



COMP30810

Intro to Text Analytics

Dr. Binh Thanh Le

thanhbinh.le@ucd.ie

Insight Centre for Data Analytics

School of Computer Science

University College Dublin

Today Goals:

- Understand Dimensionality Reduction
- Understand Feature Extraction / Selection
- Understand: What / When / Why to use Feature Reduction

What is Dimensionality Reduction?

- Significant improvements can be achieved by first **mapping** the data into a **lower-dimensional** space.

$$x = \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_N \end{bmatrix} \dashrightarrow \text{reduce dimensionality} \dashrightarrow y = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_K \end{bmatrix} \quad (K \ll N)$$

- Dimensionality can be reduced by:
 - Combining features using a **linear** or **non-linear** transformations.
 - Selecting a subset of features (i.e., **feature selection**).

Why Dimensionality Reduction?

- Some features may be irrelevant
- We want to visualize high dimensional data
- “Intrinsic” dimensionality may be smaller than the number of features



Curse of Dimensionality



Figure 2. A single feature does not result in a perfect separation of our training data.

→ Add another feature <Feature2>
e.g. the average 'green' color in the image

Figure 2 shows that we do not obtain a perfect classification result if only a single feature <Feature1> is used.

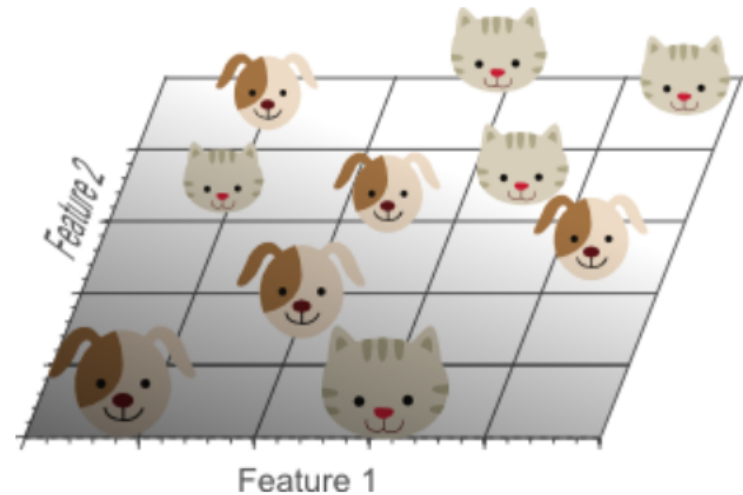


Figure 3. Adding a second feature still does not result in a linearly separable classification problem: No single line can separate all cats from all dogs in this example.

Curse of Dimensionality

→ Add a third feature <Feature3>
E.g. the average 'blue' color in the image,
→ three-dimensional feature space

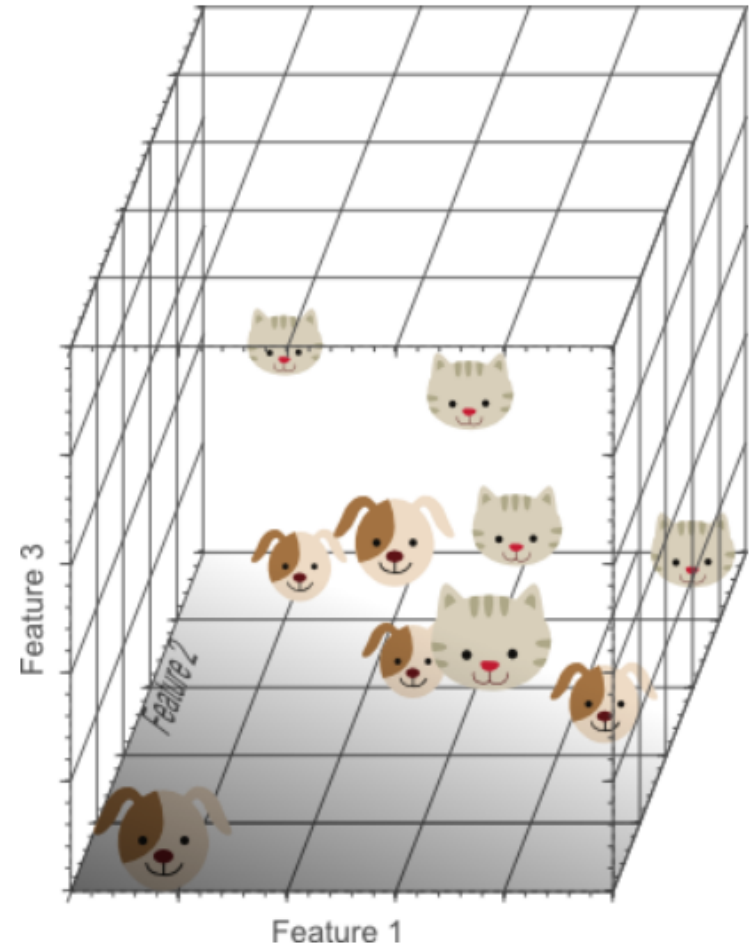


Figure 4. Adding a third feature results in a linearly separable classification problem in our example. A plane exists that perfectly separates dogs from cats.

Curse of Dimensionality

- ➔ Now, can find a plane that perfectly separates dogs from cats.
- ➔ This means that a linear combination of the three features can be used to obtain perfect classification results on our training data of 10 images.

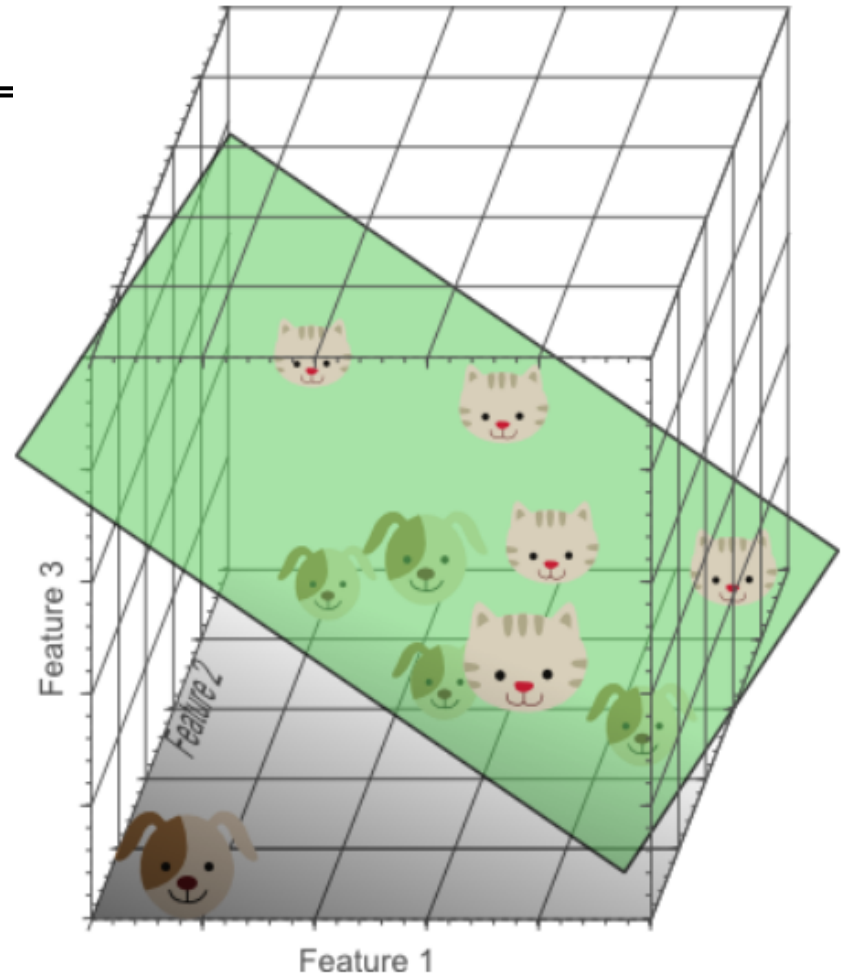


Figure 5. The more features we use, the higher the likelihood that we can successfully separate the classes perfectly.

Curse of Dimensionality

- If we would keep adding features, *the dimensionality of the feature space grows, and becomes sparser and sparser.*
- Due to this **sparsity**, it becomes much easier to find a separable hyperplane.

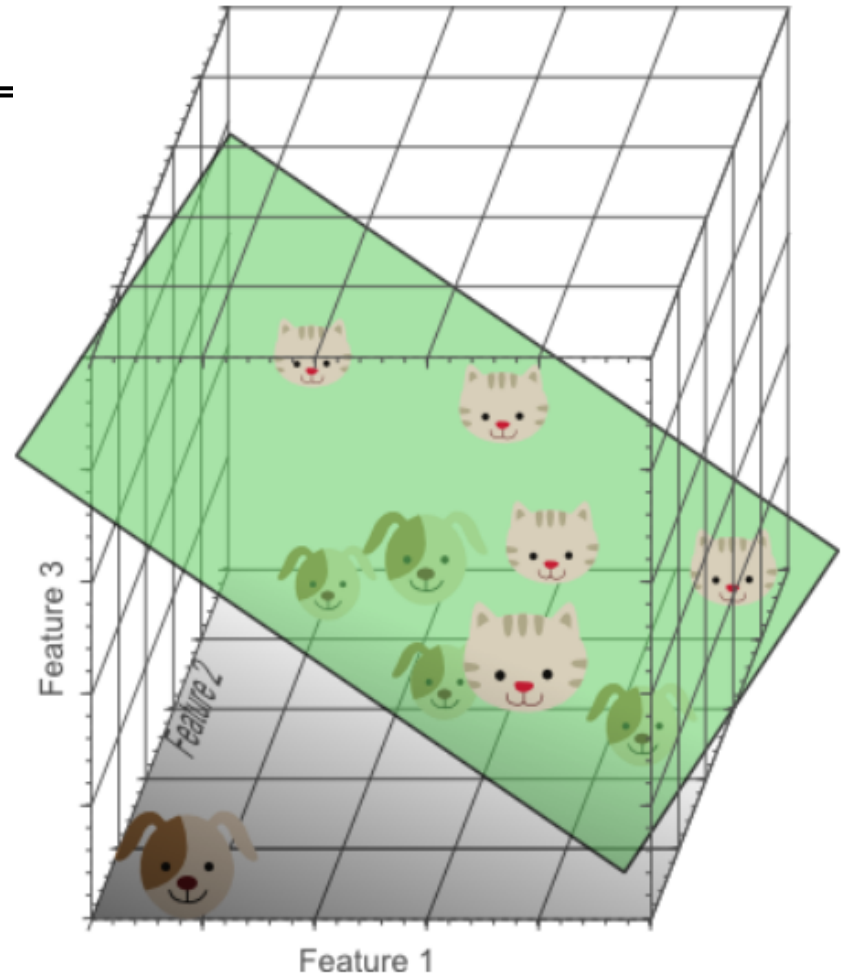


Figure 5. The more features we use, the higher the likelihood that we can successfully separate the classes perfectly.

Curse of Dimensionality

However, if we project the highly dimensional classification result back to a lower dimensional space, **a serious problem** associated with this approach becomes evident

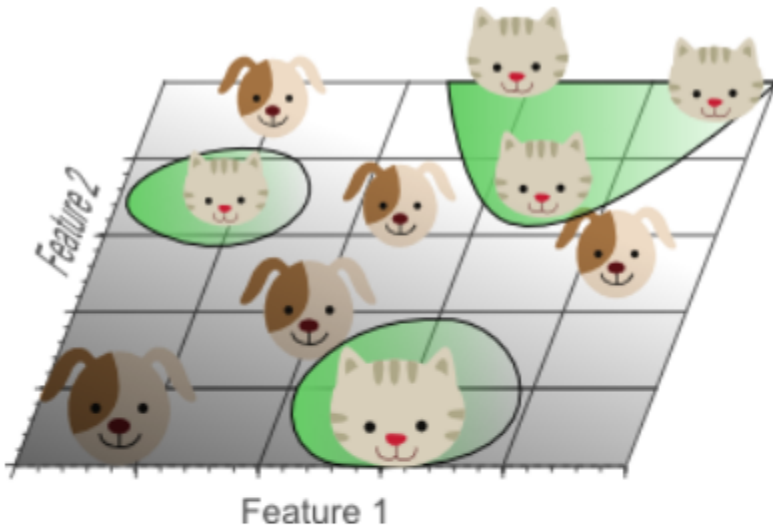


Figure 6. Using too many features results in overfitting. The classifier starts learning exceptions that are specific to the training data and do not generalize well when new data is encountered.

- The resulting classifier would fail on real-world data, consisting of an infinite amount of unseen cats and dogs.
- **OVERFITTING**

This concept is called **overfitting** and is a direct result of the **Curse of Dimensionality**.

Curse of Dimensionality

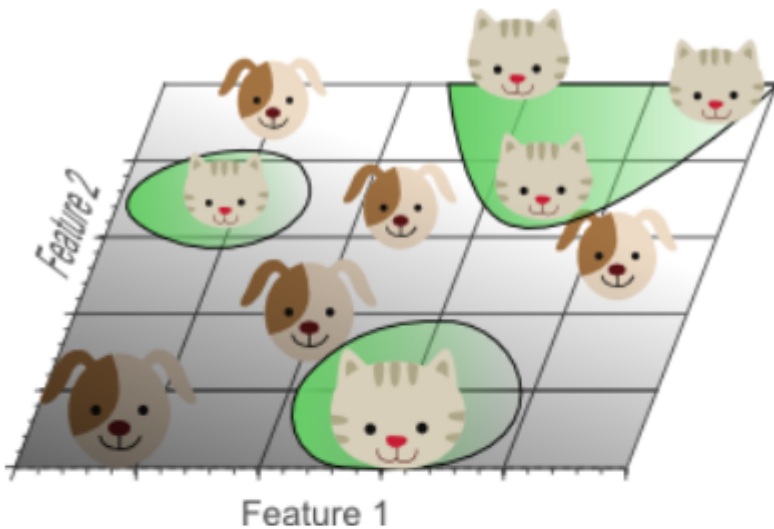


Figure 6. Using too many features results in overfitting. The classifier starts learning exceptions that are specific to the training data and do not generalize well when new data is encountered.

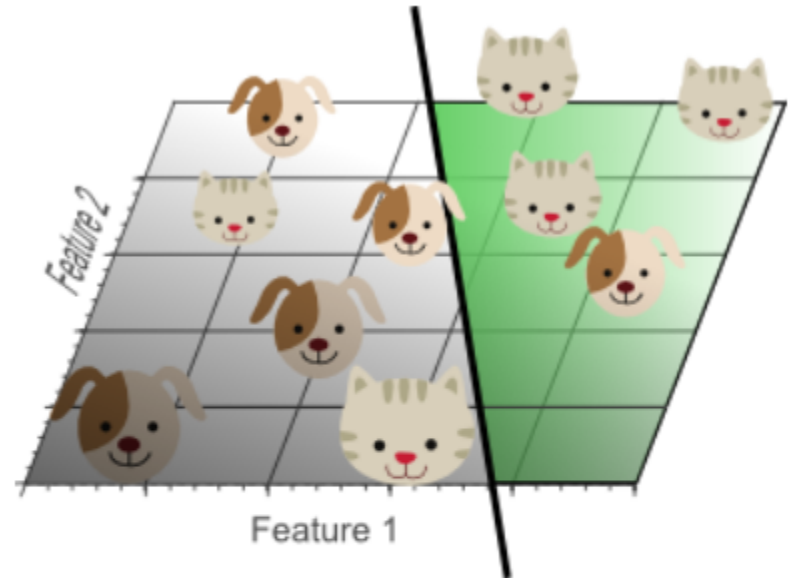
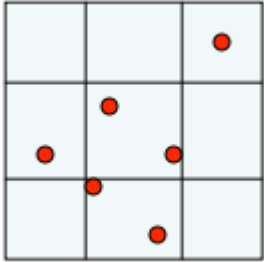
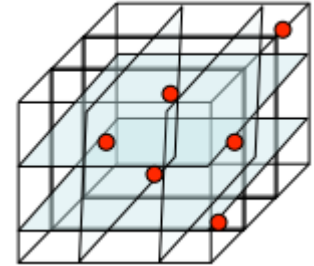


Figure 7. Although the training data is not classified perfectly, this classifier achieves better results on unseen data than the one from figure 5.

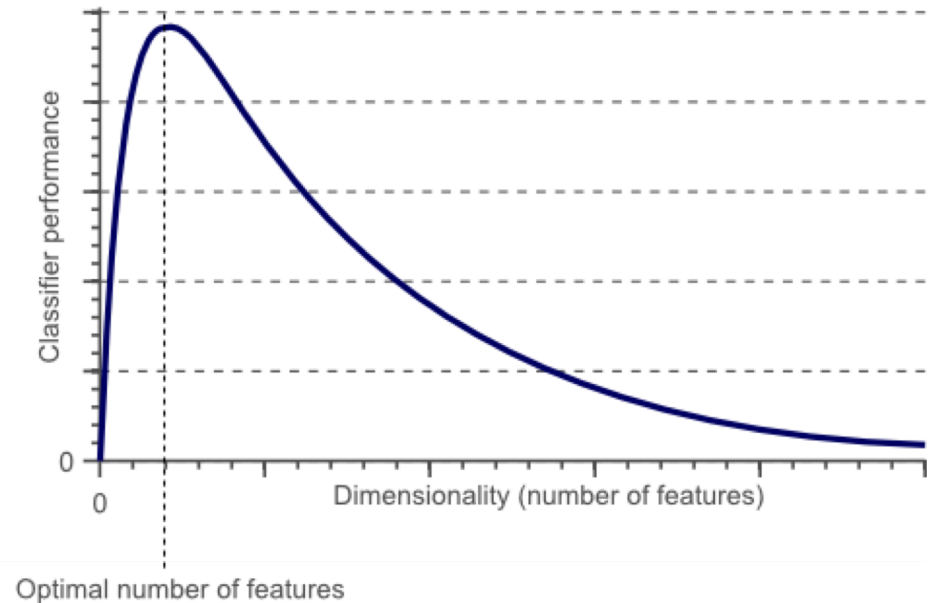
Curse of Dimensionality



- Same number of examples
- Fill more of the available space
- Back to the performance when the dimensionality is low



- Increase more #dimensions, while keep #samples
 - Overfitting
 - Performance decrease



What about Text Data ?

- While we tend to think of columns or features as dimensions, the text data works a bit differently.
- In text, every word or term becomes a dimension - it can get to 10000s of dimensions quickly.
- Just trying to find what topic a paragraph of text has, you are probably looking at 50 to 100 dimensions right there !
- Therefore text models **need a lot of data to train with!**
- When doing classification on some unknown piece of text, the models that you use are probably trained on a large corpus of data to minimize the effects of the Curse of Dimensionality.

Extraction vs Selection

X1	X2	X3	...	Xn

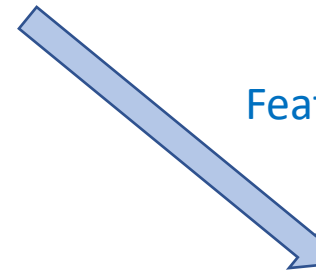
Feature Extraction



X'1	X'2	X'3	...	X'k

$$(X'_1, X'_2, \dots, X'_k) = f(X_1, X_2, \dots, X_n)$$

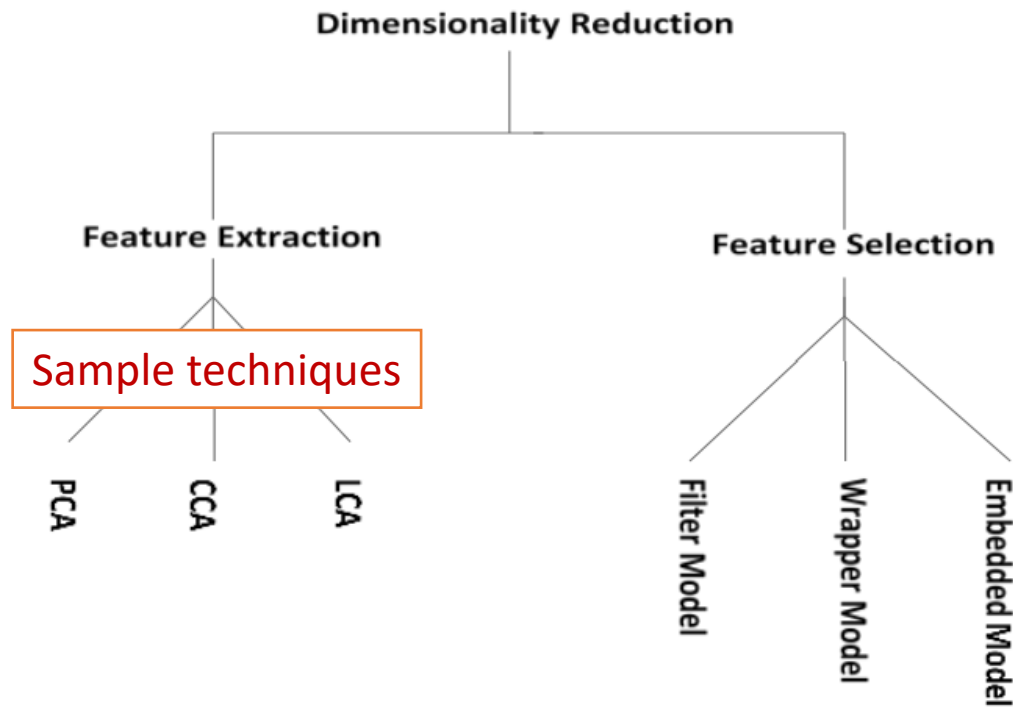
Feature Selection



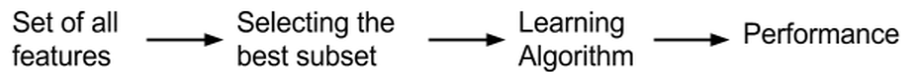
X1	X3	X4	...	Xk

$$(X_1, X_3, \dots, X_k) \subset (X_1, X_2, \dots, X_n)$$

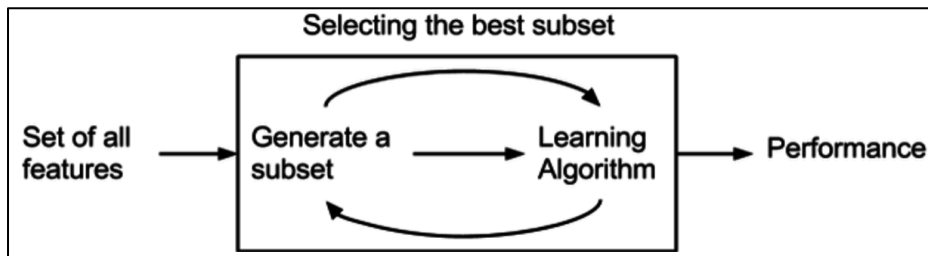
The key difference between **feature selection** and **extraction** is that *feature selection keeps a subset of the original features* while *feature extraction creates brand new ones*.



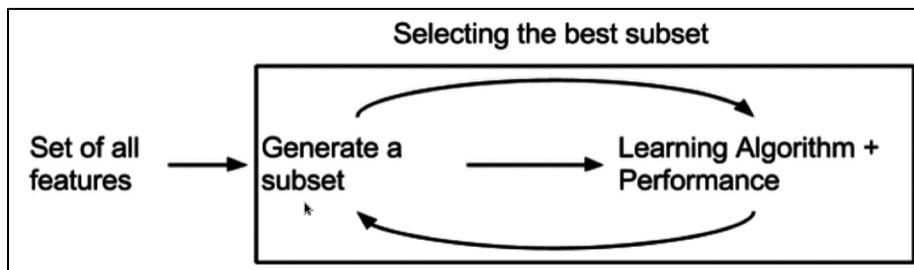
Filter method



Wrapper method

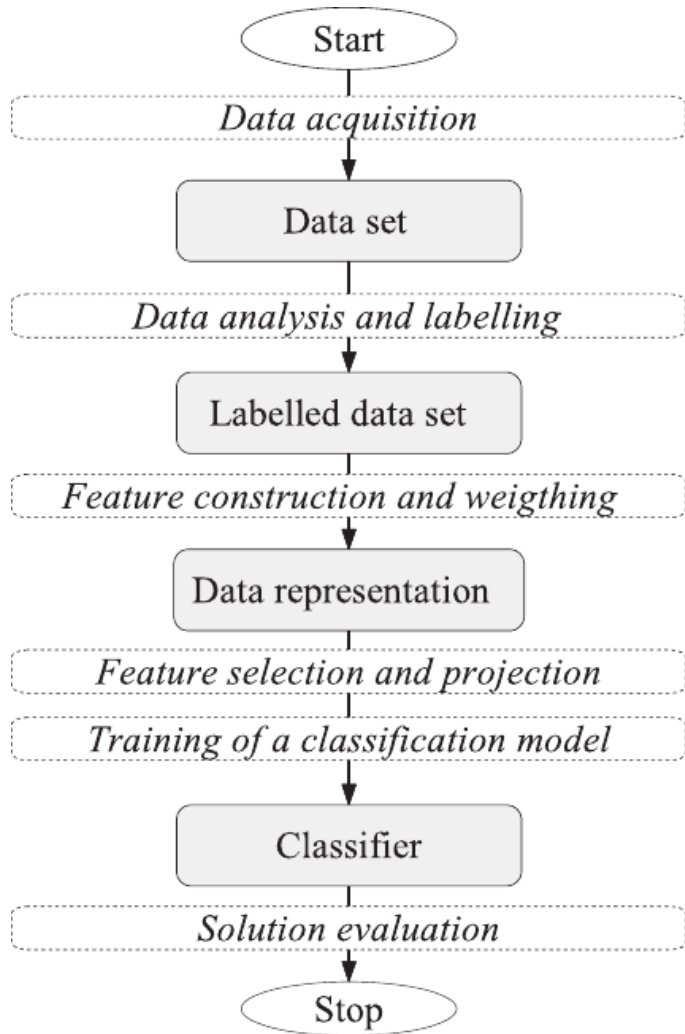


Embedded method



Evaluation methods:

- Accuracy / Probabilities with cross-validation
- Mean Square Error
- Weighted cost
- Information gain / Entropy
- ...



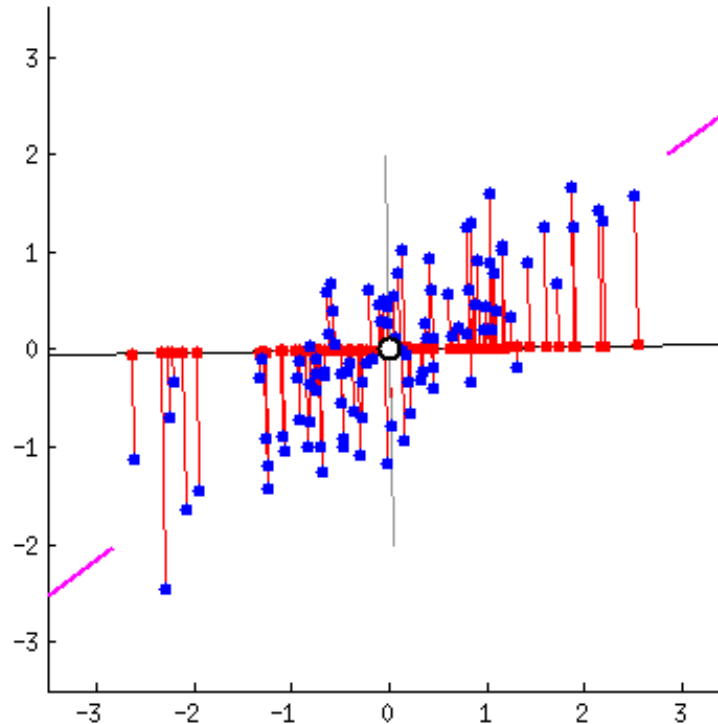
Questions:

- 1) Feature selection → Split train|test ?
- 2) Normalization → Feature selection?

Flowchart of the text classification process

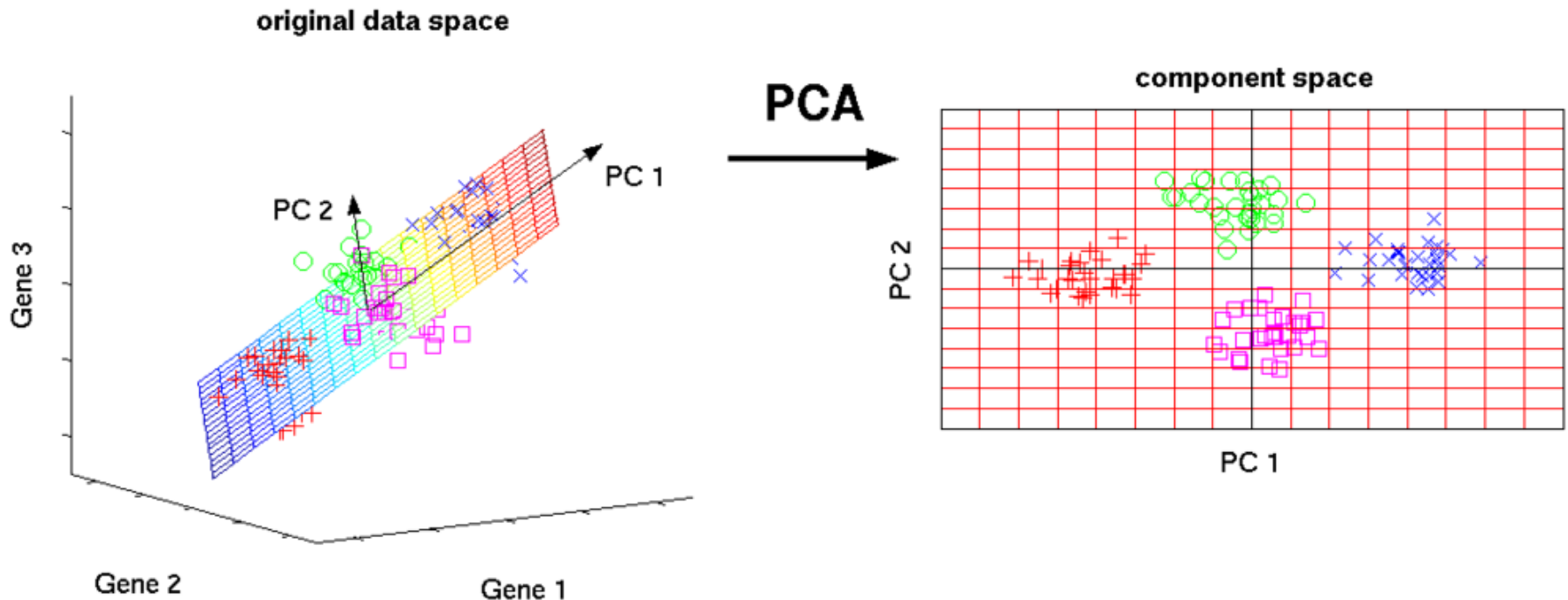
Principle Component Analysis (PCA)

Key idea: Generate the component axes that maximize the variance of data



Principle Component Analysis (PCA)

Key idea: Generate the component axes that maximize the variance of data



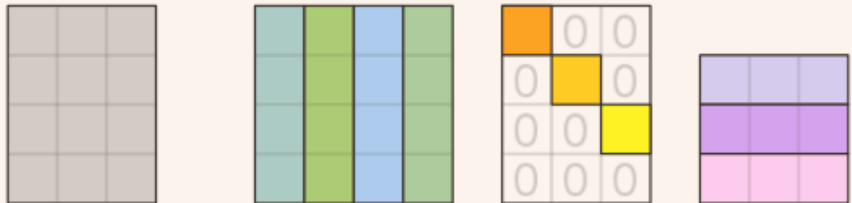
<https://plot.ly/ipython-notebooks/principal-component-analysis/>

http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf

Latent Semantic Analysis (LSA)

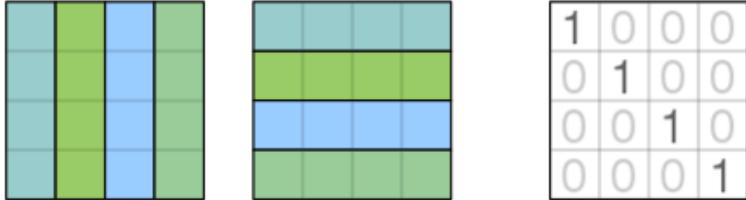
(aka Truncated SVD)

- $U : (m \times m)$ **Orthogonal Eigenvectors** of MM^T
- Σ : **diagonal** $(m \times n)$ matrix with non-negative real numbers on the diagonal \rightarrow Eigenvalues $\lambda_1 \dots \lambda_r$
- $V : (n \times n)$ **Orthogonal Eigenvectors** of M^TM



$$\begin{matrix} \text{4x3} & & \text{4x4} & & \text{4x3} & & \text{3x3} \\ \mathbf{M} & = & \mathbf{U} & & \mathbf{\Sigma} & & \mathbf{V}^* \\ m \times n & & m \times m & & m \times n & & n \times n \end{matrix}$$

Now, we truncate:



$$\begin{matrix} \text{4x4} & & \text{4x4} & & \text{4x4} \\ \mathbf{U} & & \mathbf{U}^* & = & \mathbf{I}_m \\ & & & & \\ \text{3x3} & & \text{3x3} & & \text{3x3} \\ \mathbf{V} & & \mathbf{V}^* & = & \mathbf{I}_n \end{matrix}$$

<https://blog.applied.ai/feature-reduction-using-svd/>

<http://cs229.stanford.edu/proj2017/final-reports/5163902.pdf>

<https://www.nmr.mgh.harvard.edu/PMI/toolbox/Documentation/tsvd.xhtml>

Latent Semantic Analysis (LSA)

(aka Truncated SVD)

Now, we truncate:

Full SVD

$$\begin{bmatrix} X \end{bmatrix} = \underbrace{\begin{bmatrix} \tilde{U} & \hat{U}_{\text{rem}} \end{bmatrix}}_{\hat{U}} \begin{bmatrix} \tilde{\Sigma} & \hat{\Sigma}_{\text{rem}} \\ 0 & \end{bmatrix} \begin{bmatrix} \tilde{V}^* \\ V_{\text{rem}} \end{bmatrix}$$

Truncated SVD

$$\approx \begin{bmatrix} \tilde{U} \end{bmatrix} \begin{bmatrix} \tilde{\Sigma} \\ \end{bmatrix} \begin{bmatrix} \tilde{V}^* \end{bmatrix}$$



Summary

We use Feature Extraction / Selection when:

- Avoid the Curse of Dimensionality
- Transform features
- Set up the consistent feature space for data
- Have a visualization for data