CSI5180. Machine Learning for Bioinformatics Applications

Unsupervised Learning

by

Marcel Turcotte

Version **November 6, 2019**

Preamble

Preamble

Unsupervised Learning

In this lecture, we consider several aspects of **unsupervised learning**. This is important since **most available examples are unlabelled**. This is considered to be essential for **artificial general intelligence** (**AGI**). **Transcriptomics** is an area that benefits from unsupervised learning. We consider various **clustering** algorithms as well as the concepts behind **dimensionality reduction**.

General objective :

- **Describe** the main concepts and algorithms of unsupervised learning

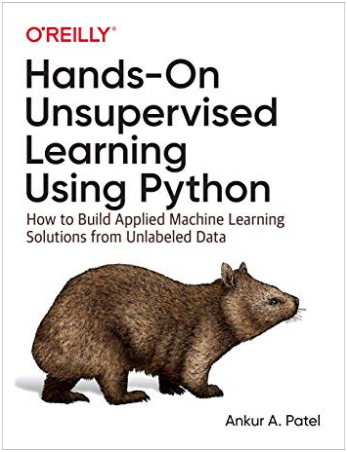
Learning objectives

- **Explain** in your own words what unsupervised learning is
- **Discuss** the problem of determining the optimal number of clusters
- **Describe** the main algorithms seen in class as well as their limitation
- **Present** the main concepts behind dimensionality reduction

Reading:

- G Kerr, H J Ruskin, M Crane, and P Doolan. Techniques for clustering gene expression data. *Comput Biol Med*, **38**(3):28393, Mar 2008.
- Jelili Oyelade, Itunuoluwa Isewon, Funke Oladipupo, Olufemi Aromolaran, Efosa Uwoghiren, Faridah Ameh, Moses Achas, and Ezekiel Adebiyi. Clustering algorithms: Their application to gene expression data. *Bioinform Biol Insights* **10**:237253, 2016.

Applied textbook



- Ankur A. Patel. Hands-On Unsupervised Learning Using Python. OReilly Media, 2019.

Plan

Unsupervised Learning - Andrew Ng

Introduction

Unsupervised Learning

https://youtu.be/jAA2g9ItoAc

Yann Lecun, AI Scientist, Facebook

*"If intelligence is a cake, **the bulk of the cake** is **unsupervised learning**, the **icing** on the cake is **supervised learning**, and the **cherry** on the cake is **reinforcement learning**."*

- **Source:** NIPS 2016 - https://www.youtube.com/watch?v=0unt2Y4qxQo&t=1072s

Introduction

Unsupervised learning

- The **data set** is a collection of **unlabelled** examples.

Unsupervised learning

- The **data set** is a collection of **unlabelled** examples.
  - $\{(x_i)\}_{i=1}^{N}$

Unsupervised learning

- The **data set** is a collection of **unlabelled** examples.
  - $\{(x_i)\}_{i=1}^{N}$
    - Each x_i is a **feature (attribute) vector** with D dimensions.

Unsupervised learning

- The **data set** is a collection of **unlabelled** examples.
  - $\{(x_i)\}_{i=1}^{N}$
    - Each x_i is a **feature (attribute) vector** with D dimensions.
    - $x_i^{(j)}$ is the value of the **feature** j of the example i, for $j \in 1 \ldots D$ and $i \in 1 \ldots N$.

Unsupervised learning

- The **data set** is a collection of **unlabelled** examples.
  - $\{(x_i)\}_{i=1}^{N}$
    - Each x_i is a **feature (attribute) vector** with D dimensions.
    - $x_i^{(j)}$ is the value of the **feature** j of the example i, for $j \in 1 \ldots D$ and $i \in 1 \ldots N$.
- **Problem**: find the underlying **"structure"** of the data.

Unsupervised learning

- The **data set** is a collection of **unlabelled** examples.
  - $\{(x_i)\}_{i=1}^{N}$
    - Each x_i is a **feature (attribute) vector** with D dimensions.
    - $x_i^{(j)}$ is the value of the **feature** j of the example i, for $j \in 1 \dots D$ and $i \in 1 \dots N$.
- **Problem**: find the underlying **"structure"** of the data.
  - The problem is **vaguely defined** compared to supervised learning.

Unsupervised learning

- The **data set** is a collection of **unlabelled** examples.
  - $\{(x_i)\}_{i=1}^{N}$
    - Each x_i is a **feature (attribute) vector** with D dimensions.
    - $x_i^{(j)}$ is the value of the **feature** j of the example i, for $j \in 1 \ldots D$ and $i \in 1 \ldots N$.
- **Problem**: find the underlying **"structure"** of the data.
  - The problem is **vaguely defined** compared to supervised learning.
  - Likewise, measuring **performance** will also be problematic.

Unsupervised learning

- The **data set** is a collection of **unlabelled** examples.
  - $\{(x_i)\}_{i=1}^{N}$
    - Each x_i is a **feature (attribute) vector** with D dimensions.
    - $x_i^{(j)}$ is the value of the **feature** j of the example i, for $j \in 1 \dots D$ and $i \in 1 \dots N$.
- **Problem**: find the underlying **"structure"** of the data.
  - The problem is **vaguely defined** compared to supervised learning.
  - Likewise, measuring **performance** will also be problematic.
  - However, the framework is very flexible.

Problem

Transcriptomics technologies

Several high-throughput **technologies** exist to measure the **expression** levels of (RNA) transcripts.

Transcriptomics technologies

Several high-throughput **technologies** exist to measure the **expression** levels of (RNA) transcripts.

- Expressed Sequence Tag (**EST**)
- Serial and cap analysis of gene expression (**SAGE**/**CAGE**)
- DNA **Microarrays** (GeneChips)
- **RNA-Seq**

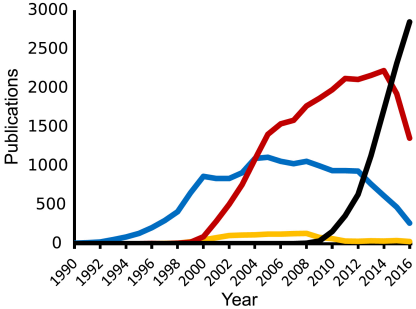


Fig 1. Transcriptomics method use over time. Published papers since 1990, referring to RNA sequencing (black), RNA microarray (red), expressed sequence tag (blue), and serial/cap analysis of gene expression (yellow)[12].

https://doi.org/10.1371/journal.pcbi.1005457.g001

Transcriptomics technologies

Several high-throughput **technologies** exist to measure the **expression** levels of (RNA) transcripts.

- Expressed Sequence Tag (**EST**)
- Serial and cap analysis of gene expression (**SAGE**/**CAGE**)
- DNA **Microarrays** (GeneChips)
- **RNA-Seq**

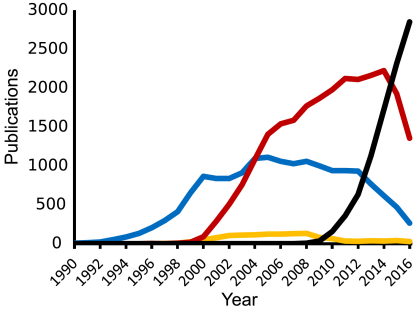
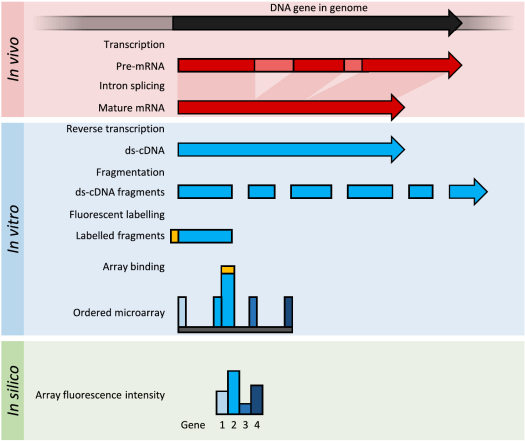


Fig 1. Transcriptomics method use over time. Published papers since 1990, referring to RNA sequencing (black), RNA microarray (red), expressed sequence tag (blue), and serial/cap analysis of gene expression (yellow)[1,2].

https://doi.org/10.1371/journal.pcbi.1005457.g001

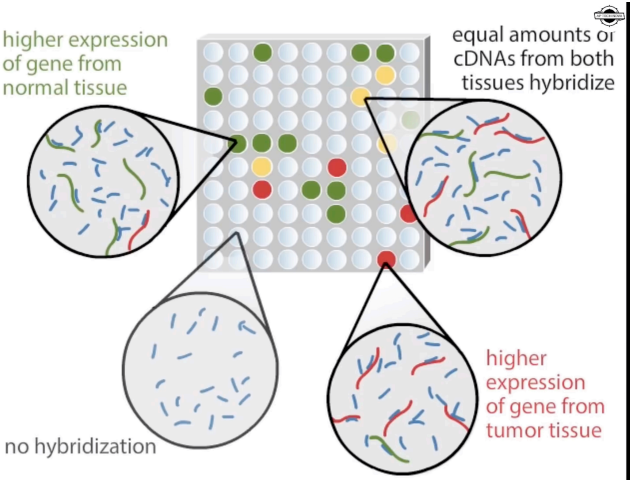
- Lowe, R., Shirley, N., Bleackley, M., Dolan, S. & Shafee, T. Transcriptomics technologies. *PLoS Comput Biol* **13**, (2017).

DNA Microarrays



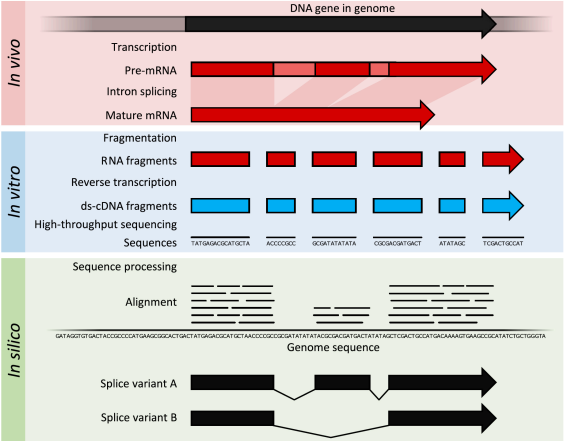
- Lowe, R., Shirley, N., Bleackley, M., Dolan, S. & Shafee, T. Transcriptomics technologies. *PLoS Comput Biol* **13**, (2017).

DNA Microarrays



https://youtu.be/yzBVOCwRanI

RNA-Seq



- Lowe, R., Shirley, N., Bleackley, M., Dolan, S. & Shafee, T. Transcriptomics technologies. *PLoS Comput Biol* **13**, (2017).

Data

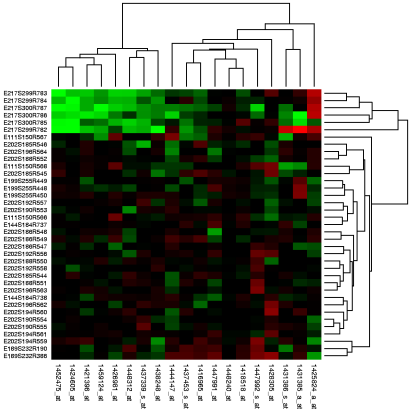
- $\{(x_i)\}_{i=1}^{N}$
  - Each x_i represents the **expression** of a given **gene** under different **conditions**, **individuals**/**tissues**/**cell types** - a **feature vector** with D dimensions.
  - $x_i^{(j)}$ is the value of the **feature** j of the example i, for $j \in 1 \dots D$ and $i \in 1 \dots N$. This is the **expression level** of **gene** i for **samples** j.

Data (alternative interpretation)

- $\{(x_i)\}_{i=1}^{N}$
  - Each x_i represents the **expression** of D **genes** for given **condition** - a **feature vector** with D dimensions.
  - $x_i^{(j)}$ is the value of the **feature** j of the example i, for $j \in 1 \ldots D$ and $i \in 1 \ldots N$. This is the **expression level** of **gene** j for **sample** i.

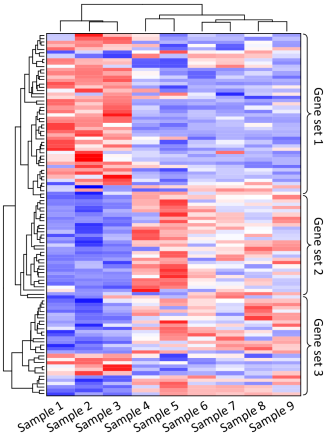
- Michael Molla, Michael Waddell, David Page, and Jude W. Shavlik. Using machine learning to design and interpret gene-expression microarrays. *AI Magazine*, **25**(1):2344, 2004.

Gene expression profiling



Source: https://upload.wikimedia.org/wikipedia/commons/4/48/Heatmap.png

Gene expression profiling



Source: `https://en.wikipedia.org/wiki/Transcriptomics_technologie`

Gene expression profiling - applications

- Identifying the **unknown function** of genes - **guilt by association**
- Diagnostics and disease profiling
- Human and pathogen transcriptomes
- Responses to environment

Transcriptomic databases

Name	Host	Data	Description
Gene Expression Omnibus [142]	NCBI	Microarray RNA-Seq	First transcriptomics database to accept data from any source. Introduced MIAME and MINSEQE community standards that define necessary experiment metadata to ensure effective interpretation and repeatability [143][144].
ArrayExpress [145]	ENA	Microarray	Imports datasets from the Gene Expression Omnibus and accepts direct submissions. Processed data and experiment metadata are stored at ArrayExpress, while the raw sequence reads are held at the ENA. Complies with MIAME and MINSEQE standards [144][145].
Expression Atlas [146]	EBI	Microarray RNA-Seq	Tissue-specific gene expression database for animals and plants. Displays secondary analyses and visualisation, such as functional enrichment of Gene Ontology terms, InterPro domains, or pathways. Links to protein abundance data where available.
Genevestigator [147]	Privately curated	Microarray RNA-Seq	Contains manual curations of public transcriptome datasets, focusing on medical and plant biology data. Individual experiments are normalised across the full database, to allow comparison of gene expression across diverse experiments. Full functionality requires licence purchase, with free access to a limited functionality.
RefEx [148]	DDBJ	All	Human, mouse, and rat transcriptomes from 40 different organs. Gene expression visualised as heatmaps projected onto 3D representations of anatomical structures.
NONCODE [149]	noncode.org	RNA-Seq	ncRNAs excluding tRNA and rRNA.

DDBJ, DNA Data Bank of Japan; EBI, European Bioinformatics Institute; ENA, European Nucleotide Archive; MIAME, Minimum Information About a Microarray Experiment; MINSEQE, Minimum Information about a high-throughput nucleotide SEQuencing Experiment; NCBI, National Center for Biotechnology Information; ncRNAs, noncoding RNAs; RNA-Seq, RNA sequencing.

https://doi.org/10.1371/journal.pcbi.1005457.t005

- Lowe, R., Shirley, N., Bleackley, M., Dolan, S. & Shafee, T. Transcriptomics technologies. *PLoS Comput Biol* **13**, (2017).

Problems

Unsupervised learning - problems

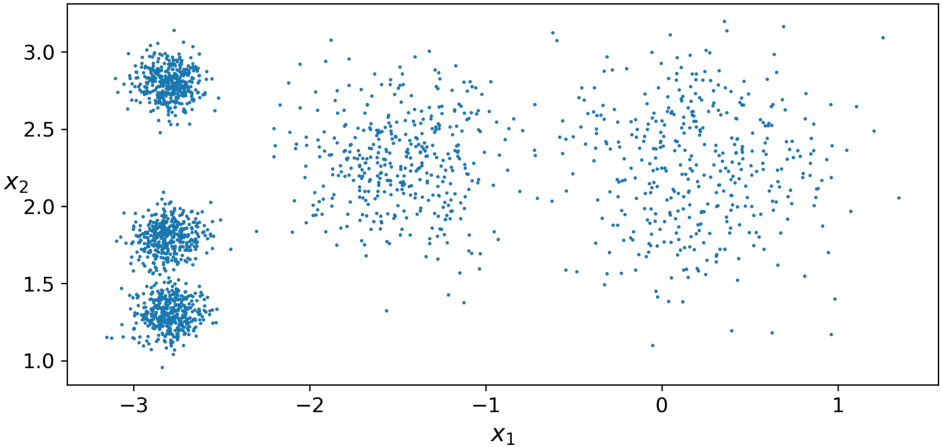
- **Dimensionality reduction**
  - Principal Component Analysis (PCA), t-Distributed Stochastic Neighbor Embedding (t-SNE)
- **Clustering**
  - K-Means, DBSCAN, hierarchical
- **Anomaly detection**
  - One-class SVM

Unsupervised learning - problems

- Later, we will consider **unsupervised learning** methods based on **deep learning**, namely the **auto-encoders**.

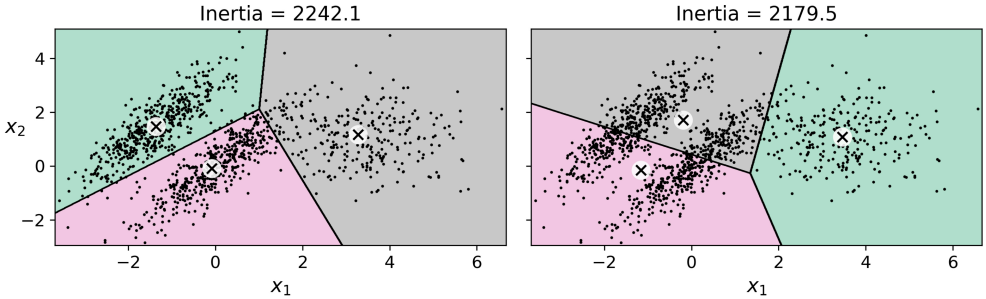
Clustering

What is a cluster?



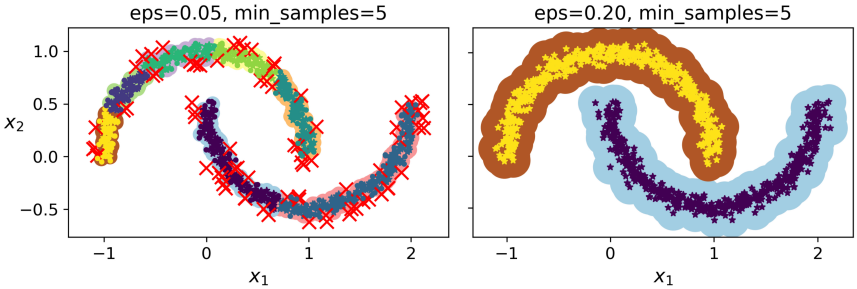
Source: [1] Figure 9.2

What is a cluster?



Source: [1] Figure 9.11

What is a cluster?



Source: [1] Figure 9.14

What is a cluster?

- The **definition** of a cluster might depend on the **application**.

What is a cluster?

- The **definition** of a cluster might depend on the **application**.
  - Round, elongated, complex.

What is a cluster?

- The **definition** of a cluster might depend on the **application**.
  - Round, elongated, complex.
- The **algorithms** have their own **bias** and **limitations**.

What is a cluster?

- The **definition** of a cluster might depend on the **application**.
  - Round, elongated, complex.
- The **algorithms** have their own **bias** and **limitations**.
  - Producing **spherical** clusters.

What is a cluster?

- The **definition** of a cluster might depend on the **application**.
  - Round, elongated, complex.
- The **algorithms** have their own **bias** and **limitations**.
  - Producing **spherical** clusters.
  - Looking for **densely packed** and **contiguous** regions.

What is a cluster?

- The **definition** of a cluster might depend on the **application**.
  - Round, elongated, complex.
- The **algorithms** have their own **bias** and **limitations**.
  - Producing **spherical** clusters.
  - Looking for **densely packed** and **contiguous** regions.
- It is important to match the **right algorithm** with the **right problem**.

What is a cluster?

- The **definition** of a cluster might depend on the **application**.
  - Round, elongated, complex.
- The **algorithms** have their own **bias** and **limitations**.
  - Producing **spherical** clusters.
  - Looking for **densely packed** and **contiguous** regions.
- It is important to match the **right algorithm** with the **right problem**.
- **Hard** vs. **soft** clusters:

What is a cluster?

- The **definition** of a cluster might depend on the **application**.
  - Round, elongated, complex.
- The **algorithms** have their own **bias** and **limitations**.
  - Producing **spherical** clusters.
  - Looking for **densely packed** and **contiguous** regions.
- It is important to match the **right algorithm** with the **right problem**.
- **Hard** vs. **soft** clusters:
  - For some algorithms/applications, each element is assigned to **one and only one cluster**, this is called **hard clustering**.

What is a cluster?

- The **definition** of a cluster might depend on the **application**.
  - Round, elongated, complex.
- The **algorithms** have their own **bias** and **limitations**.
  - Producing **spherical** clusters.
  - Looking for **densely packed** and **contiguous** regions.
- It is important to match the **right algorithm** with the **right problem**.
- **Hard** vs. **soft** clusters:
  - For some algorithms/applications, each element is assigned to **one and only one cluster**, this is called **hard clustering**.
  - The alternative is to **estimate the probability** that a given example belongs to a given cluster, this is called **soft clustering**.

sklearn.cluster.KMeans

By now, you are familiar with **KMeans**.

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=5)
y_pred = kmeans.fit_predict(X)
```

```
>>> y_pred
array([4, 0, 1, ..., 2, 1, 0], dtype=int32)
```

- **KMeans** has one **mandatory** hyperparameter, **K**, the number of clusters.
- Determining the **number of clusters** is one of the main challenges for clustering.

sklearn.cluster.KMeans

```
>>> kmeans.cluster_centers_
array([[-2.80389616, 1.80117999],
       [ 0.20876306, 2.25551336],
       [-2.79290307, 2.79641063],
       [-1.46679593, 2.28585348],
       [-2.80037642, 1.30082566]])

>>> X_new = np.array([[0, 2], [3, 2], [-3, 3], [-3, 2.5]])

>>> kmeans.predict(X_new)
array([1, 1, 2, 2], dtype=int32)
```

Source: [1] §9

Algorithm - KMeans

1. **Randomly select K examples** — these are the initial K centroids

Algorithm - KMeans

1. **Randomly select** K **examples** — these are the initial K centroids
2. **For each** example:

Algorithm - KMeans

1. **Randomly select** K **examples** — these are the initial K centroids
2. **For each** example:
   2.1 Calculate the distance between **this example** and **all** k **centroids**.

Algorithm - KMeans

1. **Randomly select** K **examples** — these are the initial K centroids
2. **For each** example:
   2.1 Calculate the distance between **this example** and **all** k **centroids**.
   2.2 Find the centroid with **minimum distance** to **this example**.

Algorithm - KMeans

1. **Randomly select** K **examples** — these are the initial K centroids
2. **For each** example:
   2.1 Calculate the distance between **this example** and **all** k **centroids**.
   2.2 Find the centroid with **minimum distance** to **this example**.
   2.3 Assign the **label** of that centroid to this example.

Algorithm - KMeans

1. **Randomly select** K **examples** — these are the initial K centroids
2. **For each** example:
   2.1 Calculate the distance between **this example** and **all** k **centroids**.
   2.2 Find the centroid with **minimum distance** to **this example**.
   2.3 Assign the **label** of that centroid to this example.
3. **For each** cluster:

Algorithm - KMeans

1. **Randomly select** K **examples** — these are the initial K centroids
2. **For each** example:
   2.1 Calculate the distance between **this example** and **all** k **centroids**.
   2.2 Find the centroid with **minimum distance** to **this example**.
   2.3 Assign the **label** of that centroid to this example.
3. **For each** cluster:
   3.1 Update the **centroid**.

Algorithm - KMeans

1. **Randomly select** K **examples** — these are the initial K centroids
2. **For each** example:
   2.1 Calculate the distance between **this example** and **all** k **centroids**.
   2.2 Find the centroid with **minimum distance** to **this example**.
   2.3 Assign the **label** of that centroid to this example.
3. **For each** cluster:
   3.1 Update the **centroid**.
4. If the centroids have **moved**, **repeat** from 2, else **stop**.

Algorithm - KMeans

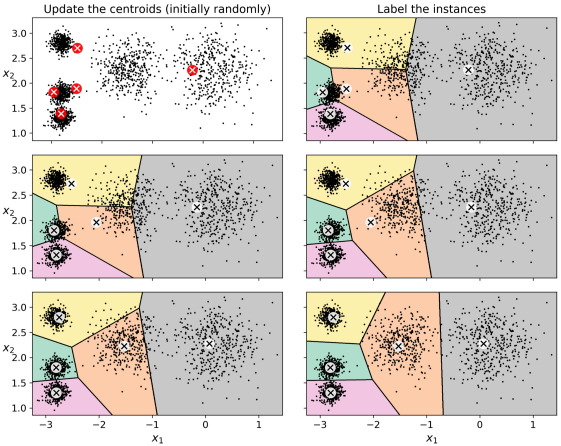
1. **Randomly select** K **examples** — these are the initial K centroids
2. **For each** example:
   2.1 Calculate the distance between **this example** and **all** k **centroids**.
   2.2 Find the centroid with **minimum distance** to **this example**.
   2.3 Assign the **label** of that centroid to this example.
3. **For each** cluster:
   3.1 Update the **centroid**.
4. If the centroids have **moved**, **repeat** from 2, else **stop**.

Algorithm - KMeans

1. **Randomly select** K **examples** — these are the initial K centroids
2. **For each** example:
   2.1 Calculate the distance between **this example** and **all** k **centroids**.
   2.2 Find the centroid with **minimum distance** to **this example**.
   2.3 Assign the **label** of that centroid to this example.
3. **For each** cluster:
   3.1 Update the **centroid**.
4. If the centroids have **moved**, **repeat** from 2, else **stop**.

The algorithm is **guaranteed to converge** (to stop in finite [small] number of steps).

Algorithm - KMeans



Source: [1] Figure 9.4

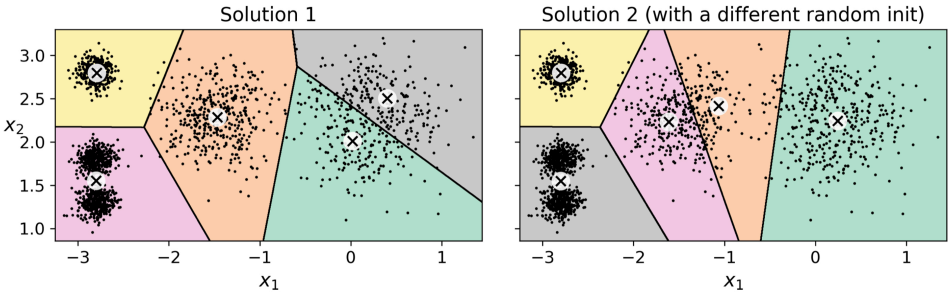
Discussion

- What **shape** of clusters would **KMeans** produce?

Discussion

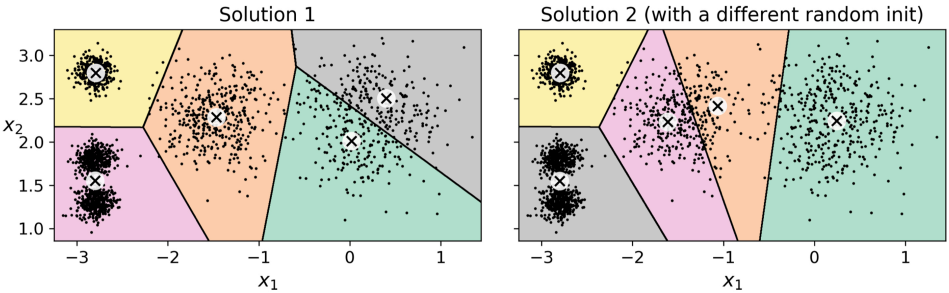
- What **shape** of clusters would **KMeans** produce?
- Would you expect the solution to be the **same** at every run?

Local optima



Source: [1] Figure 9.5

Local optima



Source: [1] Figure 9.5

Solutions

- Run KMeans multiple times, **n_init=10**.

Objective function - inertia

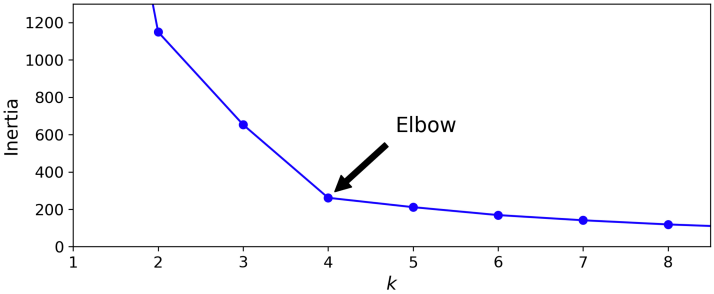
$$\sum_{i=1}^{N} \min_{\mu_j \in C} (||x_i - \mu_j||^2) \tag{1}$$

- For a fixed K, run KMeans multiple times, **n_init=10**, select the solution minimizing inertia (distortion).

KMeans++

- **KMeans++** was introduced in 2006.
- Selects the initial centroids in a way that all the centroids are **as far as possible one from another**.
- **Default** initialization method with **Scikit-Learn**.

Finding the optimal number of clusters



Source: [1] Figure 9.8

- **Inertia** cannot be used as a criterion to find the optimal number of clusters since the **more clusters** there are, **the closer each instance will be to the centroid of its cluster** — think about the case where $K = N$.

Finding the optimal number of clusters

Let i be an example and C_i its cluster, $a(i)$ is the mean intra-cluster distance:

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, j \neq j} d(i, j) \tag{2}$$

Finding the optimal number of clusters

Let i be an example and C_i its cluster, $a(i)$ is the mean intra-cluster distance:

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, j \neq j} d(i, j) \tag{2}$$

Let i be an example and C_i its cluster, $b(i)$ is the mean distance to examples from the closest cluster (different that C_i):

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j) \tag{3}$$

Finding the optimal number of clusters

Let i be an example and C_i its cluster, $a(i)$ is the mean intra-cluster distance:

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, j \neq j} d(i, j) \tag{2}$$

Let i be an example and C_i its cluster, $b(i)$ is the mean distance to examples from the closest cluster (different that C_i):

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j) \tag{3}$$

The **silhouette coefficient** of example i, $s(i)$ is:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}, if |C_i| > 1, s(i) = 0, if |C_i| = 1 \tag{4}$$

Finding the optimal number of clusters

Let i be an example and C_i its cluster, $a(i)$ is the mean intra-cluster distance:

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, j \neq j} d(i, j) \tag{2}$$

Let i be an example and C_i its cluster, $b(i)$ is the mean distance to examples from the closest cluster (different that C_i):

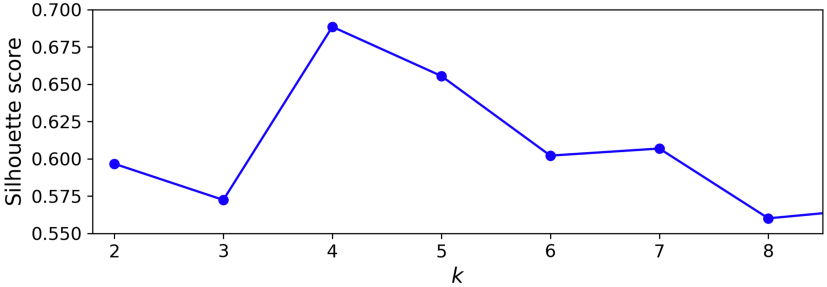
$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j) \tag{3}$$

The **silhouette coefficient** of example i, $s(i)$ is:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}, if |C_i| > 1, s(i) = 0, if |C_i| = 1 \tag{4}$$

The **silhouette score** is the mean value of $s(i)$, for all i.

Finding the optimal number of clusters



Source: [1] Figure 9.9

- The value of K maximizing the **silhouette score** is a "good indication" of the optimal number of clusters.

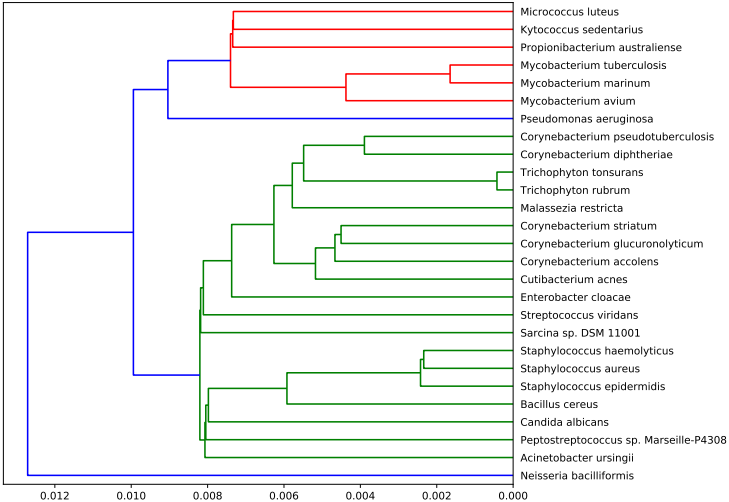
Hierarchical clustering

- **Hierarchical clustering** is sometimes called **UPGMA** (Unweighted Pair Group Method using Arithmetic) in bioinformatics.

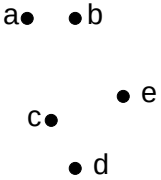
```
from scipy.cluster.hierarchy import dendrogram, linkage

linked = linkage(X, 'single')
dendrogram(linked, orientation='left',labels=names)
```

Dendrogram

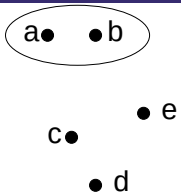

Micrococcus luteus
Kytococcus sedentarius
Propionibacterium australiense
Mycobacterium tuberculosis
Mycobacterium marinum
Mycobacterium avium
Pseudomonas aeruginosa
Corynebacterium pseudotuberculosis
Corynebacterium diphtheriae
Trichophyton tonsurans
Trichophyton rubrum
Malassezia restricta
Corynebacterium striatum
Corynebacterium glucuronolyticum
Corynebacterium accolens
Cutibacterium acnes
Enterobacter cloacae
Streptococcus viridans
Sarcina sp. DSM 11001
Staphylococcus haemolyticus
Staphylococcus aureus
Staphylococcus epidermidis
Bacillus cereus
Candida albicans
Peptostreptococcus sp. Marseille-P4308
Acinetobacter ursingii
Neisseria bacilliformis
0.012
0.010
0.008
0.006
0.004
0.002
0.000

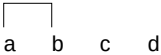
Algorithm - hierarchical clustering



	a	b	c	d	e
a	0	10	21	32	25
b		0	21	30	18
c			0	11	16
d				0	18
e					0

Algorithm - hierarchical clustering

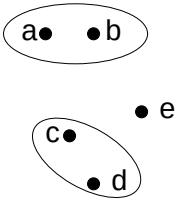




	{a,b}	c	d	e
{a,b}	0	21	31	21.5
c		0	11	16
d			0	18
e				0

$$d_{\{ab\},e} = (d_{ae} + d_{be})/2 = (25 + 18)/2$$

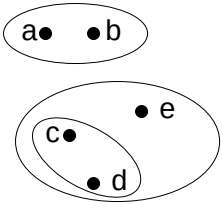
Algorithm - hierarchical clustering

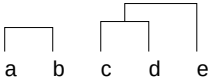


	{a,b}	{c,d}	e
{a,b}	0	26	21.5
{c,d}		0	17
e			0

$$d_{\{ab\},\{cd\}} = (d_{ac} + d_{ad} + d_{bc} + d_{bd})/4 = (21 + 32 + 21 + 30)/4$$

Algorithm - hierarchical clustering

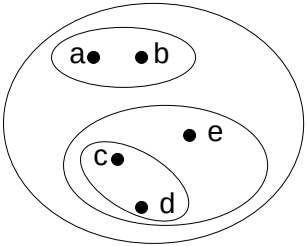


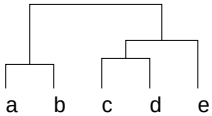


	{a,b}	{c,d,e}
{a,b}	0	24.5
{c,d,e}		0

$$d_{\{ab\},\{cde\}} = (d_{ac} + d_{ad} + d_{ae} + d_{bc} + d_{bd} + d_{bd})/6$$

Algorithm - hierarchical clustering





	{a,b,c,d,e}
{a,b,c,d,e}	0

UPGMA - distance measures

Average distance (produces clusters with same variance):

$$d_{ij} = \frac{1}{|C_i||C_j|} \sum_{p \in C_i, q \in C_j} d_{pq}$$

Complete linkage (produces compact clusters):

$$d_{ij} = \max_{p \in C_i, q \in C_j} d_{pq}$$

Single linkage (picks up elongated/irregular clusters):

$$d_{ij} = \min_{p \in C_i, q \in C_j} d_{pq}$$

DBSCAN

```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=0.05, min_samples=5)
dbscan.fit(X)
```

Algorithm:

1. **For each** example:

DBSCAN

```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=0.05, min_samples=5)
dbscan.fit(X)
```

Algorithm:

1. **For each** example:
   1.1 Count **how many** examples are located at a distance ϵ is less — this the ϵ-**neighbourhood**.

DBSCAN

```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=0.05, min_samples=5)
dbscan.fit(X)
```

Algorithm:

1. **For each** example:
   1.1 Count **how many** examples are located at a distance ϵ is less — this the ϵ-**neighbourhood**.
   1.2 If the **count** is **min_samples** or more, then this example is **core**.

DBSCAN

```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=0.05, min_samples=5)
dbscan.fit(X)
```

Algorithm:

1. **For each** example:
   1.1 Count **how many** examples are located at a distance ϵ is less — this the ϵ-**neighbourhood**.
   1.2 If the **count** is **min_samples** or more, then this example is **core**.
2. Let all the examples in the ϵ-**neighbourhood** of a **core** be part of the **same cluster**.

DBSCAN

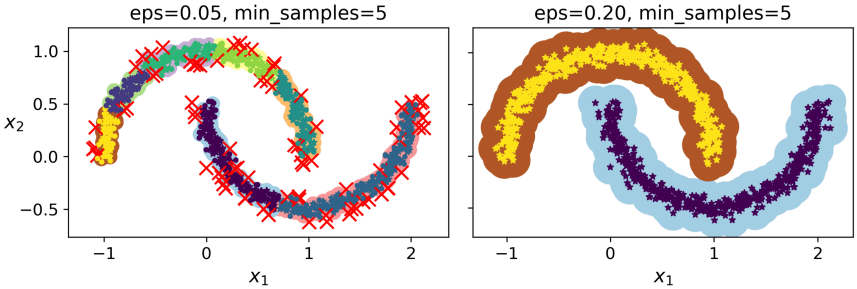
```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=0.05, min_samples=5)
dbscan.fit(X)
```

Algorithm:

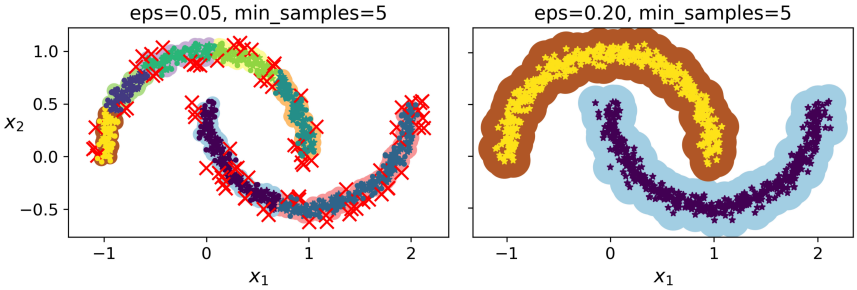
1. **For each** example:
   1.1 Count **how many** examples are located at a distance ϵ is less — this the ϵ-**neighbourhood**.
   1.2 If the **count** is **min_samples** or more, then this example is **core**.
2. Let all the examples in the ϵ-**neighbourhood** of a **core** be part of the **same cluster**.
3. Any example that is **not** a **core example and** does **not** have a core example in its ϵ-**neighbourhood** is an **anomaly**.

The hyperparameter epsilon



Source: [1] Figure 9.14

The hyperparameter epsilon



Source: [1] Figure 9.14

- **Pros:** simple, detects clusters with complex shapes, robust to outliers.
- **Cons:** challenged if the clusters have widely diverse density.

Gaussian mixture model, density estimation

- In **2020**, present **Gaussian mixture model** as well as **density estimation**.

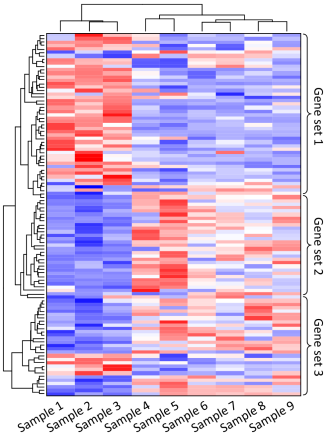
Resources (videos)

Lectures on **Machine Learning** by **Andrew Ng** - specifically, the lectures on unsupervised learning and clustering.

1. **Introduction** (3 m 17 s)
2. **KMeans algorithm** (12 m 32 s)
3. **Clustering objective function** (7 m 4 s)
4. **Clustering random initialization** (7 m 50 s)
5. **Choosing the number of clusters** (8 m 22 s)

Dimensionality reduction

Gene expression profiling



Source: https://en.wikipedia.org/wiki/Transcriptomics_technologie

Expression data

- $\{(x_i)\}_{i=1}^{N}$
  - Each x_i represents the **expression** of a given **gene** under different **conditions**, **individuals**/**tissues**/**cell types** - a **feature vector** with D dimensions.
  - $x_i^{(j)}$ is the value of the **feature** j of the example i, for $j \in 1 \dots D$ and $i \in 1 \dots N$. This is the **expression level** of **gene** i for **samples** j.

Expression data (alternative interpretation)

- $\{(x_i)\}_{i=1}^{N}$
  - Each x_i represents the **expression** of D **genes** for a given **condition** - a **feature vector** with D dimensions.
  - $x_i^{(j)}$ is the value of the **feature** j of the example i, for $j \in 1 \ldots D$ and $i \in 1 \ldots N$. This is the **expression level** of **gene** j for **sample** i.

- Michael Molla, Michael Waddell, David Page, and Jude W. Shavlik. Using machine learning to design and interpret gene-expression microarrays. *AI Magazine*, **25**(1):2344, 2004.

Segmentation of budding yeast cells

- Imagine designing a software system to **label** and **track live-cells** in bright-field microscopy **images**.
- A **classifier** must be trained to label these images.
- Assuming that each image is 512 by 512 pixels, this means a total 262,144 pixels or features!

See:

- Versari, C. et al. Long-term tracking of budding yeast cells in brightfield microscopy: CellStar and the Evaluation Platform. *Journal of The Royal Society Interface* **14**:127, (2017).
- `https://github.com/kaernlab/YeastNet`

Curse of dimensionality

- Consider features with m **discrete** values, the **volume** of the **input space grows as** m^D!

Curse of dimensionality

- Consider features with m **discrete** values, the **volume** of the **input space grows as** m^D!
- "Because of "**the curse of dimensionality**," many statistical methods **lack power** when applied to high-dimensional data. Even if the **number of collected data points is large**, they remain sparsely submerged in a voluminous high-dimensional space that is practically impossible to explore exhaustively" [23]

Curse of dimensionality

- Consider features with m **discrete** values, the **volume** of the **input space grows as** m^D!
- "Because of "**the curse of dimensionality**," many statistical methods **lack power** when applied to high-dimensional data. Even if the **number of collected data points is large**, they remain sparsely submerged in a voluminous high-dimensional space that is practically impossible to explore exhaustively" [23]
- "Generalizing correctly becomes **exponentially harder** as the dimensionality (number of features) of the examples grows, because **a fixed-size training** set covers a **dwindling fraction of the input space**" [24]

Curse of dimensionality

- Consider features with m **discrete** values, the **volume** of the **input space grows as** m^D!
- "Because of "**the curse of dimensionality**," many statistical methods **lack power** when applied to high-dimensional data. Even if the **number of collected data points is large**, they remain sparsely submerged in a voluminous high-dimensional space that is practically impossible to explore exhaustively" [23]
- "Generalizing correctly becomes **exponentially harder** as the dimensionality (number of features) of the examples grows, because **a fixed-size training** set covers a **dwindling fraction of the input space**" [24]

Curse of dimensionality

- Consider features with m **discrete** values, the **volume** of the **input space grows as** m^D!
- "Because of "**the curse of dimensionality**," many statistical methods **lack power** when applied to high-dimensional data. Even if the **number of collected data points is large**, they remain sparsely submerged in a voluminous high-dimensional space that is practically impossible to explore exhaustively" [23]
- "Generalizing correctly becomes **exponentially harder** as the dimensionality (number of features) of the examples grows, because **a fixed-size training** set covers a **dwindling fraction of the input space**" [24]

$\Rightarrow$ As the **number of dimensions increases**, the **number of examples needs to grow exponentiallly!**

Dimensionality reduction

- Our **intuition** fails us beyond three (3) dimensions — yet many machine learning problems comprise tens, hundreds, thousands, even millions of features!

Dimensionality reduction

- Our **intuition** fails us beyond three (3) dimensions — yet many machine learning problems comprise tens, hundreds, thousands, even millions of features!
- "Dimensionality reduction **removes redundant** or **highly correlated features**; it also reduces the noise in the data — all that contributes to the **interpretability** of the model." [2]

Dimensionality reduction

- Our **intuition** fails us beyond three (3) dimensions — yet many machine learning problems comprise tens, hundreds, thousands, even millions of features!
- "Dimensionality reduction **removes redundant** or **highly correlated features**; it also reduces the noise in the data — all that contributes to the **interpretability** of the model." [2]
- In a sense, **dimensionality reduction** can be seen a way to **compress** the data - going from 10,000 features down to 100 features.

Dimensionality reduction

- Our **intuition** fails us beyond three (3) dimensions — yet many machine learning problems comprise tens, hundreds, thousands, even millions of features!
- "Dimensionality reduction **removes redundant** or **highly correlated features**; it also reduces the noise in the data — all that contributes to the **interpretability** of the model." [2]
- In a sense, **dimensionality reduction** can be seen a way to **compress** the data - going from 10,000 features down to 100 features.

Dimensionality reduction

- Our **intuition** fails us beyond three (3) dimensions — yet many machine learning problems comprise tens, hundreds, thousands, even millions of features!
- "Dimensionality reduction **removes redundant** or **highly correlated features**; it also reduces the noise in the data — all that contributes to the **interpretability** of the model." [2]
- In a sense, **dimensionality reduction** can be seen a way to **compress** the data - going from 10,000 features down to 100 features.

See also:

- Lan Huong Nguyen and Susan Holmes. Ten quick tips for effective dimensionality reduction. *PLoS Comput Biol*, 15(6):e1006907, Jun 2019.

Objectives

Dimensionality reduction serves **two main objectives**:

- Data **visualization** and **exploration**
- **Speeding-up** machine learning experiments

Objectives

Dimensionality reduction serves **two main objectives**:

- Data **visualization** and **exploration**
- **Speeding-up** machine learning experiments

Some people use **dimensionality reduction** techniques to address the problem of **overfitting**. However, this is not considered to be the right approach. Instead, **rugularization** should be applied.

A word of caution

According to **Aurélien Géron**, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow* 2019, § 8:

> *"In **some cases**, reducing the dimensionality of the training data **may filter out some noise** and unnecessary details and thus result in **higher performance**, but **in general** it **won't**; it will just **speed** up training."*

A word of caution

According to **Aurélien Géron**, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow* 2019, § 8:

> *"In **some cases**, reducing the dimensionality of the training data **may filter out some noise** and unnecessary details and thus result in **higher performance**, but **in general** it **won't**; it will just **speed** up training."*

Here is what **Ankur S. Patel**, *Hands-on Unsupervised Learning with Python* 2019, has to say (§ 1):

> *"With **dimensionality reduction**, we can find the most salient features in the original feature set, reduce the number of dimensions to a more manageable number while **losing very little important information** in the process, and then apply supervised algorithms to more efficiently perform the search for a good function approximation."*

Advice

- **Dimensionality reduction** makes easier to **visualize** your data and gain **insights into its structure** — **#DataVisualization**, **#DataExploration**.

Dimensionality reduction

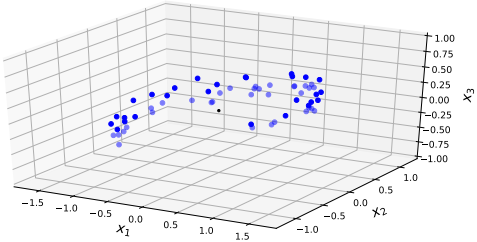
- **Projection** — Principal Component Analysis (PCA)
- **Manifold Lerning**

Dimensionality reduction

- **Projection** — Principal Component Analysis (PCA)
- **Manifold Lerning**

Projection - intuition

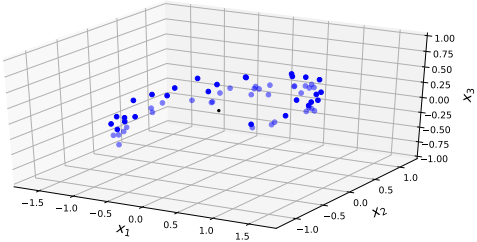
- Imagine the extreme situation where you would like to use a **single dimension** to represent the data.



Source: Adapted [1] Figure 8.2

Projection - intuition

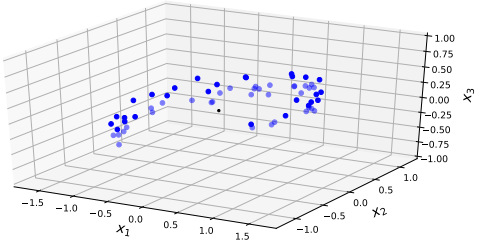
- Imagine the extreme situation where you would like to use a **single dimension** to represent the data.
  - Here, I do **not** mean selecting a **single (most informative) feature** — this would be called **feature selection**.



Source: Adapted [1] Figure 8.2

Projection - intuition

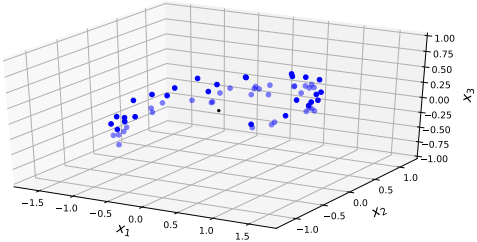
- Imagine the extreme situation where you would like to use a **single dimension** to represent the data.
  - Here, I do **not** mean selecting a **single (most informative) feature** — this would be called **feature selection**.
  - I mean a **new representation**, where each example i, is represented by a vector, Z_i, with only one column.



Source: Adapted [1] Figure 8.2

Projection - intuition

- Imagine the extreme situation where you would like to use a **single dimension** to represent the data.
  - Here, I do **not** mean selecting a **single (most informative) feature** — this would be called **feature selection**.
  - I mean a **new representation**, where each example i, is represented by a vector, Z_i, with only one column.
- **How** would you do that?



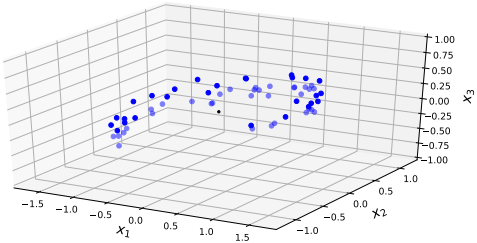
Source: Adapted [1] Figure 8.2

Projection - intuition

- We are looking for a vector (Z_1) (a line) **minimizing the average squared projection error**:

$$\frac{1}{N}\sum_{i=1}^{N}||x_i - \overline{x}_i||^2$$

where $\overline{x}_i$ is the projection of x_i onto that vector.



Source: Adapted [1] Figure 8.2

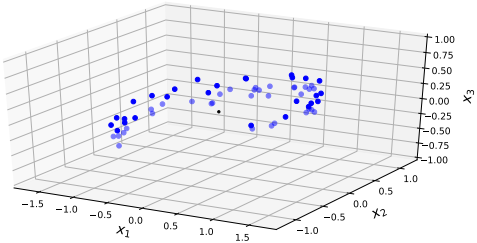
Projection - intuition

- We are looking for a vector (Z_1) (a line) **minimizing the average squared projection error**:

$$\frac{1}{N}\sum_{i=1}^{N}||x_i - \overline{x}_i||^2$$

where $\overline{x}_i$ is the projection of x_i onto that vector.

- **What** would this line look like?



Source: Adapted [1] Figure 8.2

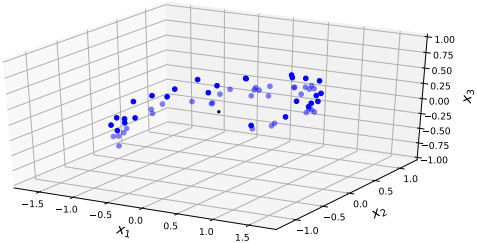
Projection - intuition

- We are looking for a vector (Z_1) (a line) **minimizing the average squared projection error**:

$$\frac{1}{N}\sum_{i=1}^{N}||x_i - \overline{x}_i||^2$$

where $\overline{x}_i$ is the projection of x_i onto that vector.

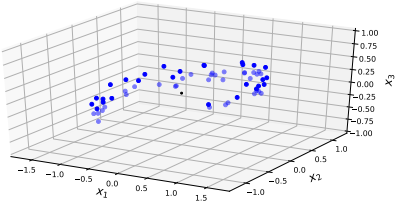
- **What** would this line look like?
  - This would be a **projection** that preserves as much of the **variance** as possible.



Source: Adapted [1] Figure 8.2

Projection - intuition

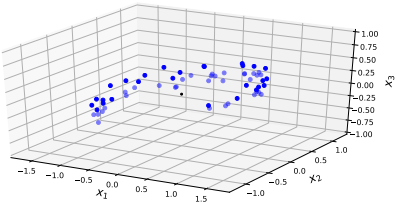
- You now would now like to use **two dimensions** to represent the data.



Source: Adapted from [1] Figure 8.2

Projection - intuition

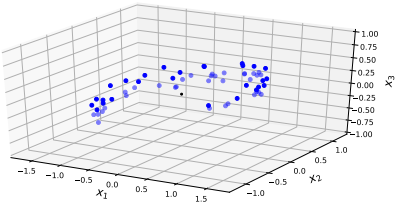
- You now would now like to use **two dimensions** to represent the data.
- Given our first choice of vector, **how** would you select a **second vector**?



Source: Adapted from [1] Figure 8.2

Projection - intuition

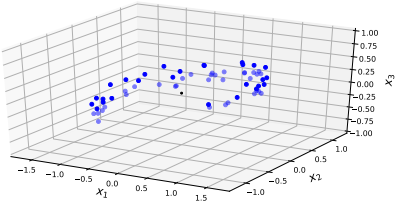
- You now would now like to use **two dimensions** to represent the data.
- Given our first choice of vector, **how** would you select a **second vector**?
  - You want this vector to be **orthogonal** to the first the vector. Do you see **why**?



Source: Adapted from [1] Figure 8.2

Projection - intuition

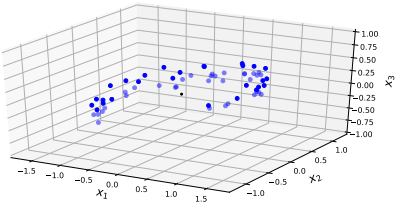
- You now would now like to use **two dimensions** to represent the data.
- Given our first choice of vector, **how** would you select a **second vector**?
  - You want this vector to be **orthogonal** to the first the vector. Do you see **why**?
  - Otherwise, there would a (linear) dependency between the vectors, which is what we are trying to eliminate. We are looking for **independent** features.



Source: Adapted from [1] Figure 8.2

Projection - intuition

- You now would now like to use **two dimensions** to represent the data.
- Given our first choice of vector, **how** would you select a **second vector**?
  - You want this vector to be **orthogonal** to the first the vector. Do you see **why**?
  - Otherwise, there would a (linear) dependency between the vectors, which is what we are trying to eliminate. We are looking for **independent** features.
  - Our **first** and **second** vector are now forming a **plane**.



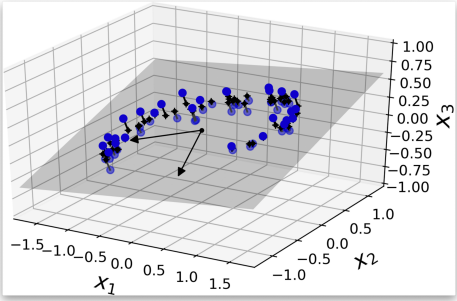
Source: Adapted from [1] Figure 8.2

Projection - intuition

- We are looking for a **second** vector (Z_2) **minimizing the average squared projection error**:

$$\frac{1}{N}\sum_{i=1}^{N}||x_i - \overline{x}_i||^2$$

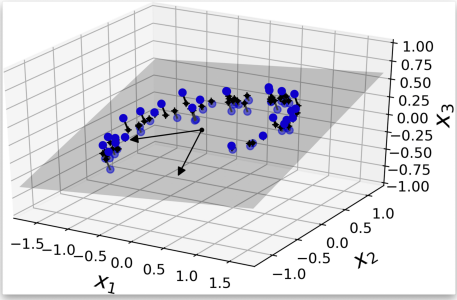
where $\overline{x}_i$ is the projection of x_i onto the plane formed by the two selected vectors.



Source: [1] Figure 8.2

Projection - intuition

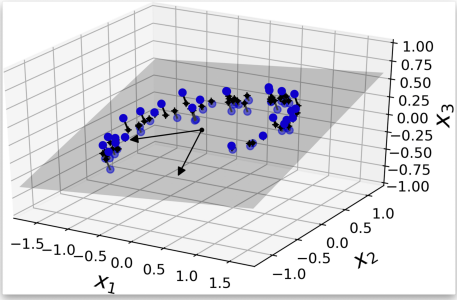
- If our data had more than three (3) dimension, $D >> 3$, we could continue adding new vectors.



Source: [1] Figure 8.2

Projection - intuition

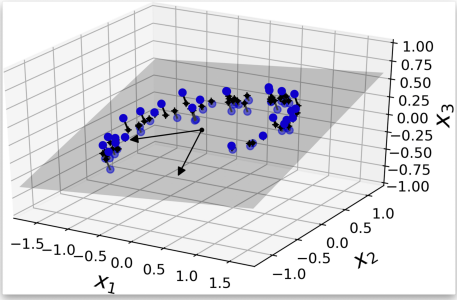
- If our data had more than three (3) dimension, $D >> 3$, we could continue adding new vectors.
- We would select a **third** vector (Z_3) so as to minimize the **avererage squared projection error**. The data would now be projected onto a **cube**.



Source: [1] Figure 8.2

Projection - intuition

- If our data had more than three (3) dimension, $D >> 3$, we could continue adding new vectors.
- We would select a **third** vector (Z_3) so as to minimize the **avererage squared projection error**. The data would now be projected onto a **cube**.
- This process can be repeated for all possible $k < D$. We would now be projecting the data in a k dimensional space, which we cannot easily visualize if $k > 3$.



Source: [1] Figure 8.2

Projection - remarks

- Notice that, the **first** component (Z_1) explains most of the variation (**variance**).

Projection - remarks

- Notice that, the **first** component (Z_1) explains most of the variation (**variance**).
- The **second** component (Z_2) explains less of the **variance** than the first component, but more than the third.

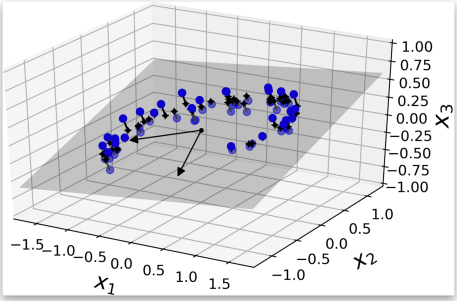
Projection - remarks

- Notice that, the **first** component (Z_1) explains most of the variation (**variance**).
- The **second** component (Z_2) explains less of the **variance** than the first component, but more than the third.
- As **more and more** components are added, the projection error decreases, **more and more** of the variation **variance** in the data is explained.

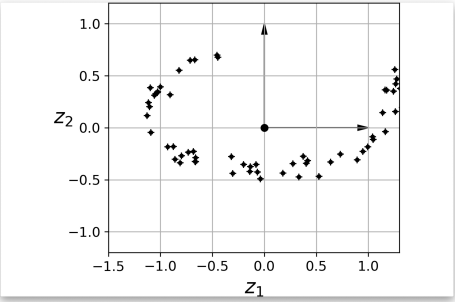
Projection - remarks

- Notice that, the **first** component (Z_1) explains most of the variation (**variance**).
- The **second** component (Z_2) explains less of the **variance** than the first component, but more than the third.
- As **more and more** components are added, the projection error decreases, **more and more** of the variation **variance** in the data is explained.
- Using $k = D$ components would reduce the variance to 0 [is this true?].

Projection



$\Rightarrow$

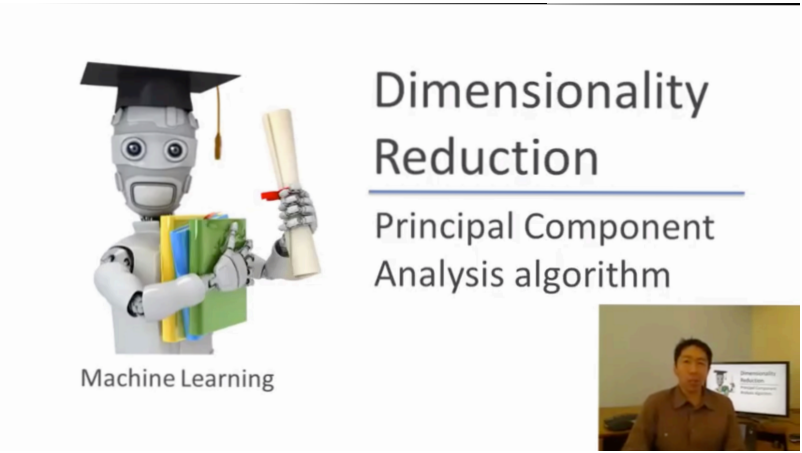


Source: [1] Figures 8.2 & 8.3

Before applying PCA

- The data should be **centered** (mean normalization) and possibly **scaled**.
- **Scikit-Learn** will take care of centering the data for you.

PCA algorithm - Andrew Ng



https://youtu.be/rng04VJxUt4

Choosing k - Andrew Ng

Choosing k (number of principal components)

Average squared projection error: $\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - x_{approx}^{(i)}\|^2$

Total variation in the data: $\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2$

Typically, choose k to be smallest value so that

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2} \leq 0.01 \qquad (1\%)$$

"99% of variance is retained"

Andrew Ng

https://youtu.be/5aHWplWElcc

sklearn.decomposition.PCA

```
from sklearn.decomposition import PCA

pca = PCA(n_components = 2)
Z = pca.fit_transform(X)
```

```
>>> pca.explained_variance_ratio_
array([0.84248607, 0.14631839])
```

Source: [1] §8

Principal Component Analysis

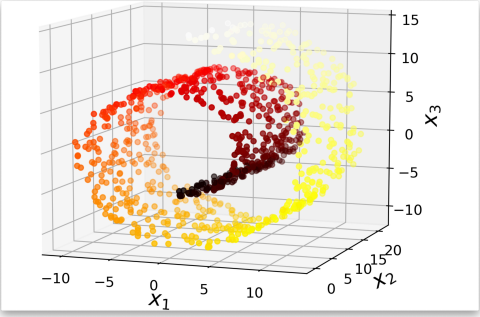
- Ma, S. & Dai, Y. Principal component analysis based methods in bioinformatics studies. *Brief Bioinform* **12,** 714722 (2011).
  - "Variable selection approaches search for a subset of genes to represent the effects of all genes."
  - "In contrast, dimension reduction approaches search for a small number of metagenes, which are often linear combinations of all genes."
  - "The dimensionality of gene expressions needs to be reduced prior to regression and many other types of analyses."
  - "In contrast, in gene profiling studies, only a small number of genes profiled are expected to be associated with the response variables and the majority of the genes are noises."
  - For future reference: Supervised and sparse PCA are said to be more effective than standard PCA.

Principal Component Analysis

- K Y Yeung and W L Ruzzo, Principal component analysis for clustering gene expression data, *Bioinformatics* **17** (2001), no. 9, 76374.
- Michael Lenz, Franz-Josef Müller, Martin Zenke, and Andreas Schuppert, Principal components analysis and the reported low intrinsic dimensionality of gene expression microarray data, *Sci Rep* **6** (2016), 25696.
- Lever, J., Krzywinski, M. & Atman, N. POINTS OF SIGNIFICANCE Principal component analysis. Nat Meth 14, 641642 (2017).
- Ringnér, M. What is principal component analysis? *Nat Biotechnol* **26**, 303304 (2008).
- Joseph C Roden, Brandon W King, Diane Trout, Ali Mortazavi, Barbara J Wold, and Christopher E Hart, Mining gene expression data by interpreting principal components, **BMC Bioinformatics 7** (2006), 194.
- https://www.kaggle.com/crawford/principle-component-analysis-gene-expression

Manifold Learning

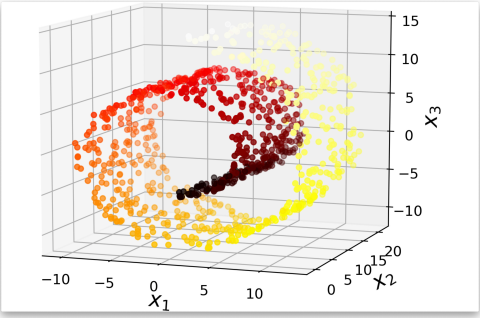
- "A manifold is a topological space that is locally Euclidean (...)". Wolfram MathWorld
- "Put simply, a 2D manifold is a 2D shape that can be bent and twisted in a higher-dimensional space. More generally, a d-dimensional manifold is a part of an n-dimensional space (where d $<$ n) that locally resembles a d-dimensional hyperplane." [1]



Source: [1] Figure 8.4

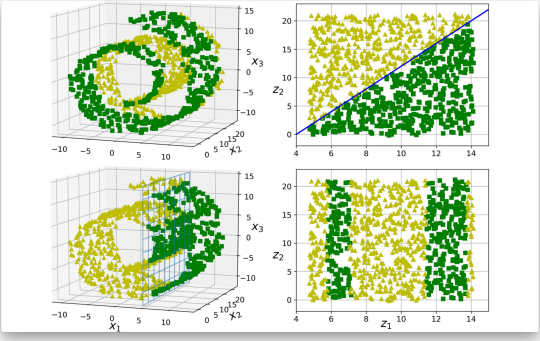
Manifold Learning

- "[T]he manifold hypothesis, which holds that most real-world high-dimensional datasets lie close to a much lower-dimensional manifold." [1]
- "The manifold assumption is often accompanied by another implicit assumption: that the task at hand (e.g., classification or regression) will be simpler if expressed in the lower-dimensional space of the manifold." [1]



Source: [1] Figure 8.4

Easier?



Source: [1] Figure 8.6

- "The decision boundary may not always be simpler with lower dimensions."

Summary

- **Dimensionality reduction** methods are linearly (PCA) or non-linearly transforming the data so as to reduce the number of features, thus speeding-up the downstream analysis (unsupervised/supervised learning).
- **Principal Component Analysis** (PCA) can be used to explore and visualize your data.
  - For example, if most of the variation in your data can be explained with a small number of components, then your data has many redundant features.
- Only apply dimensionality **if needed** (or compare with and without data reduction).

Resources (videos)

Lectures on **Machine Learning** by **Andrew Ng** - specifically, the lectures on unsupervised learning and dimensionality reduction.

1. **Motivation I - data compression** (10 m 9 s)
2. **Motivation II - data visualization** (5 m 28 s)
3. **Principal Componant Analysis (PCA) - problem formulation** (9 m 5 s)
4. **Principal Componant Analysis (PCA) - algorithm*** (15 m 14 s)
5. **Choosing the number of principal components*** (10 m 30 s)
6. **Reconstruction from compressed representation** (3 m 54 s)
7. **Advice for applying PCA** (12 m 48 s)

I find the videos indicated with $*$ particularly insightful.

2020

- Double clustering algorithms for gene profiling.
- Presenting clustering algorithms that are specific to bioinformatics.
- Disucssion on single-cell RNA-seq data.
- Discussion on feature selection methods.
  - Jing Tang, Yunxia Wang, Jianbo Fu, Ying Zhou, Yongchao Luo, Ying Zhang, Bo Li, Qingxia Yang, Weiwei Xue, Yan Lou, Yunqing Qiu, and Feng Zhu. A critical assessment of the feature selection methods used for biomarker discovery in current metaproteomics studies. *Brief Bioinform*, Jun 2019.

Prologue

Summary

- Most of the available data is **unlabelled**.

Summary

- Most of the available data is **unlabelled**.
- **Transcriptomics** serves many purposes, including gene function annotation.

Summary

- Most of the available data is **unlabelled**.
- **Transcriptomics** serves many purposes, including gene function annotation.
- Clustering algorithms are generally **simple**. We considered KMeans, hierarchical, and DBSCAN.

Summary

- Most of the available data is **unlabelled**.
- **Transcriptomics** serves many purposes, including gene function annotation.
- Clustering algorithms are generally **simple**. We considered KMeans, hierarchical, and DBSCAN.
- Finding the **number of optimal clusters** is **not simple**

Summary

- Most of the available data is **unlabelled**.
- **Transcriptomics** serves many purposes, including gene function annotation.
- Clustering algorithms are generally **simple**. We considered KMeans, hierarchical, and DBSCAN.
- Finding the **number of optimal clusters** is **not simple**
- Removing **redundant features** will accelerate (supervised) **learning**

Summary

- Most of the available data is **unlabelled**.
- **Transcriptomics** serves many purposes, including gene function annotation.
- Clustering algorithms are generally **simple**. We considered KMeans, hierarchical, and DBSCAN.
- Finding the **number of optimal clusters** is **not simple**
- Removing **redundant features** will accelerate (supervised) **learning**
- Dimensionality reduction is more about speed than learning accuracy?

Next module

- **Linear** and **logistic** regression

References

Aurélien Géron.
Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow.
O'Reilly Media, 2nd edition, 2019.

Andriy Burkov.
The Hundred-Page Machine Learning Book.
Andriy Burkov, 2019.

Ankur A. Patel.
Hands-On Unsupervised Learning Using Python.
O'Reilly Media, 2019.

Pietro Coretto, Angela Serra, and Roberto Tagliaferri.
Robust clustering of noisy high-dimensional gene expression data for patients subtyping.
Bioinformatics, 34(23):4064–4072, 12 2018.

References

Troy P Hubbard, Jonathan D D Gama, Gabriel Billings, Brigid M Davis, and Matthew K Waldor.
Unsupervised learning approach for comparing multiple transposon insertion sequencing studies.
mSphere, 4(1), 02 2019.

Alexander Rives, Siddharth Goyal, Joshua Meier, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, and Rob Fergus.
Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences.
bioRxiv, page 622803, 2019.

Lokesh Kumar and Matthias E Futschik.
Mfuzz: a software package for soft clustering of microarray data.
Bioinformation, 2(1):5–7, May 2007.

Tian Tian, Ji Wan, Qi Song, and Zhi Wei.
Clustering single-cell RNA-seq data with a model-based deep learning approach.
Nature Machine Intelligence, 1(4):191, 2019.

References

Xiaoping Su, Gabriel G Malouf, Yunxin Chen, Jianping Zhang, Hui Yao, Vicente Valero, John N Weinstein, Jean-Philippe Spano, Funda Meric-Bernstam, David Khayat, and Francisco J Esteva.
Comprehensive analysis of long non-coding RNAs in human breast cancer clinical subtypes.
Oncotarget, 5(20):9864–76, Oct 2014.

Matthew J Michalska-Smith and Stefano Allesina.
Telling ecological networks apart by their structure: A computational challenge.
PLoS Comput Biol, 15(6):e1007076, Jun 2019.

Ren Qi, Anjun Ma, Qin Ma, and Quan Zou.
Clustering and classification methods for single-cell RNA-sequencing data.
Brief Bioinform, Jul 2019.

Benjamin T James, Brian B Luczak, and Hani Z Girgis.
MeShClust: an intelligent tool for clustering DNA sequences.
Nucleic Acids Res, 46(14):e83, Aug 2018.

References

Qiu Xiao, Jiawei Luo, Cheng Liang, Jie Cai, Guanghui Li, and Buwen Cao.
CeModule: an integrative framework for discovering regulatory patterns from genomic data in cancer.
BMC Bioinformatics, 20(1):67, Feb 2019.

Davide Chicco and Marco Masseroli.
Ontology-based prediction and prioritization of gene functional annotations.
IEEE/ACM Trans Comput Biol Bioinform, 13(2):248–60, 2016.

Debajyoti Sinha, Akhilesh Kumar, Himanshu Kumar, Sanghamitra Bandyopadhyay, and Debarka Sengupta.
dropClust: efficient clustering of ultra-large scRNA-seq data.
Nucleic Acids Res, 46(6):e36, 04 2018.

Xiangtao Li, Shixiong Zhang, and Ka-Chun Wong.
Single-cell RNA-seq interpretations using evolutionary multiobjective ensemble pruning.
Bioinformatics, 35(16):2809–2817, Aug 2019.

References

Hung Nguyen, Sangam Shrestha, Sorin Draghici, and Tin Nguyen.
PINSPlus: a tool for tumor subtype discovery in integrated genomic data.
Bioinformatics, 35(16):2843–2846, Aug 2019.

Hang Liu, Lei Peng, Joan So, Ka Hing Tsang, Chi Ho Chong, Priscilla Hoi Shan Mak, Kui Ming Chan, and Siu Yuen Chan.
TSPYL2 regulates the expression of EZH2 target genes in neurons.
Mol Neurobiol, 56(4):2640–2652, Apr 2019.

Pallavi Gaur and Anoop Chaturvedi.
Clustering and candidate motif detection in exosomal miRNAs by application of machine learning algorithms.
Interdiscip Sci, 11(2):206–214, Jun 2019.

Vladimir Yu Kiselev, Tallulah S Andrews, and Martin Hemberg.
Challenges in unsupervised clustering of single-cell RNA-seq data.
Nat Rev Genet, 20(5):273–282, 05 2019.

References

Rachel Jeitziner, Mathieu Carrière, Jacques Rougemont, Steve Oudot, Kathryn Hess, and Cathrin Brisken.
Two-Tier Mapper, an unbiased topology-based clustering method for enhanced global gene expression analysis.
Bioinformatics, 35(18):3339–3347, Sep 2019.

Michael Molla, Michael Waddell, David Page, and Jude W. Shavlik.
Using machine learning to design and interpret gene-expression microarrays.
AI Magazine, 25(1):23–44, 2004.

Lan Huong Nguyen and Susan Holmes.
Ten quick tips for effective dimensionality reduction.
PLoS Comput Biol, 15(6):e1006907, Jun 2019.

Pedro M. Domingos.
A few useful things to know about machine learning.
Commun. ACM, 55(10):78–87, 2012.

Marcel **Turcotte**

Marcel.Turcotte@uOttawa.ca

School of Electrical Engineering and **Computer Science** (EECS)
University of Ottawa