Unsupervised learning: Multi TS Features Embedding

Dragomir Nikolov
nikolovd@vmware.com

Dimira Petrova
dpetrova@vmware.com

Zhivko Kolev
zkolev@vmware.com

AMLD 26th Jan 2020

VMware Vision

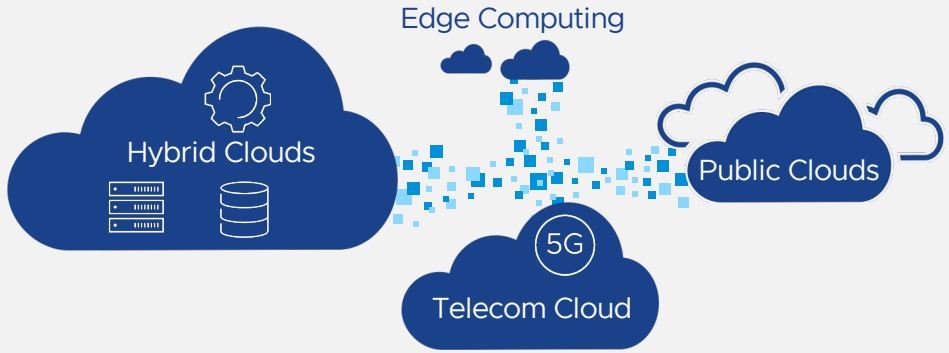
Deliver the essential, ubiquitous digital foundation

Any Device

Any App

Cloud Native | Containerized | SaaS | Traditional

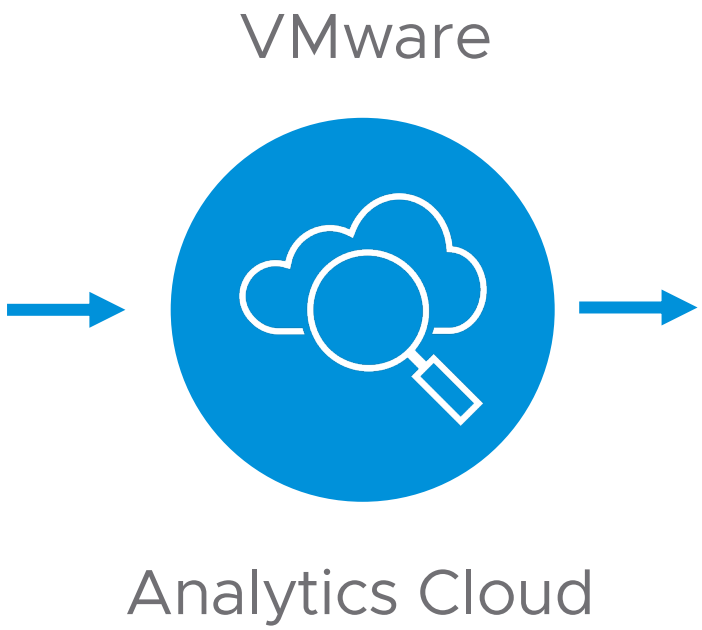
Any Cloud



vSphere Analytics

Telemetry Data

- Usage
- Performance
- Utilization
- Configuration

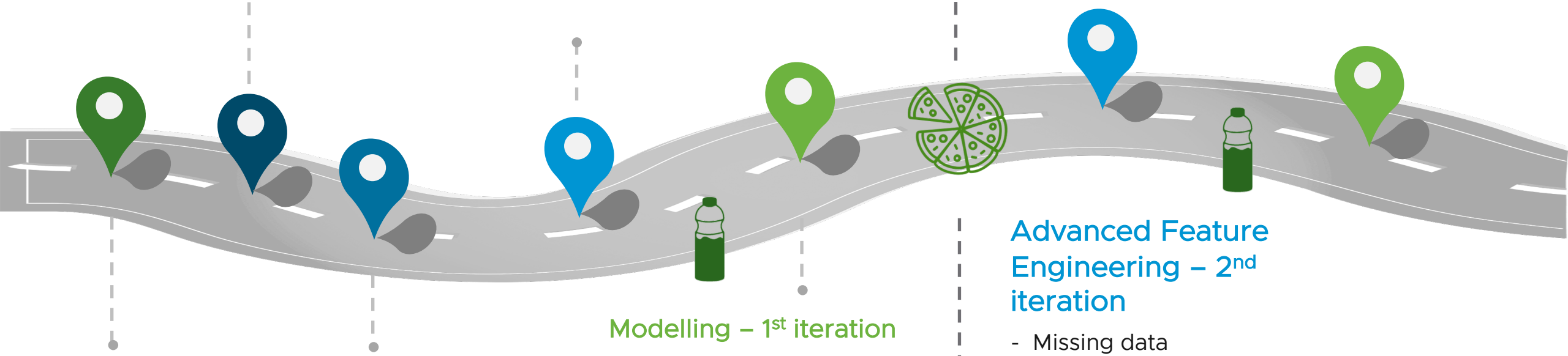


Analytics Services

- Proactive Support
- Self-Healing
- Auto-tuning

1st Half

2nd Half



Problem Description

- Vmware Stack
- Configurations
- Challenge
- The dataset

Environmental Setup

First look at the data

- Inspect a single VM series
- What is a data transformation
- How to choose which data transformation to choose
- Apply data transformation

Feature Engineering – 1st iteration

Modelling – 1st iteration

- PCA and why
- t-SNE
  - Theory
  - Out of the box application
  - Fine-tuning
- DBSCAN clustering algorithm
- Capture clusters

Advanced Feature Engineering – 2nd iteration

- Missing data imputation
- Time series overview
- Time series specific features
- Spectral Analysis

Modelling – 2nd iteration

- Correlated Features
- UMAP embedding theory
- Reiterate with the new features and algorithm

vmware

Workshop Objectives

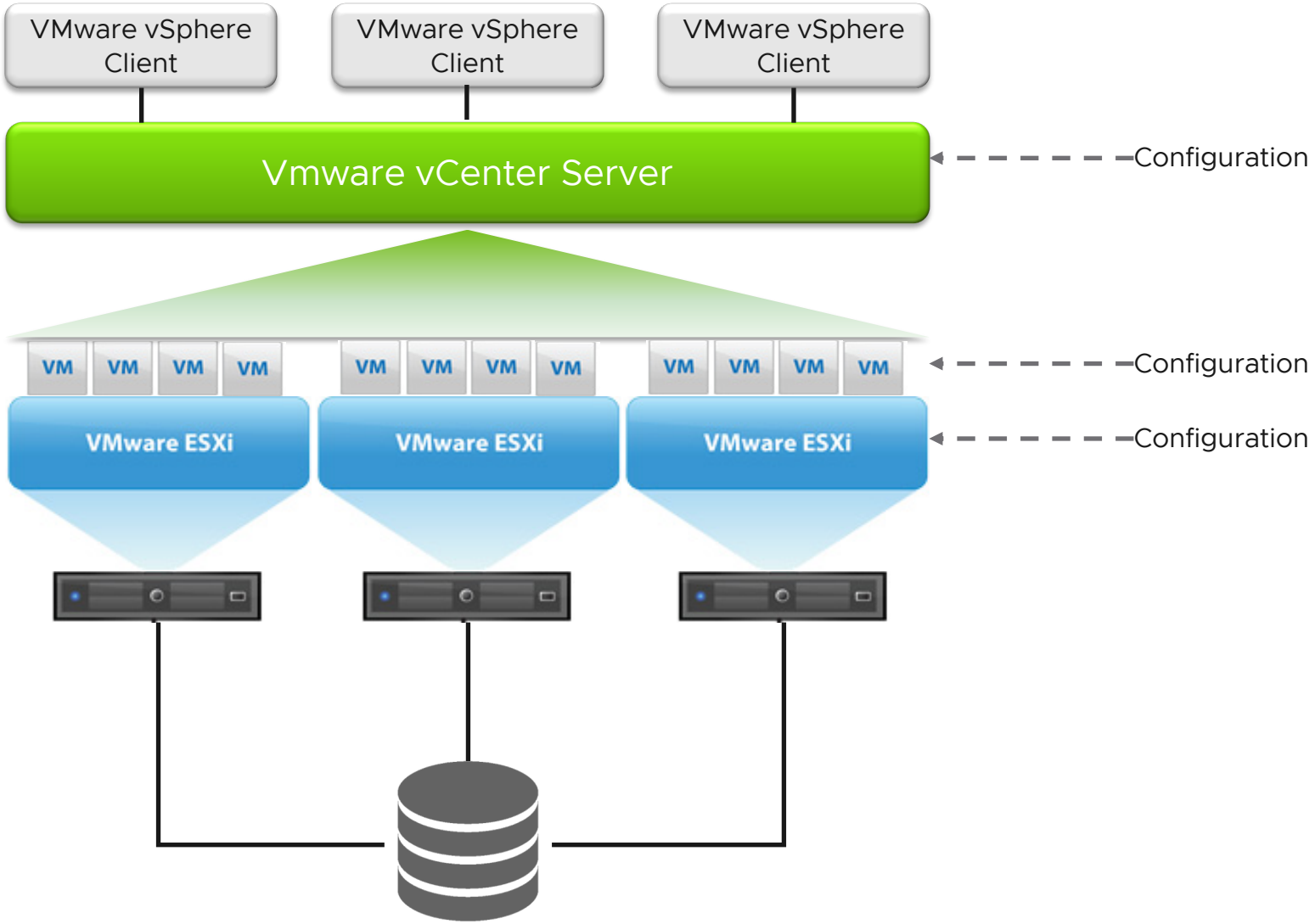
- Preparing data for embedding
- Extract features from time series
- Use common implementations of embedding algorithms with real-world data
- Interpret the results

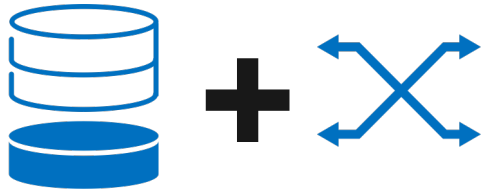
Problem Description

Still easy to drive

Problem Description – Make it relevant to us

VMware Stack – simply explained





Storage Networking

More technologies involved around networking and storage.

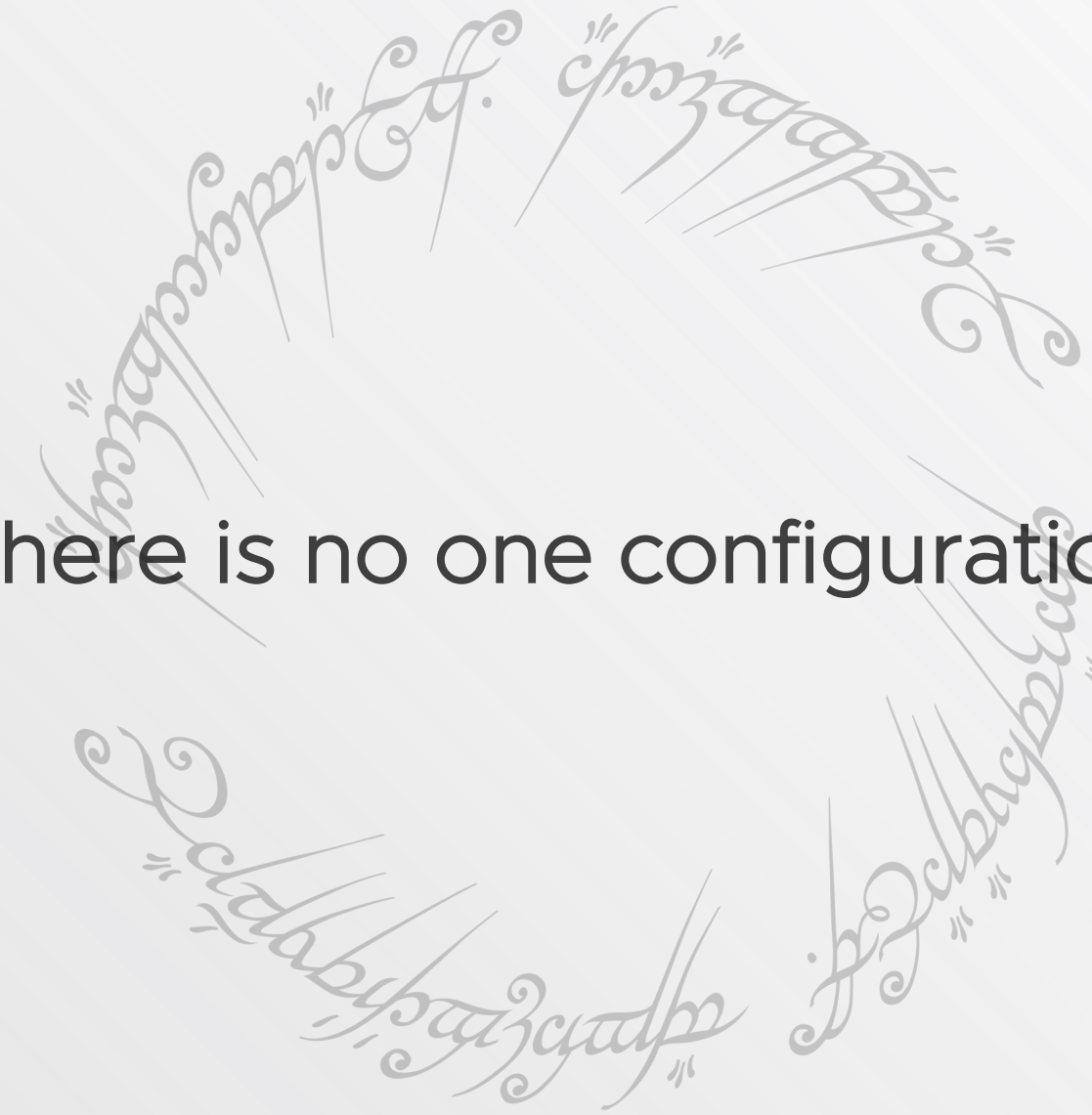
Complex architecture

Customization

Multiple customization options at different levels.


vmware
YouTube
User name
Password
Log in
vmware ESXi
High throughput traded for high latency
2:07 / 29:45

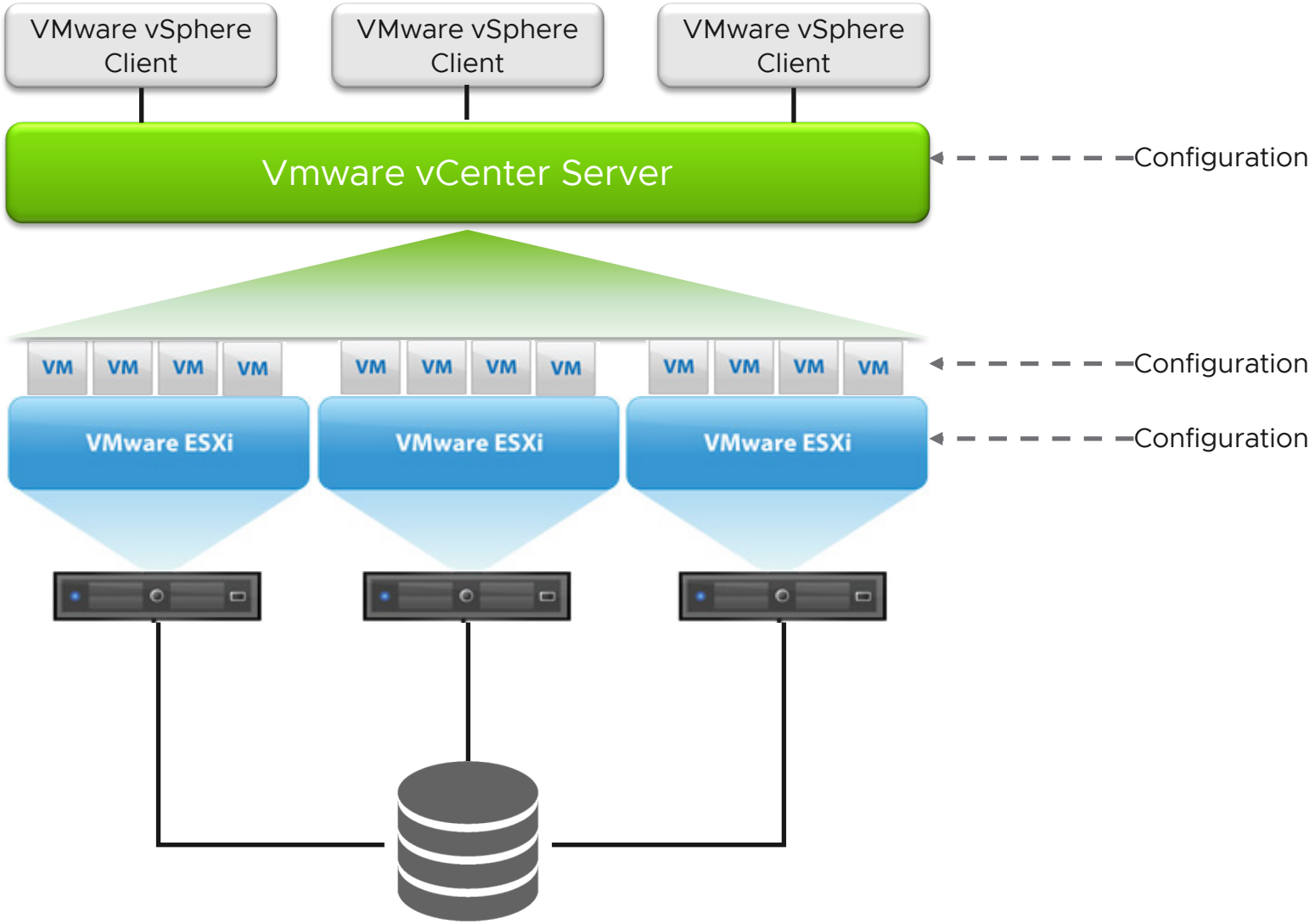
Not all workloads are created equal



And there is no one configuration to fit them all

Problem Description – The role of workload

VMware Stack – simply explained



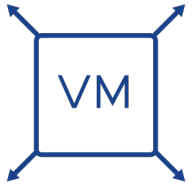
Storage **Networking**

More technologies involved around networking and storage.

Complex architecture

Customization

Multiple customization options at different levels.

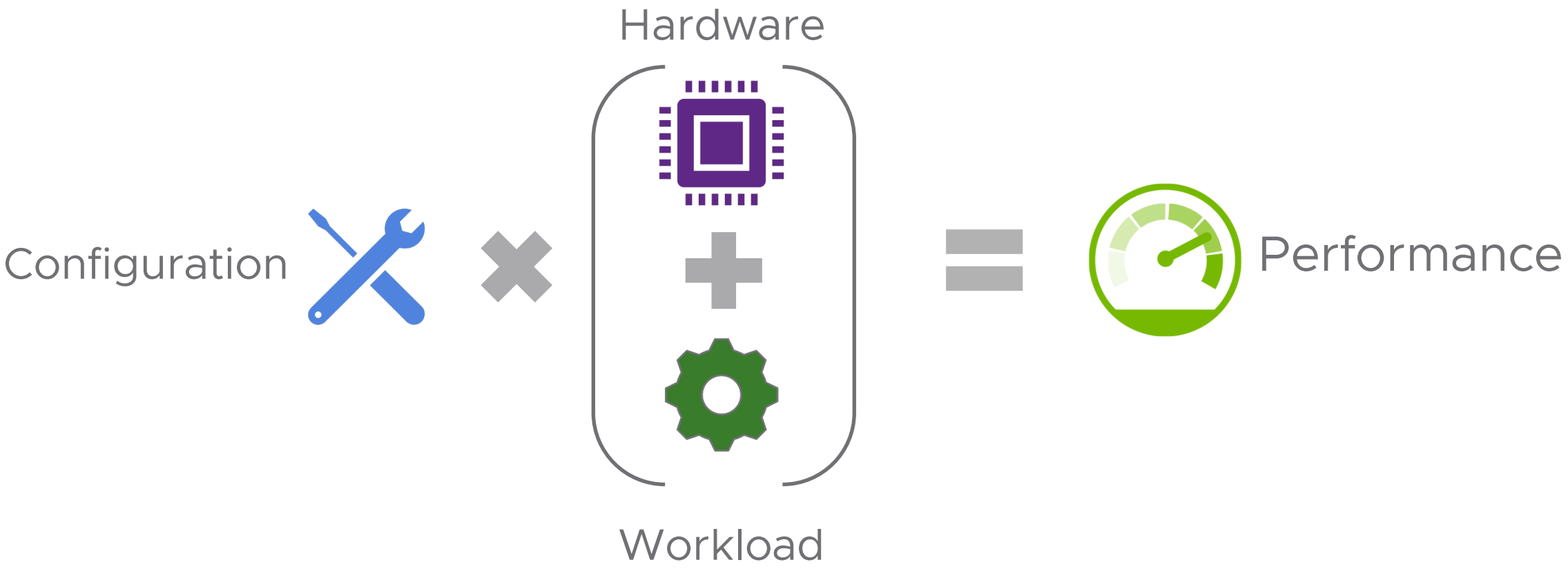


Workload

VMs working different applications and processes.

Why does it matter?

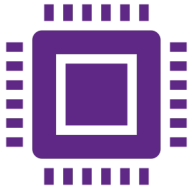
Performance Drivers



The Challenge

Configuration

Hardware

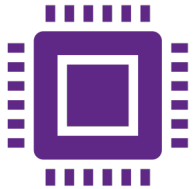


Workload

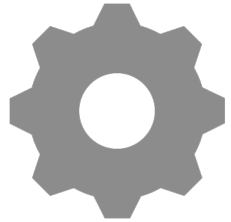
Performance

The Challenge

Configuration 

Hardware 

Performance 



Workload

The Challenge:

Find virtual machines with:

- similar properties
- utilization
- scale

What is embedding?

Representation of high dimensional vector on low dimensional scale by preserving as much information from the original vector as we can by positioning similar inputs close together on the embedding space.

vmware®

Setup the environment

https://github.com/amld-vmware/AMLD_2020

vmware

Get to know our data

The dataset

Get to know the data

Performance telemetry per VM

Data collected at 5 minutes intervals

One observation per VM

Virtual hardware dataset

$\frac{60\ min}{5\ min}$ = 12 obsr./hour

=> 12*24 = 288 obsr./day

ts	cpu_run	cpu_ready	mem_active	mem_activewrite	net_packetsRx	net_packetsTx
2019-12-06 00:00:00+00:00	3470.0	58.0	640329.0	439001.0	2809.0	90.0
2019-12-06 00:05:00+00:00	4325.0	46.0	599783.0	462770.0	2725.0	97.0
2019-12-06 00:10:00+00:00	4416.0	49.0	638930.0	535471.0	2538.0	118.0
2019-12-06 00:15:00+00:00	3269.0	62.0	610968.0	455779.0	2606.0	103.0
2019-12-06 00:20:00+00:00	4266.0	62.0	624949.0	482344.0	2638.0	141.0

CPU RUN
The time the virtual machine use the CPU

CPU READY
The time the virtual machine wants to run a program on the CPU but waited to be scheduled

Memory Active
The amount of memory actively used by the vm

Memory Active Write
Amount of memory actively being written to by the virtual machine.

Network Packets Received
Number of packets received during the interval.

Network Packets Transmitted
Number of packets transmitted during the interval.

The dataset

Get to know the data

Performance telemetry per VM

Data collected at 5 minutes intervals

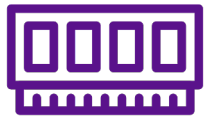
288 observations per day

One observation per VM

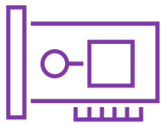
Virtual hardware dataset

Each Virtual Machine (VM) represented by one row

id	memory_mb	num_vcpus	number_of_nics	num_virtual_disks	os_fam
1	2048	2	2	1	Linux
2	12288	4	1	2	Linux
3	12288	4	1	2	Linux
4	12288	4	1	2	Linux
5	2048	2	2	1	Linux



Memory (MB)
Configured virtual RAM



Network Interface Cards
Number of network interface cards

VM's Operating System
The operating system of the VM

VCPUS
Number of virtual processor cores

Virtual Disks
Number of the configured HDD

vmware

Data Transformation

Prepare your data for analysis

What & Why transform data?

What?

Replacing the actual value X with some function of that variable f(X).

Why?

Multiple reasons for collecting data

Might not be in suitable format for predictions

Transformations help!

As a result of the transformation we would change the distribution and the relationship of this variable, which can be beneficial.

Feature Engineering

Feature Engineering – Iteration 1

Performance Telemetry

ts	cpu_run
2019-12-06 00:00:00	3470.0
2019-12-06 00:05:00	4325.0
2019-12-06 00:10:00	4416.0
.....	.....
2019-12-06 00:20:00	4266.0

Multiple records per VM

Create Summary Statistics:

- Mean, median, mode
- Standard Deviation, skewness, etc.
- Min, max
- Scaling

One record per VM

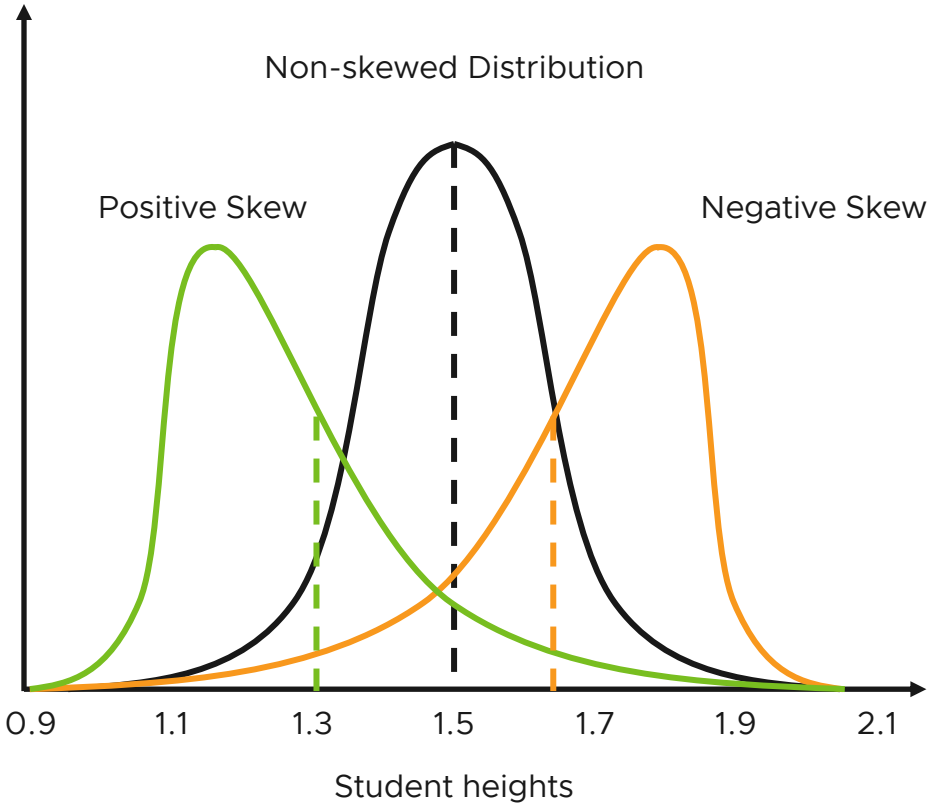
Pros:

- Not sensitive to small amounts of missing data
- East and fast to compute
- Easy to interpret

Cons:

- Not very expressive
- Completely ignore temporal dimension
- Hide a lot of information
- Some basic descriptive are very sensitive to extreme values and need preprocessing

What is skewness?



Mean = where the center of the distribution is (in terms of value).

Median = represents the ”middle” of the dataset.

Mode = The value that occur most often.

Standard deviation (sd) = the average distance between any point and the mean. It shows how thin the distribution is going to be.

Shewness – measures asymmetry

Why is it important?

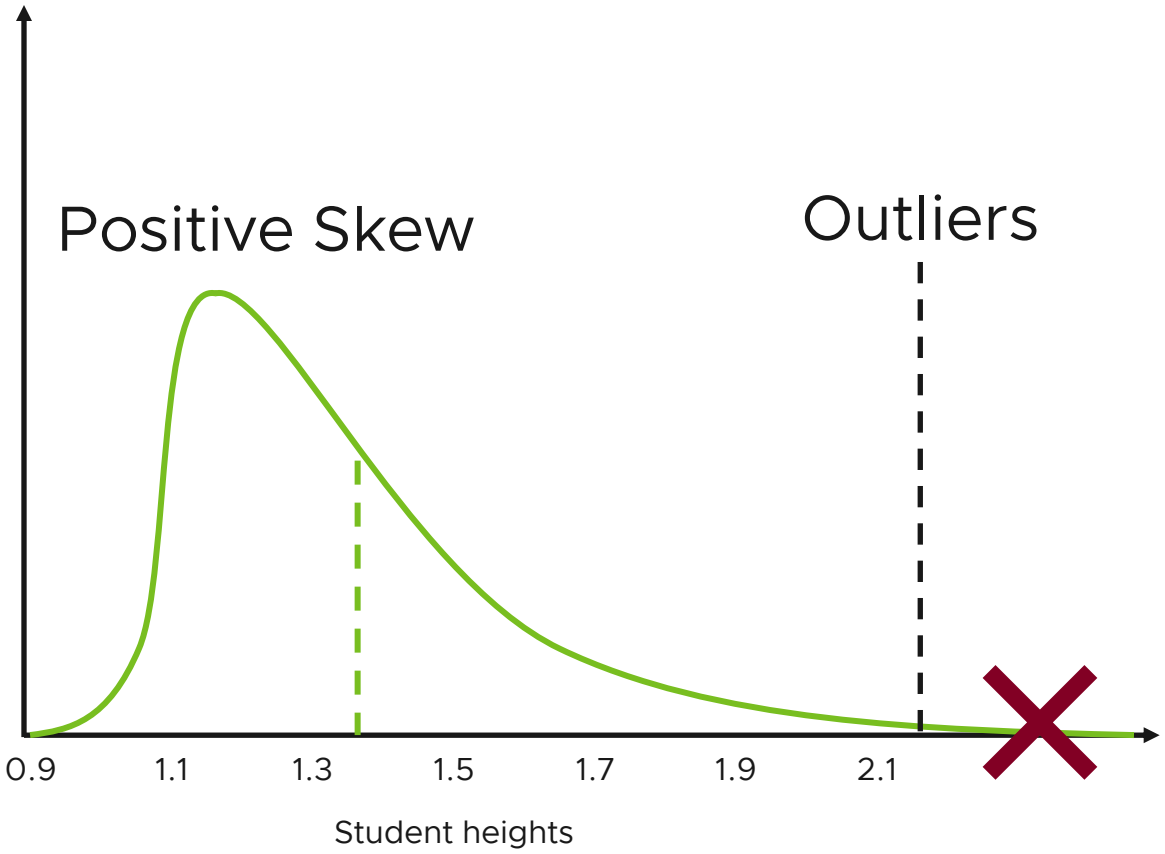
Skewed data might lead to misleading results.

”Normality is usually assumed with statistical techniques”.

How to fix this?

Square-root, cube root, logarithms (i.e. base or base 10) or Box-Cox transformation

Data Truncation



Outliers/ extreme values are introducing noise to the data.

Prevent outliers from obscuring our data.

Scaling

Sometimes we have features highly varying in magnitudes, units and ranges.

Since most machine learning use Eucledian distance, they are sensitive to the magnitude of the numbers.

I.e. In the case of 5kg and 5000gms, the higher magnitude will have more weight in distance calculations.

The problem

Standardisation

With this standardization we have each feature to have zero-mean value.

Mean normalization

$$x' = \frac{x - mean(x)}{\mathrm{O}}$$

In this case values are distributed between [-1;1].

Min-max scaling

Simplest method for normalization, rescaling in the range in [0;1] or [-1;1], selecting the target depending on the nature of the data

$$x' = \frac{x - min(x)}{\max(x) - \min(x)}$$

How to scale

Use algorithms that assume normality and your data is not normal

Use algorithms that compute distance

When to scale

Principal Component Analysis

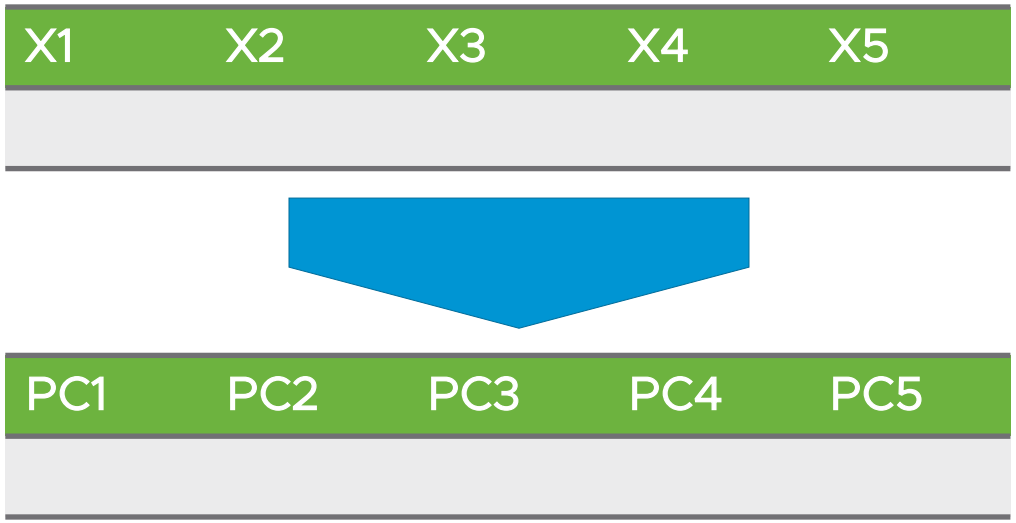
PCA Intro

In the beginning …

- Sometimes there is too much data
- It is highly correlated and redundant
- There is the need to visualize data
- The data does not have the necessary properties for statistical modeling and PCA ensure them

PCA comes to rescue

PCA Intro



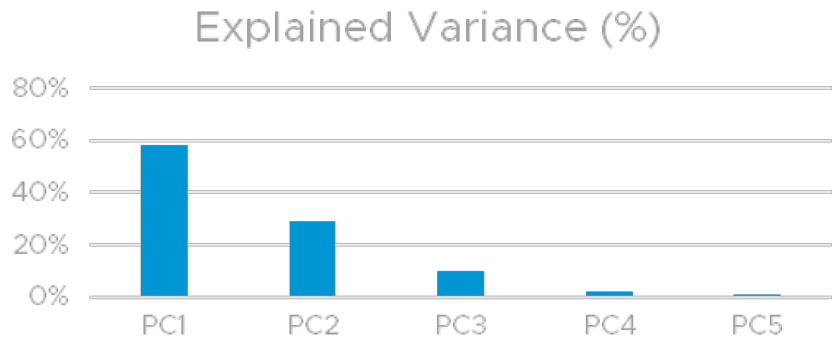
With PCA we transform our data to new features, called principal components, where each component is a linear combination of the original features.

$$PC_1 = w_{11}X1 + w_{12}X2 + w_{13}X_3 + w_{14}X_4 + w_{15}X_5$$

Ultimately, we end up with transformed data, where the total number of PC equals the number of features in our dataset …

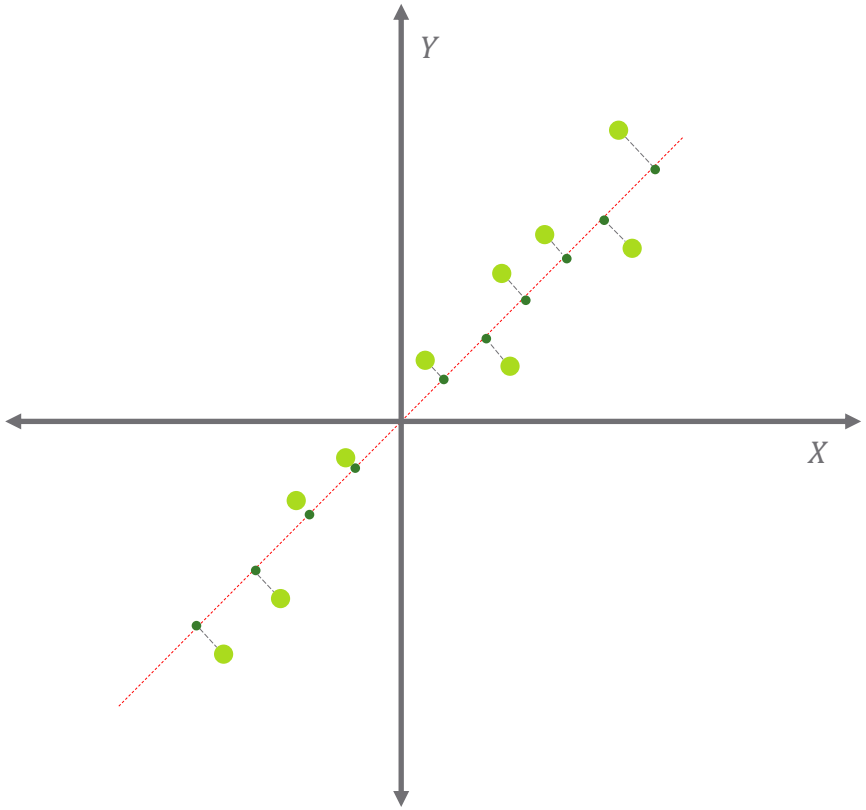
The PCs are ordered in a way, that PC1 explains largest variance of data, PC2 – second largest, PC3 – third largest etc.

Ultimately most of the variance is explained through few components



How PCA Works

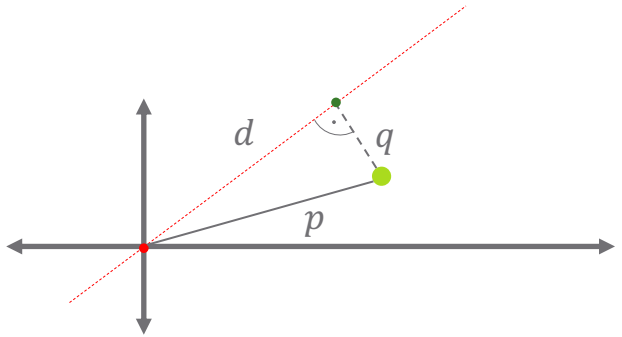
PC1



We start with some data that is centered around the origin

Then we draw a line that passes through origin and fits the best it can. We project the points on that line

How good line fits can be determined by distance from points to the projections OR from the projection to the origin of the graph



Since p is a constant

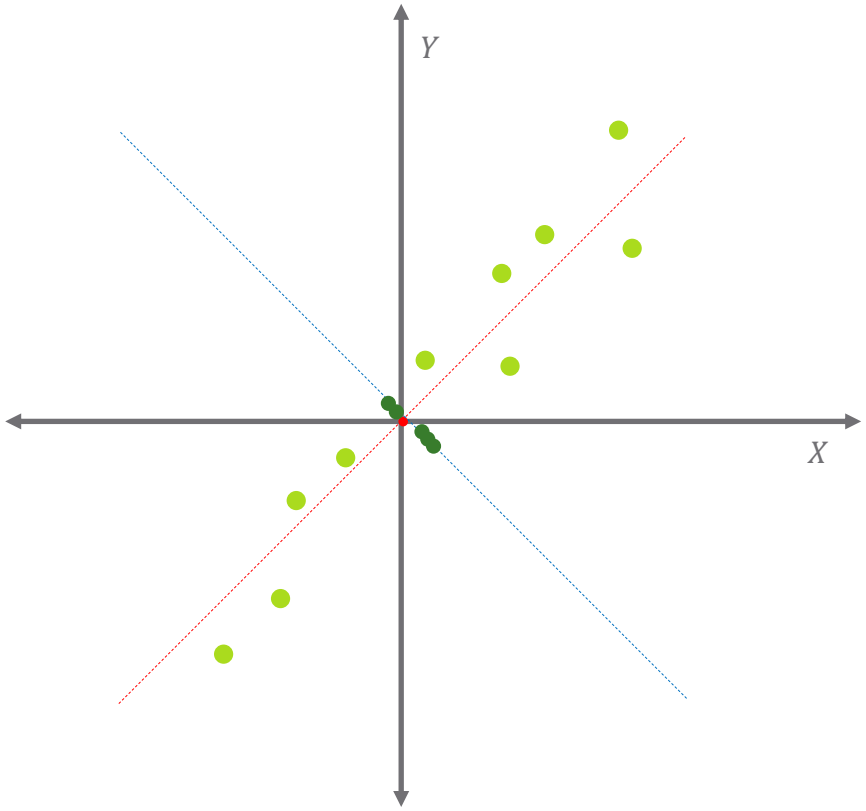
When d is large, q is small

When we divide the sum of squared distances d by the number of samples then we get the explained variance

$$Explained\ variance = \frac{\sum d^2}{n-1}$$

How PCA Works

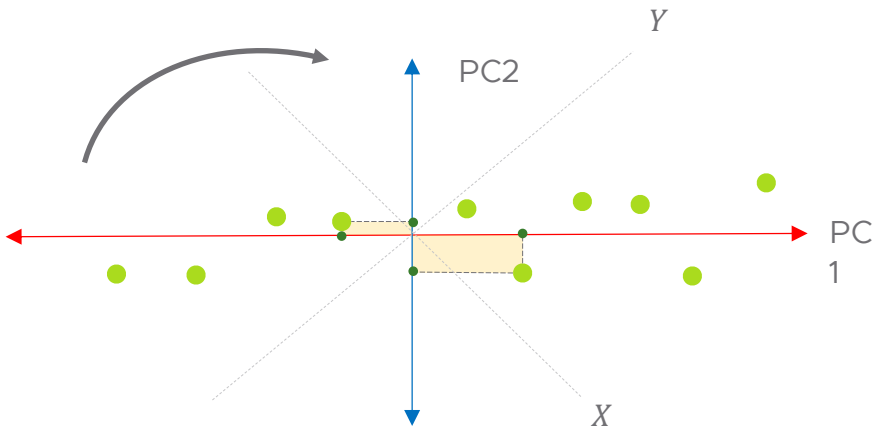
PC2



PC2 is determined by the line that is perpendicular to PC1, passes through origin and again passes as close as possible to all points (equivalent to having largest distance from the projection on the PC axis to the origin)

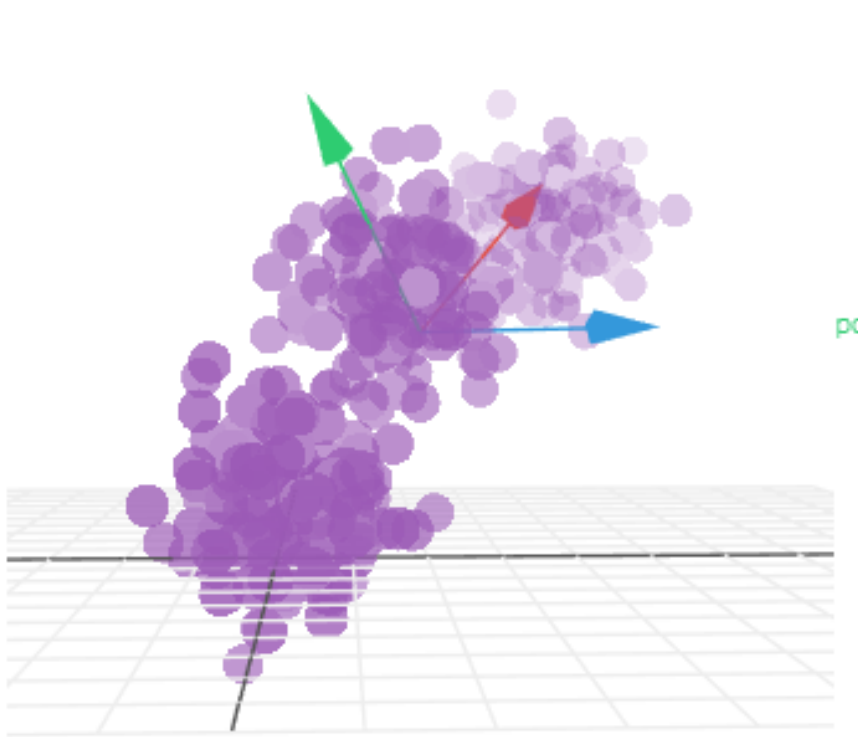
To plot the data on PC1 and PC2 we simply rotate the components, so the new axes are the PC1 and PC2

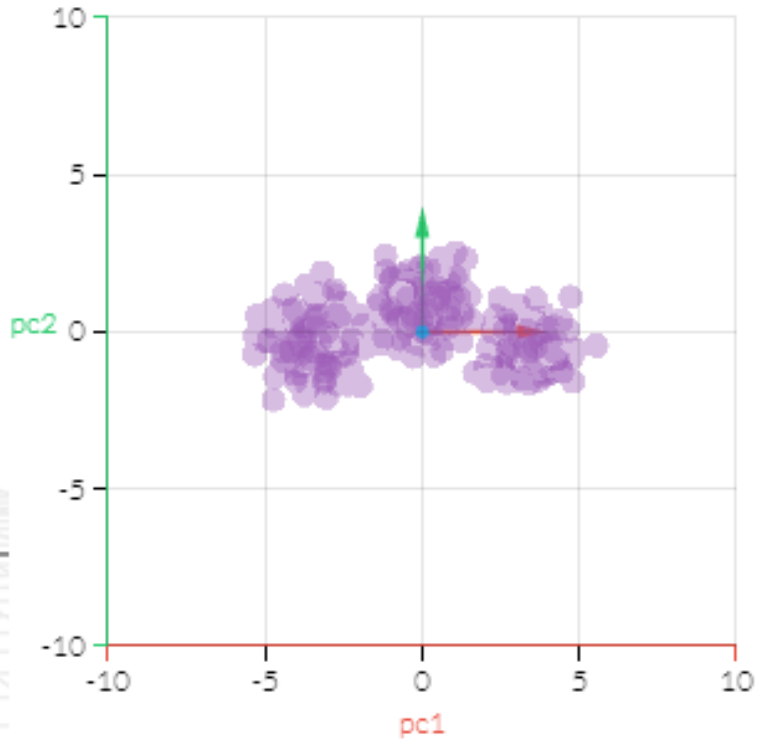
The new coordinates are given by the projections on the PCs



How PCA Works

The idea extends to higher dimensions

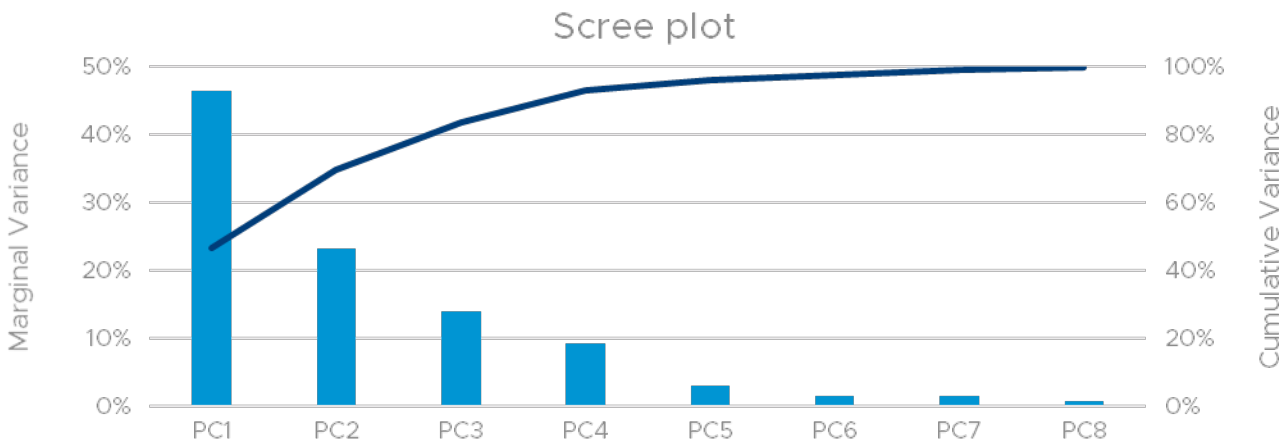




http://setosa.io/ev/principal-component-analysis/

Scree plot

Scree plot is graphical representation of the percentages of variation that each PC accounts for.



It is used to determine the reliability of the PC visualization or to determine the number of components to keep in case PCA is used as a preprocessing step

When visualizing data in 2D we are interested in the explained variance of PC1 and PC2 (and PC3 for 3D plots)

When we use PCA as a preprocessing step we are interested in the elbow of the plot (where the cumulative explained variance levels off)

This approach however is criticized due to its subjectivity.

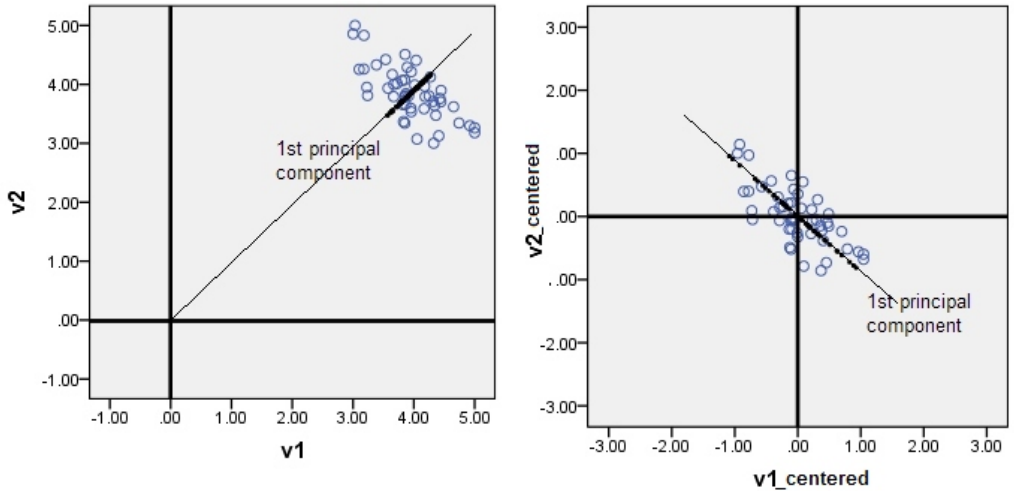
Look to the number of PCs as tunable hyperparameter !

PCA: The Good, the Bad and the Ugly…

- PCA is mature intuitive, easy and fast to implement technique. It is of great use especially when the interpretability of the transformation is important
- Cares should be taken when applying PCA. The features should be centered and put on a common scale. The interpretation varies depending on the transformation of the data

https://stats.stackexchange.com/questions/53/pca-on-correlation-or-covariance

https://stats.stackexchange.com/questions/22329/how-does-centering-the-data-get-rid-of-the-intercept-in-regression-and-pca



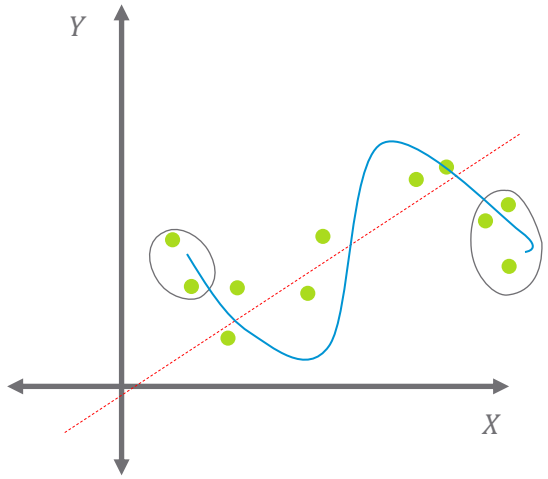
PCA: The Good, the Bad and the Ugly…

- PCA is mature intuitive, easy and fast to implement technique. It is of great use especially when the interpretability of the transformation is important
- Cares should be taken when applying PCA. The features should be centered and put on a common scale. The interpretation varies depending on the transformation of the data

https://stats.stackexchange.com/questions/53/pca-on-correlation-or-covariance

https://stats.stackexchange.com/questions/22329/how-does-centering-the-data-get-rid-of-the-intercept-in-regression-and-pca

- PCA cannot achieve good dimensions reduction when the features are not linearly related



t-Distributed Stochastic Neighbor Embedding

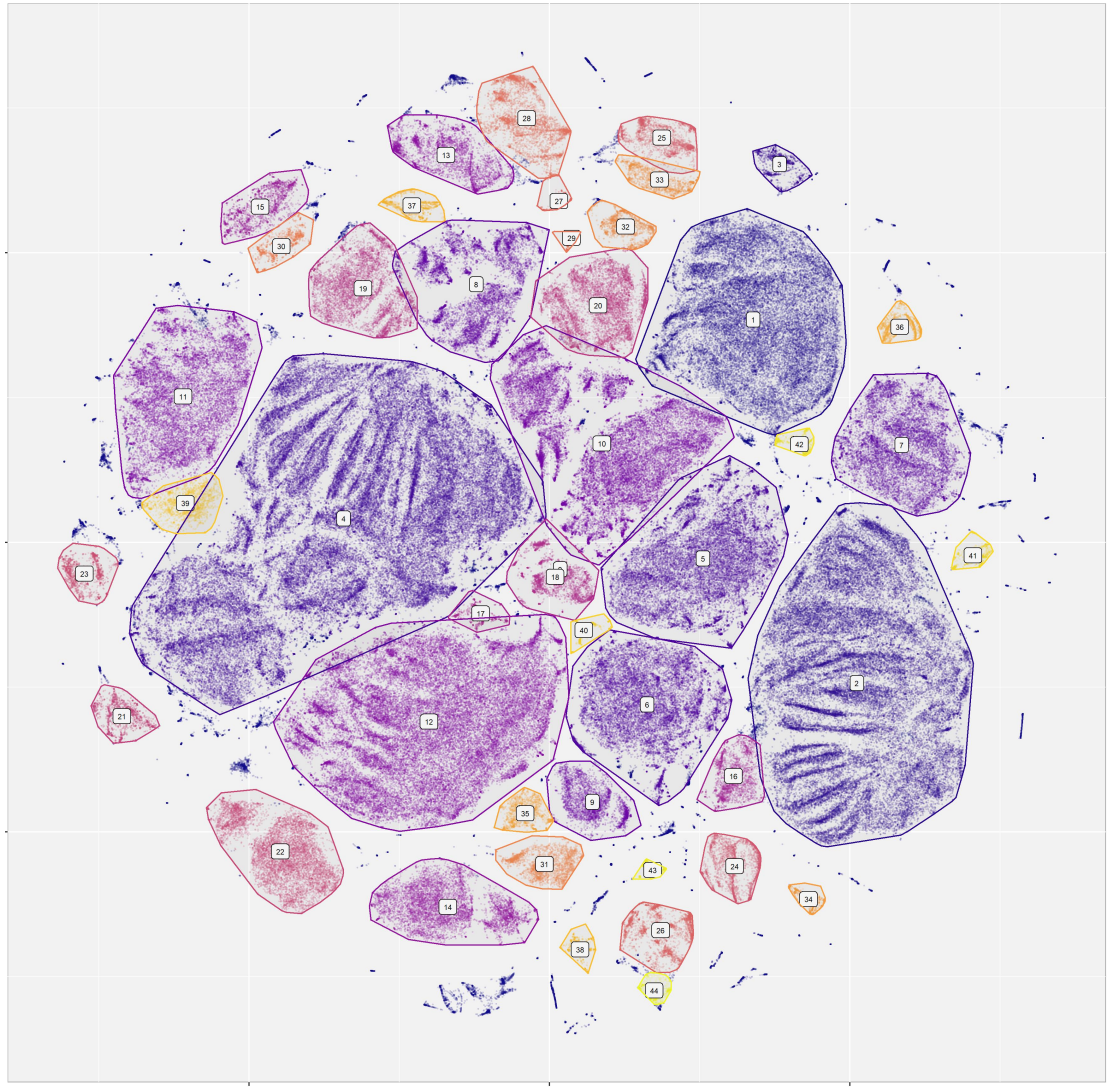
or simply t-SNE

t-Distributed Stochastic Neighbor Embedding

or simply t-SNE

*"…technique for dimensionality reduction that is particularly well suited for the **visualization** of high-dimensional datasets"*

ID	X_001	X_002	…	X_099	X_100
1	0	0.011	…	0.352	1
2	1	0.697	…	0.530	0
…	…	…	…	…	…
999	0	0.859	…	0.360	1
1000	0	0.250	…	0.303	0



“Black” box demystified

Our action plan:

We will construct simple example and will go through the process of how t-SNE plots 2d data on 1d plane

Adapted from StatQuest:

https://www.youtube.com/watch?v=NEaUSP4YerM

Consider 2d data

How can we project it to 1d

Out original 2D data:

Consider 2d data

How can we project it to 1d

Out goal is to project it on 1D:

Out original 2D data:

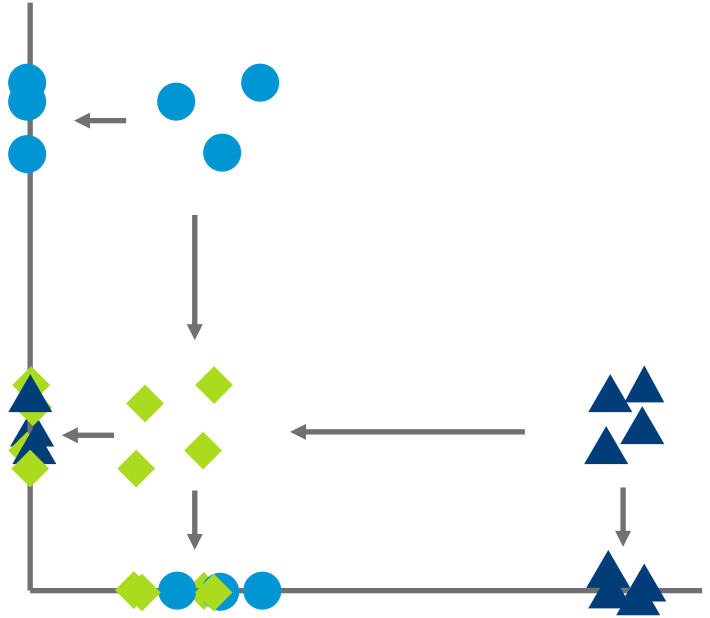
Consider 2d data

How can we project it to 1d

Out original data:

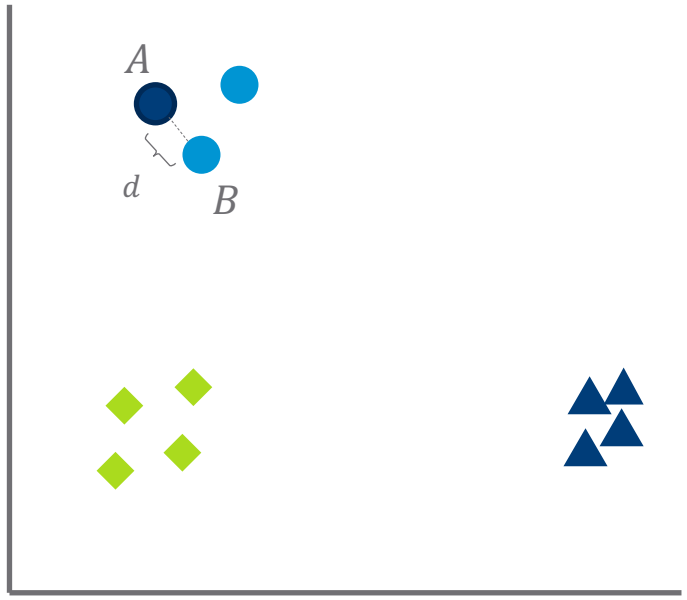
Out goal is to project it on 1D:

What we get when project directly on the axes



t-SNE Operates in two steps

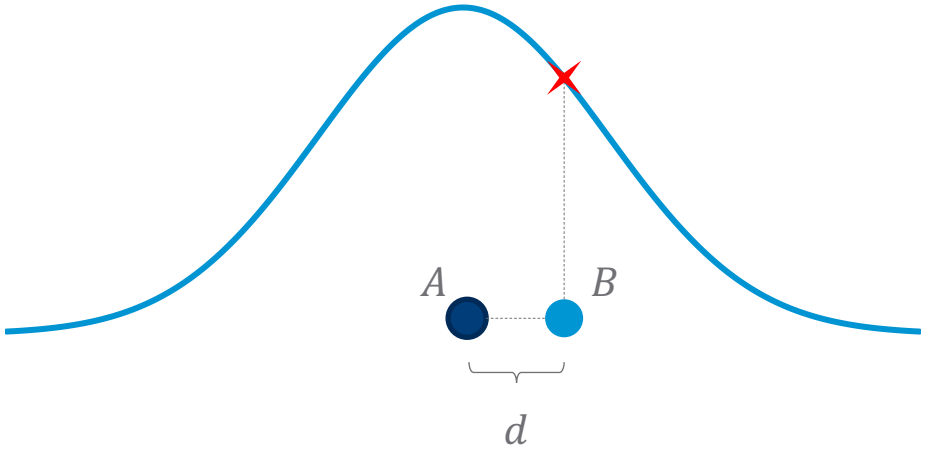
1st Building a graph in high dimensions



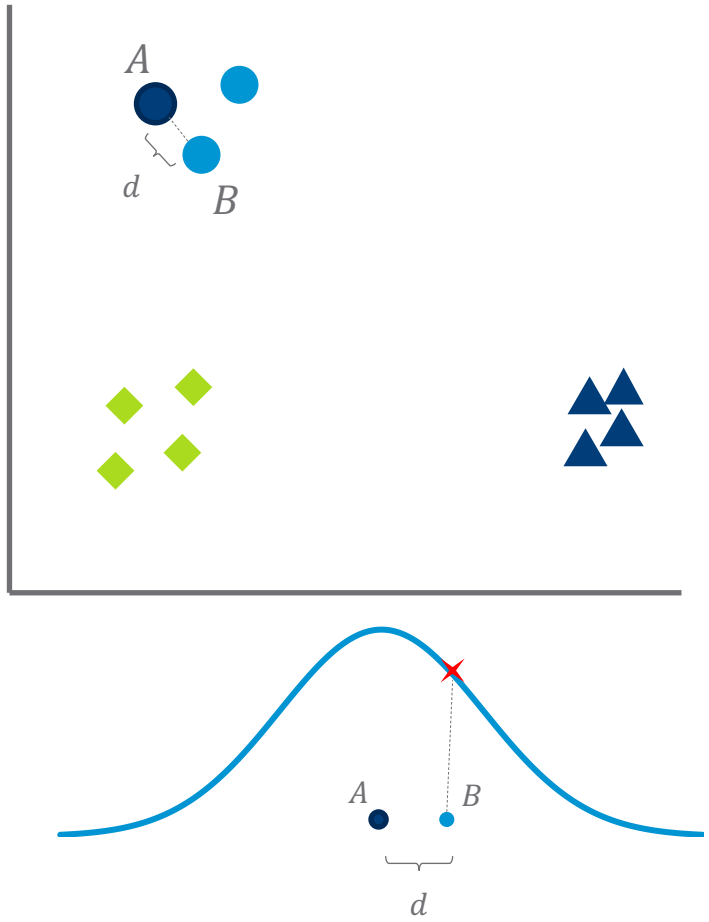
Step 1: Calculate similarities:

- Measure the distance
- Project the distance on a normal curve, centered around the point of interest

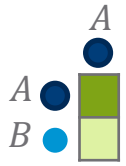
$$p_{j|i} = \exp\left(-\left\|x_i - x_j\right\|^2 / \sigma_i^2\right)$$



Building a graph in high dimensions

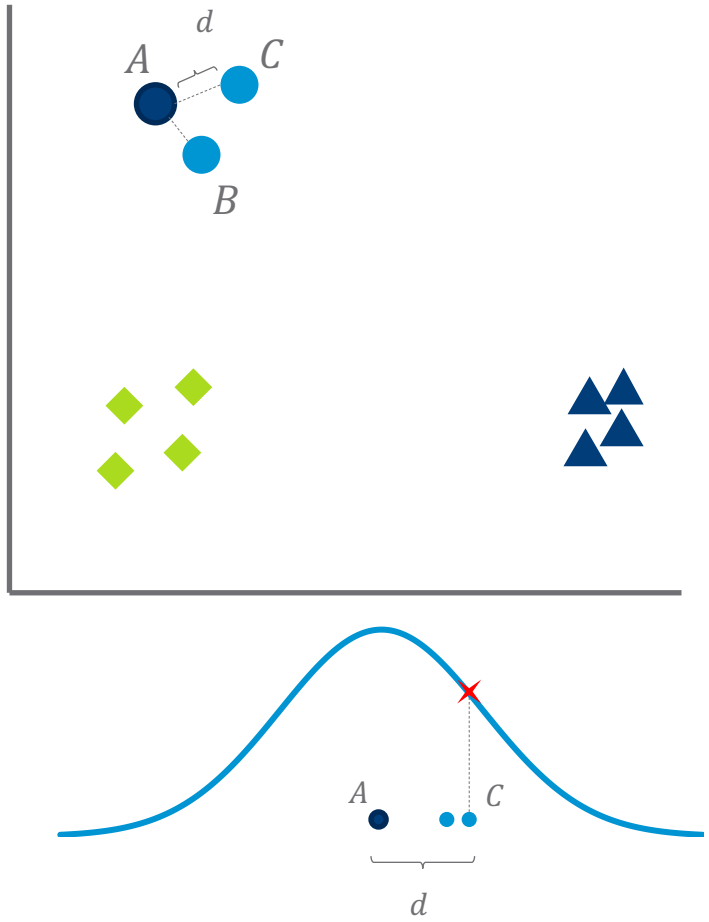


- Record the similarity into the similarity matrix

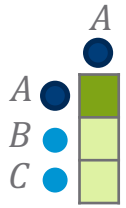


High similarity Low similarity

Building a graph in high dimensions

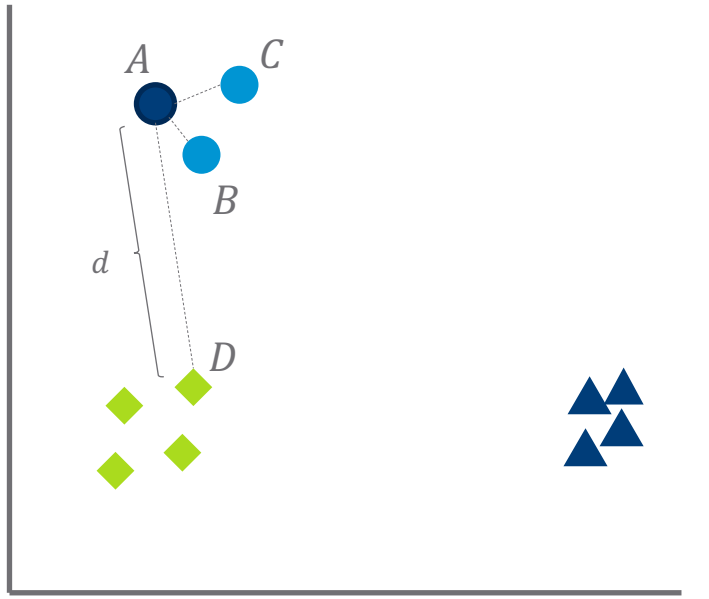


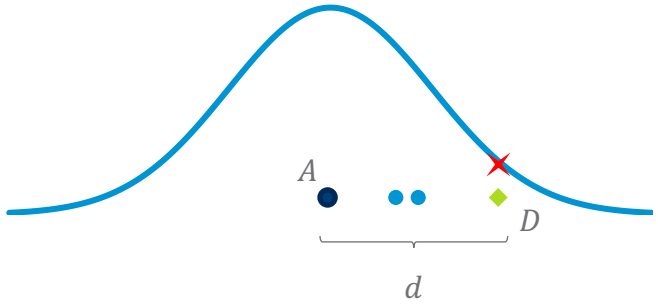
❖ Do this for point C with respect to point A



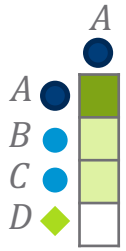
High similarity Low similarity

Building a graph in high dimensions



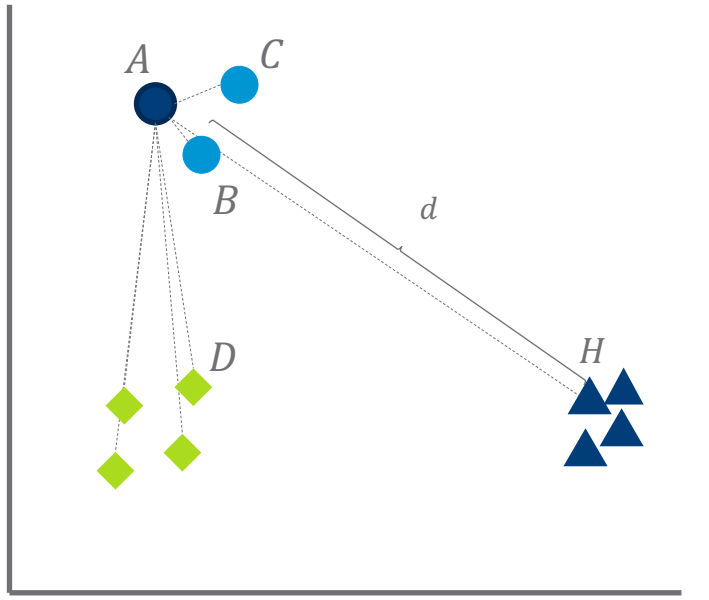


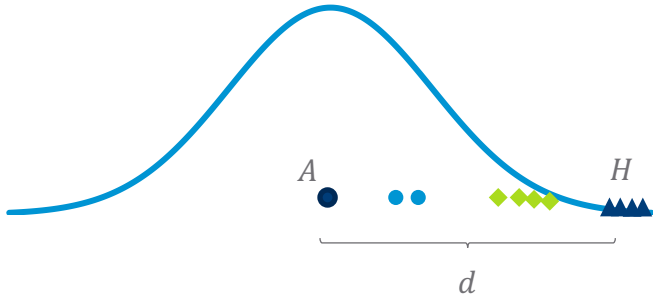
- Do this for point D with respect to point A



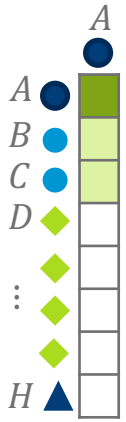
High similarity Low similarity

Building a graph in high dimensions



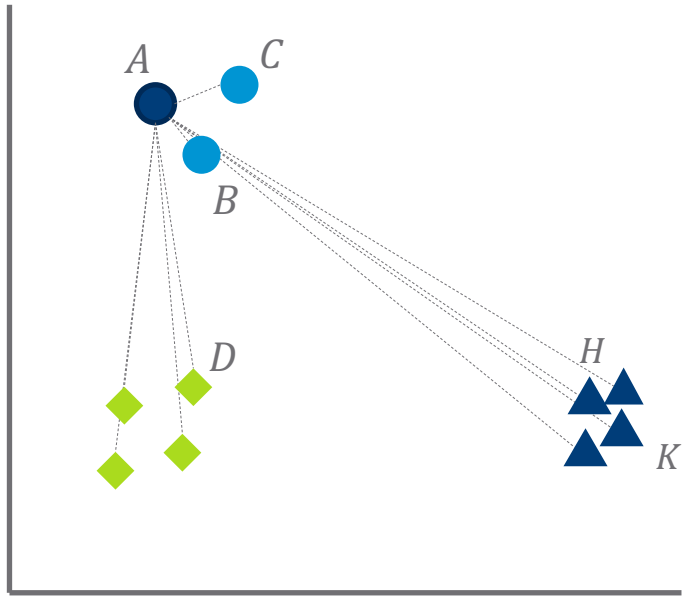


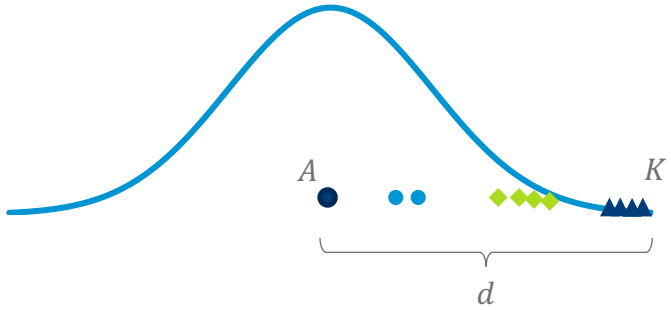
- Do this for point H with respect to point A



High similarity Low similarity

Building a graph in high dimensions





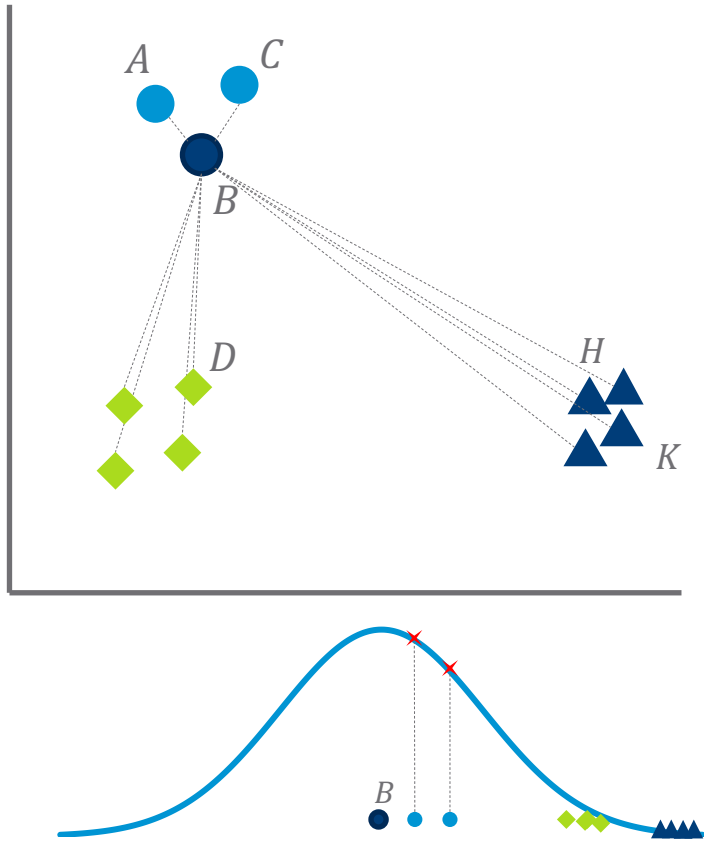
❖ Until we visit all points with respect to point A

$$p_{j|i} = \frac{\exp\left(-\left\|x_i - x_j\right\|^2 / 2\sigma_i^2\right)}{\sum_{i'} \exp\left(-\left\|x_i - x_{j'}\right\|^2 / 2\sigma_i^2\right)}$$

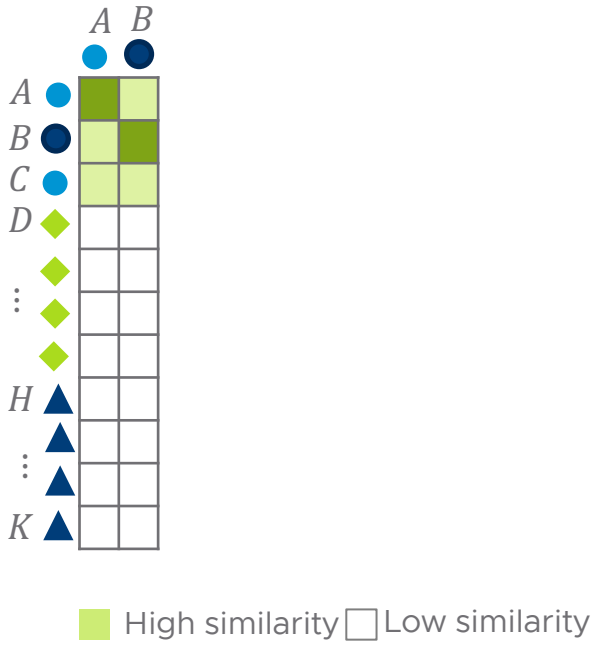
High similarity Low similarity

Once we build the similarity for all points with respect to A, we scale them, so all add up to 1 and continue …

Building a graph in high dimensions

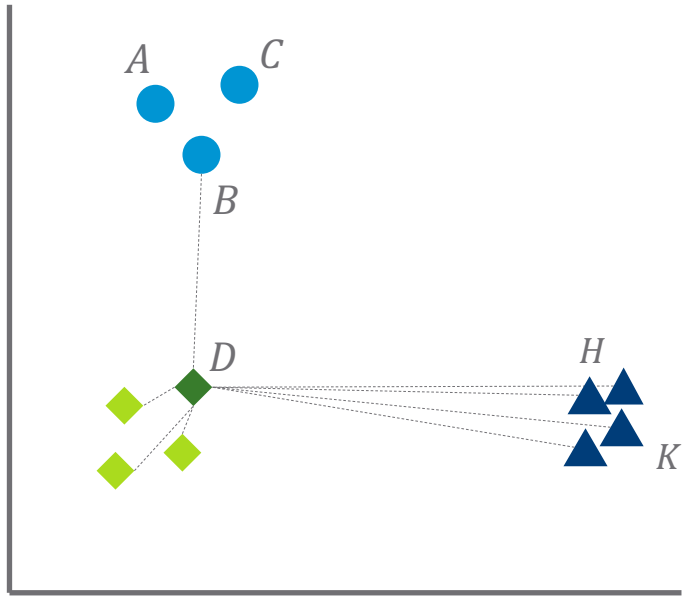


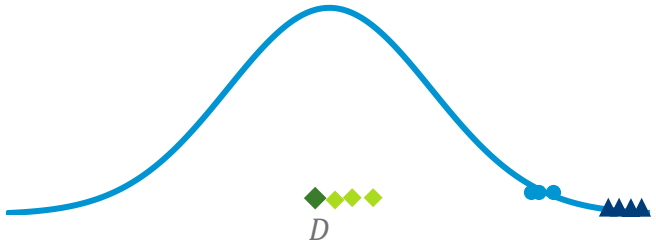
Build similarity matrix based on the original 2D data (Pi)



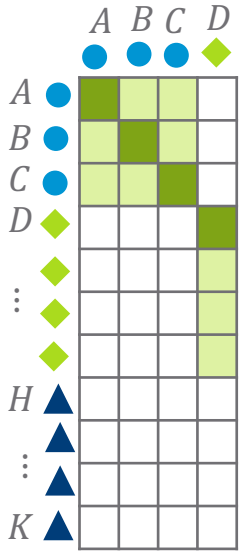
For point B

Building a graph in high dimensions





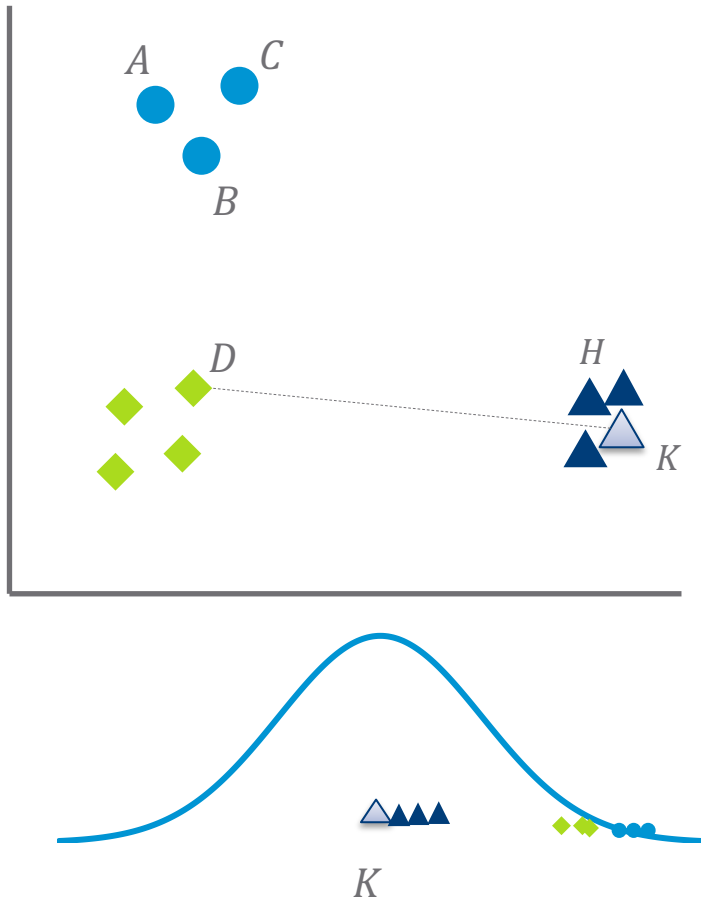
Build similarity matrix based on the original 2D data (Pi)



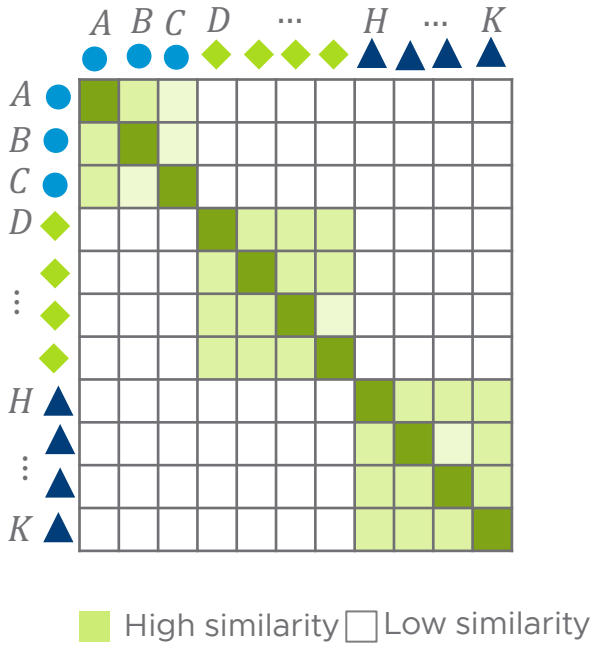
High similarity Low similarity

.. and C

Building a graph in high dimensions



Build similarity matrix based on the original 2D data (Pi)

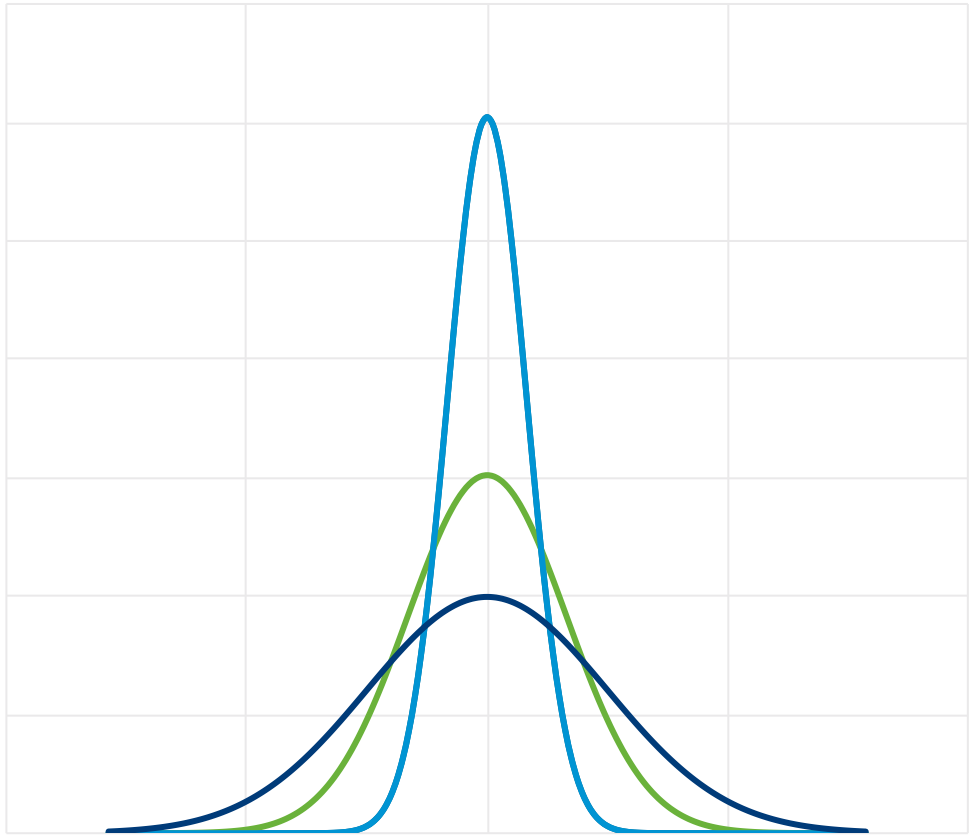


.. and all the points

Building a graph in high dimensions

Perplexity

But the shape of the normal curve depends on σ as well …

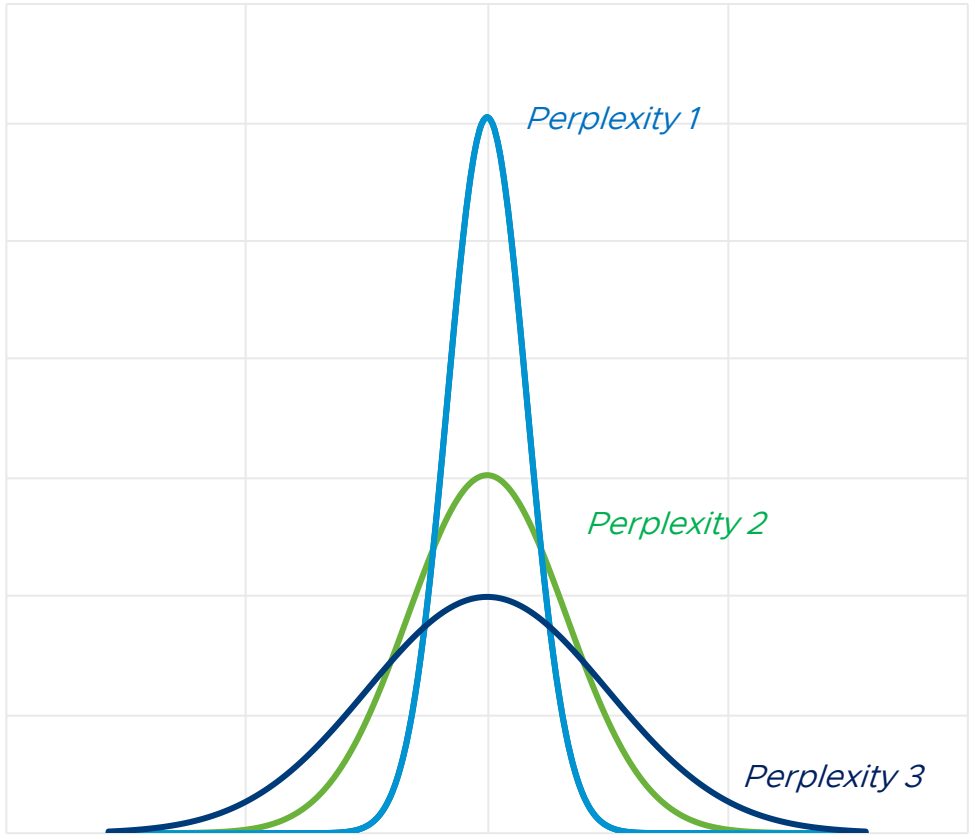


$$p_{j|i} = \exp\left(-\|x_i - x_j\|^2 / \sigma_i^2\right)$$

Building a graph in high dimensions

Perplexity

But the shape of the normal curve depends on σ as well …



And σ depends on perplexity …

$$Perplexity_1 < Perplexity_2 < Perplexity_3$$

Building a graph in high dimensions

Perplexity

Perplexity is a tunable parameter

$$Perplexity = 2^{\sum_j p_{j|i} \log_2 p_{j|i}}$$

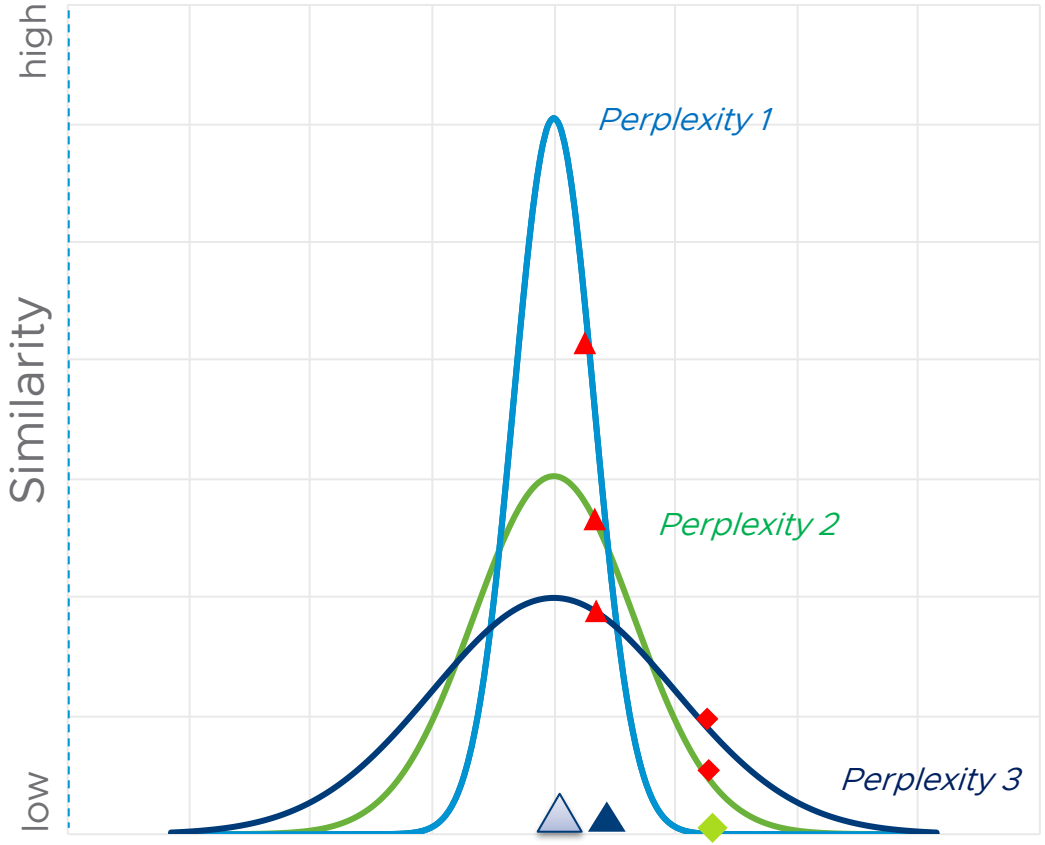
It controls the strength of local versus global similarity

A way to think of perplexity is as number of closest points in respect to the point of interest that are used to calculate sigma

Building a graph in high dimensions

Perplexity

But the shape of the normal curve depends on $\mathcal{O}$ as well …



And $\mathcal{O}$ depends on perplexity …

$$Perplexity_1 < Perplexity_2 < Perplexity_3$$

▲ Is more similar to △ under *Perplexity 1* compared to *Perplexity 2* and *Perplexity 3*

◆ Is less dissimilar to △ under *Perplexity 3* compared to *Perplexity 1* and *Perplexity 2*

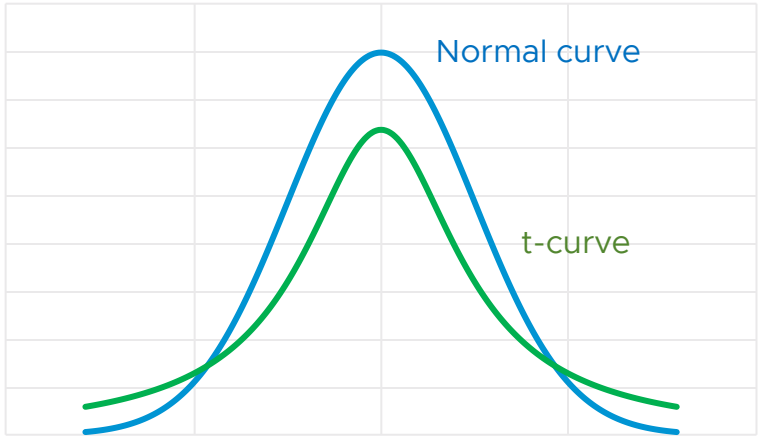
Second Step:

Build and optimize low dimensional representation

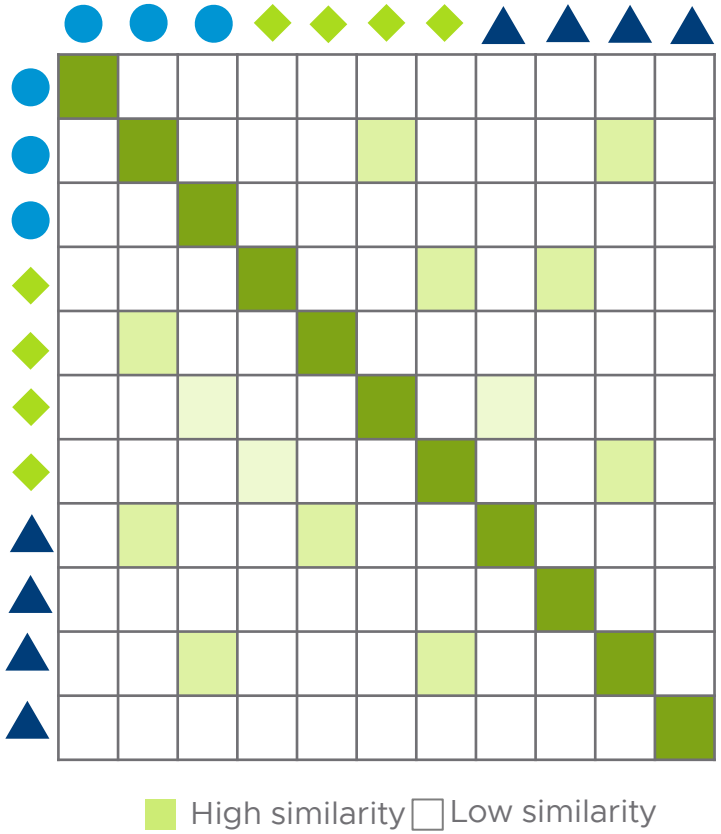
Create second matrix Q_i that represent the similarity in 1D

Initially the points are positioned randomly on the number line

Instead of normal curve this time similarity is calculated with t-distribution (where comes the t in t-SNE)



$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}}$$

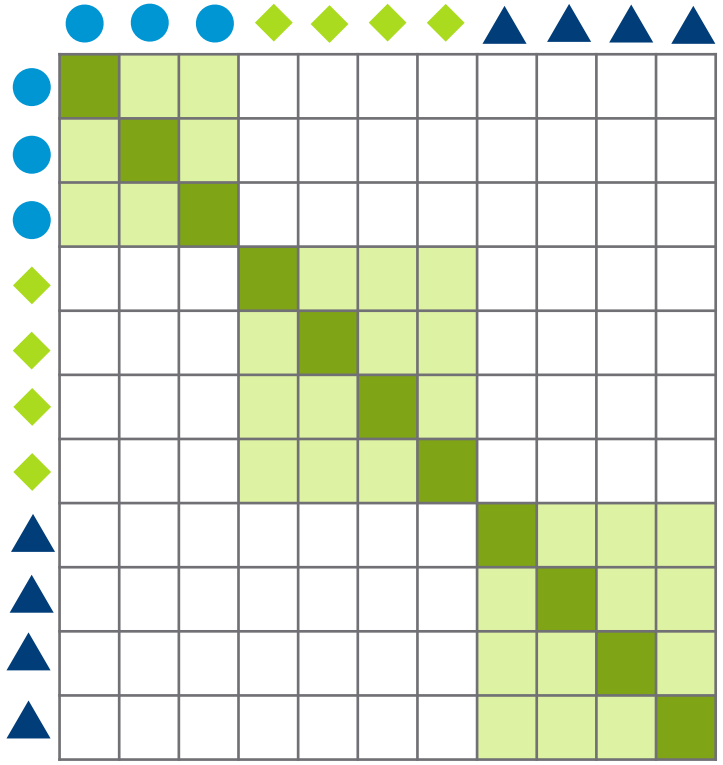


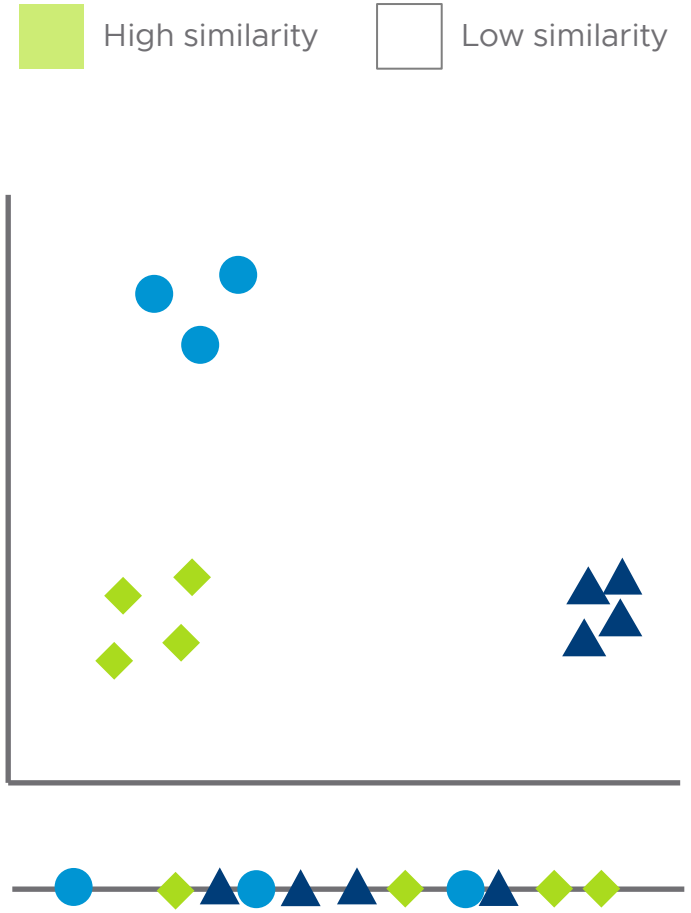
vmware

Second Step:

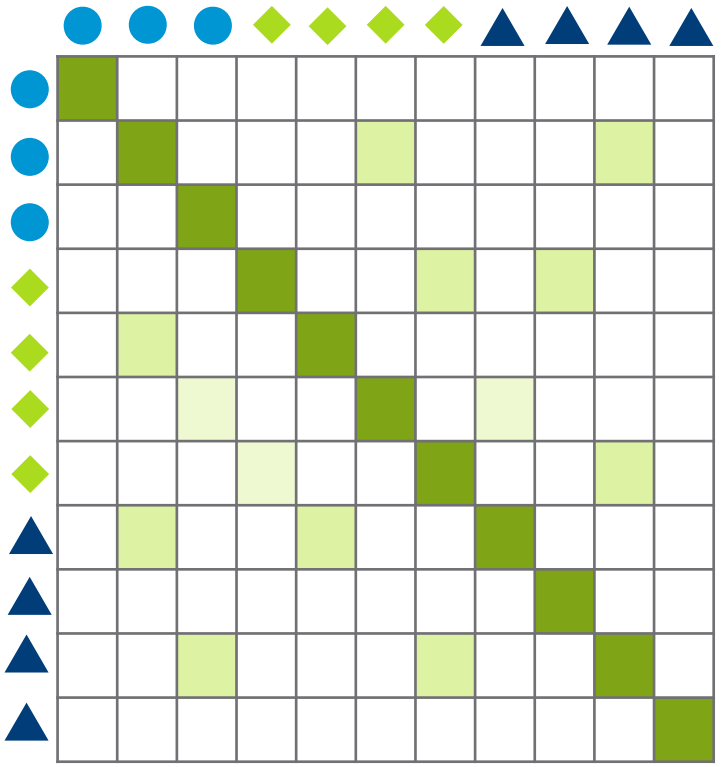
Optimize

Original space P_i



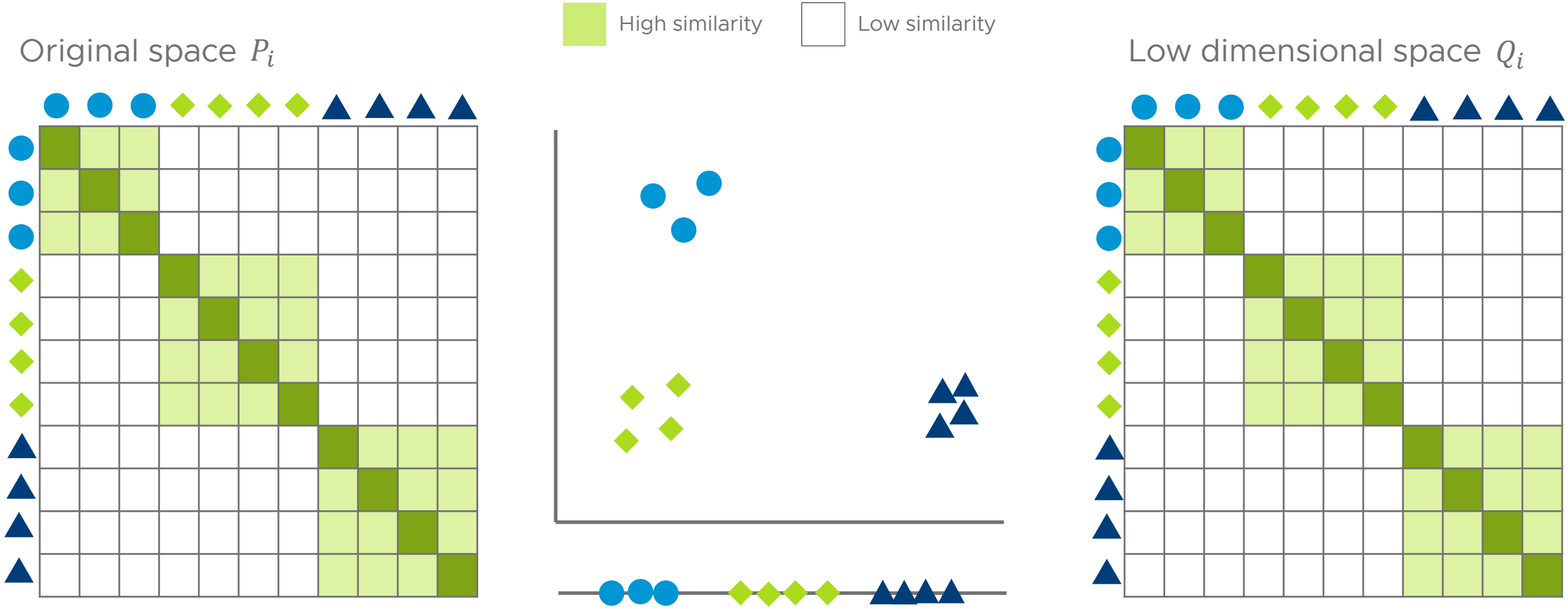


Low dimensional space Q_i



Second Step:

Optimize



Second step:

Optimize:

What we do is run gradient descent trying to optimize KL Divergence

$$KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

And the gradient is:

$$\frac{\partial KL}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)\left(1 + \|y_i - y_j\|^2\right)^{-1}$$

Some notes

- t-SNE is infamous for eliminating global distances. Clusters that appears very far away from each other might end being up very similar
- The t-SNE algorithm doesn’t always produce similar output on successive runs
- Cluster sizes in a t-SNE plot mean nothing
- Random noise doesn’t always look random (especially true for low perplexity values)

https://distill.pub/2016/misread-tsne/

Density Based Spatial Clustering of Applications with Noise

... or simply DBSCAN

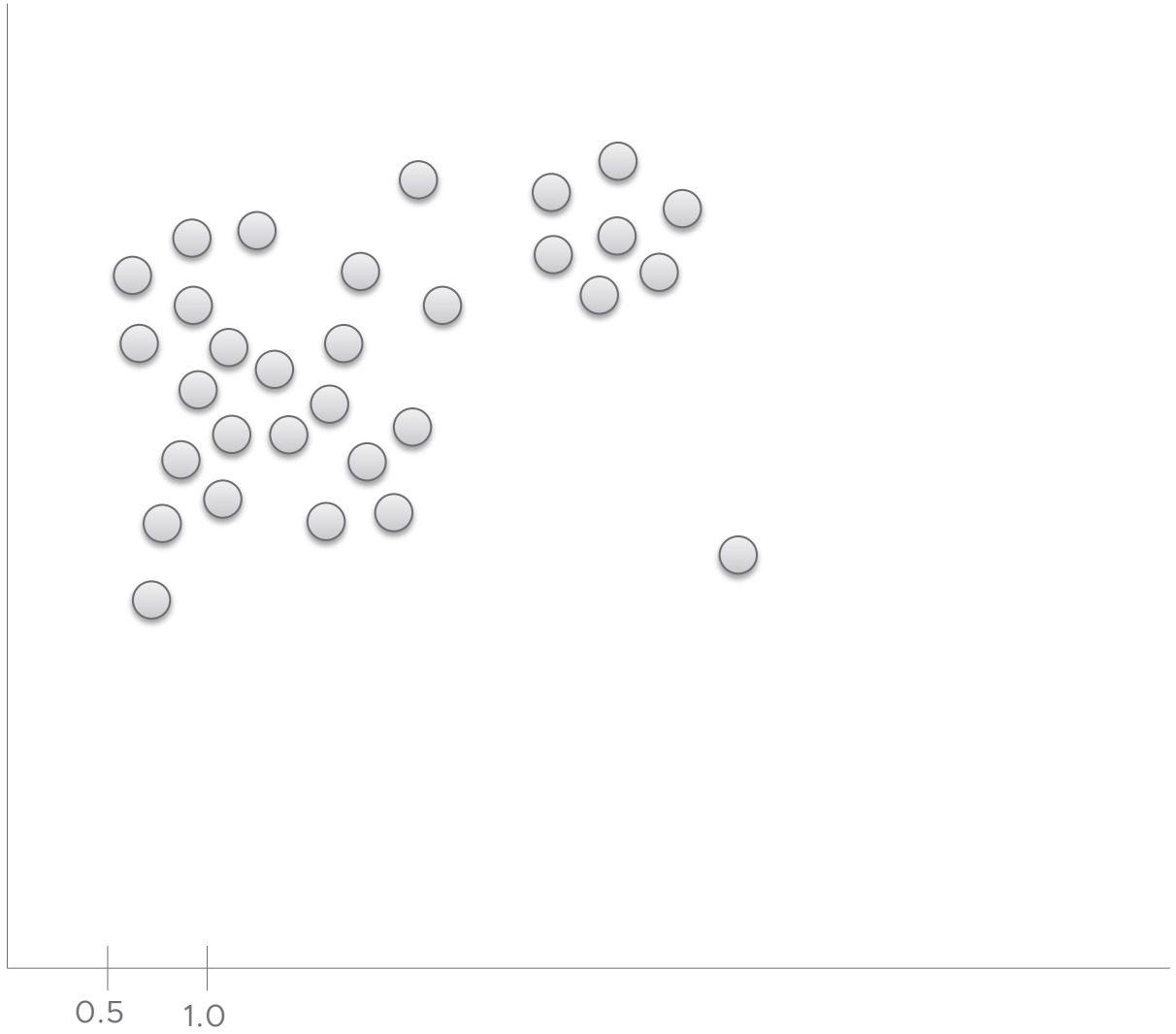
DBSCAN

- DBSCAN locates regions of high density, separated from another with low density.

- Low and high density is defined with respect to two hyperparameters:
  - Epsilon (ε) - specifies how close points should be to each other to be considered a part of a cluster
  - min # of points - how many neighbors a point should have to be included into a cluster

DBSCAN

Simple example



$\underline{Hyperparameters}$:

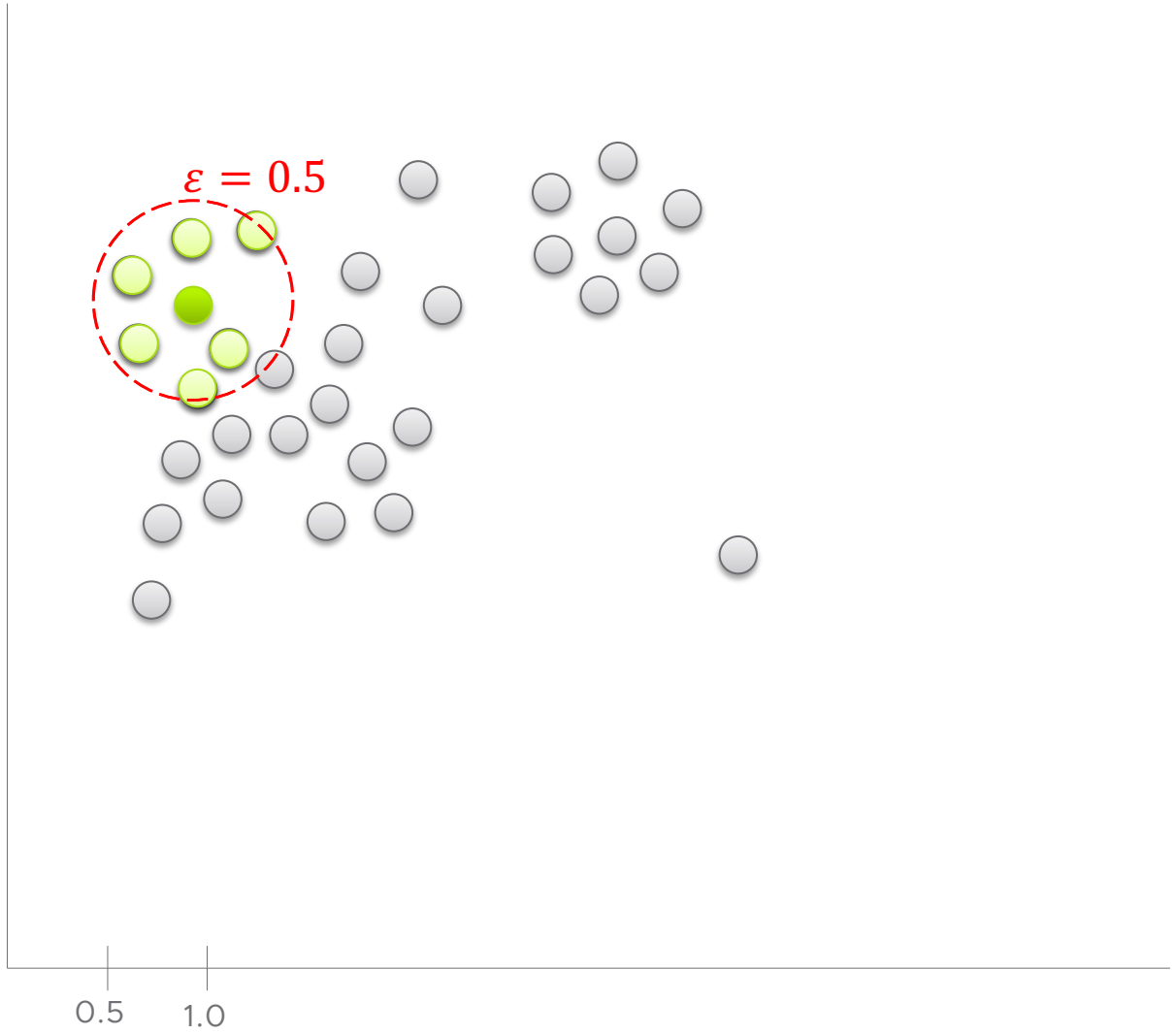
$\varepsilon = 0.5$

$\mathrm{m}inPoints = 5$

vmware

DBSCAN

Core points



$\underline{Hyperparameters}$:

$\varepsilon = 0.5$

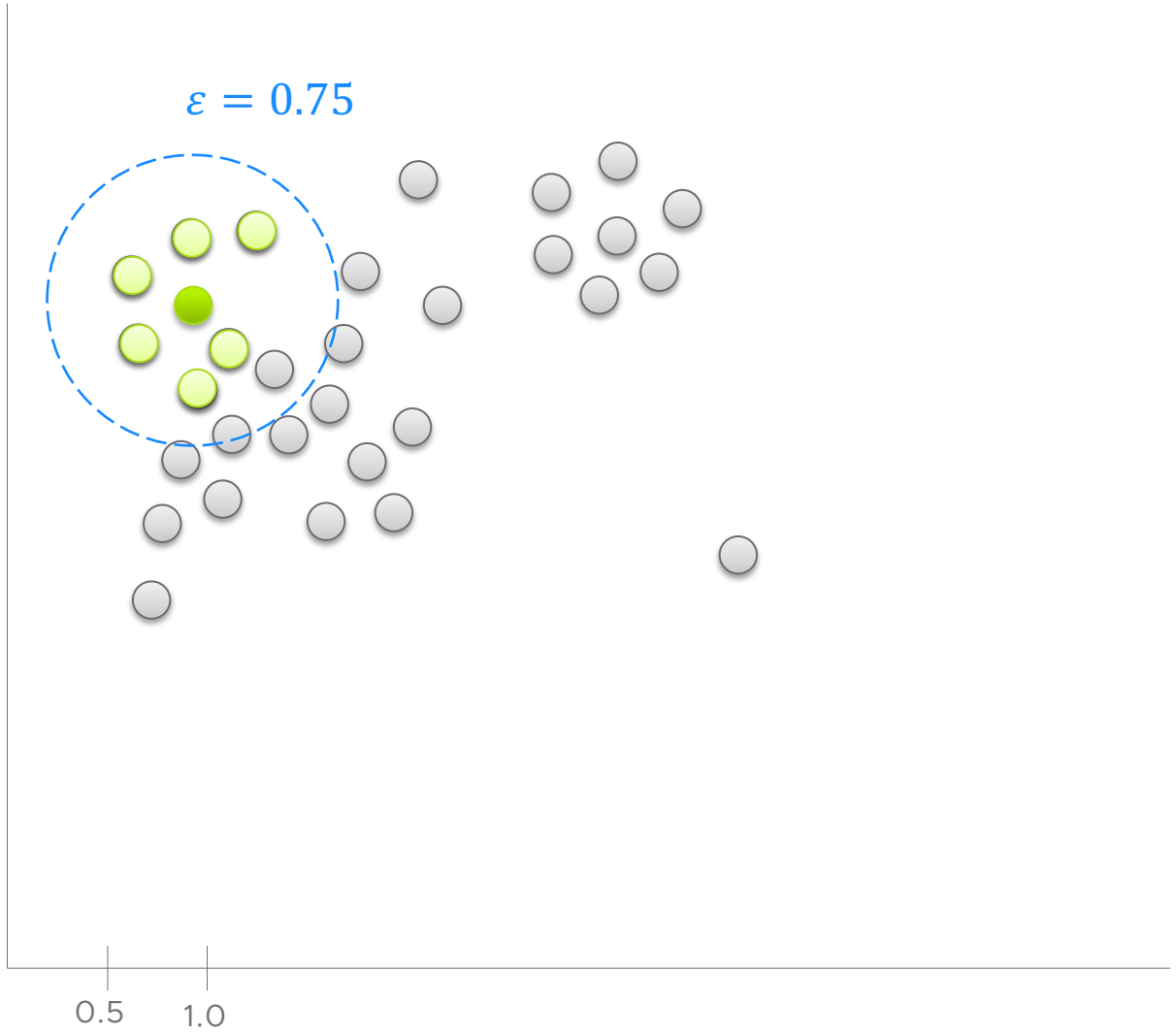
$\mathrm{m}inPoints = 5$

Core point

Within the neighborhood of a core point cluster is formed.

DBSCAN

Core points



$\underline{Hyperparameters}$:

$\varepsilon = 0.5$

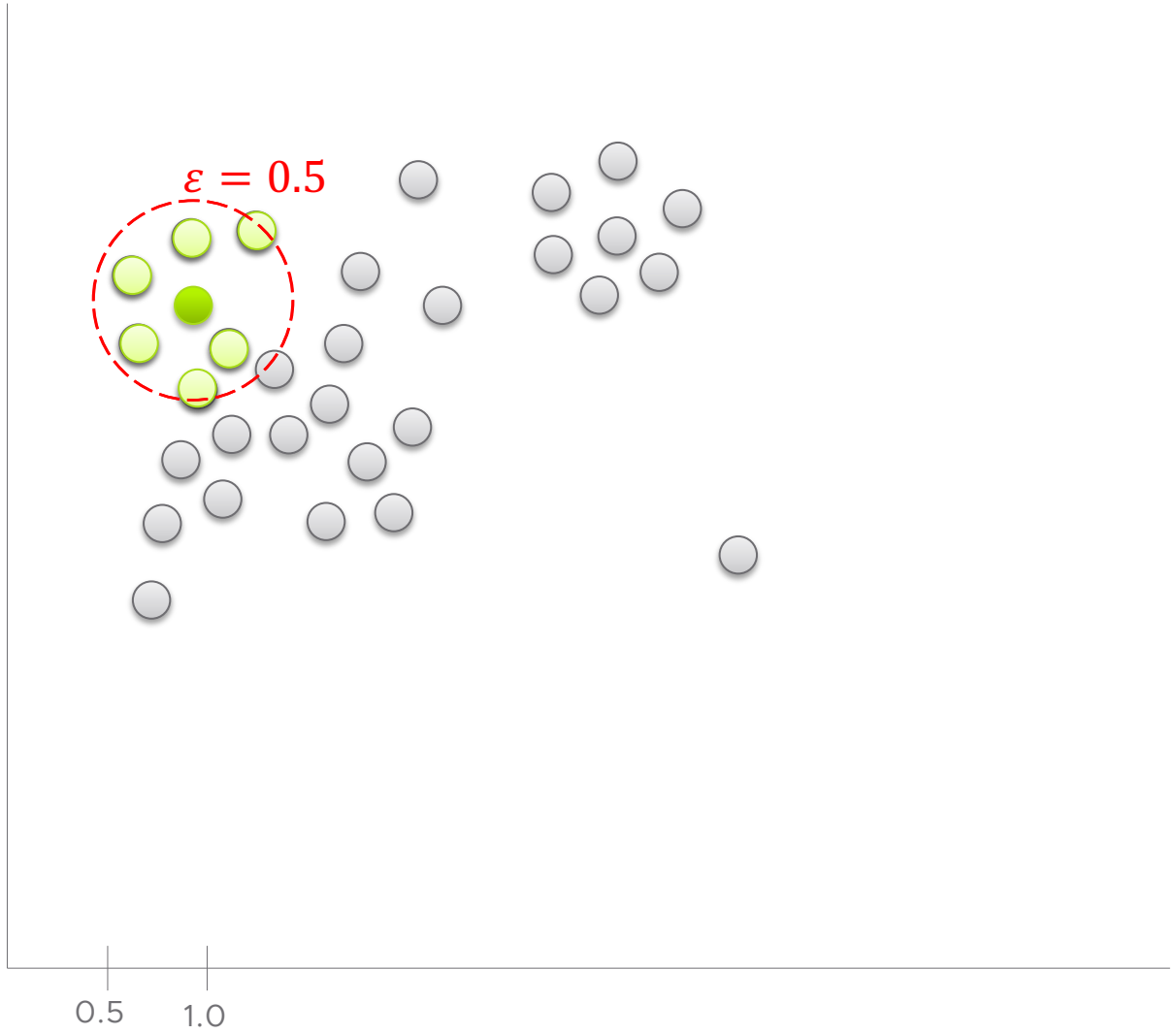
$\mathrm{m}inPoints = 5$

Core point

Within the neighborhood of a core point cluster is formed.

DBSCAN

Core points



$\underline{Hyperparameters}$:

$\varepsilon = 0.5$

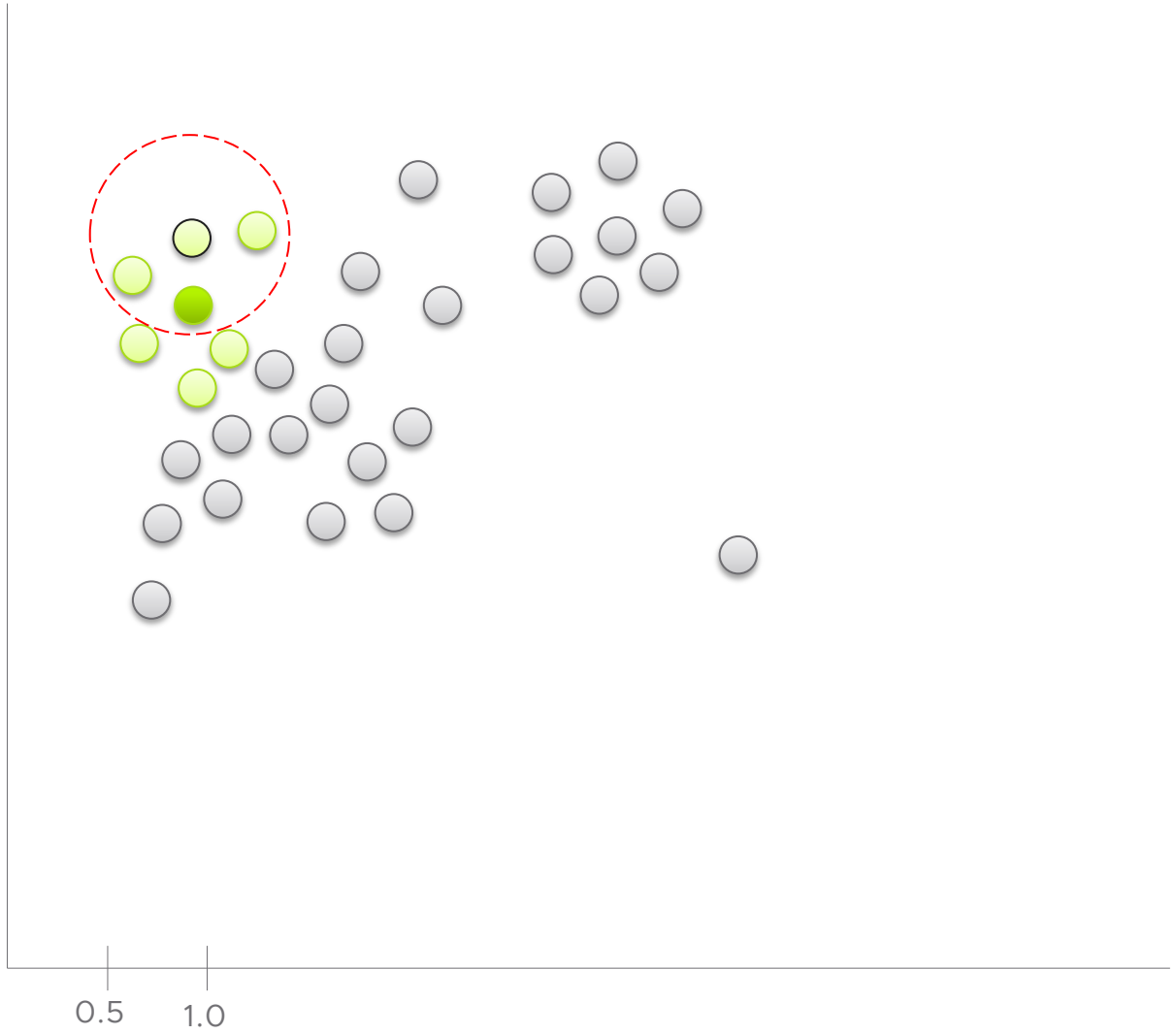
$\mathrm{m}inPoints = 5$

Core point

Within the neighborhood of a core point cluster is formed.

DBSCAN

Boarder points



$$\underline{Hyperparameters}:$$

$$\varepsilon = 0.5$$

$$\mathrm{m}inPoints = 5$$

Core point

Border point

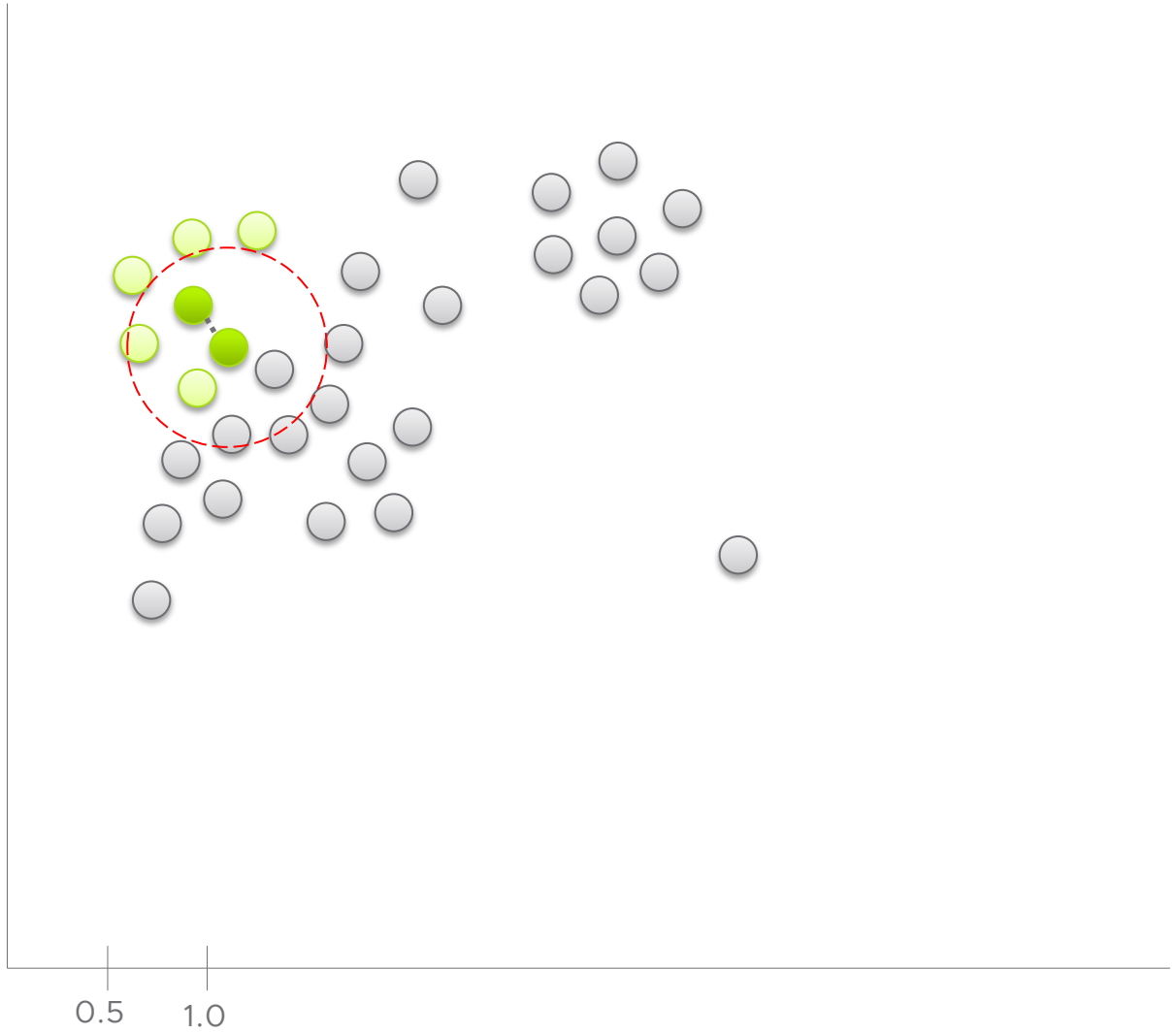
The cluster is growing by inspecting the points from the neighborhood of the core points

If given point does not contain enough points within its radius, but contains core point, it is labeled as border point

vmware

DBSCAN

Extending cluster



$\underline{Hyperparameters}$:

$\varepsilon = 0.5$

$\mathrm{m}inPoints = 5$

Core point

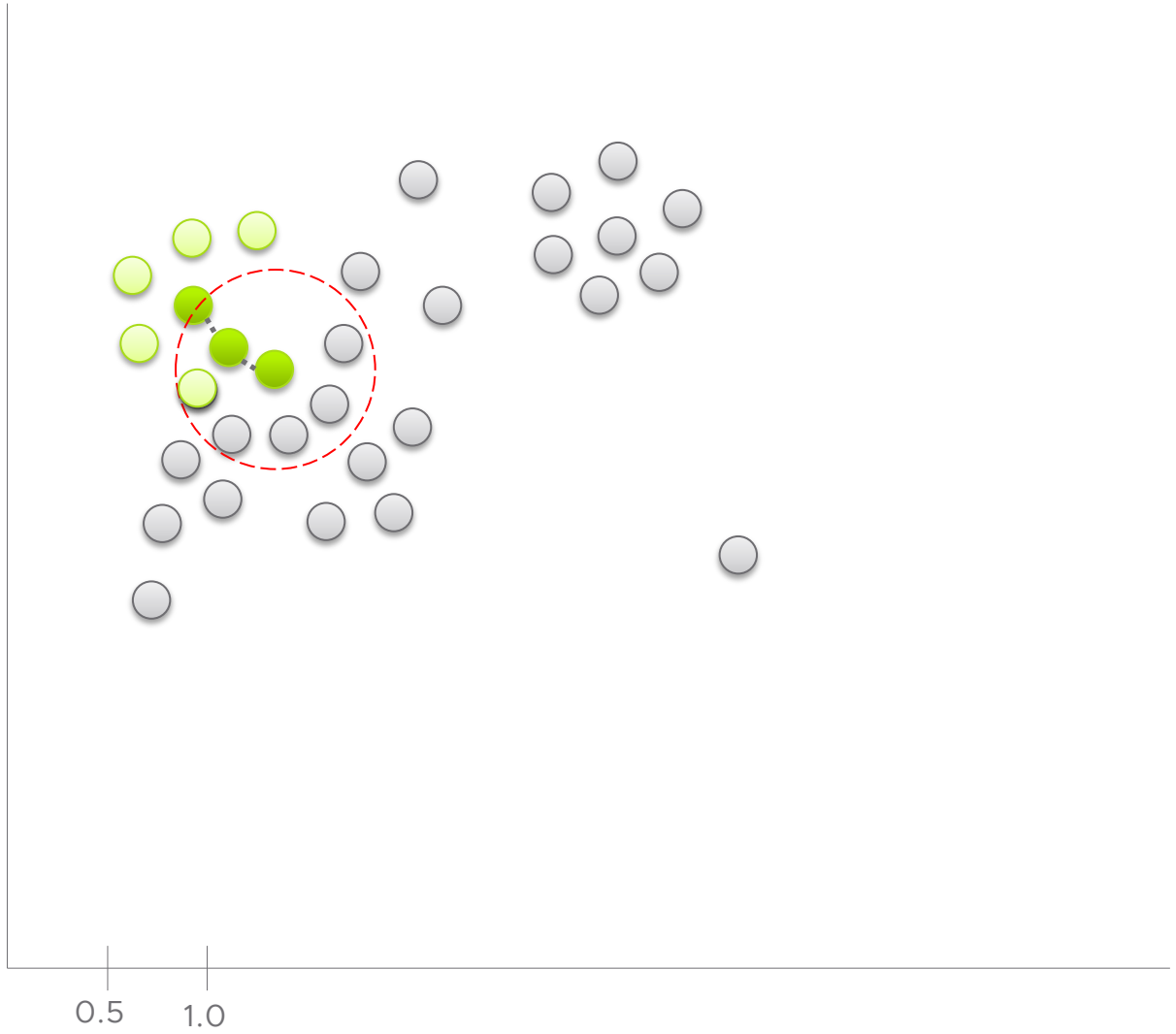
Border point

If two core points are within each other radiuses they become connected and extend the cluster

vmware

DBSCAN

Extending cluster



$\underline{Hyperparameters}$:

$\varepsilon = 0.5$

$\mathrm{m}inPoints = 5$

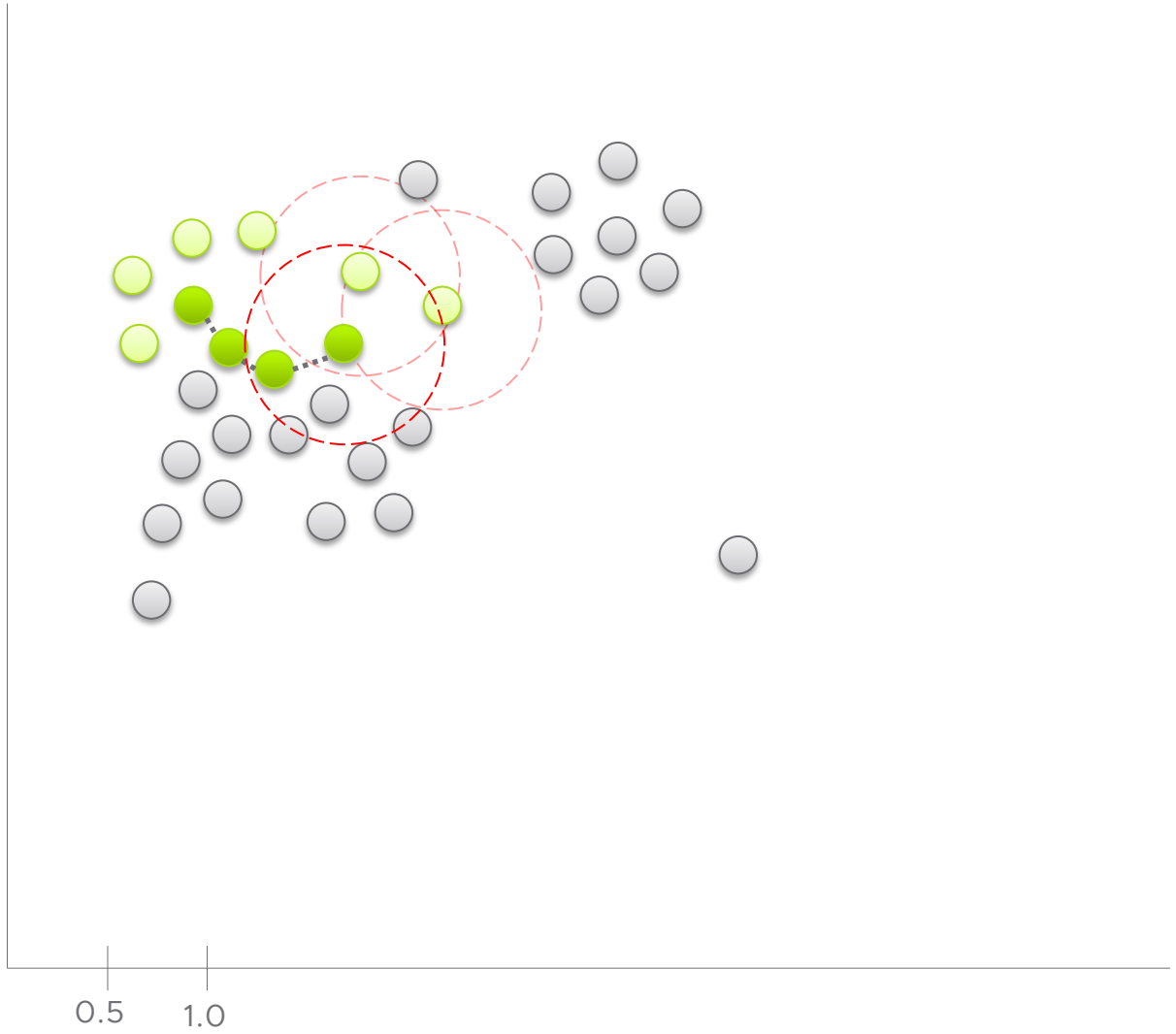
Core point

Border point

vmware

DBSCAN

Extending cluster



$\underline{Hyperparameters}$:

$\varepsilon = 0.5$

$\mathrm{m}inPoints = 5$

Core point

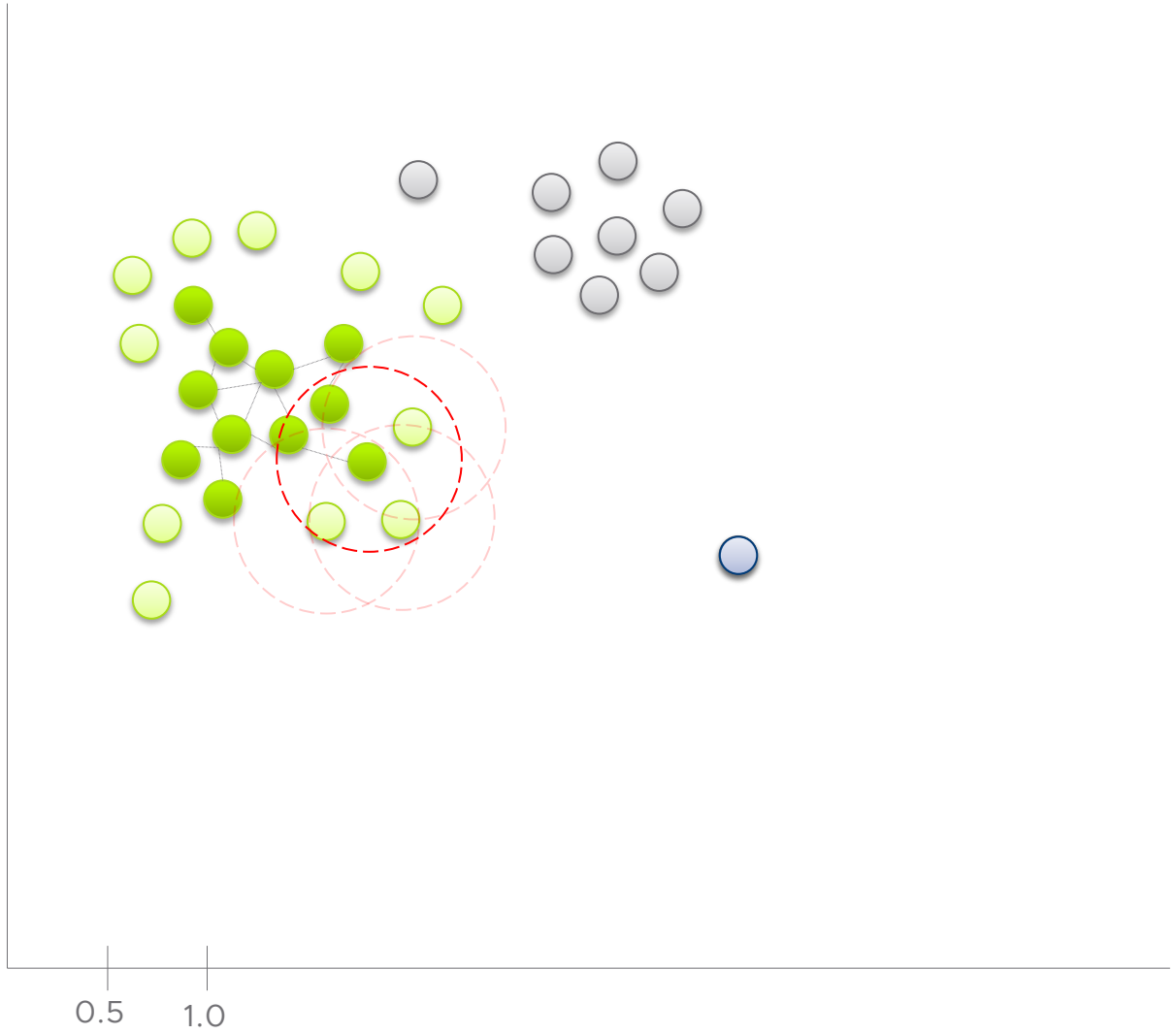
Border point

Border points define the edge of the cluster since they cannot be used to reach more points

vmware

DBSCAN

Noise



$\underline{Hyperparameters}$:

$\varepsilon = 0.5$

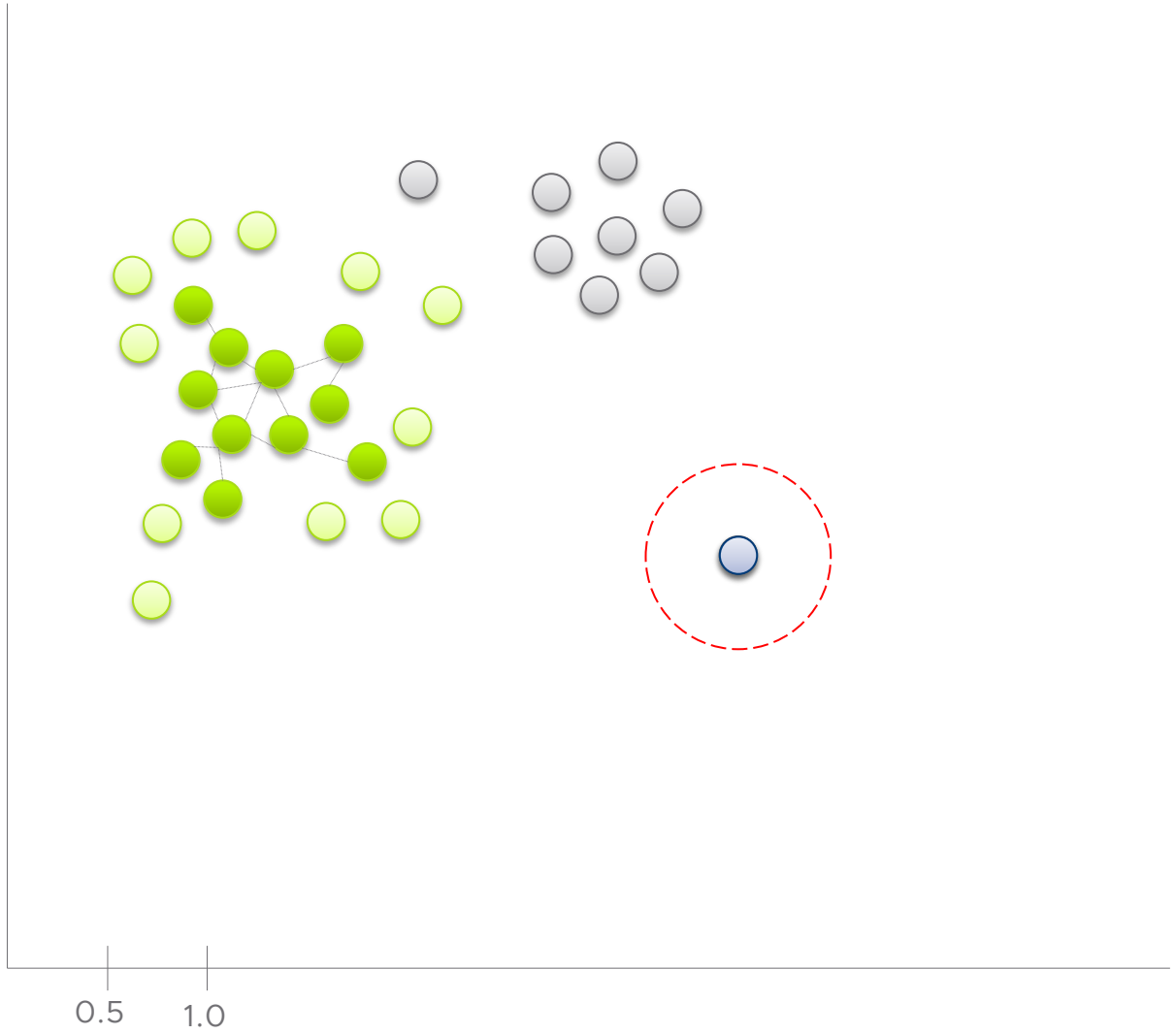
$\mathrm{m}inPoints = 5$

- Core point
- Border point
- Noise

Points, that are neither boarder nor core points are considered as noise

DBSCAN

Noise



$\underline{Hyperparameters}$:

$\varepsilon = 0.5$

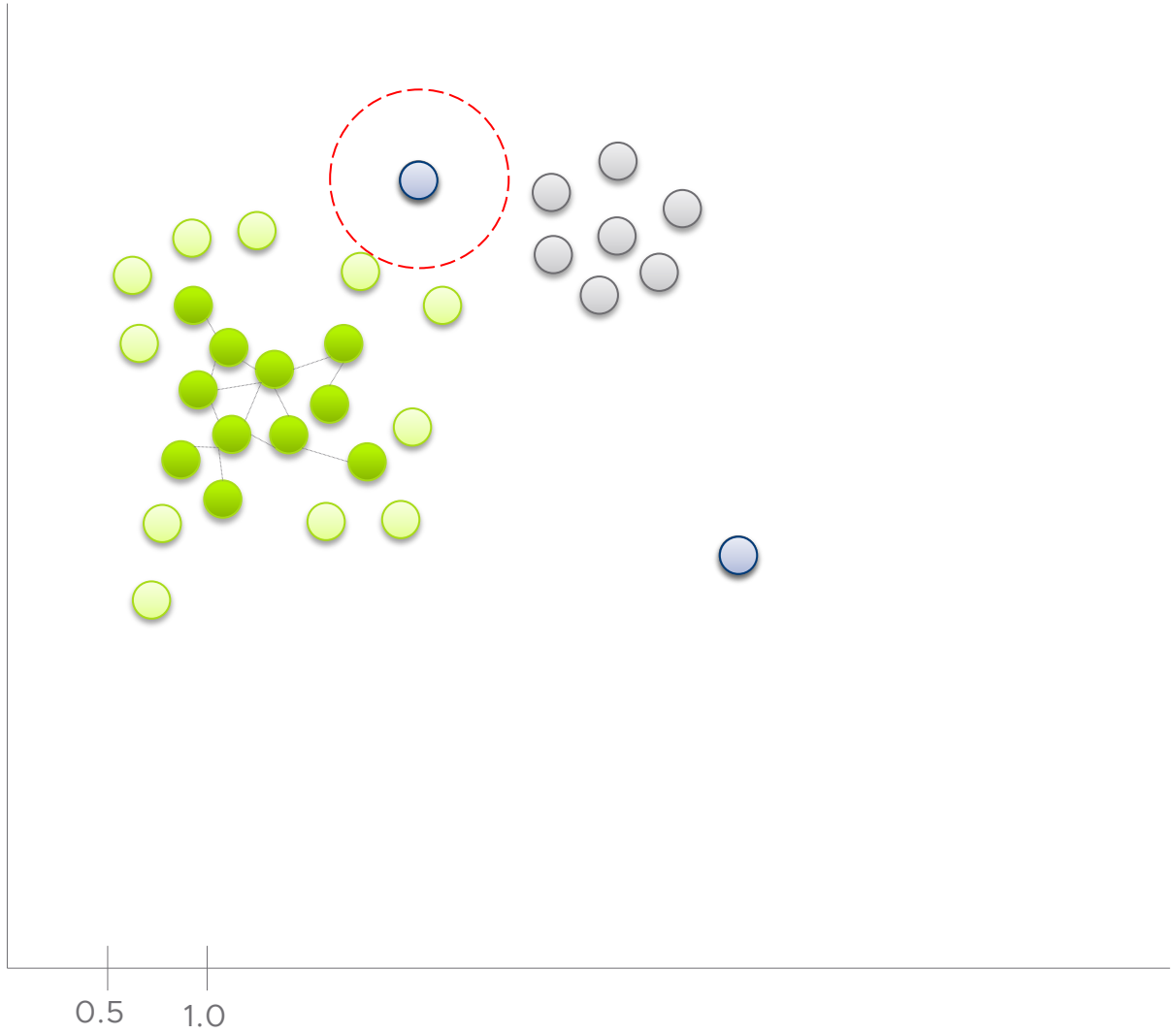
$\mathrm{m}inPoints = 5$

- Core point
- Border point
- Noise

Points, that are neither boarder nor core points are considered as noise

DBSCAN

Noise



$\underline{Hyperparameters}$:

$\varepsilon = 0.5$

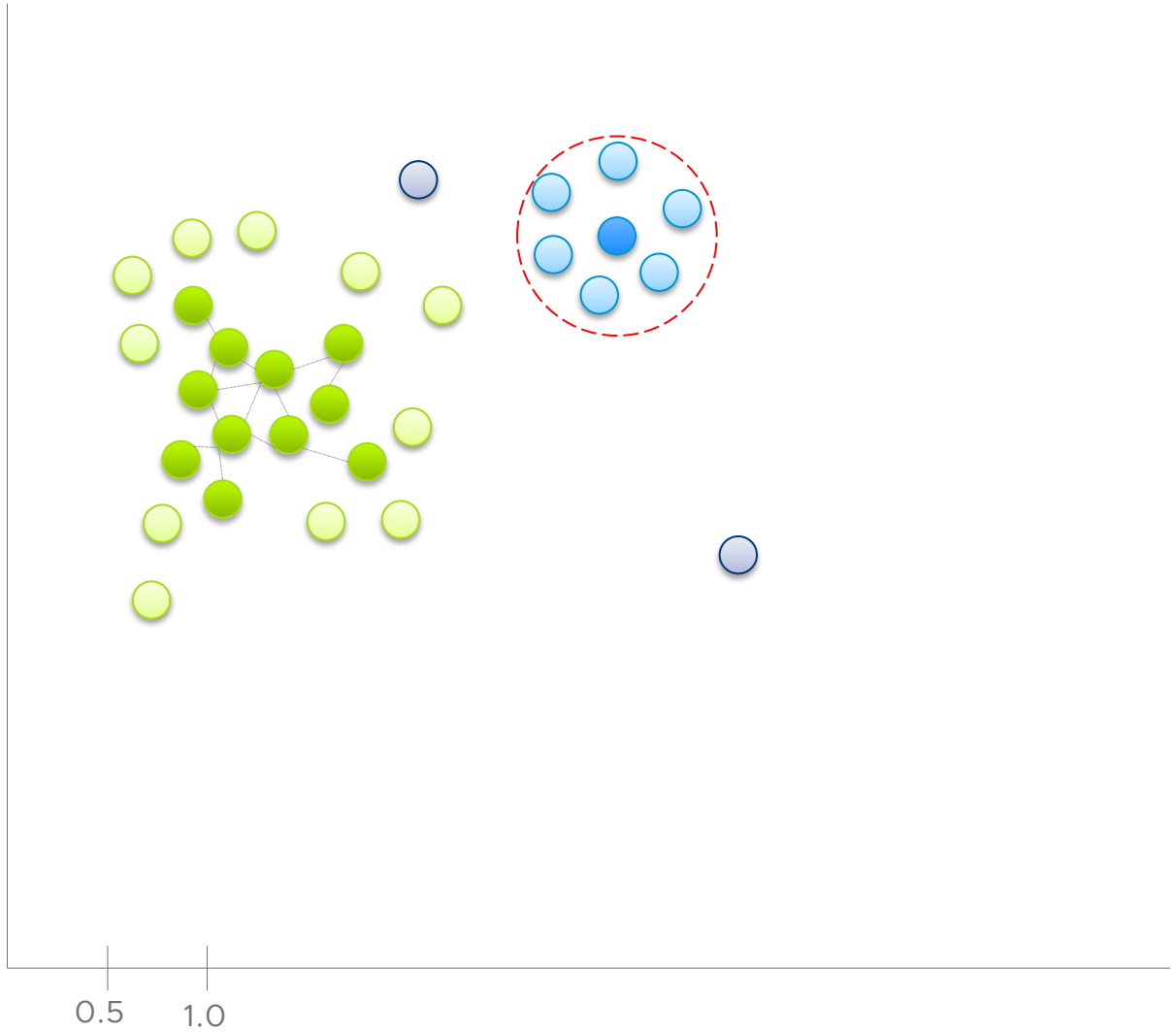
$\mathrm{m}inPoints = 5$

- Core point
- Border point
- Noise

Points, that are neither boarder nor core points are considered as noise

DBSCAN

Final clusters



$\underline{Hyperparameters}$:

$\varepsilon = 0.5$

$\mathrm{m}inPoints = 5$

Cluster 1

Cluster 2

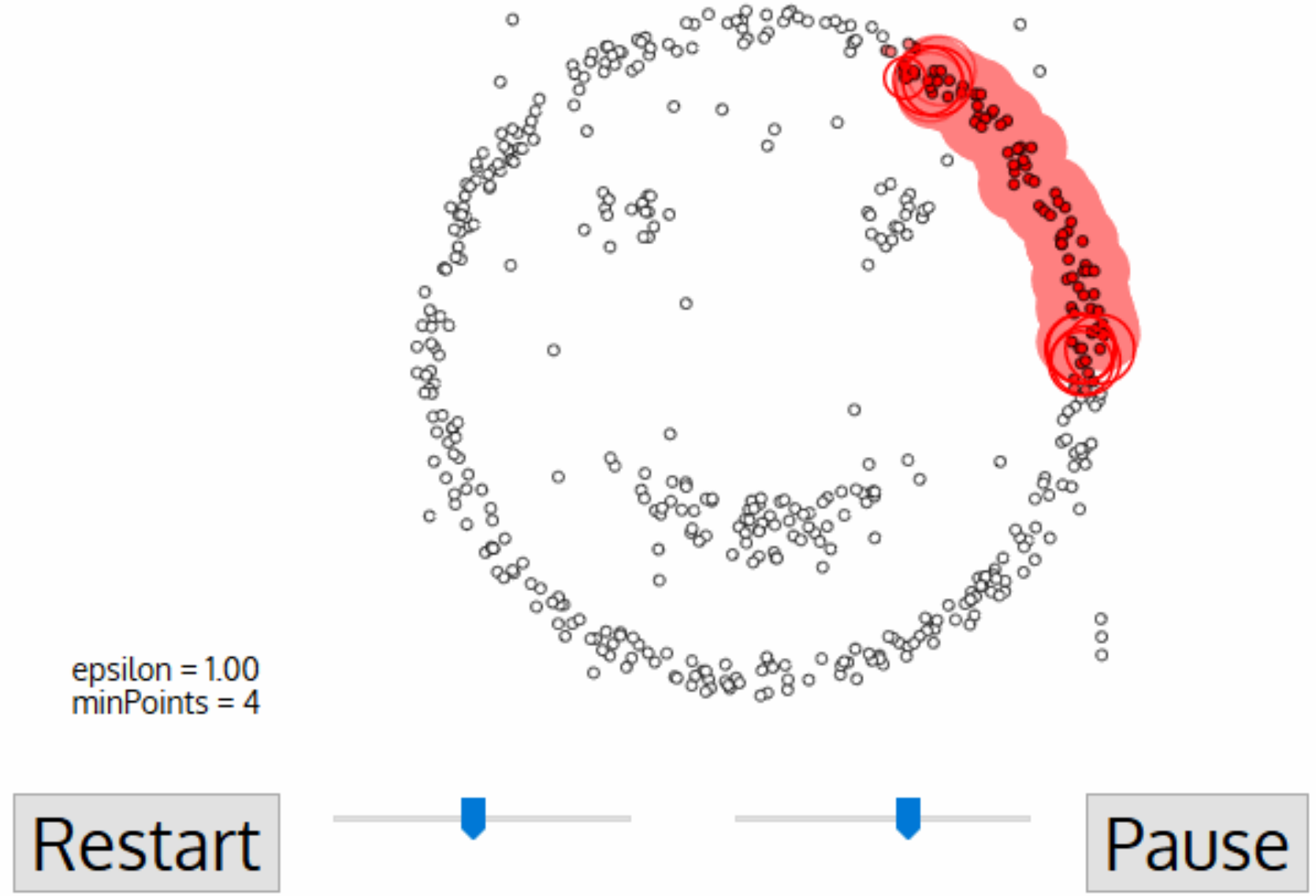
Noise

vmware

The good, the bad ….

- DBSCAN does not require one to specify the number of clusters in the data a priori
- Can find arbitrary clusters of arbitrary shape
- Has notion of noise and it is robust to outliers

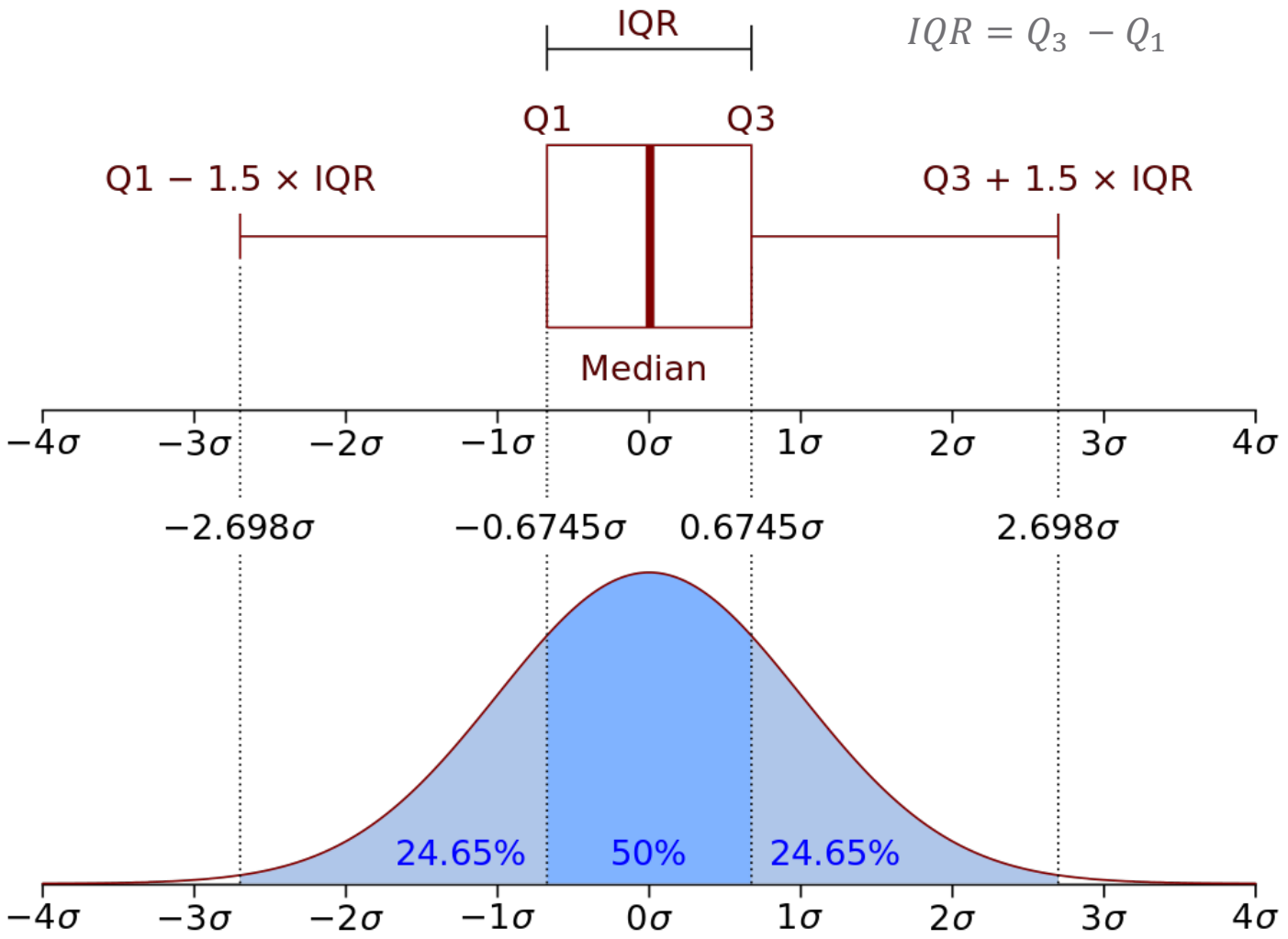
- Heavily depends on the distance measure and might be very hard to tune under high dimension
- Extremely sensitive to hyperparameter values – small changes often can substantially change the clustering output
- minPts-epsilon cannot be chosen appropriately for all clusters



https://www.quora.com/Is-there-any-alternative-algorithm-similar-to-the-K-means-algorithm-for-clustering

Boxplot

Compared to normal distribution



Lunch Break

vmware®

Missing data imputation

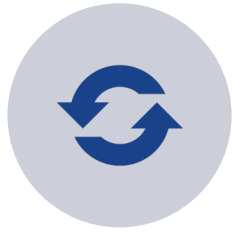
vmware®

Missing Data Imputation

Statistical tests do not tolerate missing data

Use some univariate imputation methods to impute/ replace those missing values.

Interpolation



Imputation by predefined constant

Summary statistics

Moving average

Time Series Features

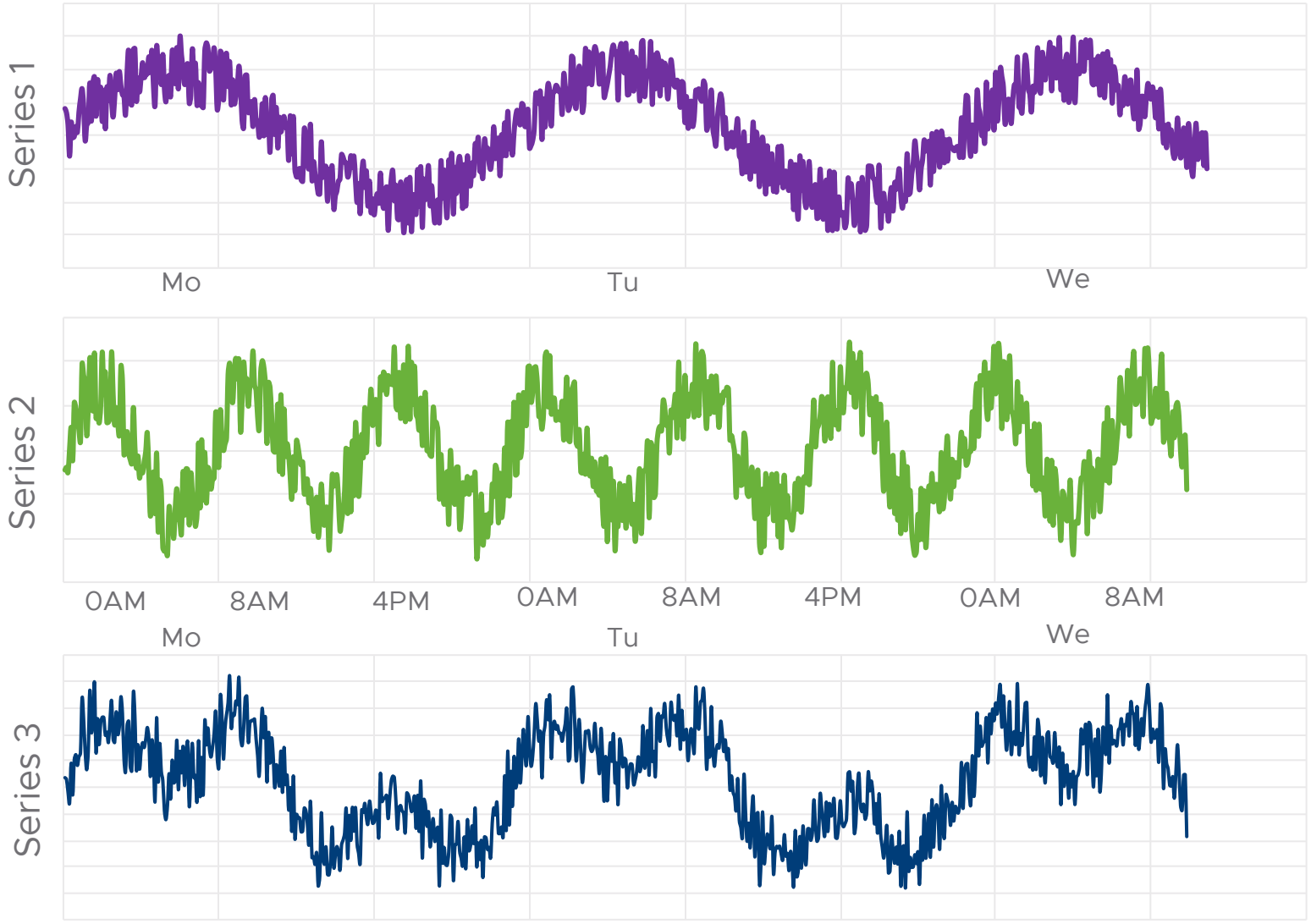
PROS
More expressive & retain the information encoded in the time domain of the data

CONS
Costly to compute hard to interpret - thus profiling cluster requires domain knowledge not trivial to ensure comparability some of the tests are very sensitive to data inconsistencies

Spectral Analysis

Running in cycles

Consider some data



Data	Daily Cycle	8 Hours Cycle
Series 1	High	Low
Series 2	Low	High
Series 3	High	High

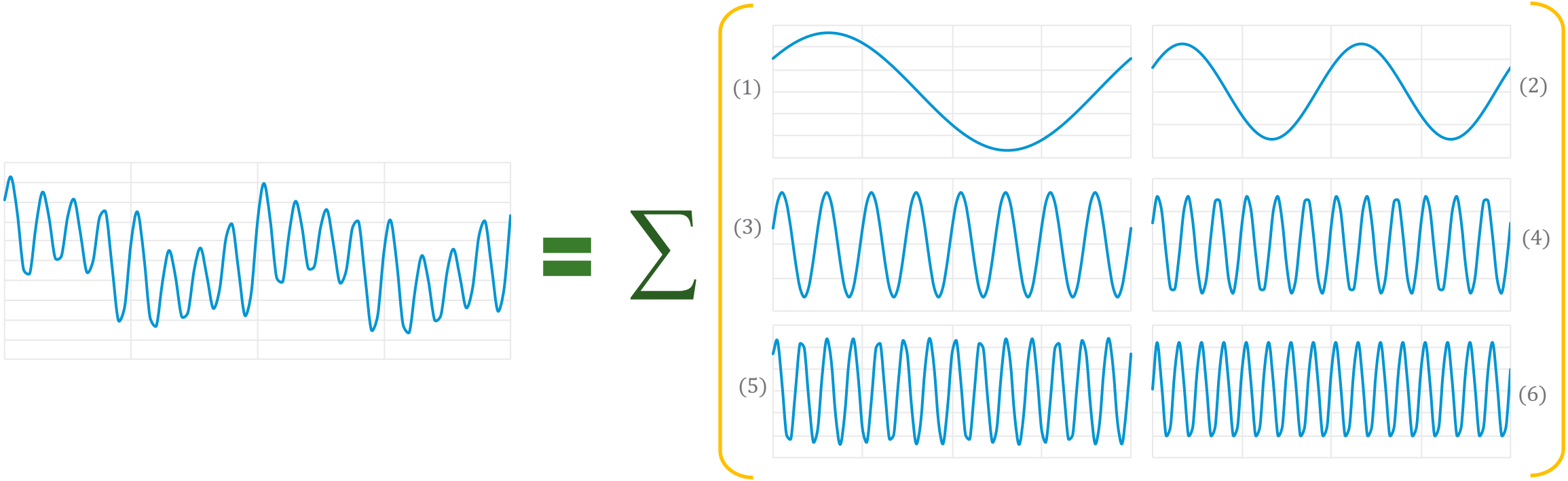
Our goal is to build set of features that accounts for cyclical patterns in our data

We try to do this through spectral analysis

vmware

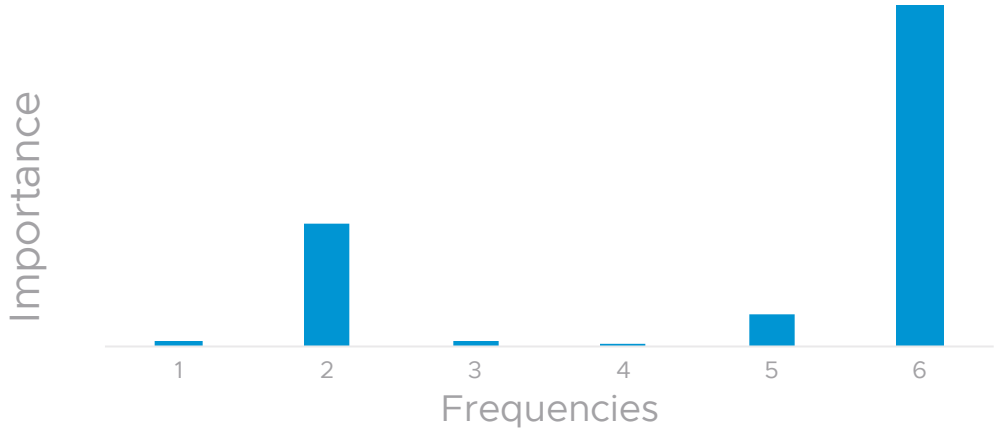
Spectral analysis

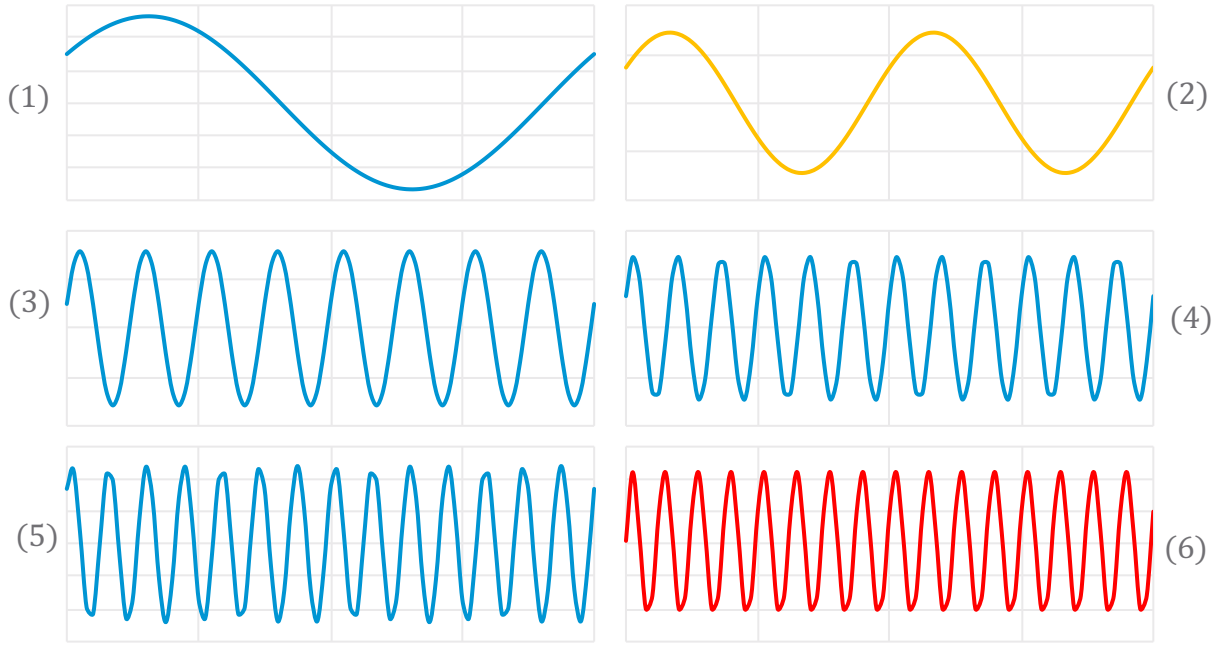
Spectral analysis decomposes our signal to many different sin and cos waves with different frequencies



Spectral analysis

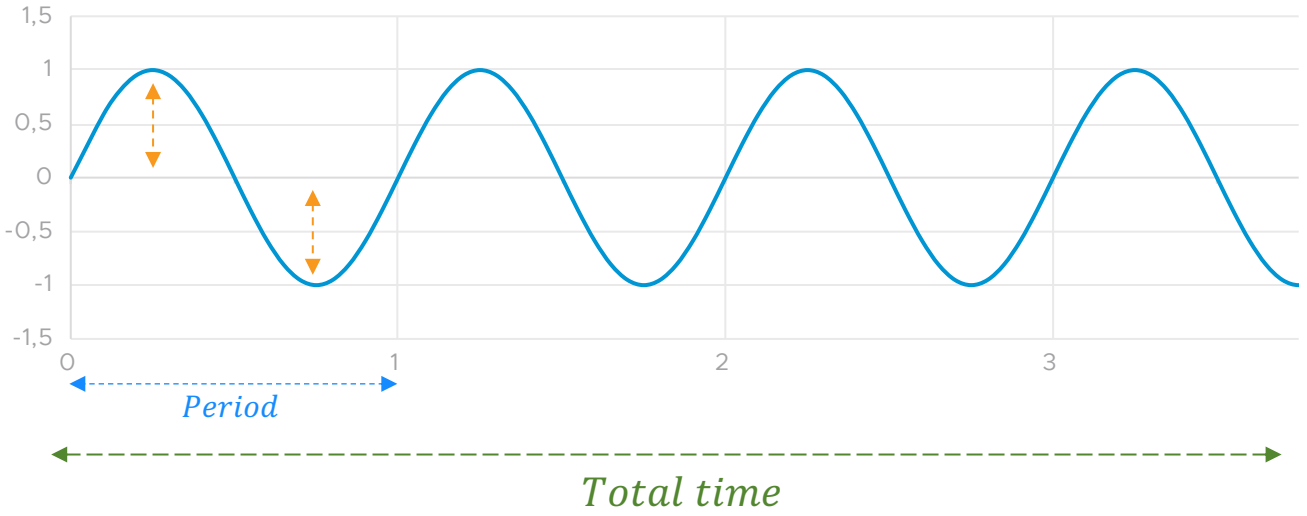
It allows us to determine those frequencies that appear particularly strong and important





Let put some common ground

Basic definitions



Total time

The time norm with respect to which we express all other durations

Also the total number of samples n

Period

The duration of time for one cycle

Amplitude

Change over 1 period

$$Frequency = \frac{Total\ Time}{Period} = \text{Number of repeats per unit time}$$

Spectral analysis

Fitting the waves

$Total\ time$

$Period$

$Amplitude$

$Frequency$

Spectral analysis can be viewed as a multiple linear regression:

- The dependent variable is our data Y_t
- Independent variables are the sine functions of all possible frequencies

$$Y_t = a_0 + \mathbf{a_1}\sin\left(\frac{Total\ time}{Period_1}\right) + \mathbf{b_1}\cos\left(\frac{Total\ time}{Period_1}\right) + \mathbf{a_2}\sin\left(\frac{Total\ time}{Period_2}\right) + \mathbf{b_2}\cos\left(\frac{Total\ time}{Period_2}\right) + \cdots$$

$Frequency_1$ $Frequency_2$

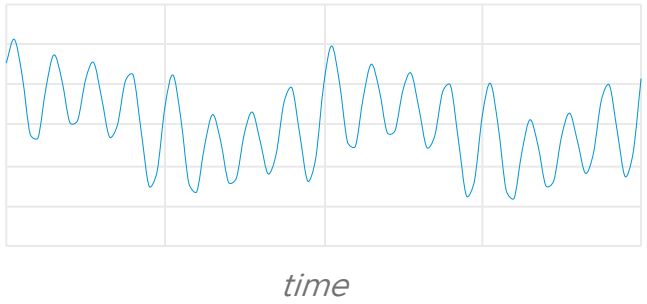
$mean(Y_t)$

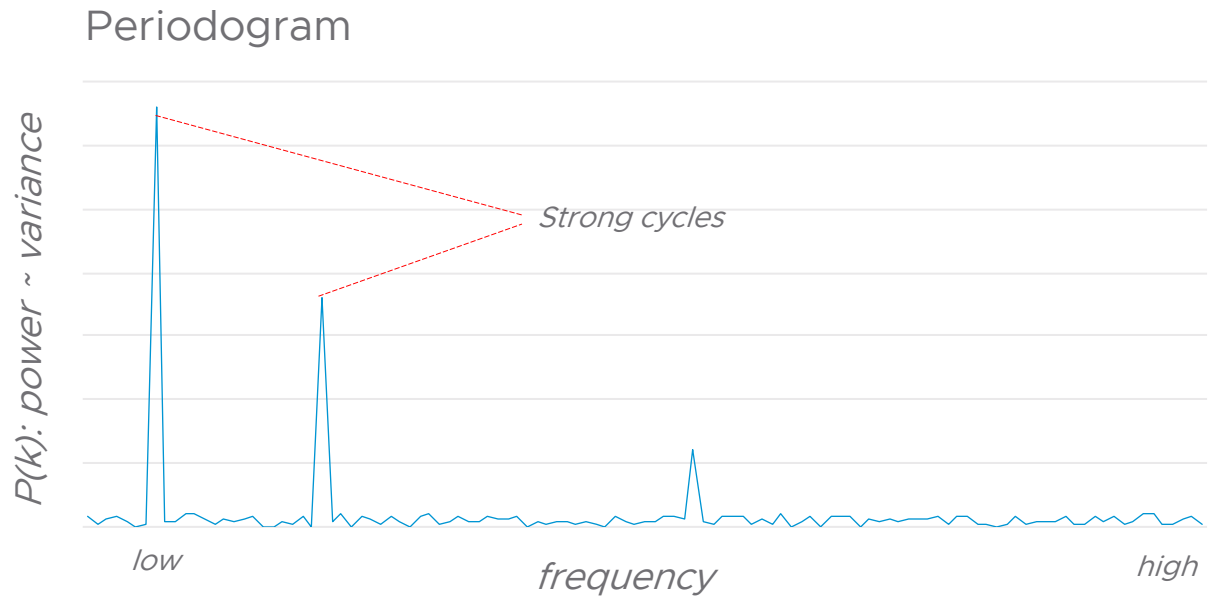
Spectral analysis

Periodogram

The periodogram quantifies the contributions of the individual frequencies to the time series regression

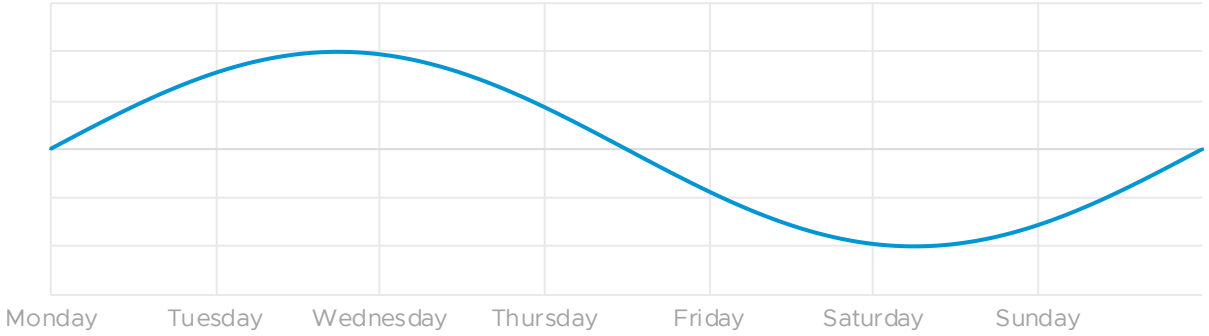
$$P_k = a_k^2 + b_k^2, for\ k\ in\ \left\{1, 2, \ldots, \frac{n}{2}\right\}$$





Spectral analysis in the context of our data

Frequency = 1

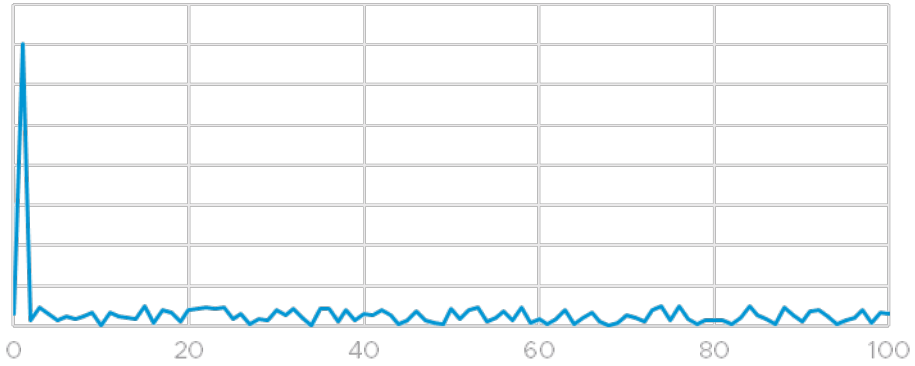


We have 7 full days of data, sampled every 5 minutes:

- There are 60 min / 5 min = 12 samples per hour
- There are 12 * 24 = 288 samples per day
- There are 7 * 288 = 2016 samples per week

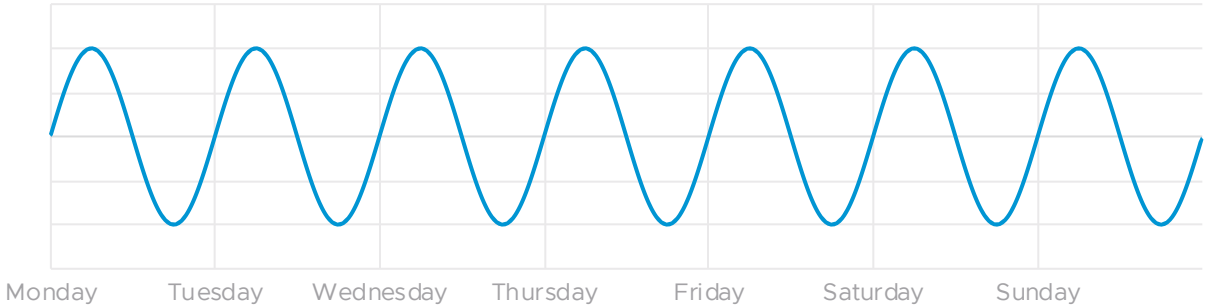
At frequency 1 we compare our data to sine wave that makes only one cycle during entire series (period = 2016 samples * 5 min)

Usually Frequency 1 captures trends or shifts in the data when the data is not stationary



Spectral analysis in the context of our data

Frequency = 7

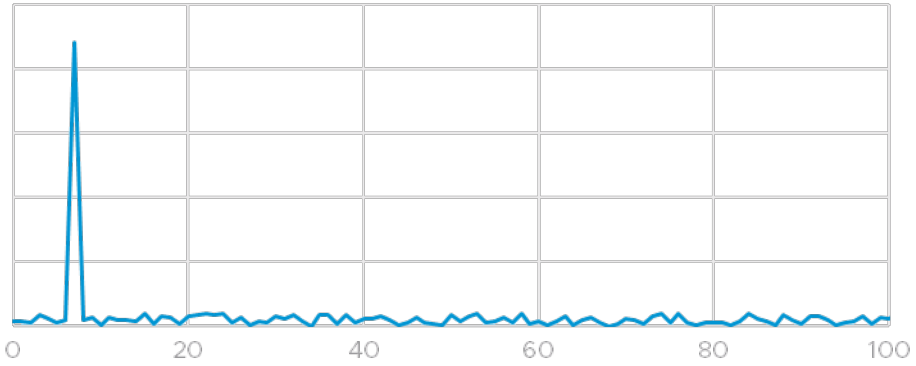


We have 7 full days of data, sampled every 5 minutes:

- There are 60 min / 5 min = 12 samples per hour
- There are 12 * 24 = 288 samples per day
- There are 7 * 288 = 2016 samples per week
- Total time = 2016 samples * 5 min

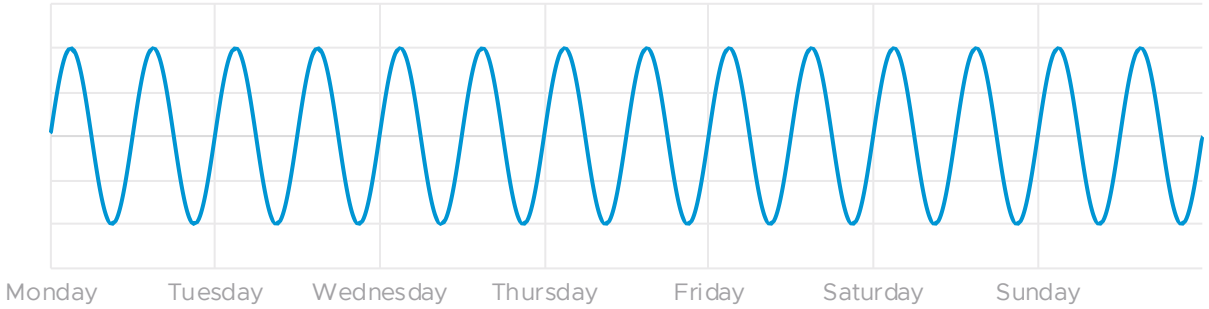
At frequency 7 we consider daily cycles

$$Period = \frac{Total\ Time}{Frequency} = \frac{2016 \times 5\ min}{7} = 288\ samples \times 5min$$



Spectral analysis in the context of our data

Frequency = 14

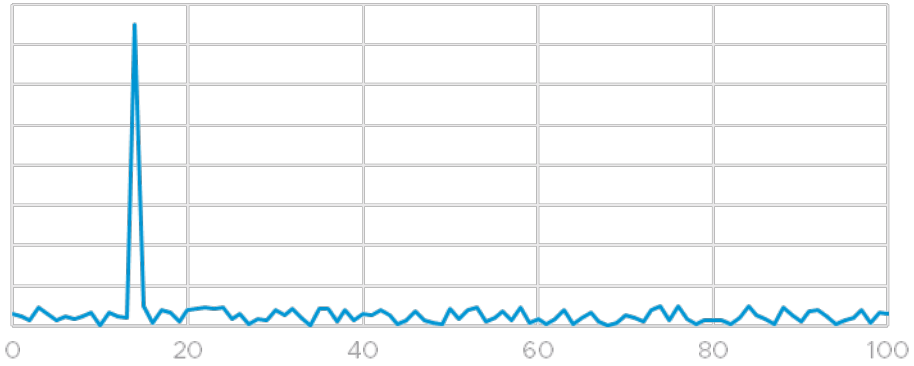


We have 7 full days of data, sampled every 5 minutes:

- There are 60 min / 5 min = 12 samples per hour
- There are 12 * 24 = 288 samples per day
- There are 7 * 288 = 2016 samples per week
- Total time = 2016 samples * 5 min

At frequency 14 we have cycles of 12 hours

$$Period = \frac{Total\ Time}{Frequency} = \frac{2016 \times 5\ min}{14} = 144\ samples \times 5min$$



Spectral analysis in the context of our data

Frequencies 21, 28, 42

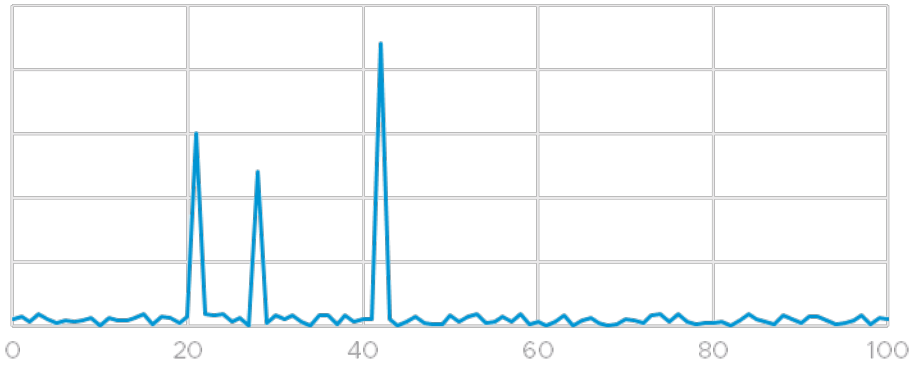
We have 7 full days of data, sampled every 5 minutes:

- There are 60 min / 5 min = 12 samples per hour
- There are 12 * 24 = 288 samples per day
- There are 7 * 288 = 2016 samples per week
- Total time = 2016 samples * 5 min

At frequency 21 we have cycles of 8 hours

At frequency 28 we have cycles of 6 hours

At frequency 42 we have cycles of 4 hours



Spectral analysis in the context of our data

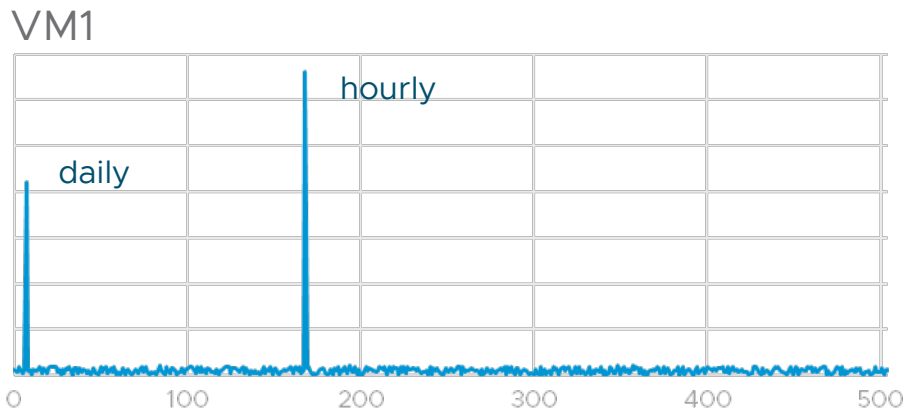
Common frequencies

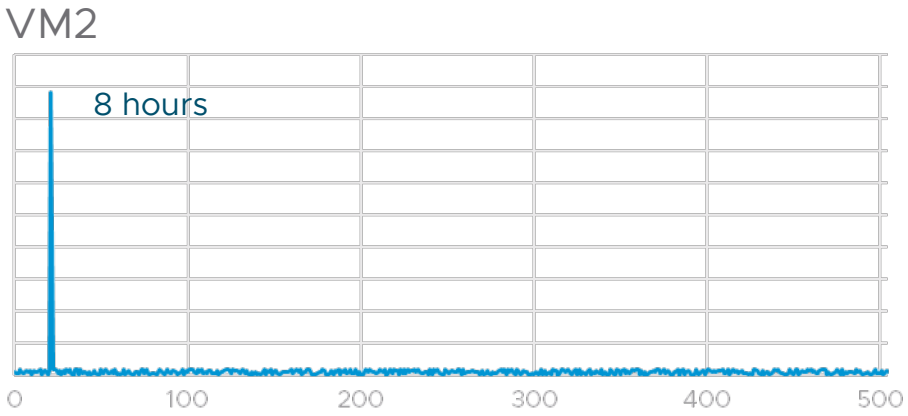
Frequency	Corresponding cycle
1	1 Week
7	1 Day
14	12 Hours
21	8 Hours
28	6 Hours
42	4 Hours
56	3 Hours
168	1 Hour
336	30 Minutes
504	20 Minutes
672	15 Minutes
1008 (highest frequency)	10 Minutes

We have 7 full days of data, sampled every 5 minutes:

- There are 60 min / 5 min = 12 samples per hour
- There are 12 * 24 = 288 samples per day
- There are 7 * 288 = 2016 samples per week
- Total time = 2016 samples * 5 min

Recall our goal





VM	Daily	12 H	8 H	6H	4H	1H	30Min
VM 1	High	Low	Low	Low	Low	High	Low
VM 2	Low	Low	High	Low	Low	Low	Low

Some practical steps

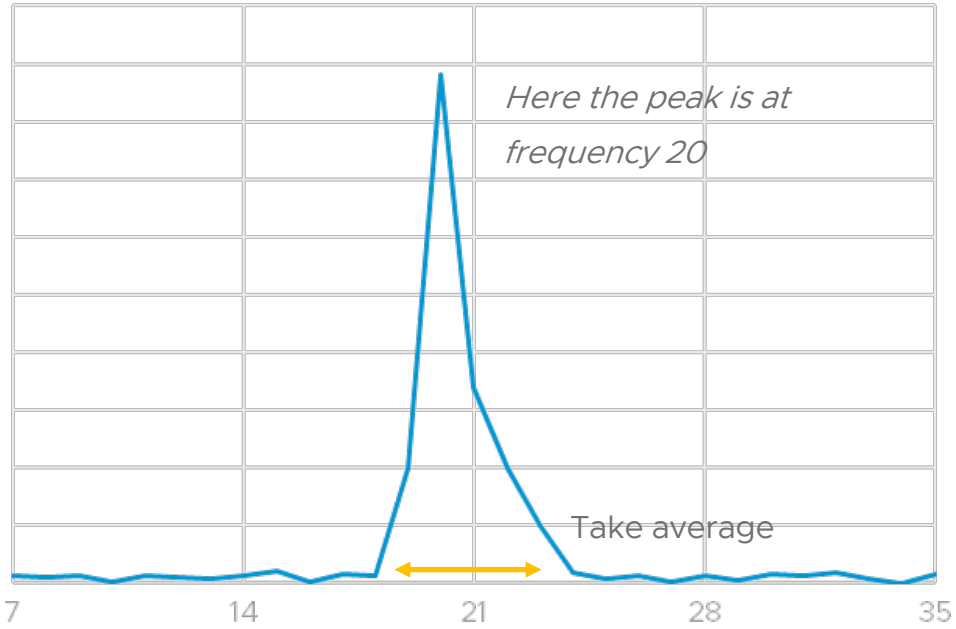
Preprocessing steps:

- Log transform
- Remove mean
- Fit polynomial and remove time trend

Classic spectral analysis requires stationary data (mean and covariance does not change in time). However some of our series violate this assumption even after transformations we applied. Thus they produce some obscured features and require more advanced techniques

Final consideration

Very often the power is spread between the neighborhood of the frequency of interest or might be shifted other frequencies within this neighborhood



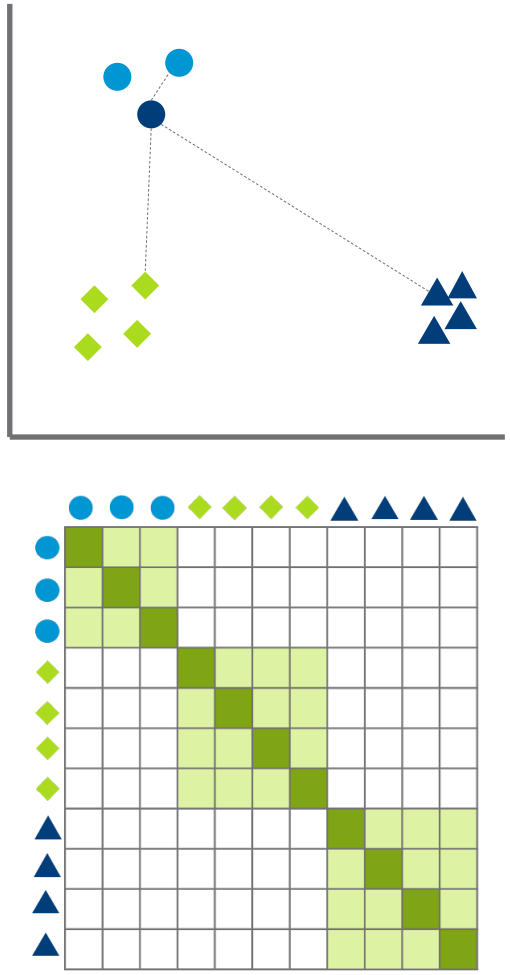
- To account for this we average over the neighborhood of the frequency of interest
- The window within which we average is defined by us usually + / - 2 frequencies
- To make things comparable between different VMs we normalize the spectrogram by the standard deviation of the signal

UMAP vs t-SNE

since we already know how t-SNE works

UMAP vs t-SNE

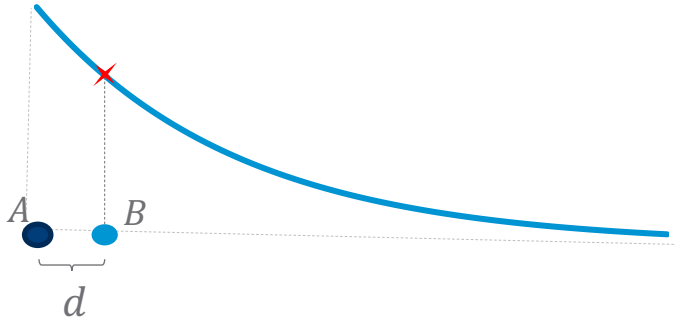
Building high dimensional graph



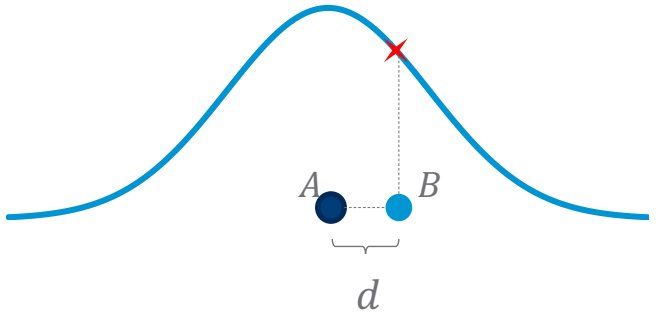
On a high level, t-SNE and UMAP are following the same steps.

However, they plug in different quantities

UMAP



t-SNE



$$p_{j|i} = exp^{-\frac{d(x_j,x_i)-\rho_j}{\sigma_j}}$$

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2/2\sigma_i^2\right)}{\sum_{i'}\exp\left(-\|x_i - x_{j'}\|^2/2\sigma_i^2\right)}$$

Locality and symmetrization

Unlike t-SNE, UMAP determines sigma directly from the nearest neighbors **("nneigh")**
"nneigh" is a tunable parameter that just like perplexity

t-SNE

$$perplexity = 2^{\sum_j p_{j|i} \log_2 p_{j|i}}$$

UMAP

$$nneigh = k = 2^{\sum_i p_{ij}}$$

Distances are symmetrized in a different way:

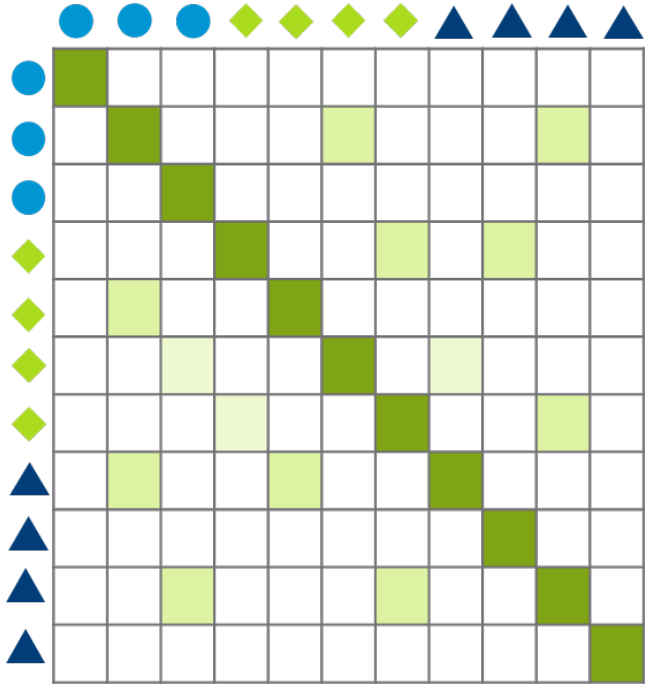
$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N}$$

$$p_{ij} = p_{i|j} + p_{j|i} - p_{i|j} * p_{j|i}$$

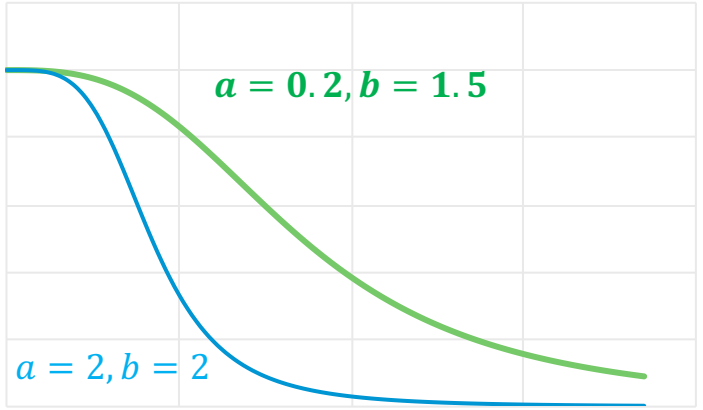
UMAP vs t-SNE

Building the low dimensional graph

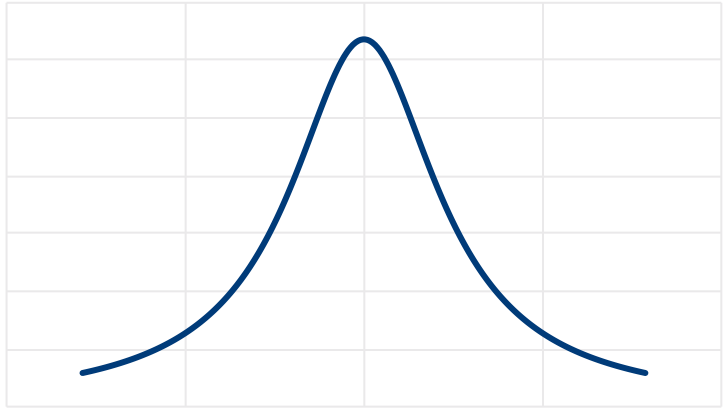
On a high level, t-SNE and UMAP are following the same steps. However, they plug in different quantities



UMAP



t-SNE



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

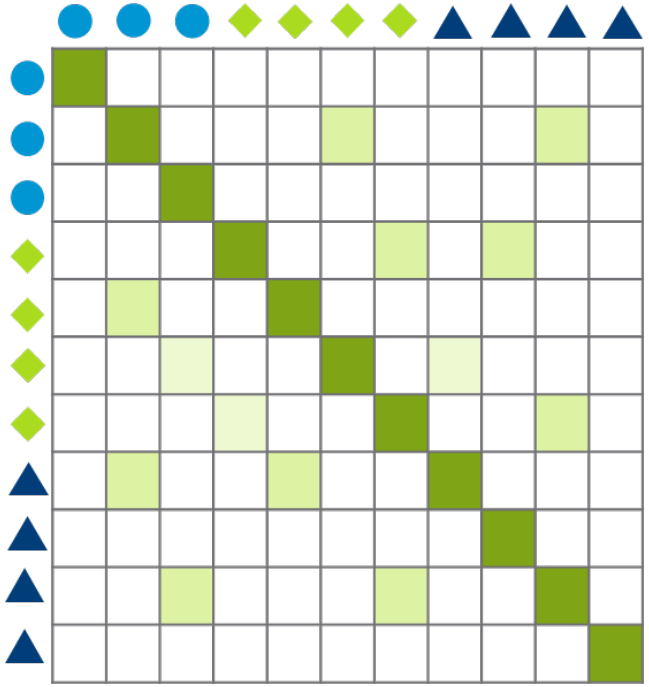
$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}}$$

UMAP vs t-SNE

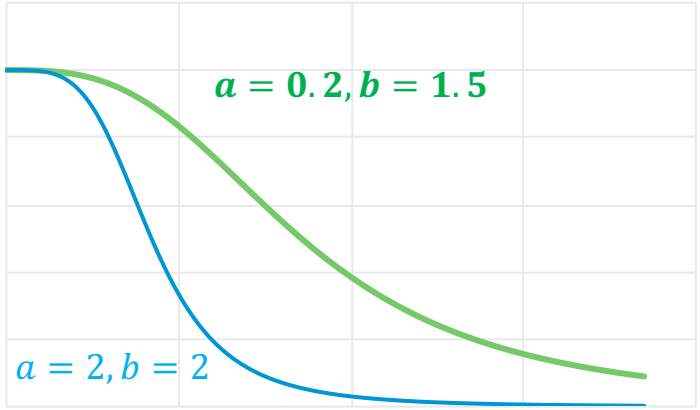
Building the low dimensional graph

On a high level, t-SNE and UMAP are following the same steps.

However, they plug in different quantities



UMAP



a and b are tunable parameters

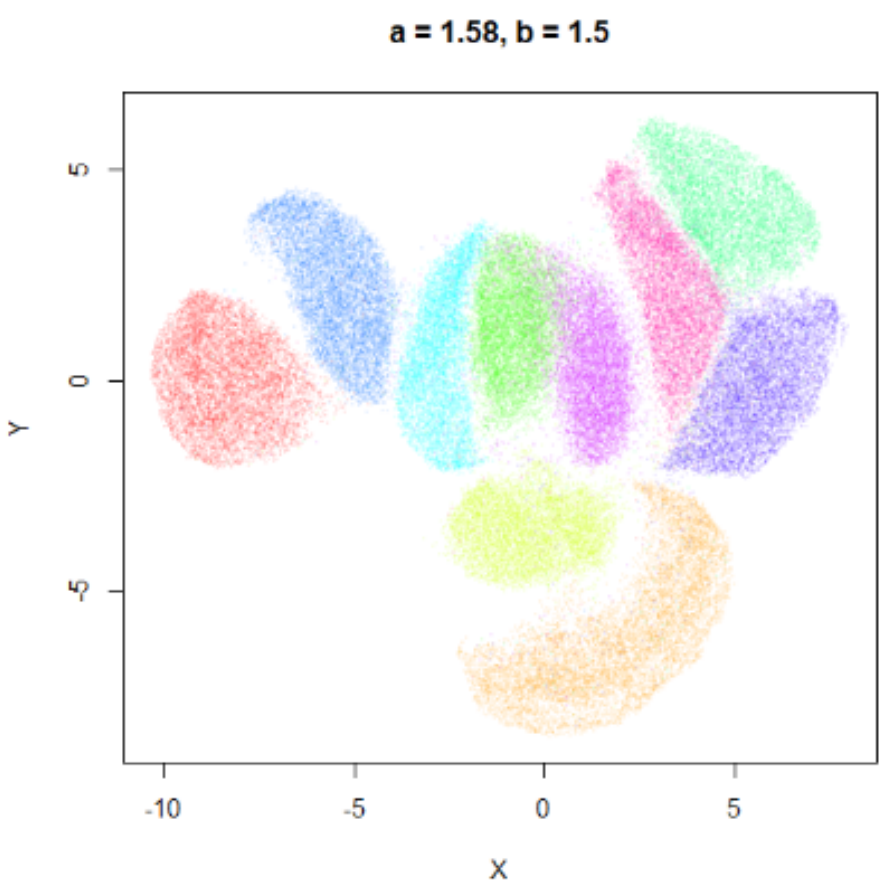
They can be tuned directly or inferred from min_dist and spread which are another parameters of the algorithm

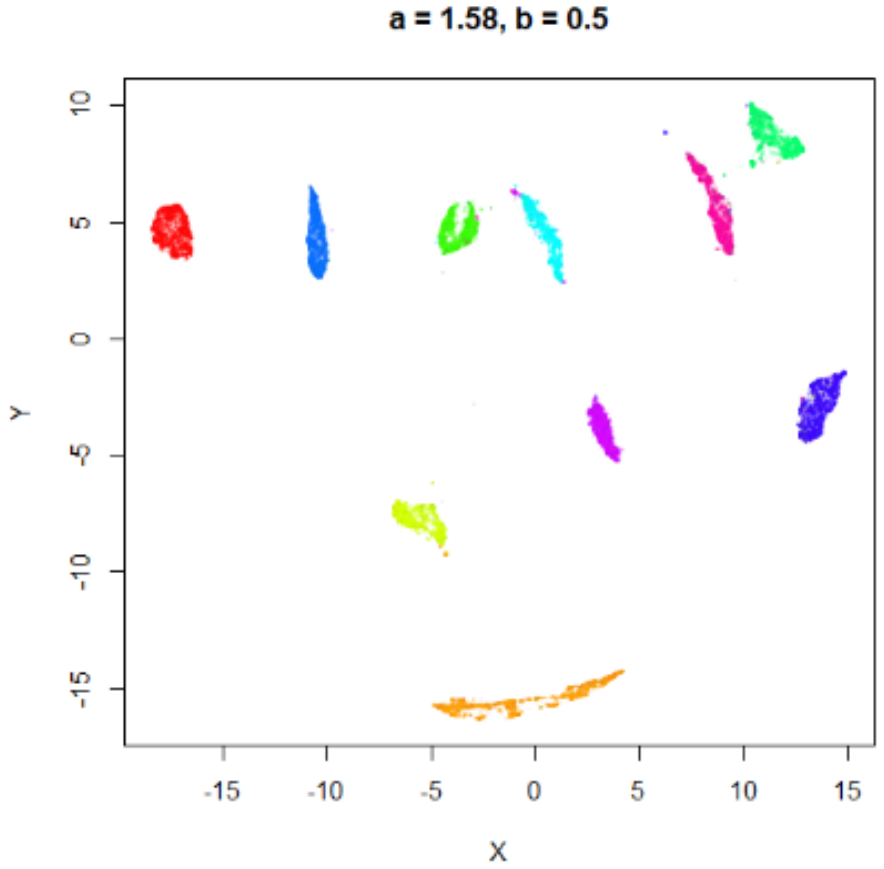
Generally these parameters controls how tightly packed the points, relative to one another

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

UMAP vs t-SNE

Embedding with different “b”

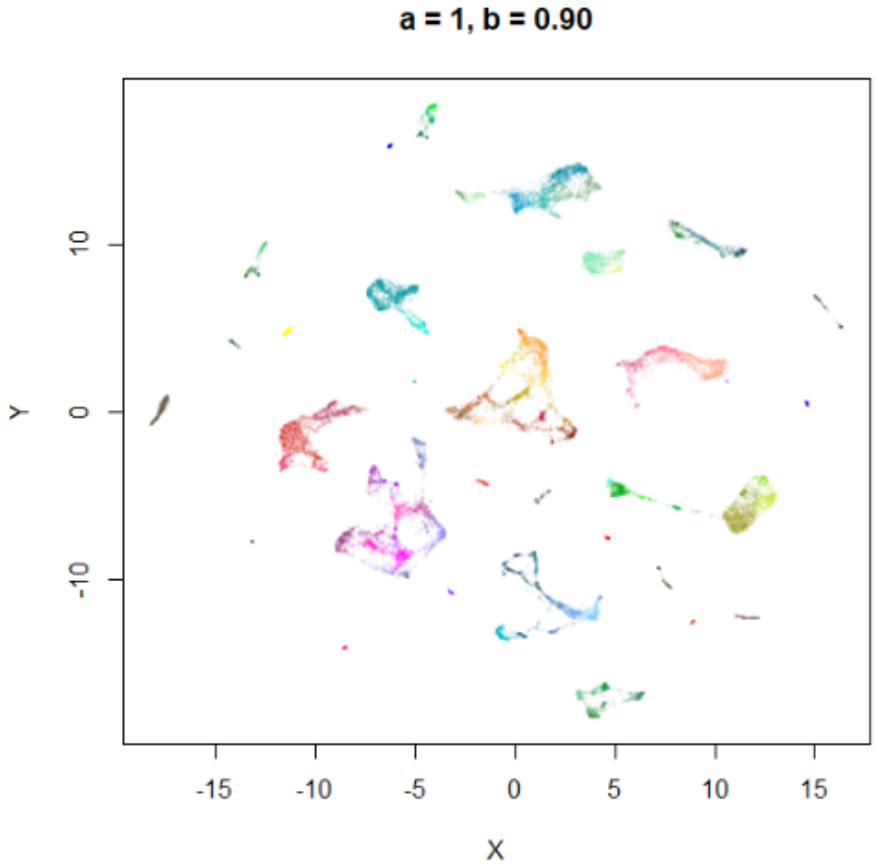


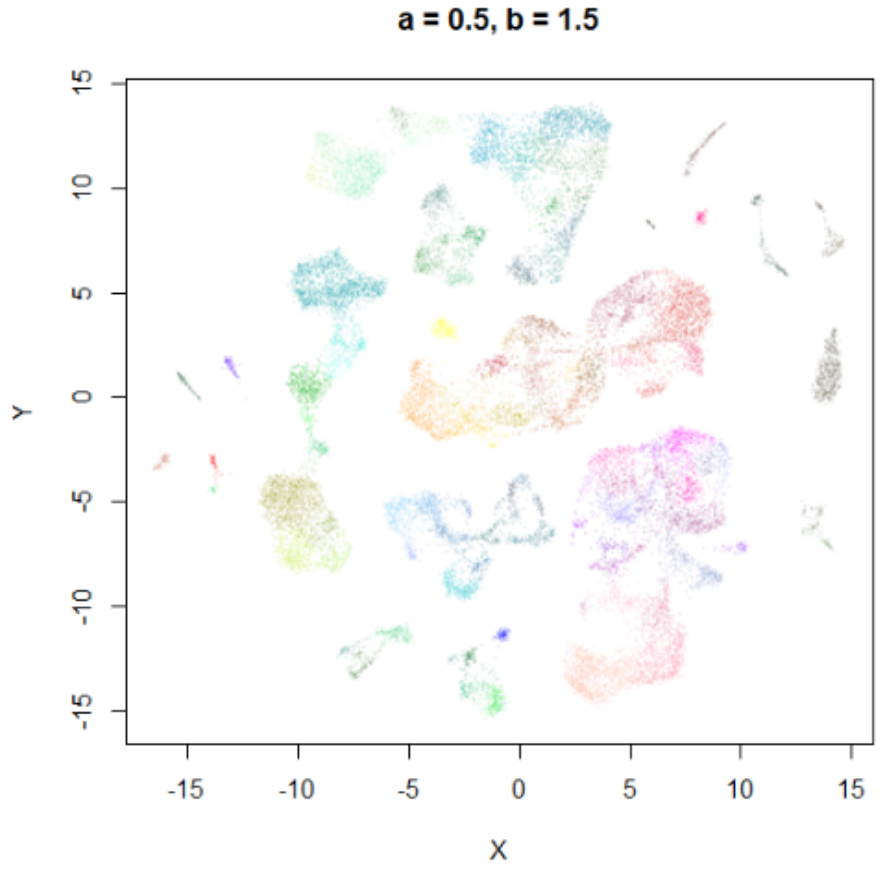


Source:https://jlmelville.github.io/uwot/abparams.html

UMAP vs t-SNE

Embedding with different “a”





Source:https://jlmelville.github.io/uwot/abparams.html

UMAP vs t-SNE

Optimization objective

Algorithm	Optimization objective
t-SNE	$KL(P_i \| \| Q_i) = \sum_i \sum_j p_{j\|i} \log \frac{p_{j\|i}}{q_{j\|i}}$ *"Kullback–Leibler divergence"* via gradient descent
UMAP	$CE(X,Y) = \sum_i \sum_j \left[p_{ij}(X) \log\left(\frac{p_{ij}(X)}{q_{ij}(Y)}\right) + \left(1 - p_{ij}(X)\right) \log\left(\frac{1 - p_{ij}(X)}{1 - q_{ij}(Y)}\right) \right]$ *"Binary cross-entropy"* with stochastic gradient descent (faster)

t-SNE vs UMAP

Summary

Metrics	t-SNE	UMAP
High dim distance	$p_{j\|i} = \frac{\exp\left(-\|x_i - x_j\|^2/2\sigma_i^2\right)}{\sum_{i'}\exp\left(-\|x_i - x_{j'}\|^2/2\sigma_i^2\right)}$	$p_{j\|i} = exp^{-\frac{d(x_j,x_i)-\rho_j}{\sigma_j}}$
Local vs global structure	$perplexity = 2^{\sum_j p_{j\|i}\log_2 p_{j\|i}}$	$nneigh = k = 2^{\sum_i p_{ij}}$
symmetrization	$p_{ij} = \frac{p_{i\|j} + p_{j\|i}}{2N}$	$p_{ij} = p_{i\|j} + p_{j\|i} - p_{i\|j} * p_{j\|i}$
Low dim distance	$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k\neq l}(1 + \|y_k - y_l\|^2)^{-1}}$	$q_{ij} = \left(1 + a\left(y_i - y_j\right)^{2b}\right)^{-1}$
Optimization objective (t-SNE)	$KL(P_i\|\|Q_i) = \sum_i\sum_j p_{j\|i}\log\frac{p_{j\|i}}{q_{j\|i}}$	
Optimization objective (UMAP)	$CE(X,Y) = \sum_i\sum_j\left[p_{ij}(X)\log\left(\frac{p_{ij}(X)}{q_{ij}(Y)}\right) + \left(1 - p_{ij}(X)\right)\log\left(\frac{1 - p_{ij}(X)}{1 - q_{ij}(Y)}\right)\right]$	

UMAP wins

But it is not flawless victory

Much better preservation of the global distances than t-SNE (but not as good as PCA)

Much faster – thus can be applied over quite large datasets

Operates on higher dimensions (sklearn implementation of t-SNE operates up to 3 dimensions)

Implemented with predict/transform methods – thus it can be used to extract features from unseen data.

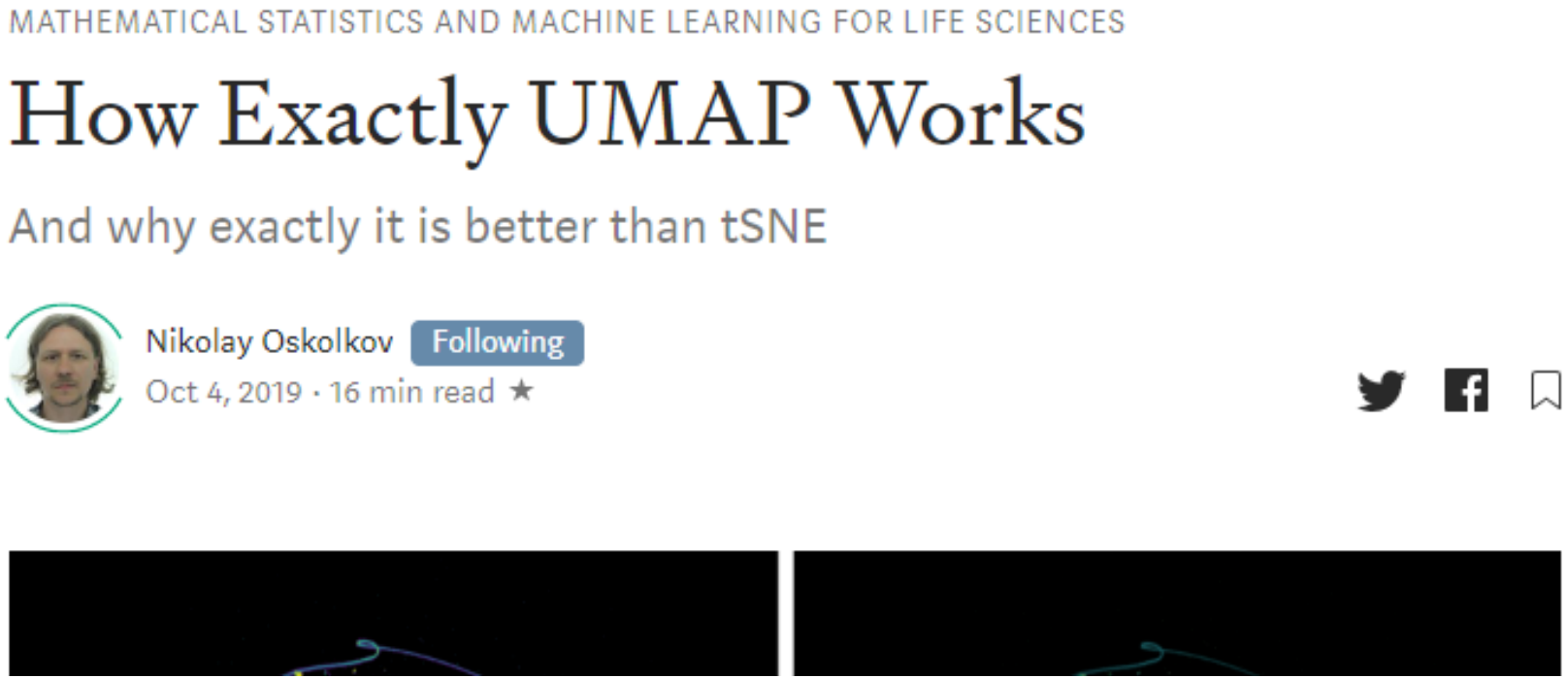
Like PCA can be used as preprocessing step prior predictive modeling

Can be used for semi supervised learning to infer classes on unlabeled observations

Credit for the main source for this section

Fantastic blog post about UMAP and T-SNE:

https://towardsdatascience.com/how-exactly-umap-works-13e3040e1668



Backup Slides

Box-Cox transformation

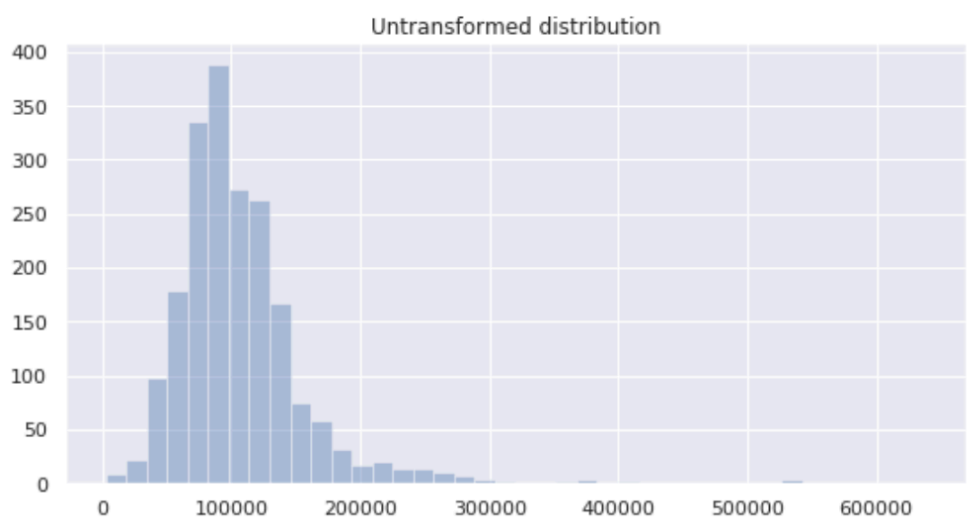
Box-Cox transformation used to stabilize variance & make data looks normal distribution – like.

Box-cox transformation aims to minimize variation.

Box-Cox is a generalization of the power transformation that can be manipulated through the Lambda parameter.

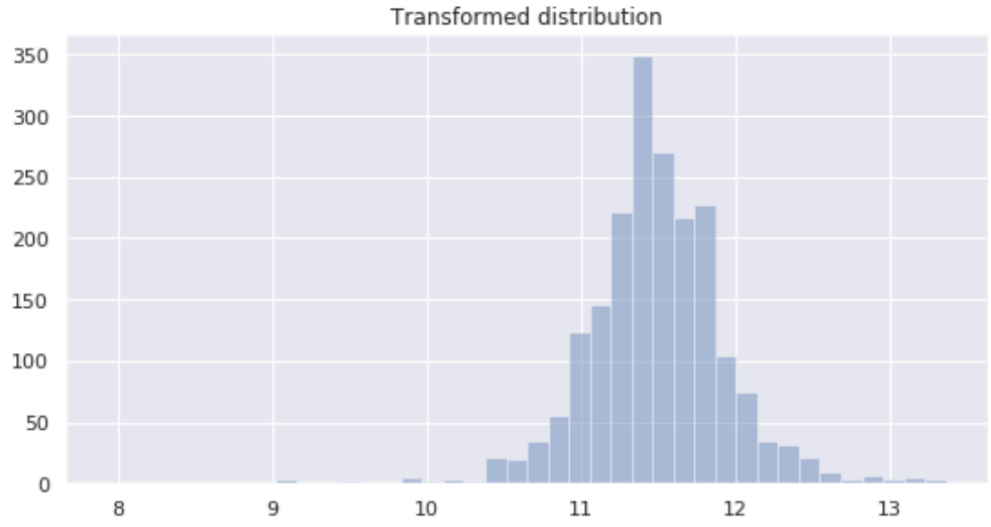
Most common Box-Cox transformation

Lambda	Y'
-2	$Y^{-2} = 1/Y^2$
-1	$Y^{-1} = 1/Y^1$
-0.5	$Y^{-0.5} = 1/(Sqrt(Y))$
	log(Y)
0.5	$Y^{0.5} = Sqrt(Y)$
1	$Y^1 = Y$
2	Y^2



Memory active write

VM 100



Logarithmic transformation

exponent

$\log_a(y) = x$

base

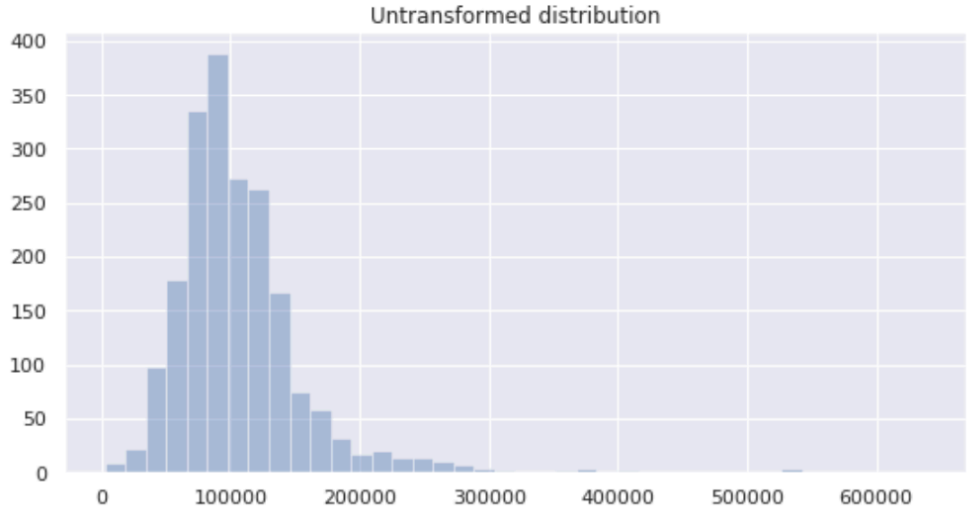
$\log_{10}(100) = 2$

$\log_2(8) = 3$

Commonly exploted in the analysis of the time series

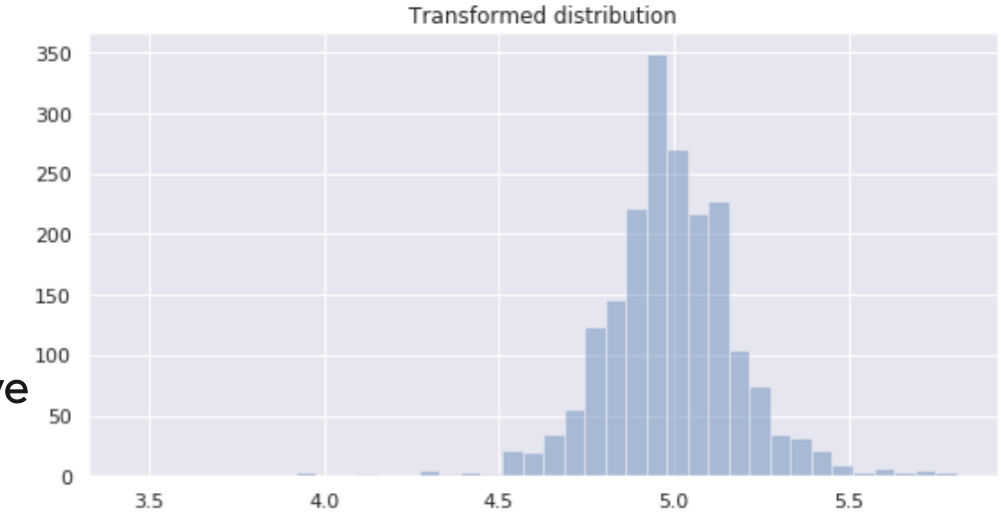
It is easier to interpret the results and you can draw conclusions back on the original scale.

Private case of Box-Cox transformation, but Box-Cox allows to find the optimal power transformation.



Memory active write

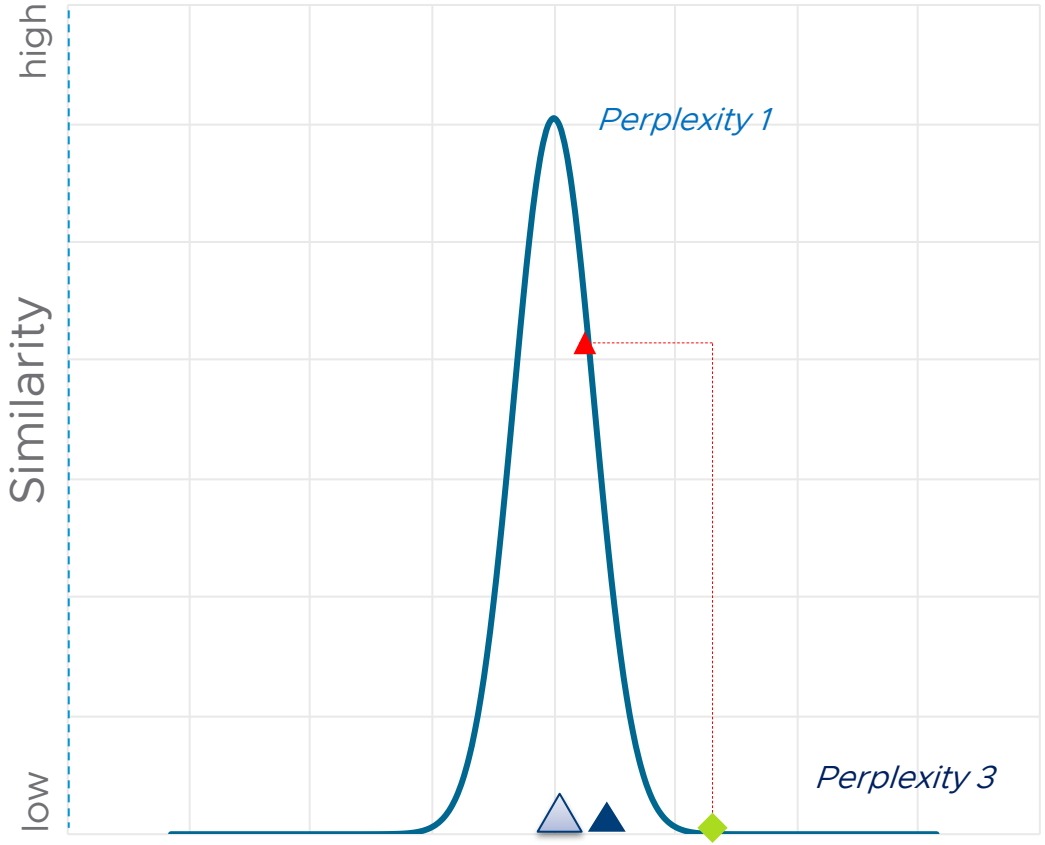
VM 100



Building a graph in high dimensions

Perplexity

But the shape of the normal curve depends on $\mathcal{O}$ as well …



And $\mathcal{O}$ depends on perplexity …

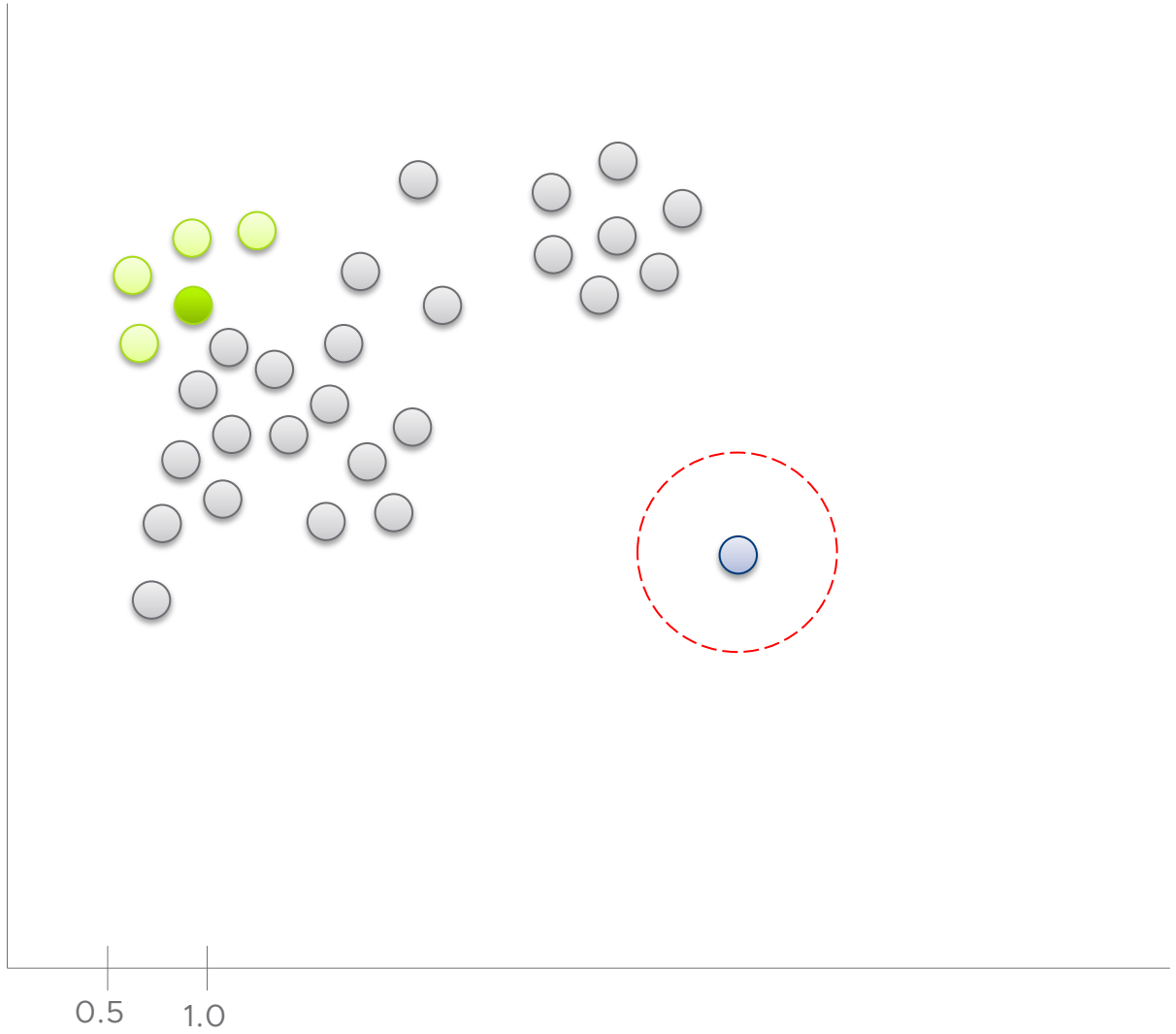
$$Perplexity_1 < Perplexity_2 < Perplexity_3$$

▲ Is more similar to △ under *Perplexity 1* compared to *Perplexity 2* and *Perplexity 3*

◆ Is less dissimilar △ under *Perplexity 3* compared to *Perplexity 1* and *Perplexity 2*

DBSCAN

Type of points



$\underline{Hyperparameters:}$

$\varepsilon = 0.5$

$\mathrm{m}inPoints = 5$

- Core point
- Border point
- Noise

vmware

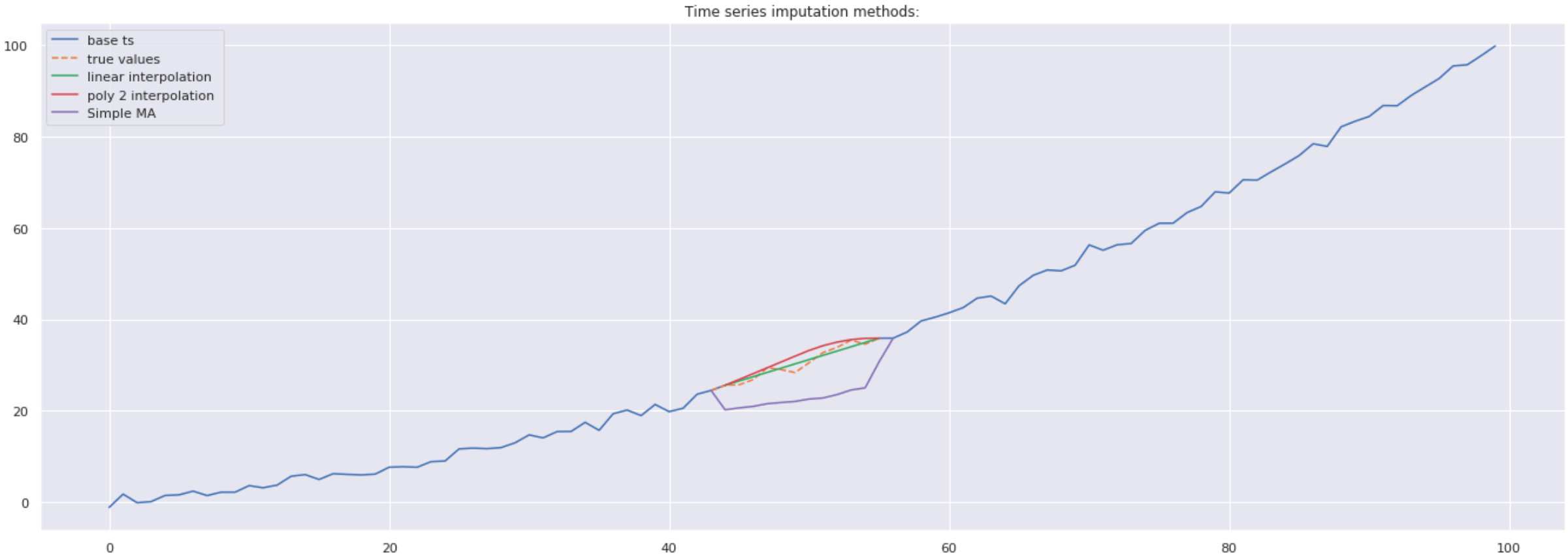
Linear Interpolation

What it does:

Concatenate linear interpolants between each missing pair of data points.

When do we try it:

Suitable when there is a linear trend.



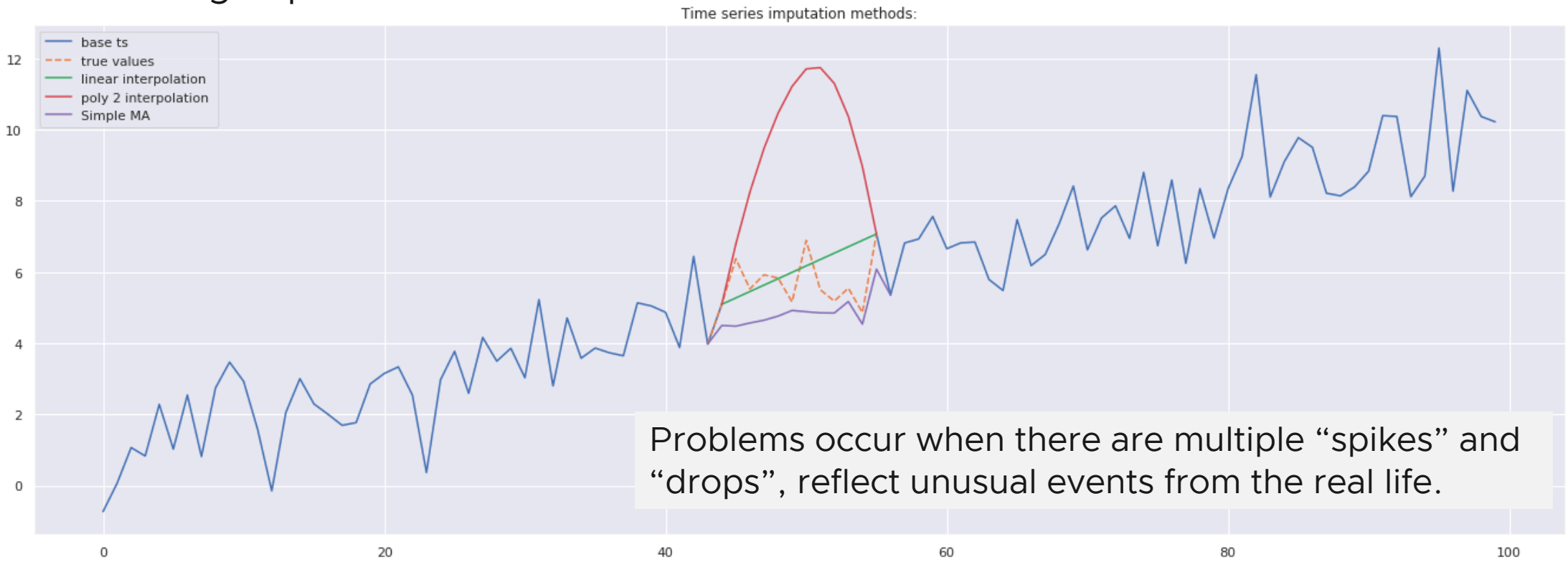
Polynomial Interpolation

What it does:

Uses mathematical expression (polynomial) to recreate the missing values using imputation.

When do we try it:

When we have a higher order trend, its worth trying polynomial interpolation.



Problems occur when there are multiple “spikes” and “drops”, reflect unusual events from the real life.

Moving Average Interpolation

What it does:

Uses mean values from a window for interpolation. Simple moving average (SMA) is an arithmetic moving average calculated by adding recent closing prices and then dividing that by the number of time periods in the calculation average.

When do we try it:

When we have a random process with NO TREND