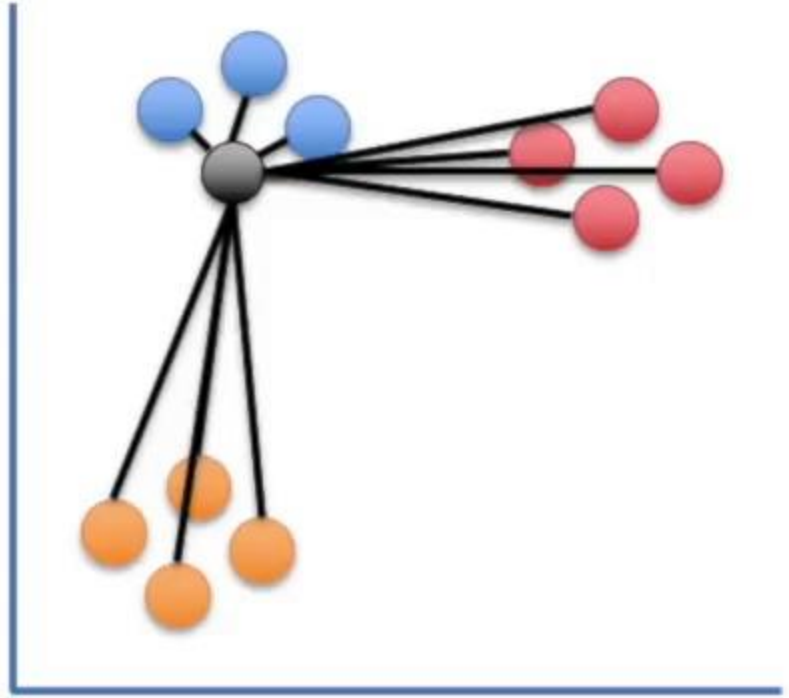


Now we do it for this point…

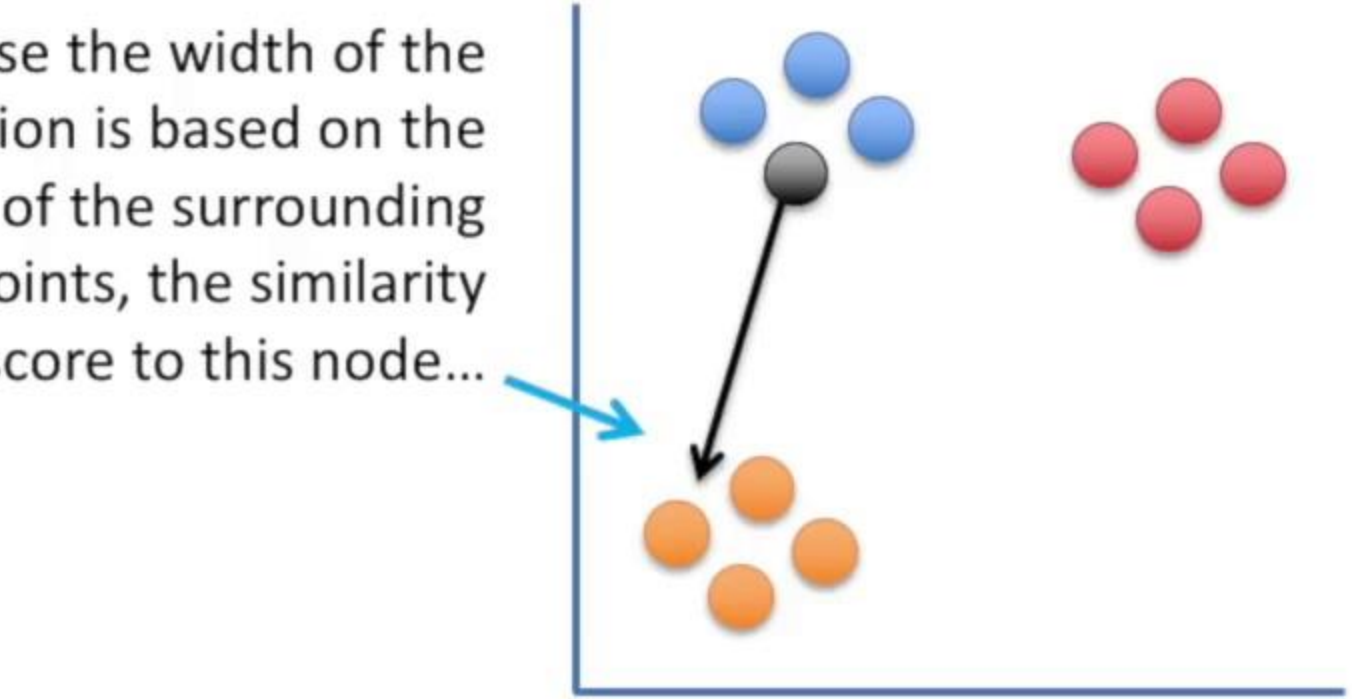
…and we do it for all the points.



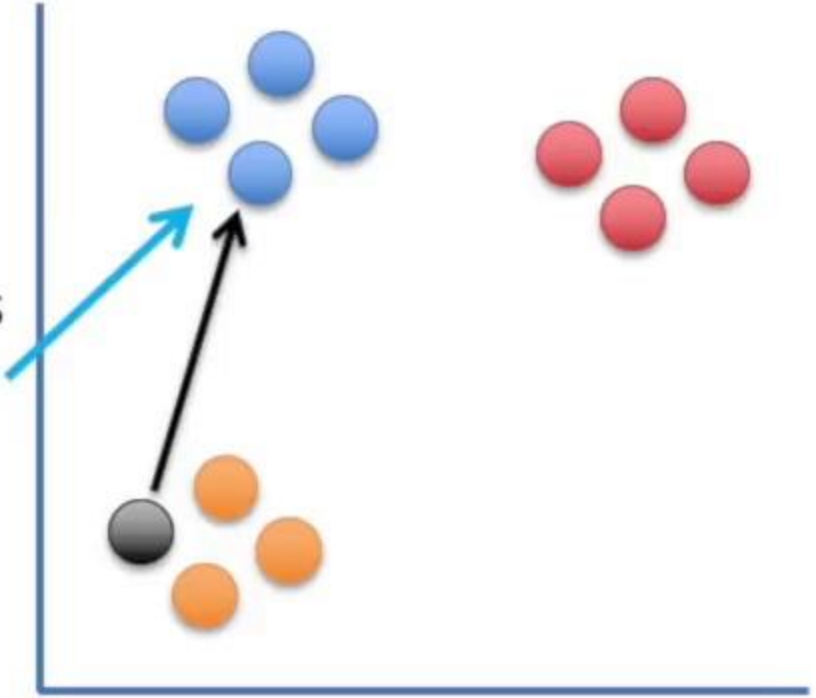
One last thing and the scatter plot will be all set with similarity scores!!!

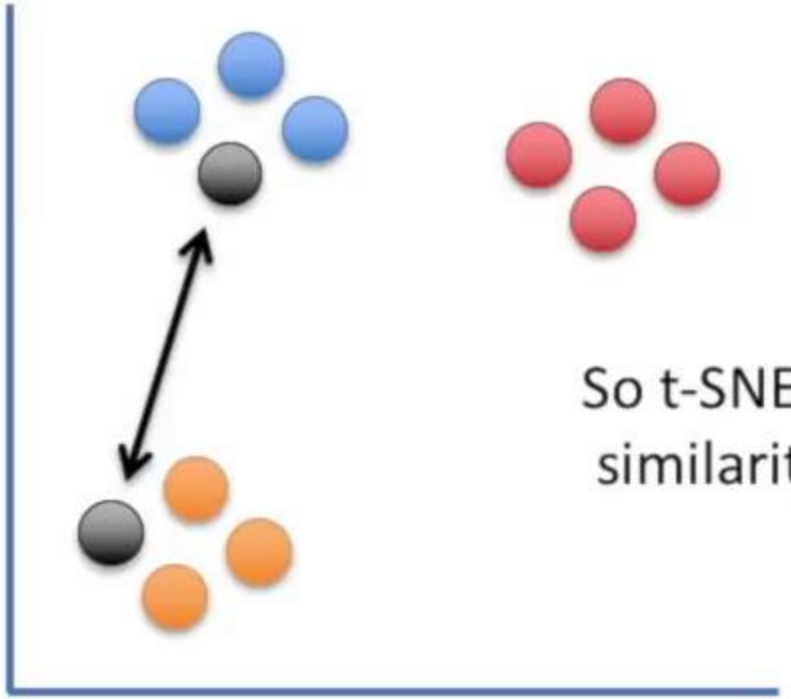


Because the width of the distribution is based on the density of the surrounding data points, the similarity score to this node…

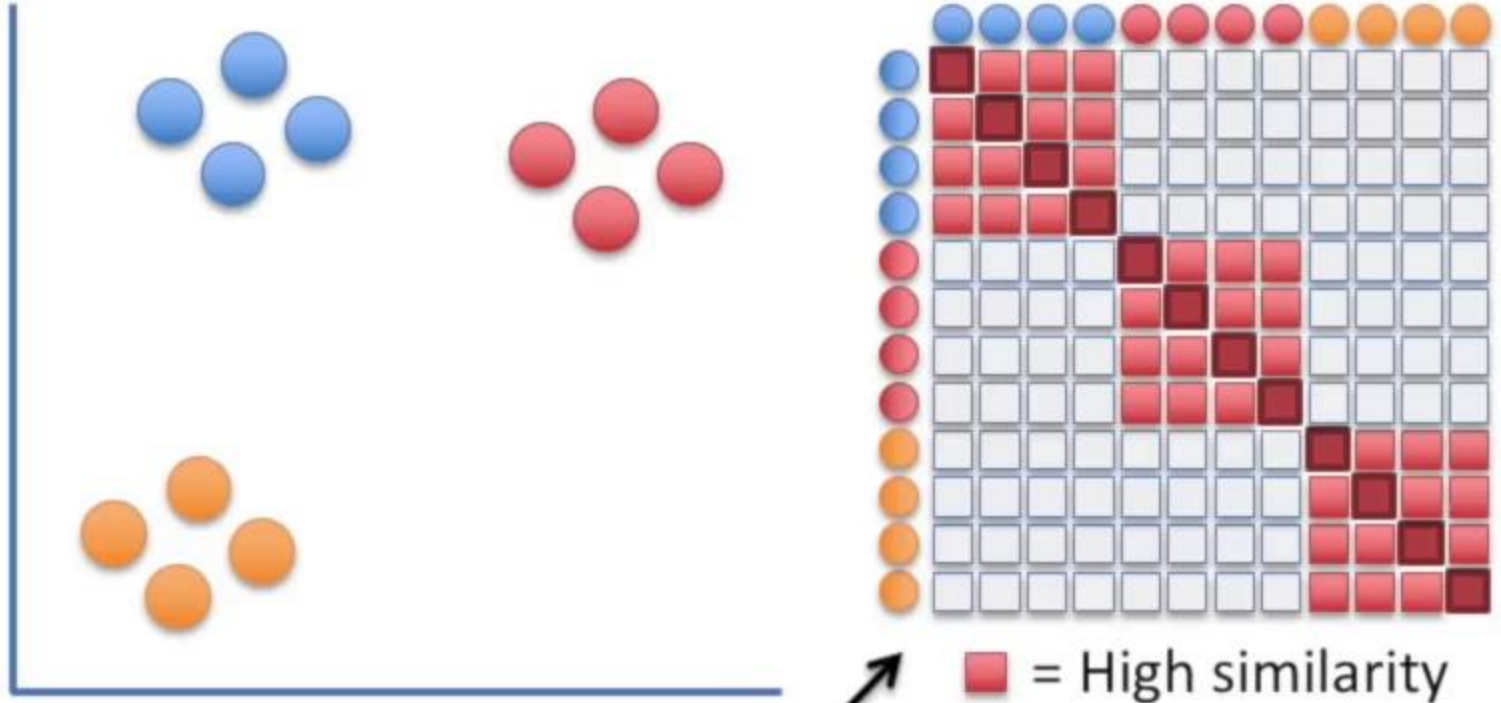


…might not be the same as the similarity to this node.

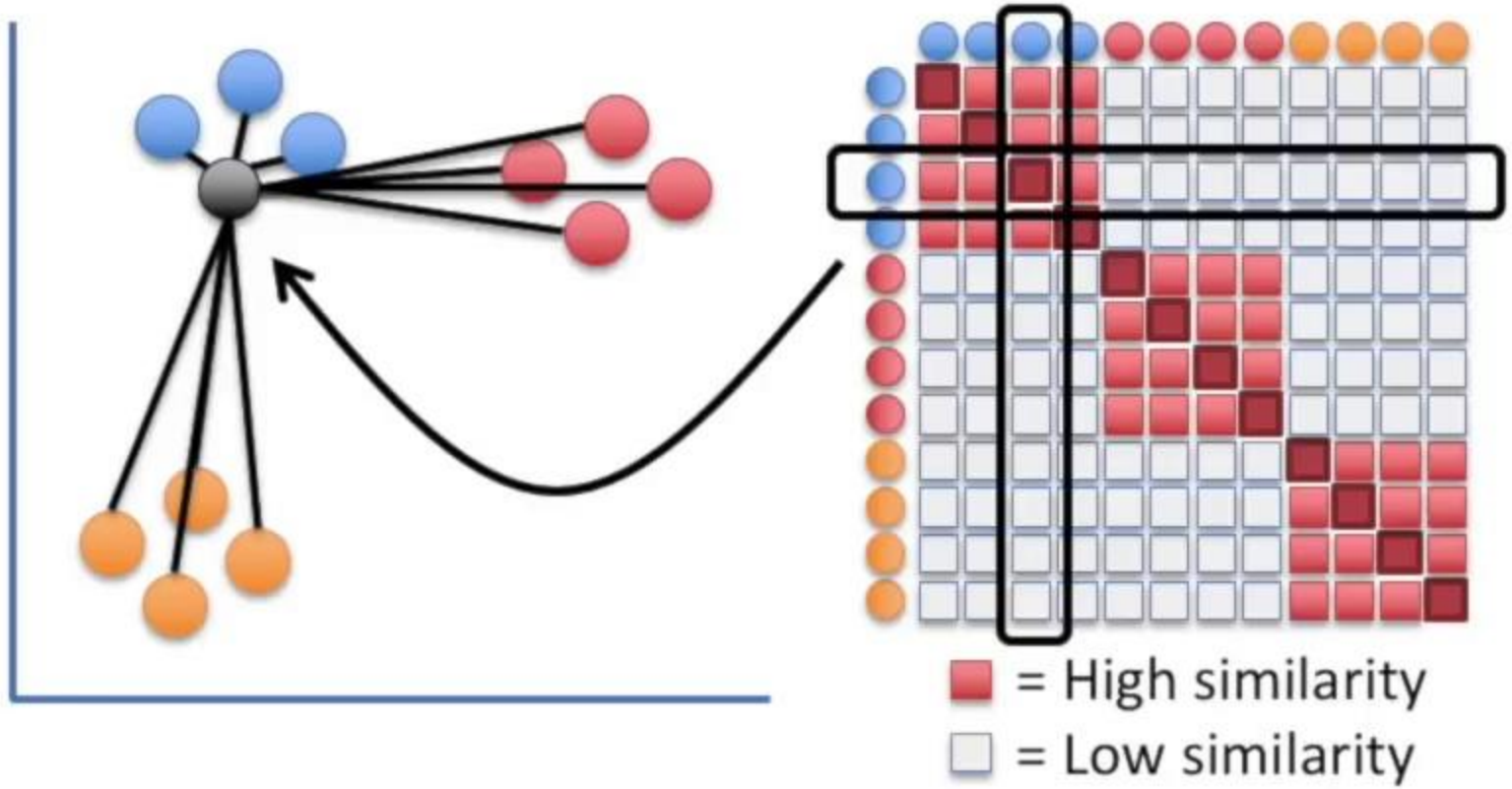




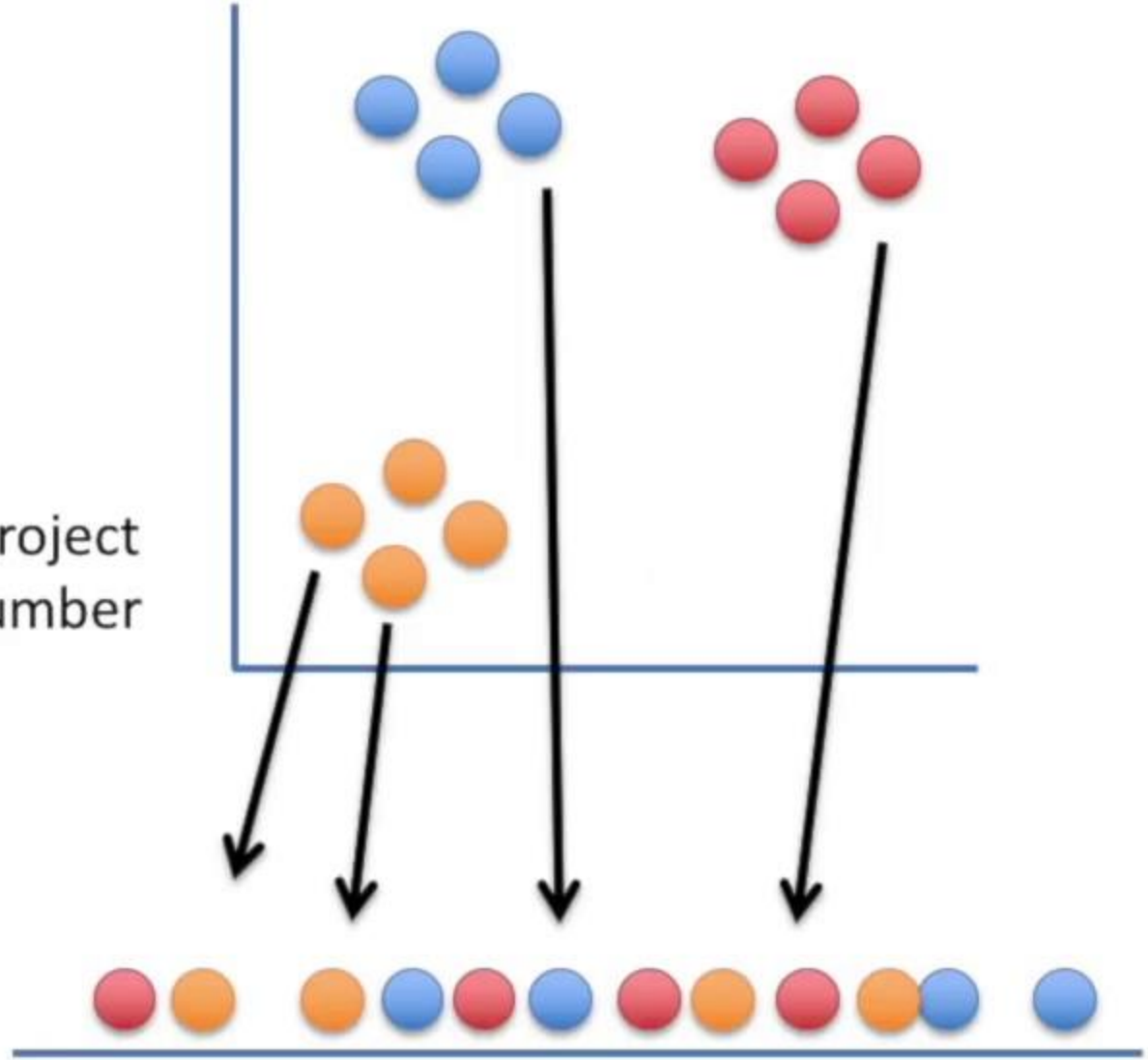
So t-SNE just averages the two similarity scores from the two directions…



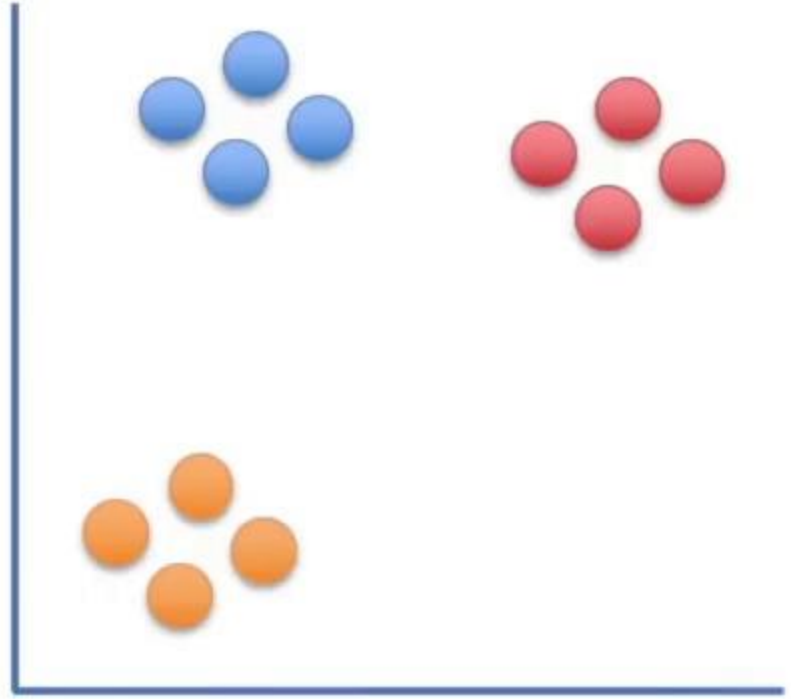
Ultimately, you end up with a matrix of similarity scores.


= High similarity
= Low similarity

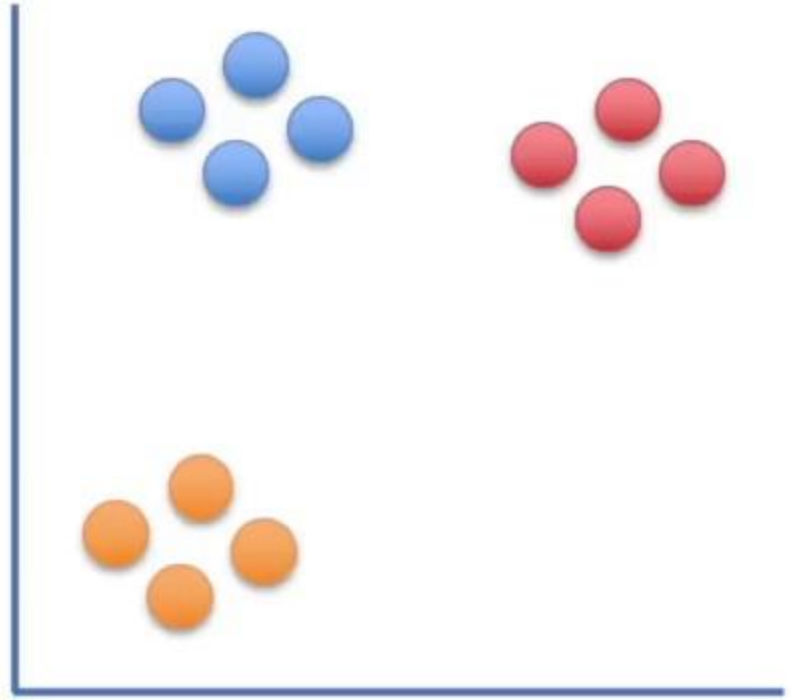
Now we randomly project the data onto the number line…



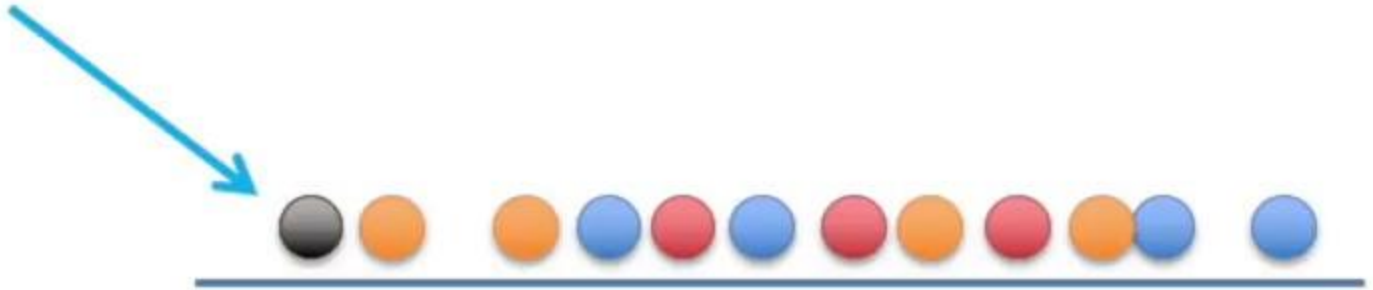
Now we randomly project the data onto the number line…

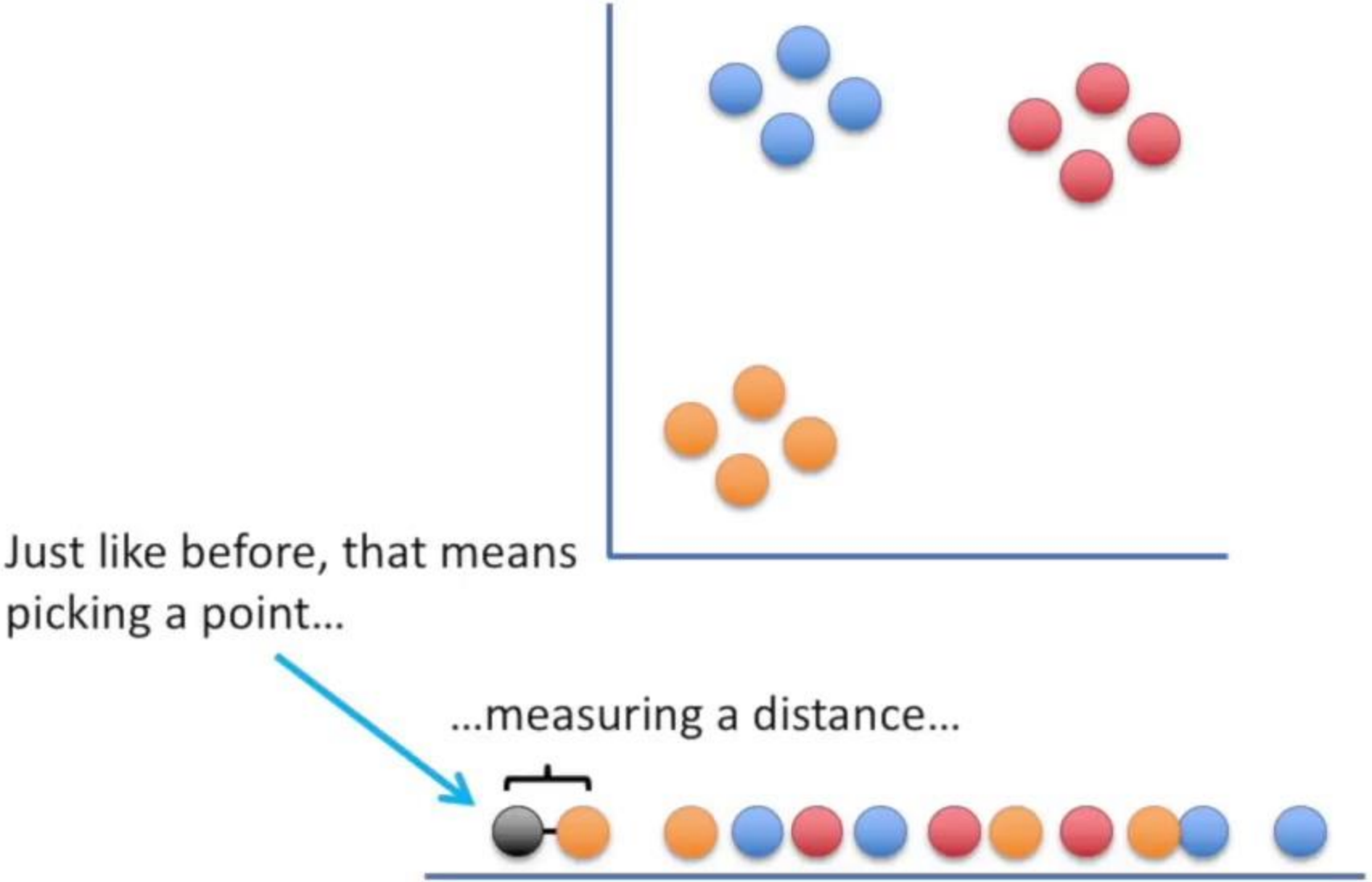


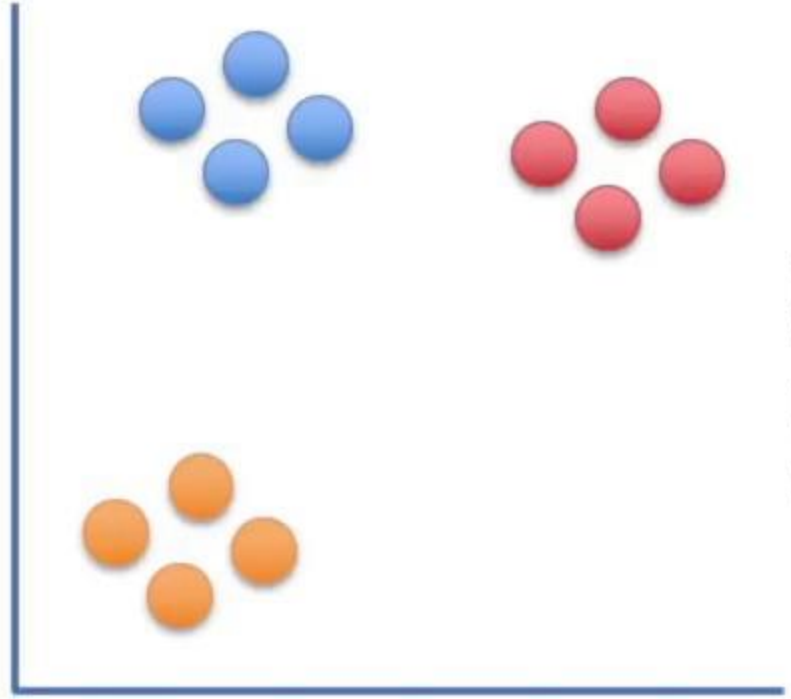
… and calculate similarity scores for the points on the number line.



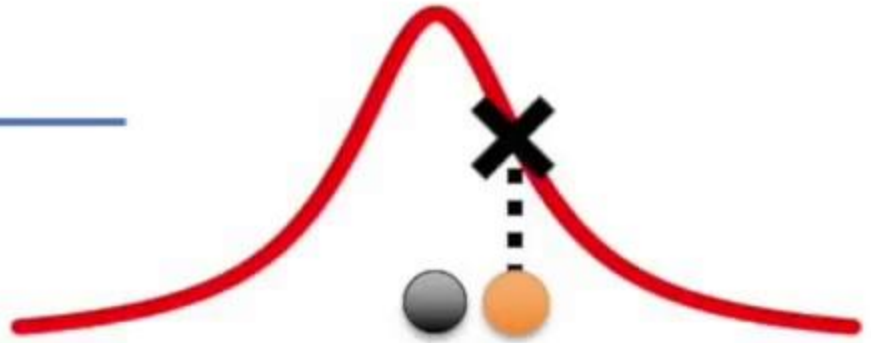
Just like before, that means picking a point…




Just like before, that means picking a point…
…measuring a distance…



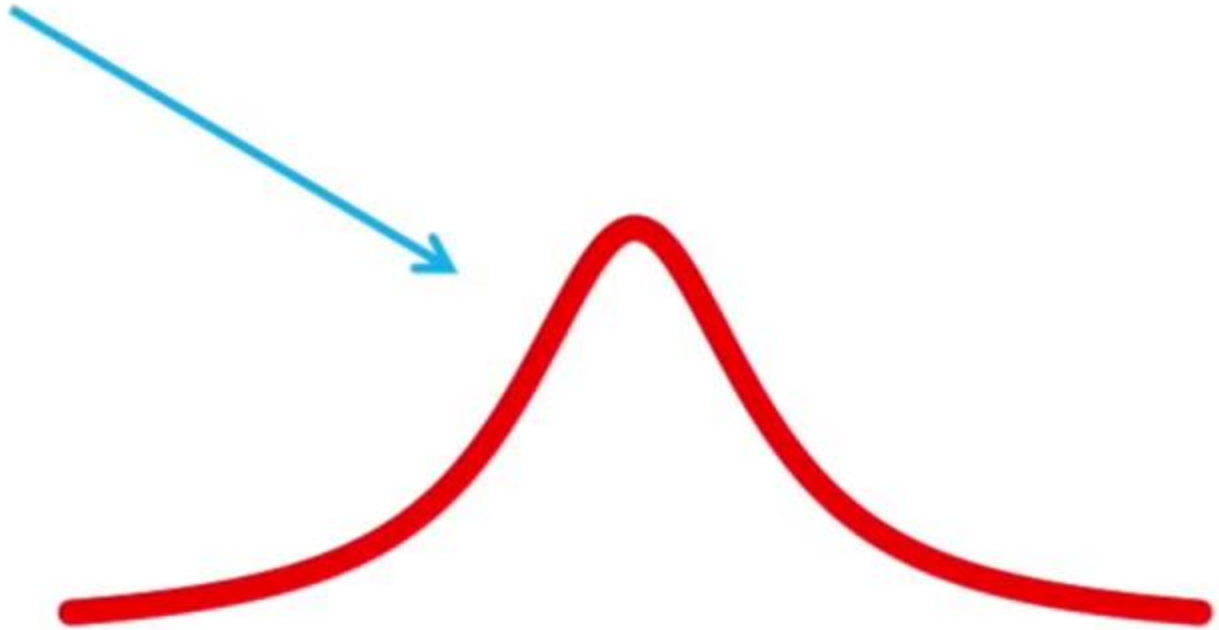
…and lastly, drawing a line from the point to a curve. However, this time we're using a "t-distribution".



Just like before, that means picking a point…

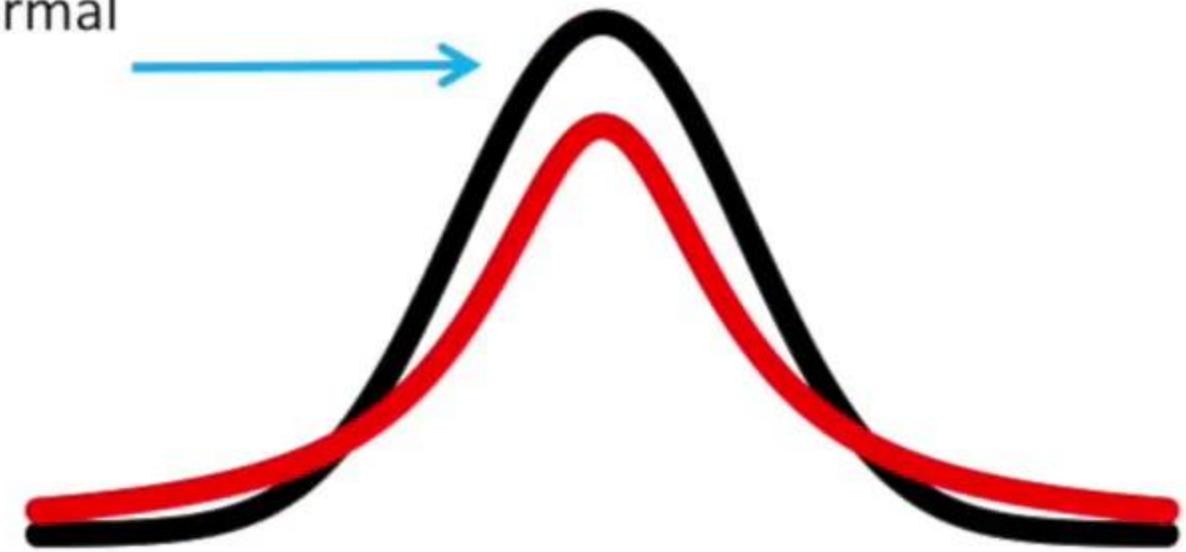
…measuring a distance…

A “t-distribution”…



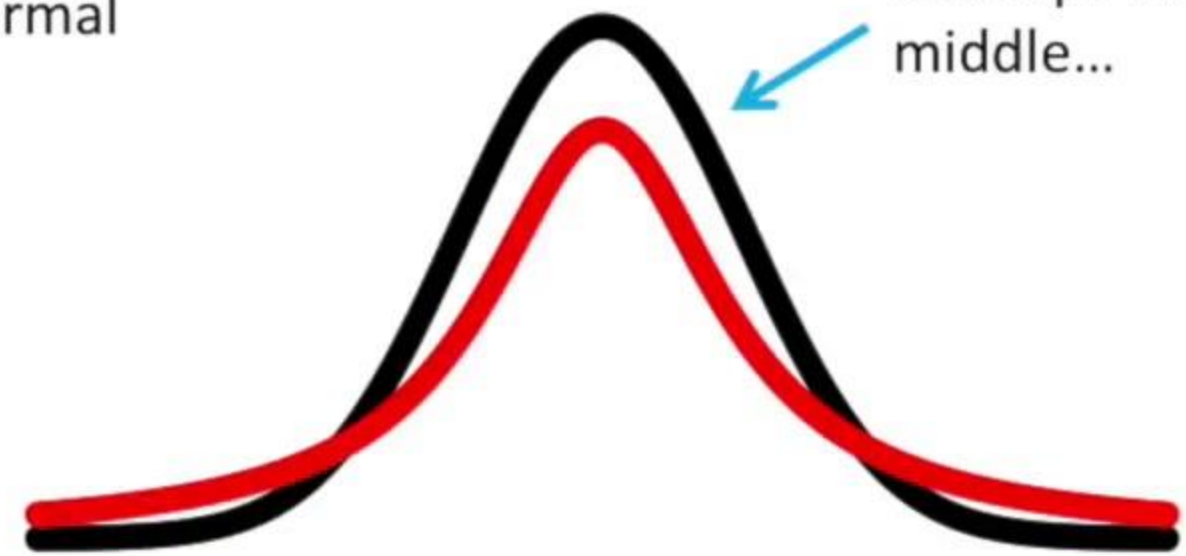
A “t-distribution”…

…is a lot like a normal distribution



A “t-distribution”…

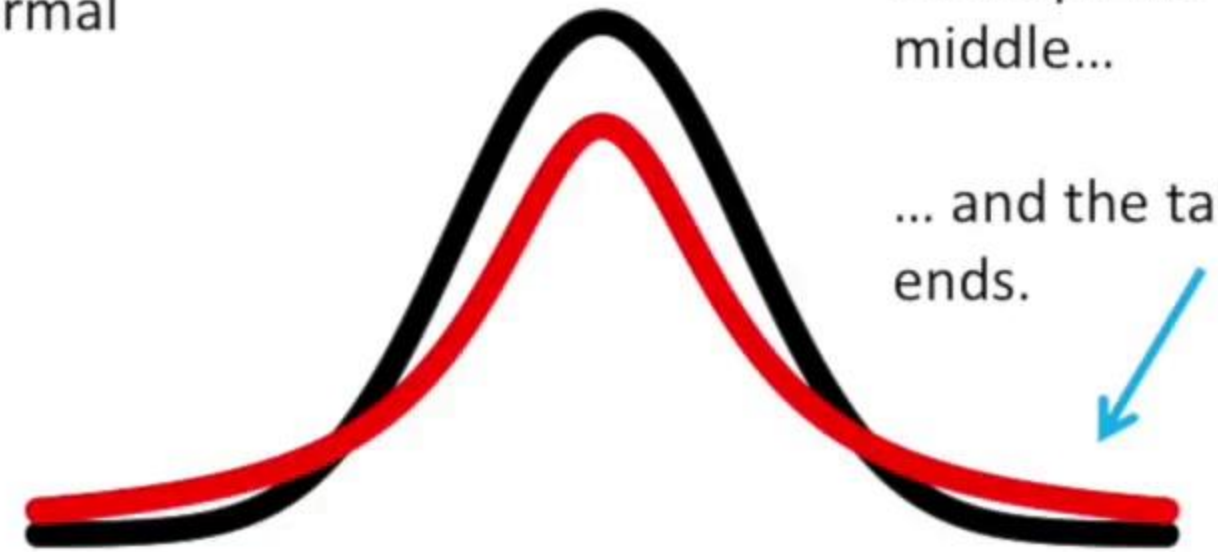
…is a lot like a normal distribution…



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

A “t-distribution”…

…is a lot like a normal distribution…

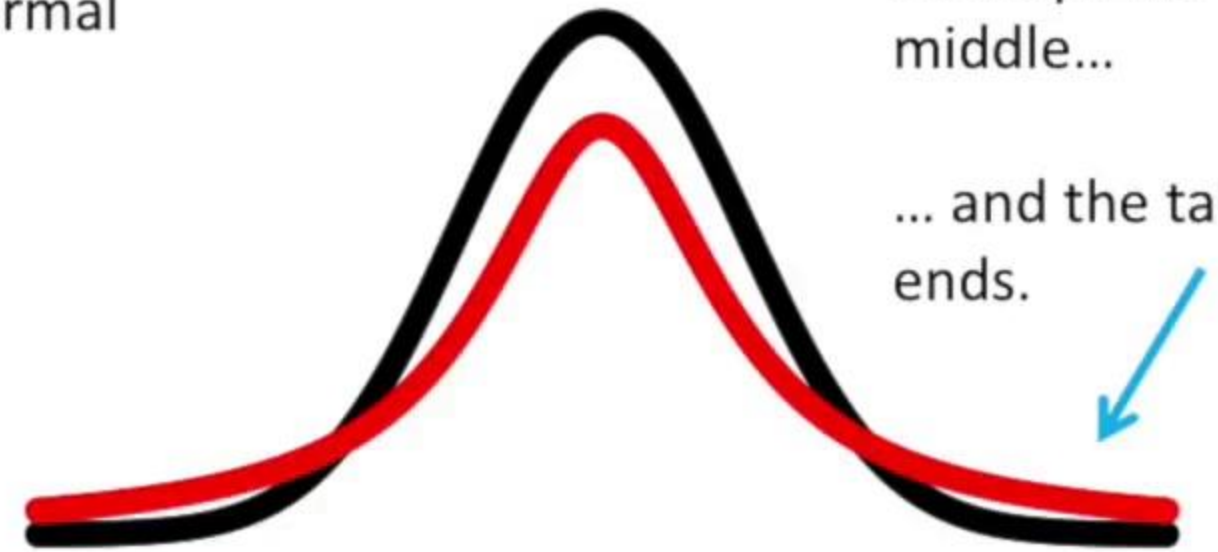


…except the “t” isn’t as tall in the middle…

… and the tails are taller on the ends.

A “t-distribution”…

…is a lot like a normal distribution…



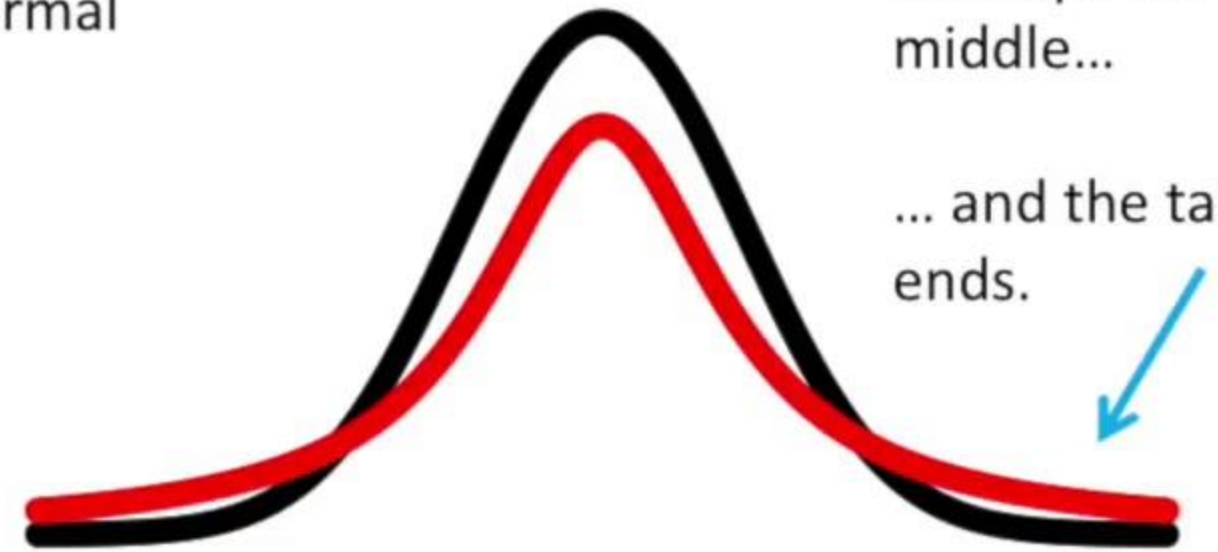
…except the “t” isn’t as tall in the middle…

… 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…



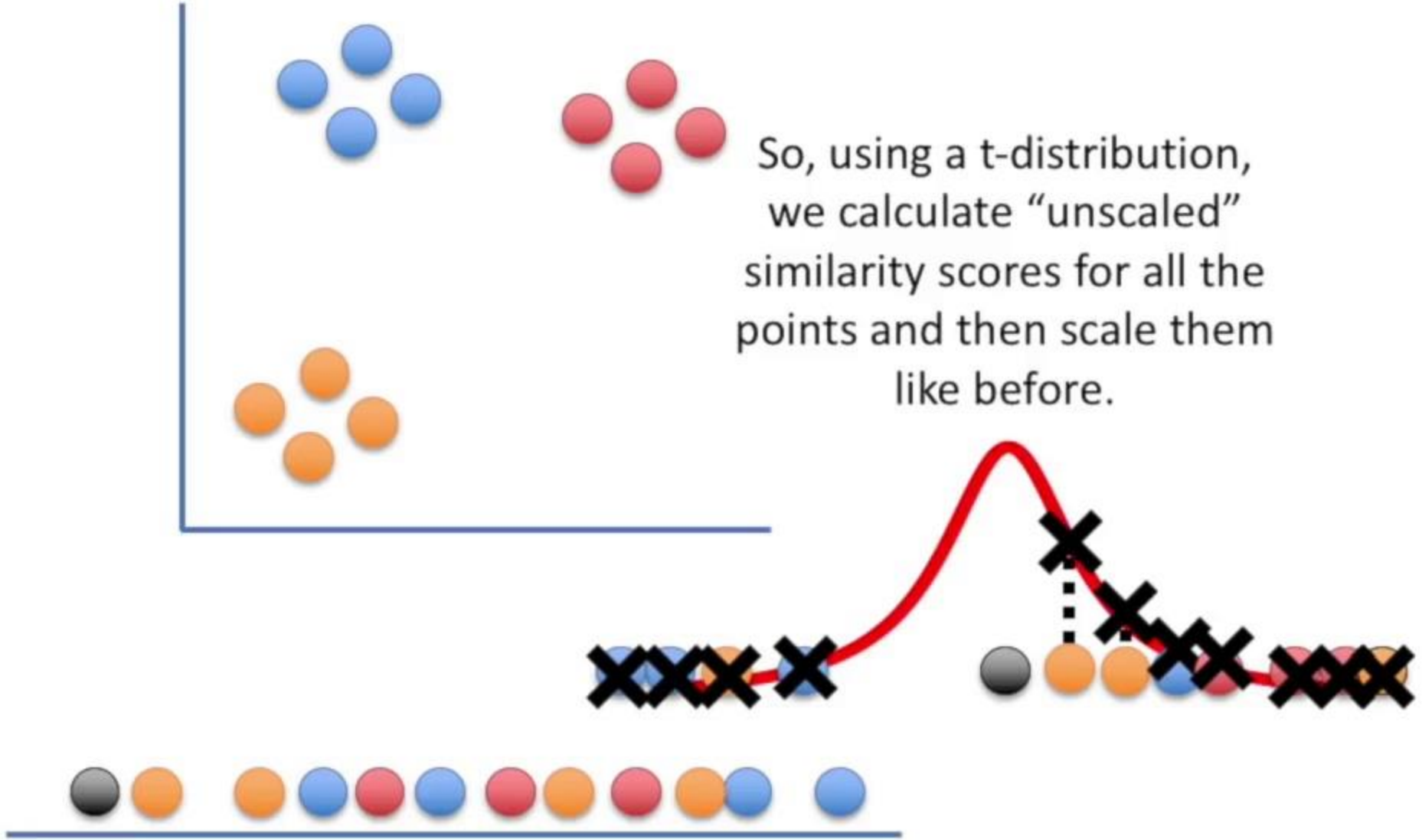
…except the “t” isn’t as tall in the middle…

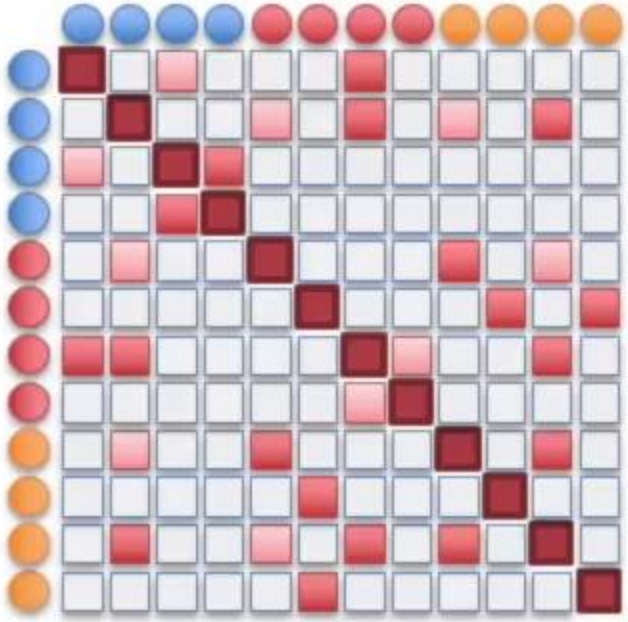
… and the tails are taller on the ends.

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.

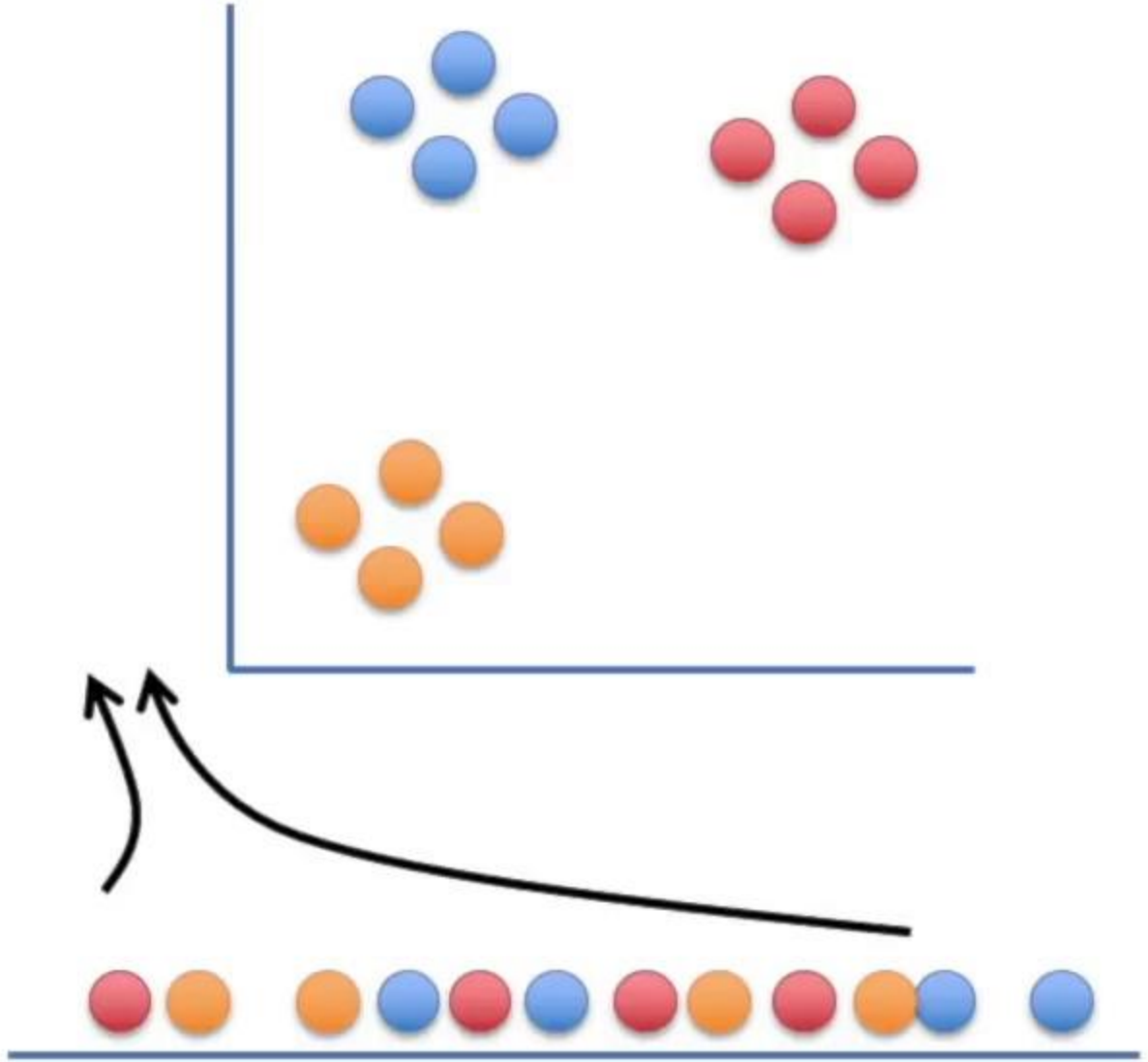


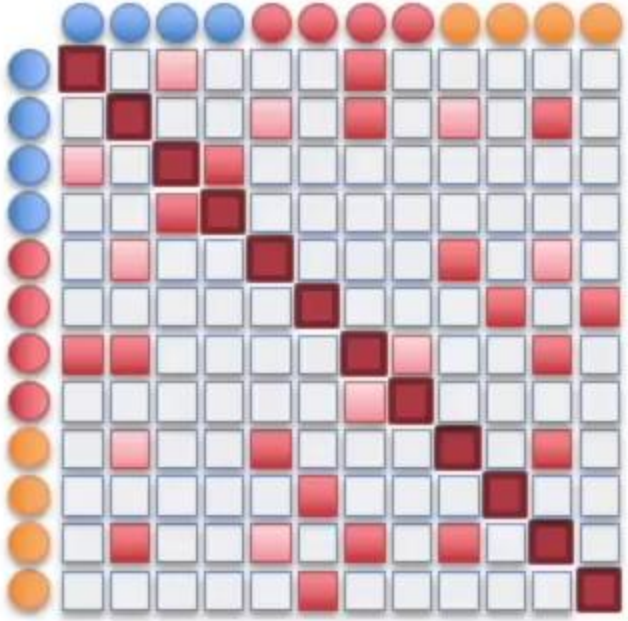


= High similarity

= Low similarity

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

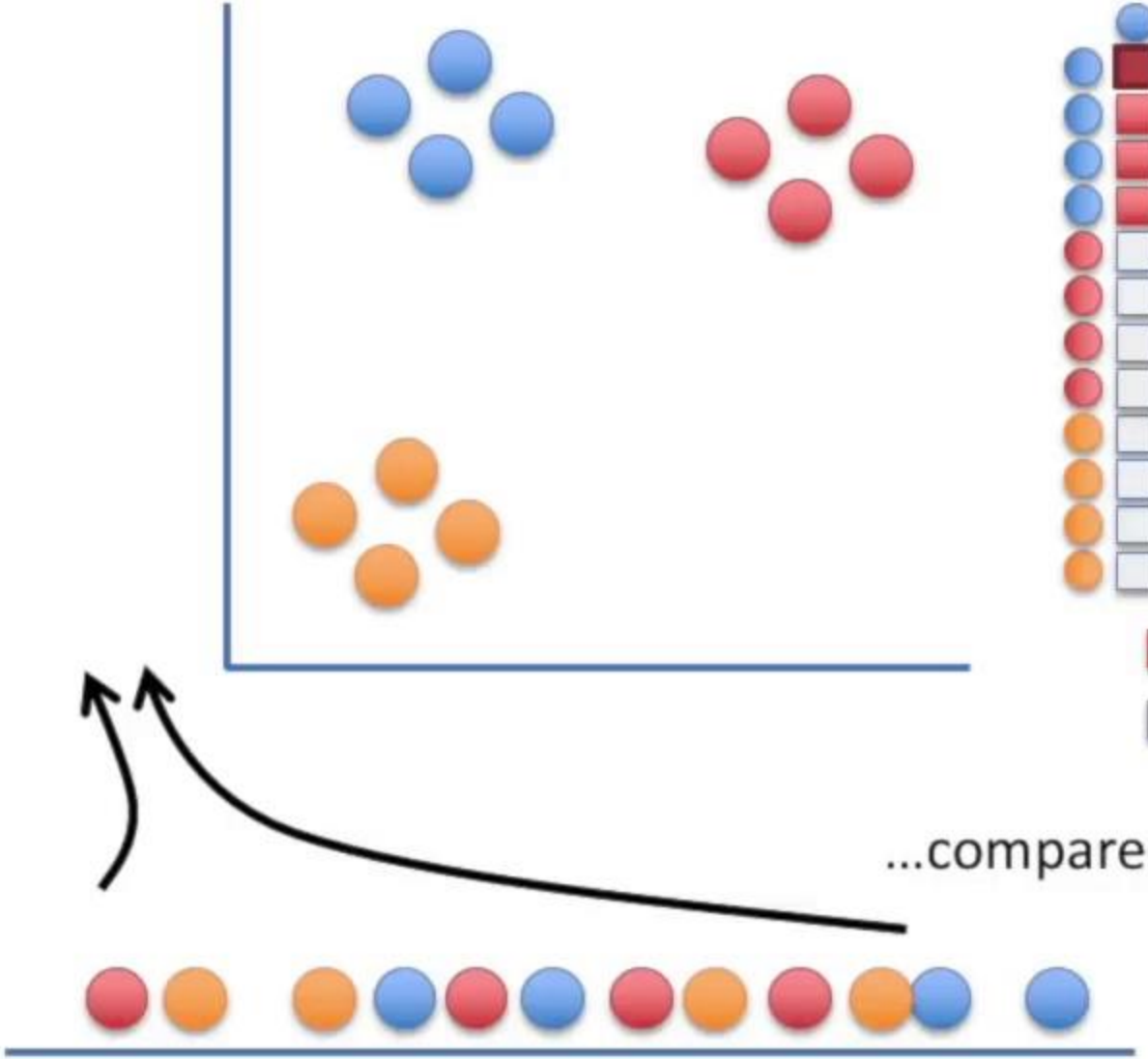


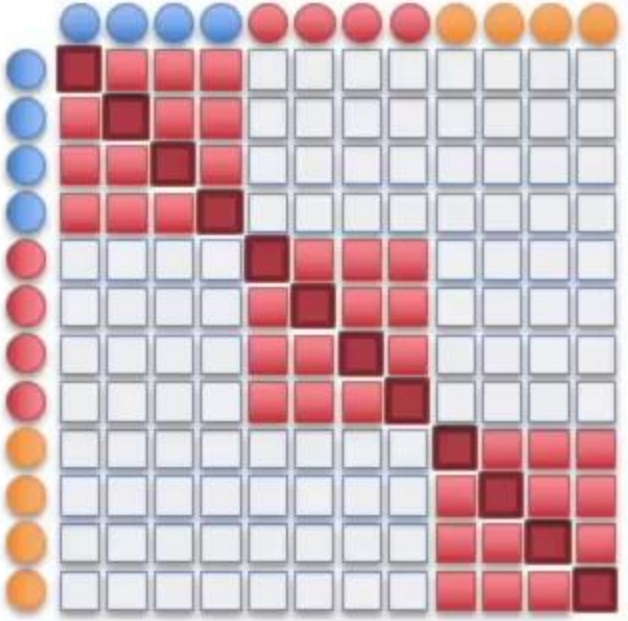


= High similarity

= Low similarity

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

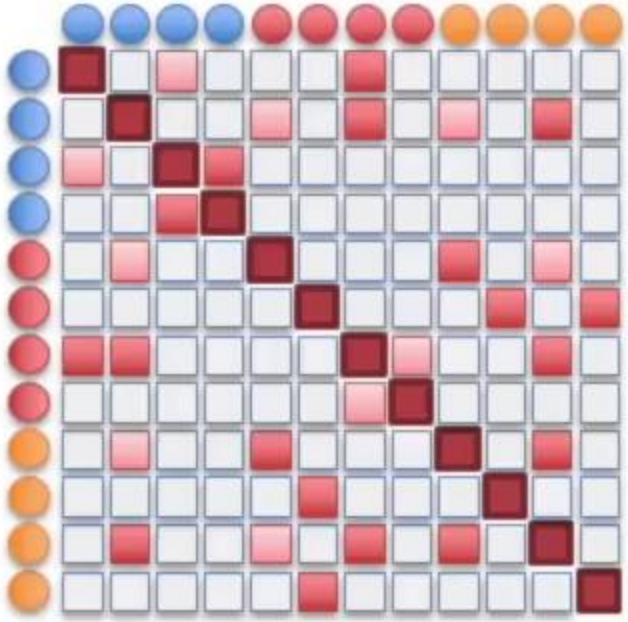


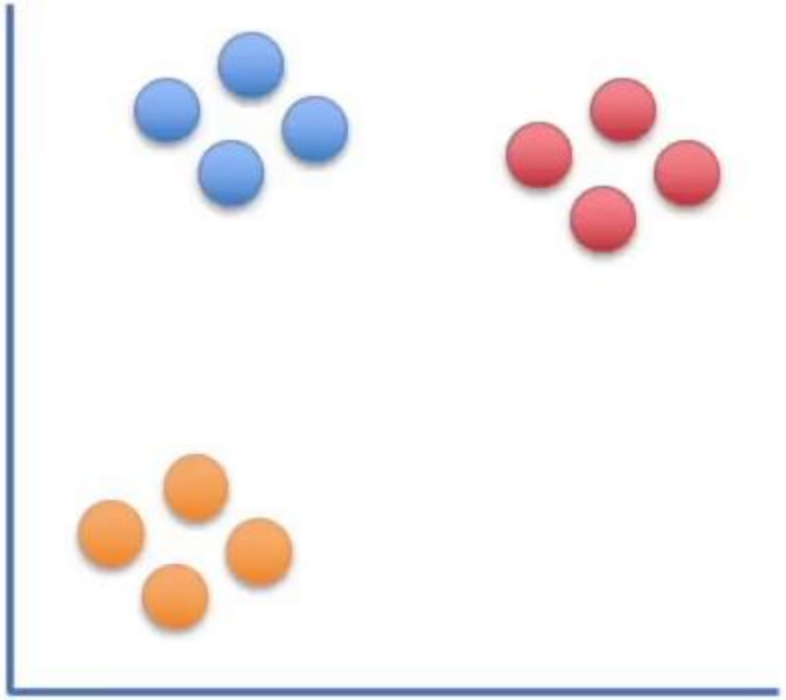


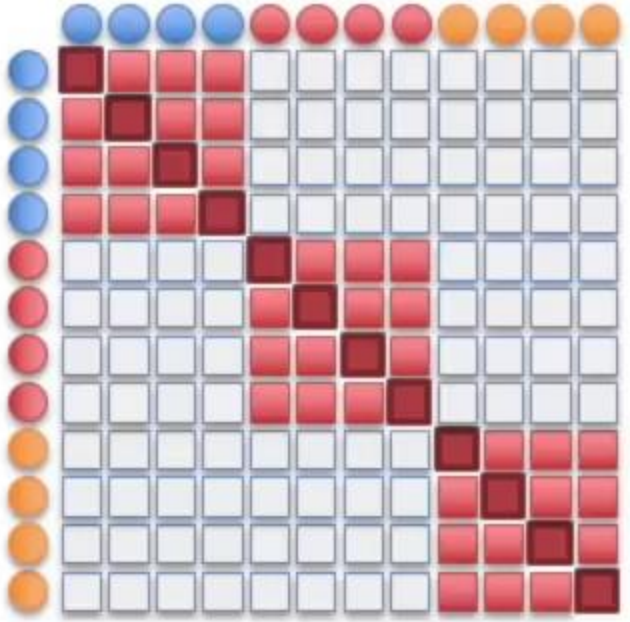
= High similarity

= Low similarity

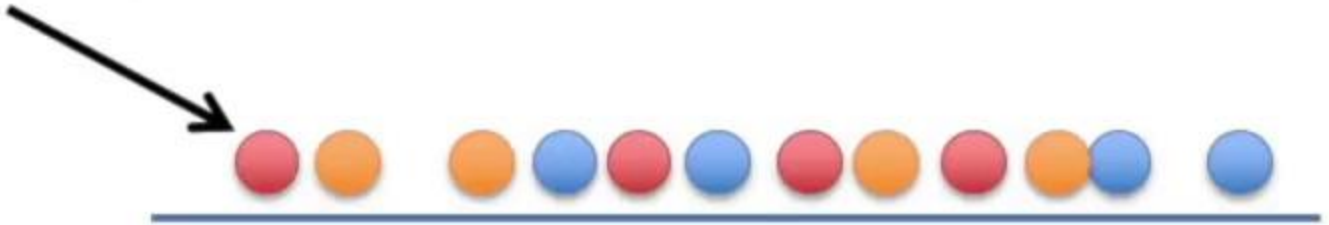
...compared to the original matrix.

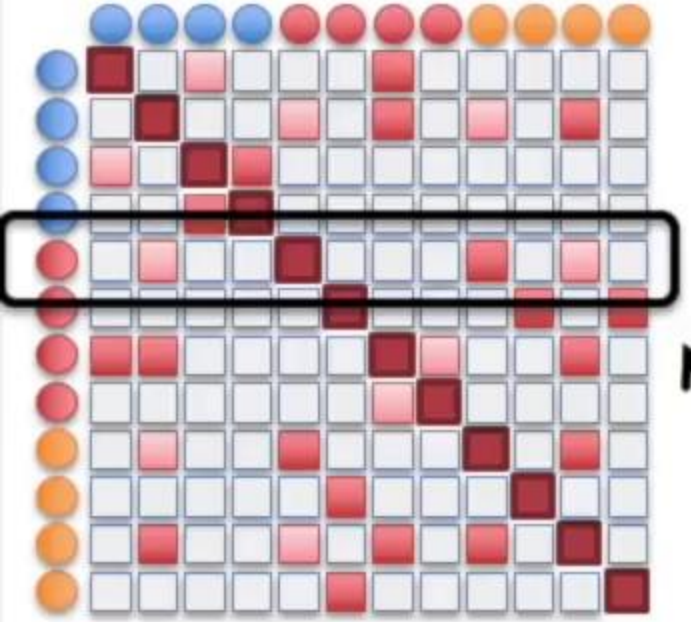


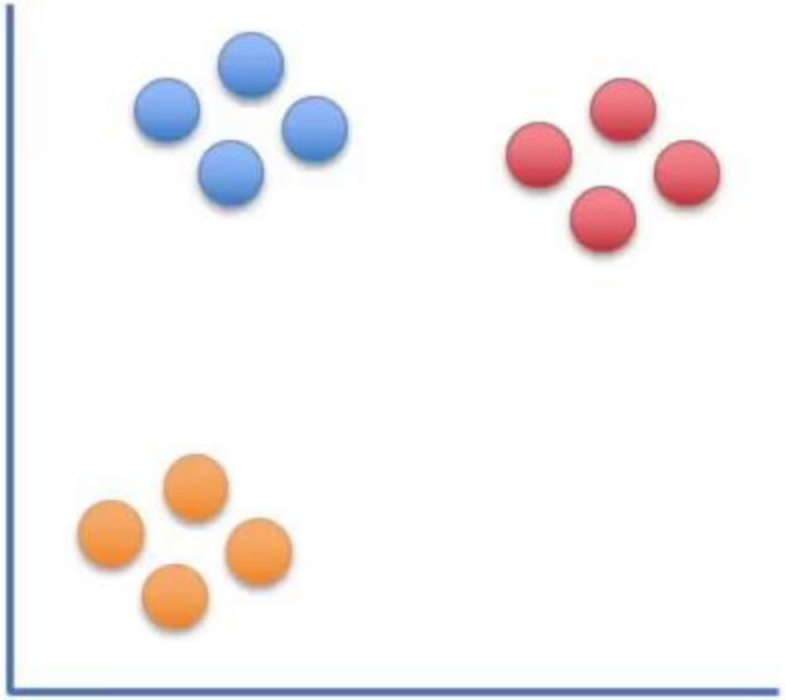


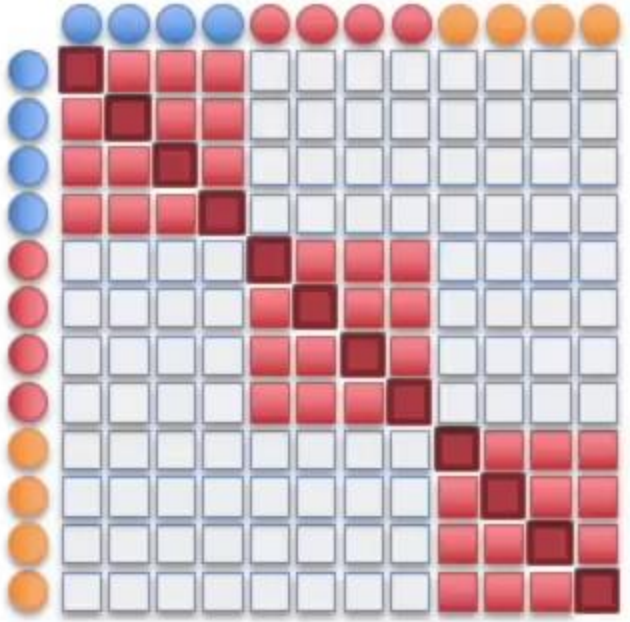


The goal of moving this point is…

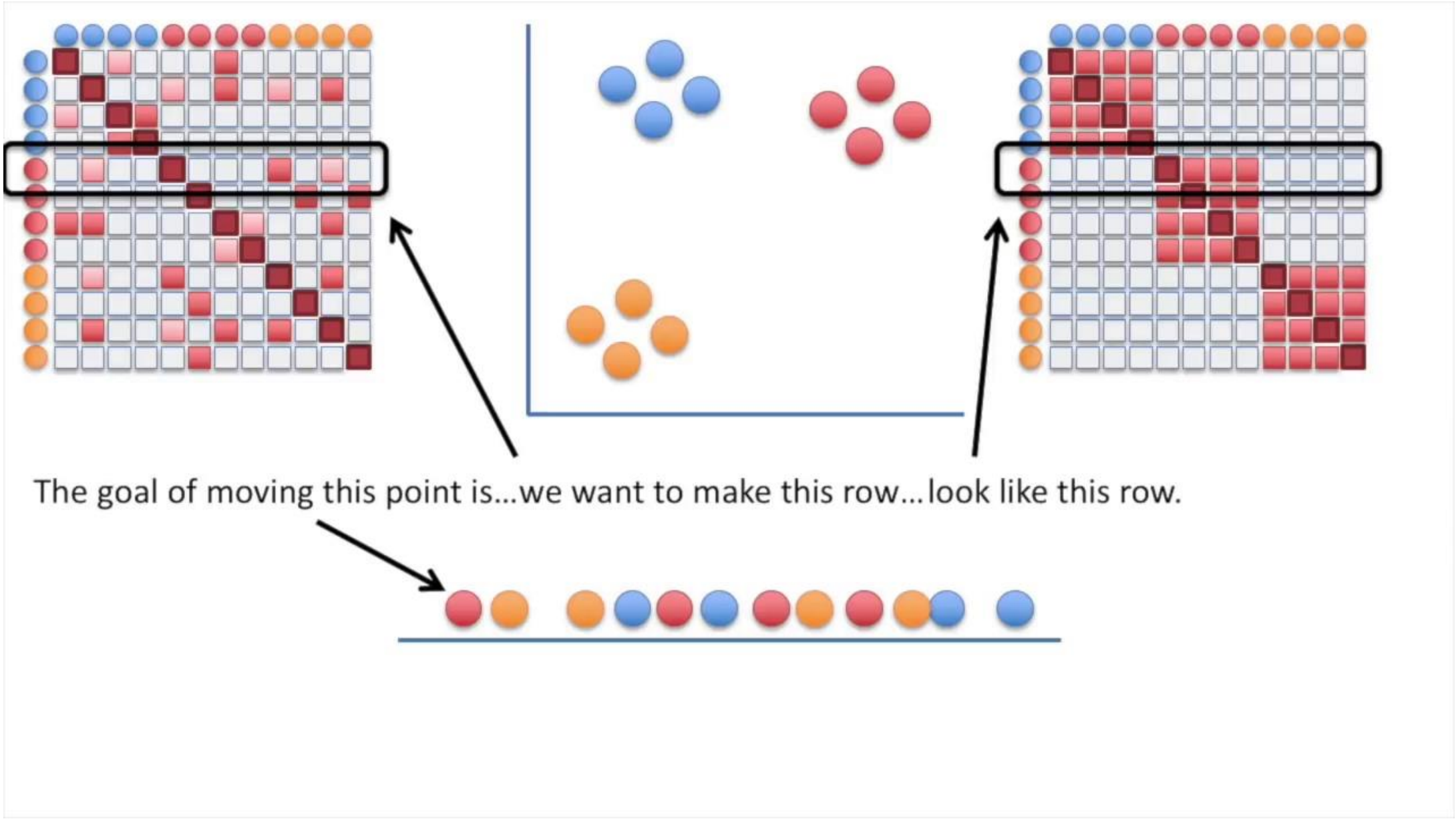


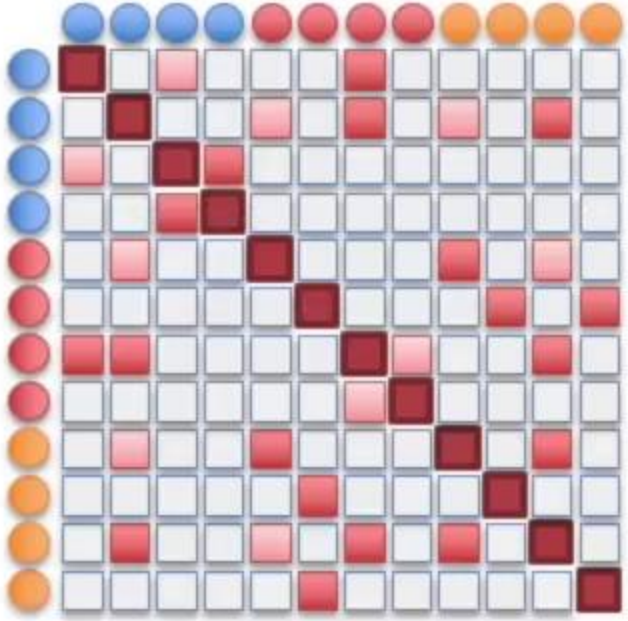


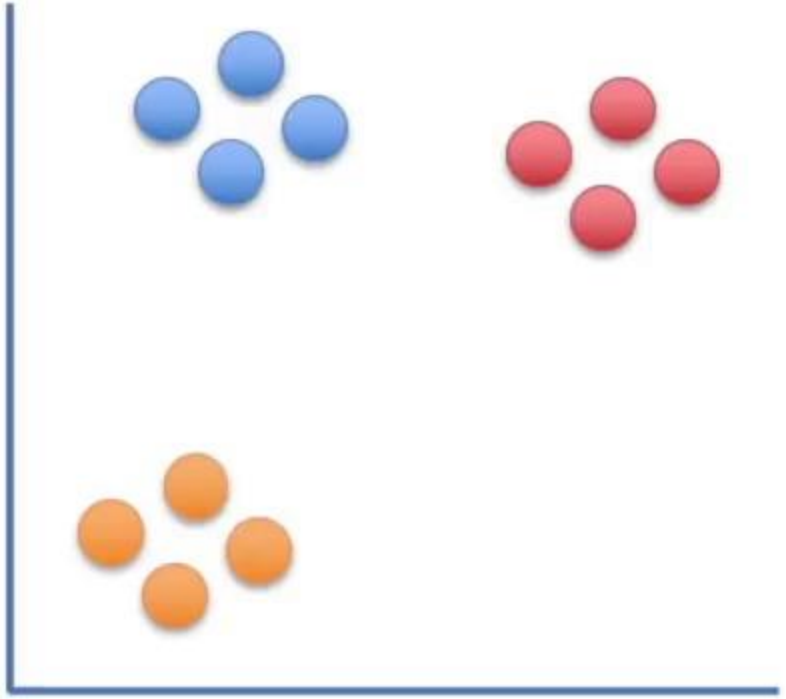


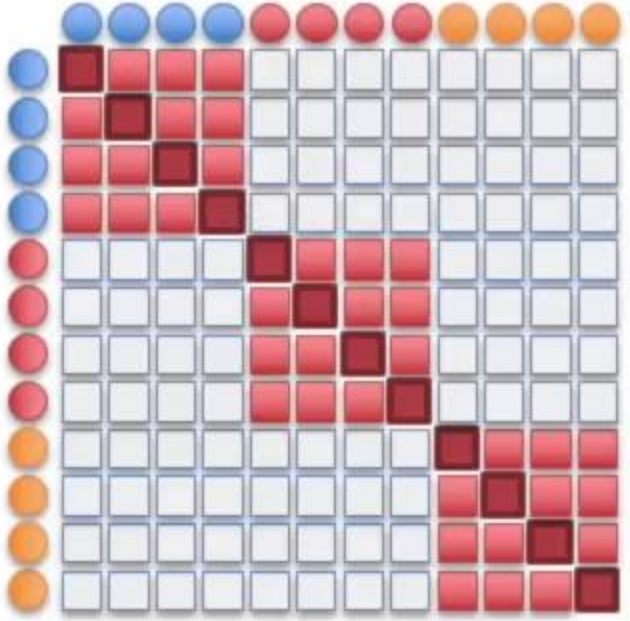


The goal of moving this point is…we want to make this row…

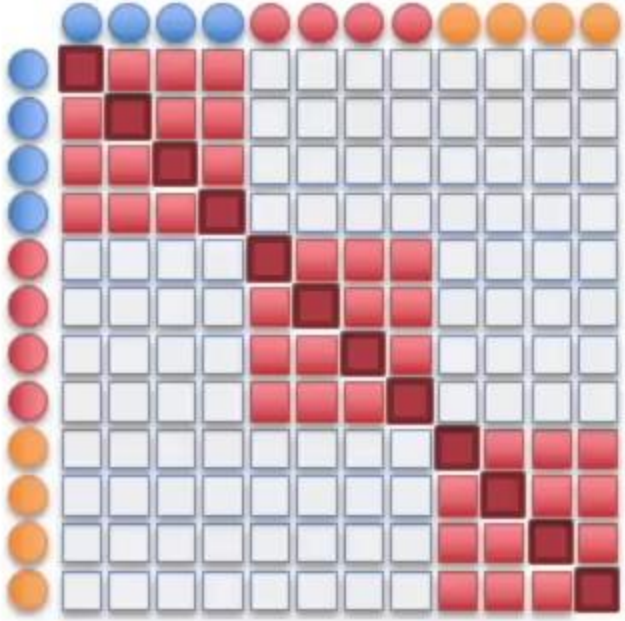

The goal of moving this point is…we want to make this row…look like this row.

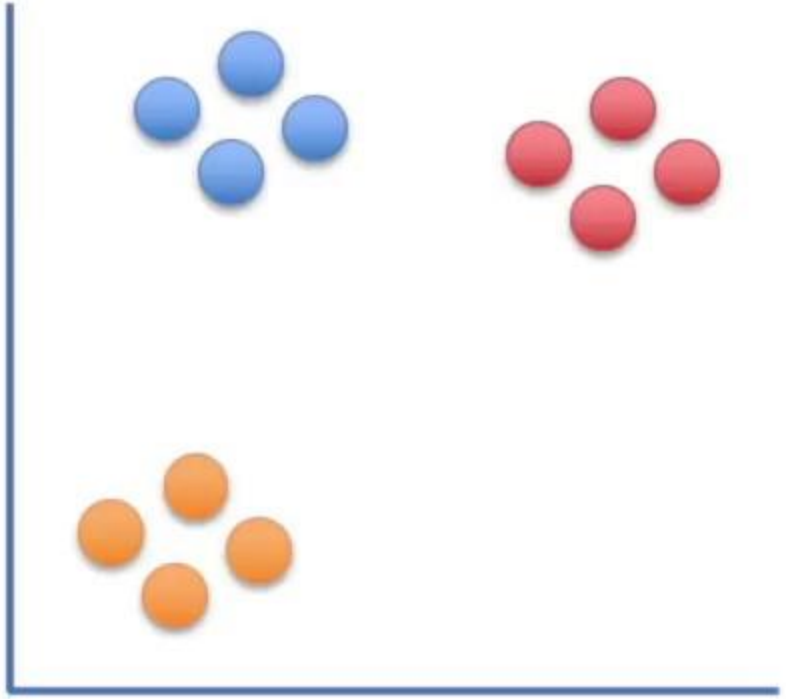


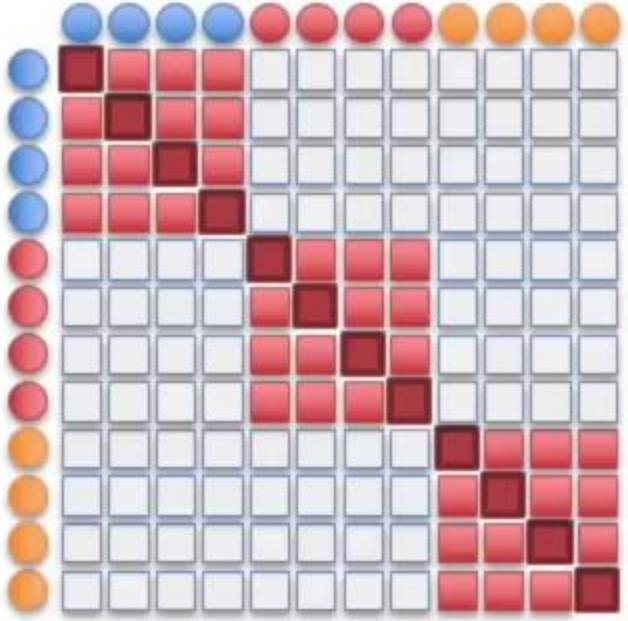




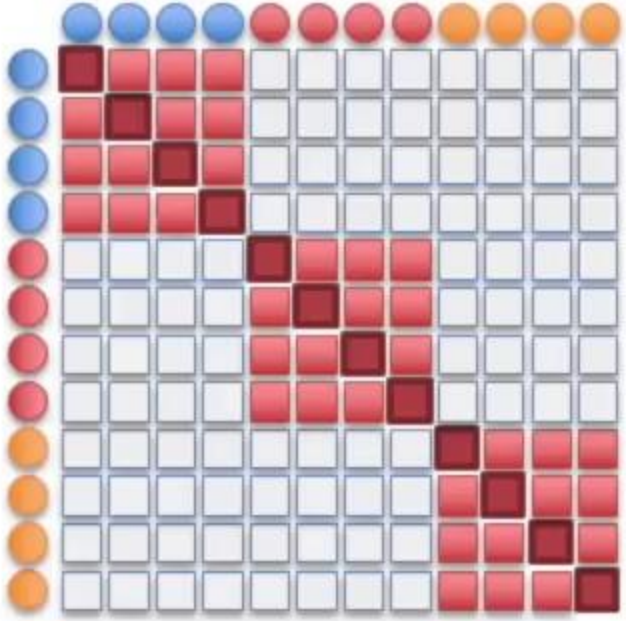
t-SNE moves the points a little bit at a time, and each step it chooses a direction that makes the matrix on the left more like the matrix on the right.

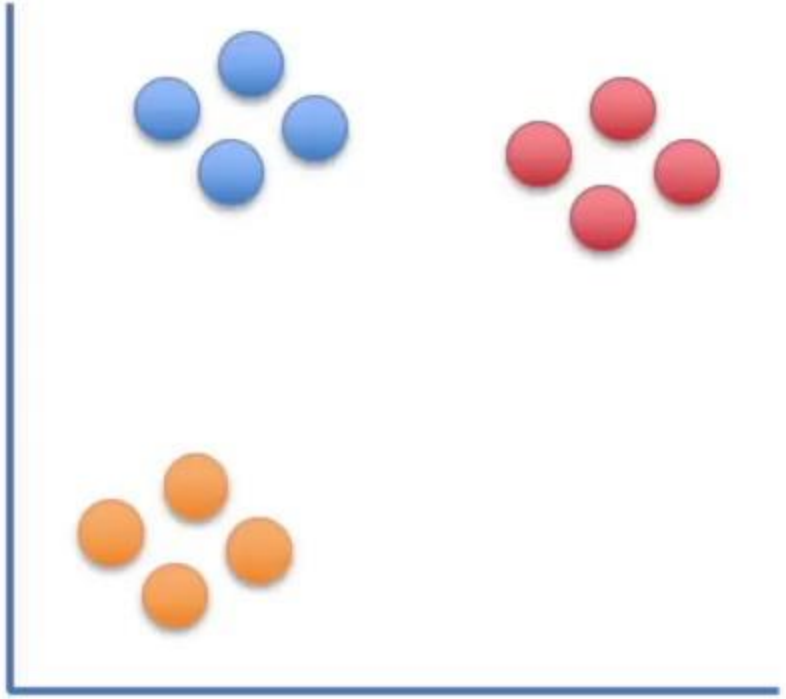


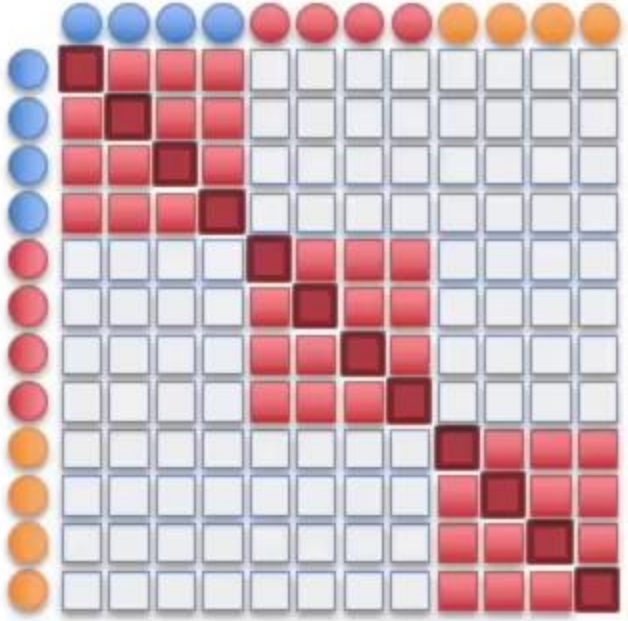




t-SNE moves the points a little bit at a time, and each step it chooses a direction that makes the matrix on the left more like the matrix on the right.







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.

Conclusion

- T-SNE is a powerful visualization technique
- It reduces the dimension by keeps the neighbors as close as possible
  - Therefore could be used to check the performance of classification algorithms