

MAT 3373

Methods of Machine Learning

Chapter 5

Additional Topics

P. Boily (uOttawa)

Fall – 2023

P. Boily (uOttawa)

Outline

5.1 – Dimension Reduction (p.4)

- Curse of Dimensionality (p.6)
- The Manifold Hypothesis (p.16)
- Nonlinear Dimensionality Reduction Methods (p.19)

5.2 – Feature Selection (p.38)

- Filter Methods (p.43)
- Wrapper Methods (p.61)
- Subset Selection Methods (p.64)
- Regularization Methods (p.66)
- SL and UL Methods (p.67)

Outline

5.3 – PCA Regression and Partial LS (p.69)

- Principal Components Analysis (p.72)
- PCA Formalism (p.87)
- PCA Regression (p.93)
- Partial Least Squares (p.96)

The Rest of the ML Picture (p.99)

References (p.100)

Main References:

- *Data Understanding, Data Analysis, and Data Science*, chapter 23.

5 – Additional Topics

Data mining is the collection of processes by which we extract actionable insights from data.

Inherent in this definition is the idea of **data reduction**: useful insights (whether in the form of summaries, sentiment analyses, etc.) ought to be “smaller” and “more organized” than the original raw data.

This chapter tackles the challenges presented by high data dimensionality in order to achieve insightful and interpretable analytical results.

The follow-up course is *MAT 4373 – Statistical Machine Learning*.

5.1 – Dimension Reduction

Advantages of working with reduced, low-dimensional data:

- **visualisation methods** of all kinds are available and readily applicable to such data (great for insight extraction);
- high-dimensional datasets are subject to the **curse of dimensionality** – when the number of features in a model increases, the number of observations required to maintain predictive power also increases, but at a **substantially faster rate**;
- in high-dimension sets, all observations are roughly **dissimilar** to one another – observations tend to be nearer the dataset's boundaries than they are to one another.

Dimension reduction techniques such as

- **principal component analysis** and **independent component analysis**;
- **factor analysis** (for numerical data) and **multiple correspondence analysis** (for categorical data)

project multi-dimensional datasets onto low-dimensional spaces with high-information content (see **Manifold Hypothesis**).

Some information is lost in the process, but the hope is that the loss is minimal and that the gains made by working with small-dimensional datasets can offset the losses.

5.1.1 – Curse of Dimensionality

A model is **local** if it depends solely on the observations near the input vector (k NN classification is **local**, linear regression is **global**).

With a large Tr , increasing k in a k NN model will yield enough points to provide a **solid approximation** to the theoretical classification boundary.

The **curse of dimensionality** (CoD) is the breakdown of this approach in **high-dimensional spaces**: going from 2 to 3 features, how many observations are required to maintain k NN's **predictive power**? Going from 2 to 10?

If the # of features increases but the # of observations does not, **local** models become **global** models (**global** models are not affected).

Manifestations of CoD

Let $x_i \sim U^1(0, 1)$ be i.i.d. for $i = 1, \dots, N$.

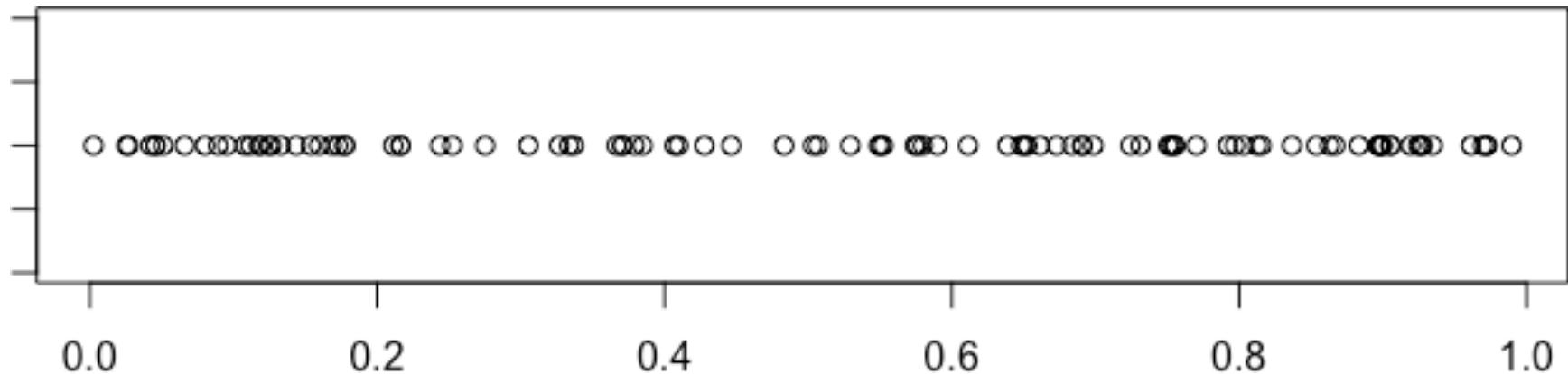
For any $z \in [0, 1]$ and $\varepsilon > 0$ such that

$$I_1(z; \varepsilon) = \{y \in \mathbb{R} : |z - y|_\infty < \varepsilon/2\} \subseteq [0, 1],$$

the expected number of observations x_i in $I_1(z; \varepsilon)$ is

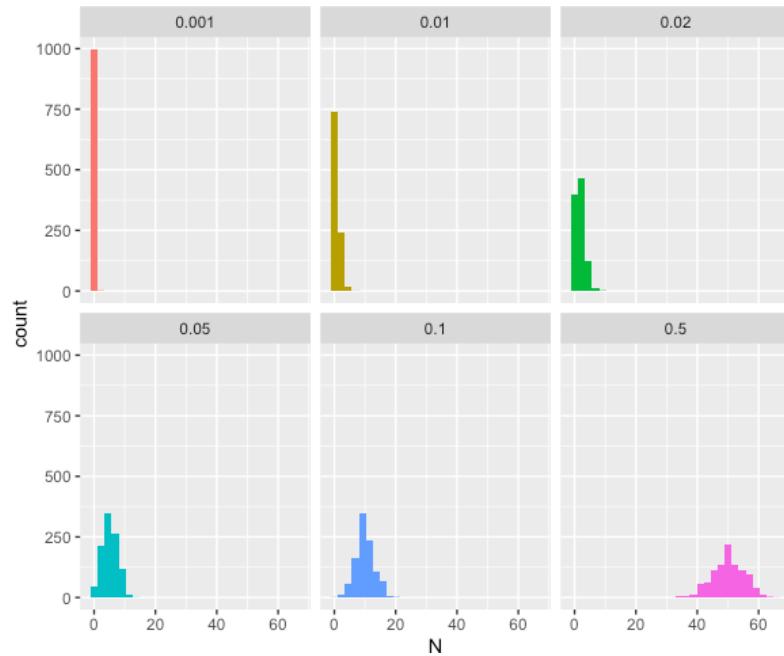
$$|I_1(z; \varepsilon) \cap \{x_i\}_{i=1}^N| \approx \varepsilon \cdot N.$$

In other words, an ε_∞ -ball subset of $[0, 1]^1$ contains $\approx \varepsilon$ of the observations in $\{x_i\}_{i=1}^N \subseteq \mathbb{R}$, on average.



In this instance ($N = 100$), the numbers of observations in $I_1(1/2, \varepsilon)$:

ε	0.001	0.01	0.02	0.05	0.1	0.5
$I_1 \cap \{x_i\}$	0	1	2	3	5	39
$E[I_1 \cap \{x_i\}] = \varepsilon N$	0.1	1	2	5	10	50



Repeating the experiment
1000 times yields:

ε	$\overline{I_1 \cap \{x_i\}}$	εN
0.001	0.09	0.1
0.01	1.01	1
0.02	2.01	2
0.05	4.99	5
0.1	9.85	10
0.5	50.10	50

Now, let $\mathbf{x}_i \sim U^2(0, 1)$ be i.i.d. for $i = 1, \dots, N$.

For any $\mathbf{z} \in [0, 1]^2$ and $\varepsilon > 0$ such that

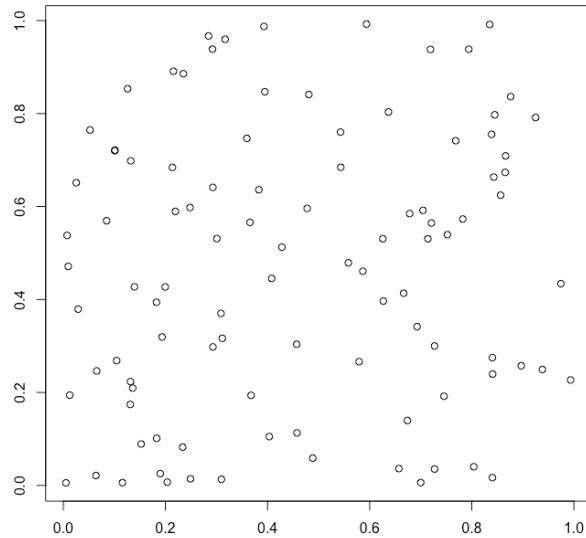
$$I_2(\mathbf{z}; \varepsilon) = \{\mathbf{y} \in \mathbb{R}^2 : \|\mathbf{z} - \mathbf{y}\|_\infty < \varepsilon/2\} \subseteq [0, 1]^2,$$

the expected number of observations \mathbf{x}_i in $I_2(\mathbf{z}; \varepsilon)$ is

$$|I_1(\mathbf{z}; \varepsilon) \cap \{\mathbf{x}_i\}_{i=1}^N| \approx \varepsilon^2 \cdot N.$$

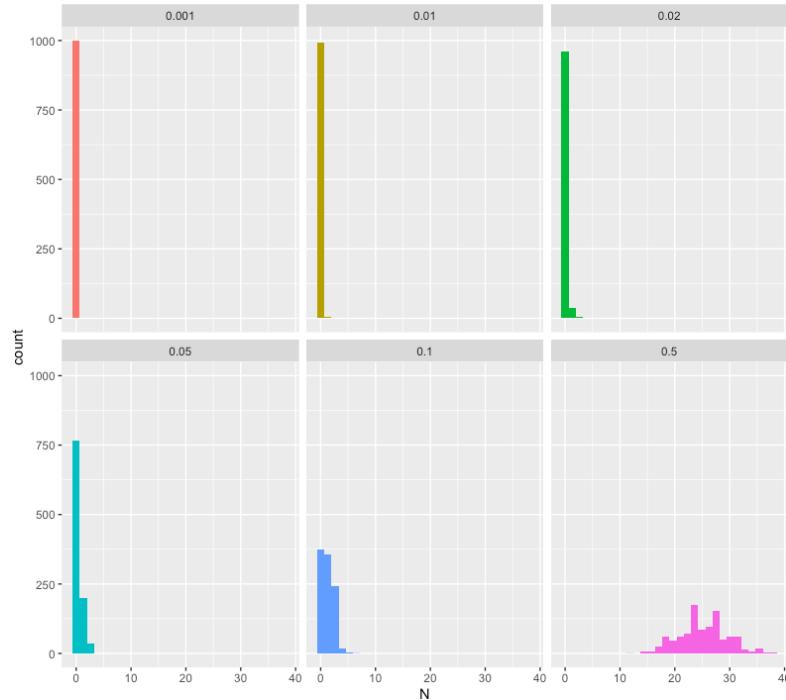
I.e., an ε_∞ -ball subset of $[0, 1]^2$ contains $\approx \varepsilon^2$ of the observations in $\{\mathbf{x}_i\}_{i=1}^N \subseteq \mathbb{R}^2$, on average.

We can also verify this statement through a data experiment.



In this instance ($N = 100$), the numbers of observations in $I_2((1/2, 1/2), \varepsilon)$:

ε	0.001	0.01	0.02	0.05	0.1	0.5
$I_2 \cap \{\mathbf{x}_i\}$	0	0	0	0	0	25
$\varepsilon^2 N$	0.0001	0.01	0.04	0.25	1	25



Repeating the experiment
1000 times yields:

ε	$I_2 \cap \{\mathbf{x}_i\}$	$\varepsilon^2 N$
0.001	0	0.0001
0.01	0.007	0.01
0.02	0.045	0.04
0.05	0.272	0.25
0.1	1.02	1
0.5	25.10	25

In general, an ε_∞ -ball subset of $[0, 1]^p \subseteq \mathbb{R}^p$ contains approximately ε^p of the observations in $\{\mathbf{x}_i\}_{i=1}^N \subseteq \mathbb{R}^p$, **on average**, if $\mathbf{x}_i \sim U^p(0, 1)$.

To capture $r\%$ of uniformly i.i.d. observations in a **unit p -hypercube**, we need, **on average**, a p -hypercube with edge

$$\varepsilon_p(r) = r^{1/p}.$$

For instance, to capture $r = 1/3$ of the observations in a unit p -hypercube in \mathbb{R} , \mathbb{R}^2 , and \mathbb{R}^{10} , we need, on average a p -hypercube with edge $\varepsilon_1(1/3) \approx 0.33$, $\varepsilon_2(1/3) \approx 0.58$, and $\varepsilon_{10}(1/3) \approx 0.90$, respectively.

As $p \nearrow$, the nearest observations to a given point $\mathbf{x}_j \in \mathbb{R}^p$ are **quite distant** from \mathbf{x}_j , in the Euclidean sense, **on average – locality is lost!** (That's not necessarily the case if observations are not uniformly i.i.d.).

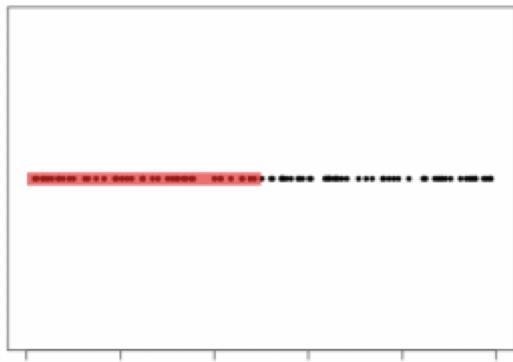
⚠ This is a problem for models and algorithms that rely on the (Euclidean) nearness of observations (k nearest neighbours, k –means clustering, etc.).

The CoD manifests itself in various ways. In datasets with a large number of features:

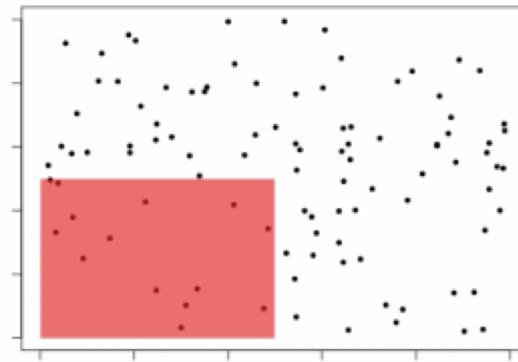
- most observations are **nearer the edge of the sample than they are to other observations**, and
- realistic training sets are necessarily **sparse**.

Imposing restrictions on models can help mitigate the effects of the CoD, but if the assumptions are not warranted the end result may be **even worse**.

42% of data is captured



14% of data is captured



7% of data is captured

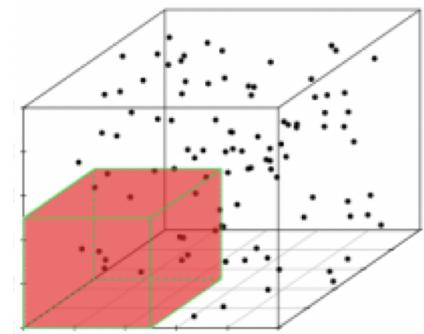


Illustration of the CoD; $N = 100$ observations are uniformly distributed on the unit hypercube $[0, 1]^p$, $p = 1, 2, 3$.

The red regions represent the smaller hypercubes $[0, 0.5]^p$, $p = 1, 2, 3$.

The proportion of captured data points is seen to **decrease** when $p \nearrow$.

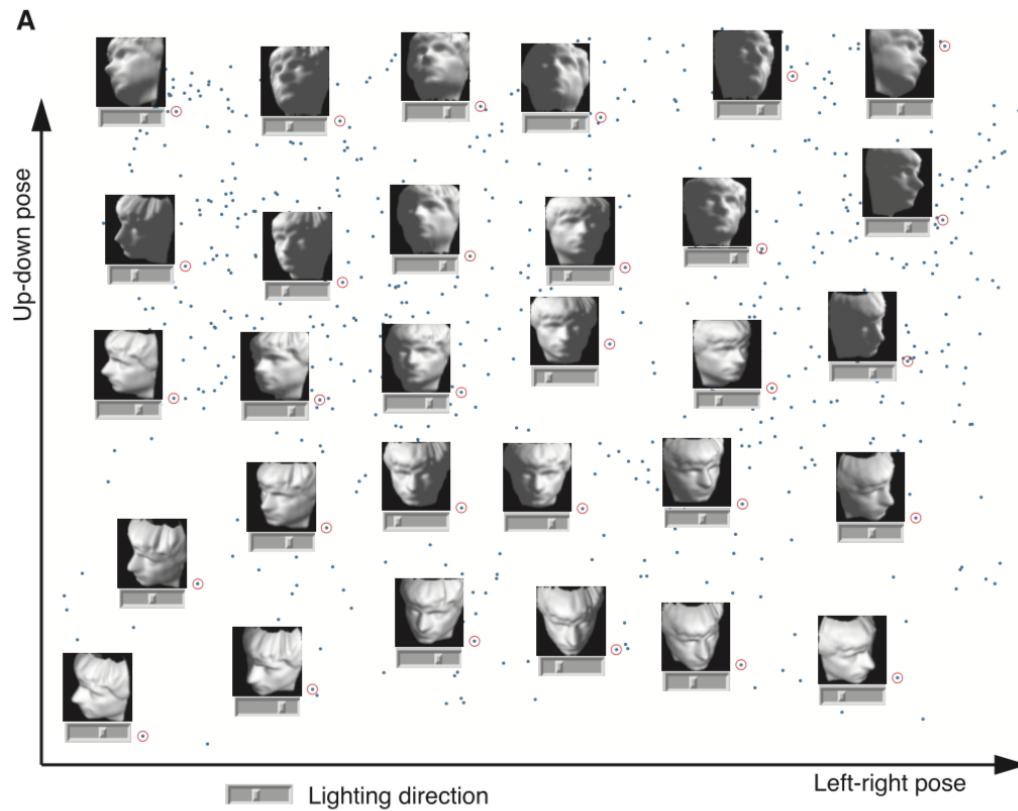
5.1.2 – The Manifold Hypothesis

Manifold learning: mapping **high-d** data to a **lower-d** manifold, such as

$$\mathbb{R}^3 \rightarrow T^2 \hookrightarrow 2D \text{ object.}$$

The problem can also be re-cast as finding a set of **degrees of freedom** (d.f.) which can reproduce most of the **variability** in a dataset.

Example: multiple photographs of a $3D$ object taken from different positions (but at the same distance) can be represented by 2 d.f.: a **horizontal** angle and a **vertical** angle.



Plots showing **degrees-of-freedom** manifolds for images of faces – 3D object [Tenenbaum, Silva, Langford].

Dimensionality reduction and manifold learning are often used:

- to **reduce the overall dimensionality** of the data while trying to **preserve its variance**;
- to **display** high-dimensional datasets, or
- to **reduce the processing time** of supervised learning algorithms by lowering the dimensionality of the processed data.

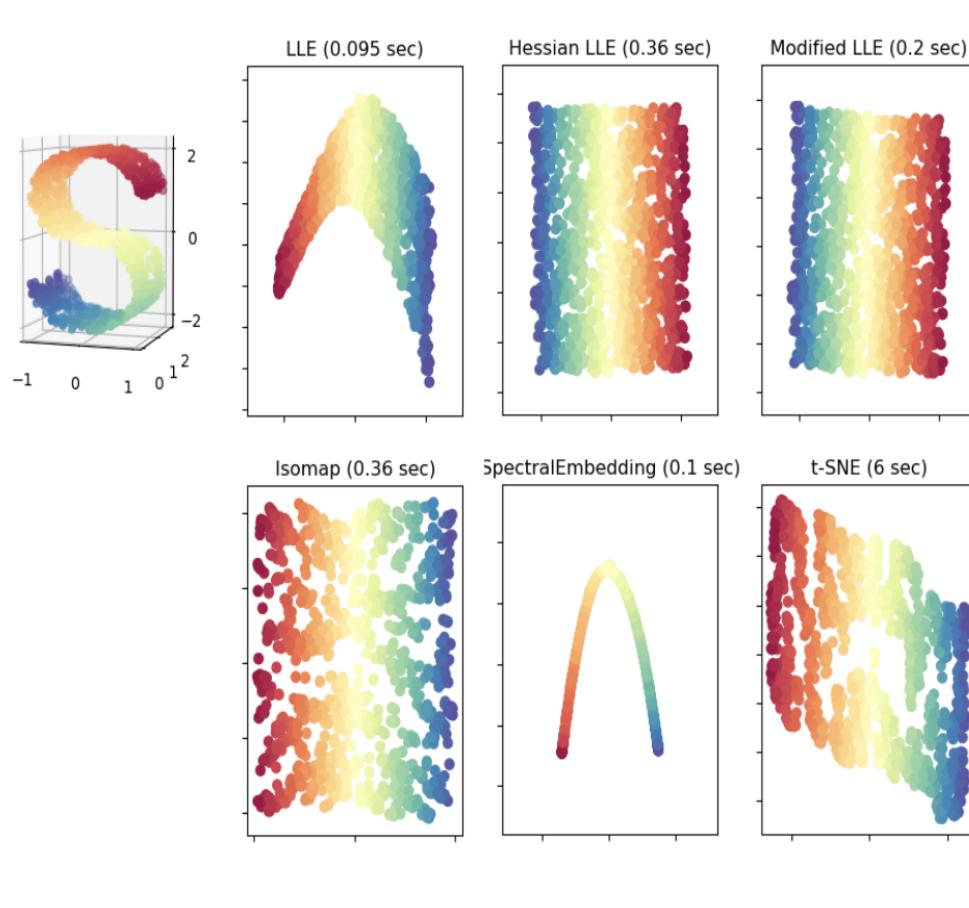
Example: **PCA** (see Chapter 5.3) provides a sequence of **best linear approximations** to high-dimensional observations. The process has interesting theoretical properties for computation and applications, but data is **not always well-approximated by a fully linear process**.

5.1.3 – Nonlinear Dimensionality Reduction Methods

We will briefly discuss some dimensionality reduction methods:

- **LLE**
- **Laplacian eigenmap**
- **isomap**
- ***semidefinite embedding**; ****t*–SNE** (see DUDADS, Chapter 22)
- ***UMAP** (ditto)

Examples: $\mathbb{R}^3 \rightarrow \mathbb{R}^2$ and $\mathbb{R}^3 \rightarrow \mathbb{R}^1$ mappings on the next page.



Comparison of manifold learning methods on an artificial dataset [Cayton].

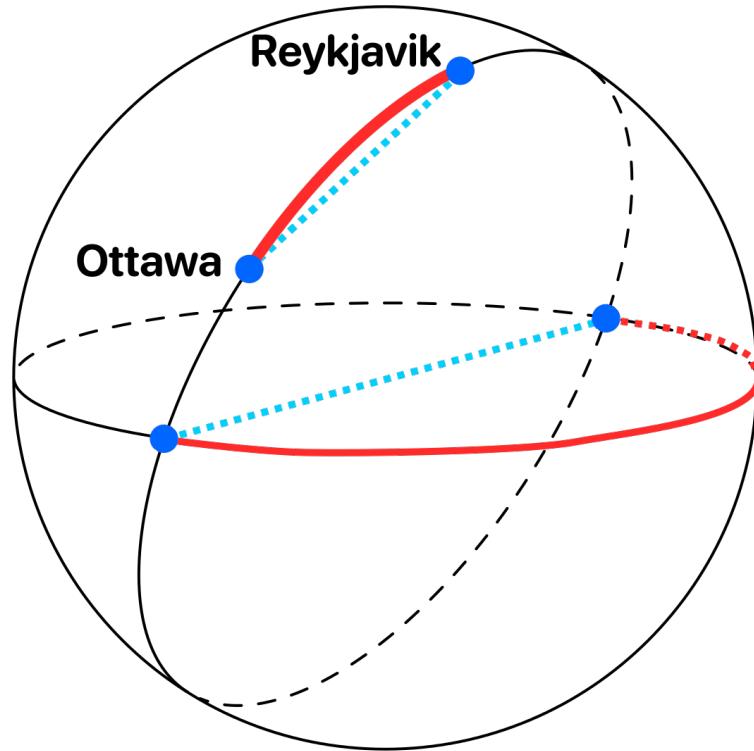
Kernel Principal Component Analysis

For high-d datasets, **linear PCA** may only weakly capture/explain the variance across the entire dataset.

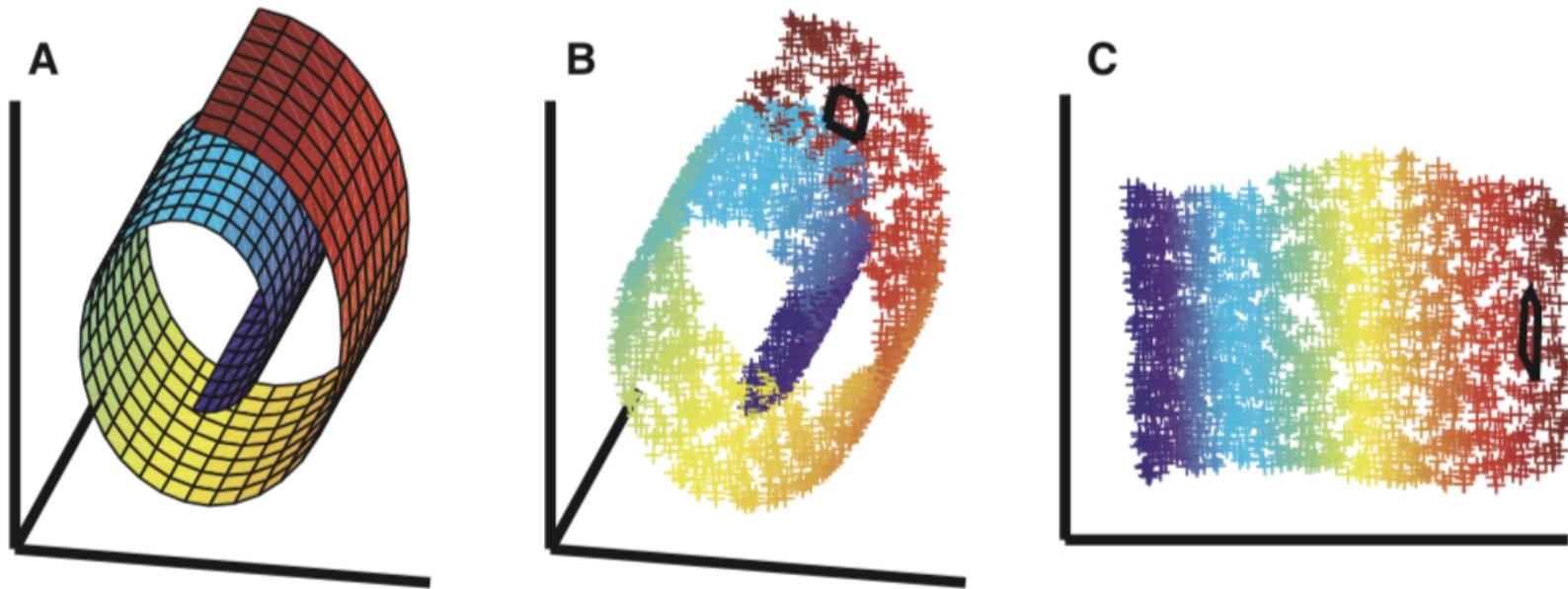
This is partly because PCA relies on Euclidean distance as opposed to **geodesic distance**: the distance between two points *along* the manifold **if it was first unrolled** (see p.23).

Example: the Euclidean distance (“**as the mole burrows**”) between Ottawa and Reykjavik is the length of the shortest tunnel joining the two cities.

The geodesic distance (“**as the crow flies**”) is the arclength of the **great circle** through the two locations.



Geodesic paths in red, Euclidean paths in light blue.



Unfolding of a high-dimensional manifold [Tenenbaum, Silva, Langford]

High-d manifolds can be **unfolded/unrolled** with the use of **transformations** $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^m$, with $m \leq N$, the number of observations.

If Φ is such that

$$\sum_{i=1}^N \Phi(\mathbf{x}_i) = \mathbf{0}$$

(i.e., the transformed data is also centered in \mathbb{R}^m), the **kernel PCA objective** in \mathbb{R}^n can be re-written as a linear PCA objective in \mathbb{R}^m :

$$\min_{\mathbf{V}_q} \left\{ \sum_{i=1}^N \left\| \Phi(\mathbf{x}_i) - \mathbf{V}_q \mathbf{V}_q^\top \Phi(\mathbf{x}_i) \right\|^2 \right\} = \min_{\mathbf{V}_q} \left\{ \sum_{i=1}^N \left\| I - \mathbf{V}_q \mathbf{V}_q^\top \right\|^2 \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_i) \right\},$$

over the set of $m \times q$ matrices \mathbf{V}_q with **orthonormal** columns, where q is the **desired dimension of the manifold**.

This is the **error reconstruction** approach and it is equivalent to the **covariance** approach.

In practice, it is difficult to determine Φ **explicitly**.

The problem can be resolved by working with **positive-definite kernel functions** $K : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}_+$ which satisfy $K(\mathbf{x}, \mathbf{y}) = K(\mathbf{y}, \mathbf{x})$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and

$$\sum_{i=1}^k \sum_{j=1}^k c_i c_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

for any integer k , coefficients $c_1, \dots, c_k \in \mathbb{R}$ and vectors $\mathbf{x}_1, \dots, \mathbf{x}_k \in \mathbb{R}^n$, with **equality** if and only if $c_1, \dots, c_k = 0$.

In general, $K(\mathbf{x}, \mathbf{w}) \rightsquigarrow \Phi(\mathbf{x})^\top \Phi(\mathbf{w})$, arising in the kernel PCA objective.

Popular data analysis kernels (see SVM) include the:

- **linear kernel** $K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}$;
- **polynomial kernel** $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^\top \mathbf{y} + r)^k$, $n \in \mathbb{N}$, $r \geq 0$, and
- **Gaussian kernel** $K(\mathbf{x}, \mathbf{y}) = \exp\left\{\frac{-\|\mathbf{x}-\mathbf{y}\|}{2\sigma^2}\right\}$, $\sigma > 0$.

If $\mathbf{x} \in \mathbb{R}^n$ is any (new) point, its **projection** onto the (non-linear) principal component $\mathbf{w}_j \in \mathbb{R}^m$ is

$$\Phi(\mathbf{x})^\top \mathbf{w}_j = \sum_{i=1}^N w_{j,i} K(\mathbf{x}, \mathbf{x}_i).$$

Most dimension reduction algorithms can be re-expressed as a **kernel PCA**.

Kernel PCA Summary

1. Pick a **kernel** K , i.e. where $K_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$, $1 \leq i, j \leq N$, and compute the $N \times N$ normalized and centered data kernel

$$\mathbf{K} = K - \frac{2}{N} \mathbf{1}_{N \times N} K + \frac{1}{N^2} \mathbf{1}_{N \times N} K \mathbf{1}_{N \times N};$$

2. find the **ordered eigenvalue decomposition** $\{\mathbf{W}, \Lambda\}$ of \mathbf{K} , and select an appropriate $d \leq m$ (using Λ , see 5.3);
3. for $1 \leq j \leq d \leq m$, the j -coordinate of $\mathbf{x} \in \mathbb{R}^n$ in $\mathcal{M} = \mathbb{R}^d \hookrightarrow \mathbb{R}^m$ is

$$y_j = \Phi(\mathbf{x})^\top \mathbf{w}_j = \sum_{i=1}^N w_{j,i} K(\mathbf{x}, \mathbf{x}_i).$$

Locally Linear Embedding

Locally linear embedding (LLE) computes a low-d, **neighbourhood-preserving** embedding of high-d data.

Main assumption: for any subset $\{\mathbf{x}_i\} \subseteq \mathbb{R}^n$ lying on well-behaved manifold \mathcal{M} , with $\dim(\mathcal{M}) = d$, each data point and its neighbours lie on a **locally linear patch** of \mathcal{M} .

Using **translations**, **rotations**, and **rescaling**, the (high-d) coordinates of each locally linear neighbourhood is mapped to a set of **global coordinates** of \mathcal{M} preserving the **neighbouring relationships** between points.

LLE Procedure:

1. identify the **punctured neighbourhood** $N_i = \{i_1, \dots, i_k\}$ of each data point \mathbf{x}_i via **k nearest neighbours**;
2. find the **weights** $z_{i,j}$ that provide the best **linear reconstruction** of each $\mathbf{x}_i \in \mathbb{R}^n$ from their respective punctured neighbourhoods, i.e., solve

$$\min_{\mathbf{Z}} \left\{ \sum_{i=1}^N \left\| \mathbf{x}_i - \sum_{j \in N_i} z_{i,j} \mathbf{x}_{N_i(j)} \right\|^2 \right\},$$

where $\mathbf{Z} = (z_{i,j})$ is an $N \times N$ matrix ($z_{i,j} = 0$ if $j \notin N_i$), and

3. find the **low-dimensional embedding** (or code) vectors $\mathbf{y}_i \in \mathcal{M} (\subseteq \mathbb{R}^d)$ and **neighbours** $\mathbf{y}_{N_i(j)} \in \mathcal{M}$ for each i which are best reconstructed by the weights determined in the previous step, i.e., solve

$$\min_{\mathbf{Y}} \left\{ \sum_{i=1}^N \left\| \mathbf{y}_i - \sum_{j \in N_i} w_{i,j} \mathbf{y}_{N_i(j)} \right\|^2 \right\} = \min_{\mathbf{Y}} \left\{ \text{Tr} \left(\mathbf{Y}^\top \mathbf{Y} \mathbf{L} \right) \right\},$$

where $\mathbf{L} = (\mathbf{I} - \mathbf{Z})^\top (\mathbf{I} - \mathbf{Z})$ and \mathbf{Y} is an $N \times d$ matrix.

We can add restrictions to ensure that the global coordinates of the sampled points are **centered at the origin**, with **unit variance in all directions**, so that \mathbf{L} has a 0 eigenvalue. The j^{th} column of \mathbf{Y} is then simply the **eigenvector** associated with the j^{th} smallest **non-zero eigenvalue** of \mathbf{L} .

Laplacian Eigenmaps

Laplacian eigenmaps are similar to LLE, except that the first step consists of constructing a **weighted graph** \mathcal{G} with N **nodes** (one per observation) and a set of **edges** connecting the neighbouring points.

The **edges** of \mathcal{G} can be obtained by finding the k **nearest neighbours** of each **node**, or by selecting all points **within some fixed radius** ε .

In practice, the edges' weights W are determined either by:

- using the **inverse exponential** with respect to the Euclidean distance $w_{i,j} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{s}\right)$, for all i, j , for some parameter $s > 0$, or
- by setting $w_{i,j} = 1$, for all i, j .

The embedding map is then provided by the following objective

$$\min_{\mathbf{Y}} \left\{ \sum_{i=1}^N \sum_{j=1}^N w_{i,j} (\mathbf{y}_i - \mathbf{y}_j)^2 \right\} = \min_{\mathbf{Y}} \left\{ \text{Tr}(\mathbf{Y} L \mathbf{Y}^\top) \right\},$$

subject to appropriate constraints, with the **Laplacian** L given by $L = D - W$, where D is the (diagonal) **degree matrix** of \mathcal{G} (the sum of weights emanating from each node), and W its **weight matrix**.

The Laplacian eigenmap construction is identical to the LLE construction, save for **the definition of L** .

A similar construction is used in **spectral clustering**.

Isomap

Isomap follows the same steps as LLE except that it uses **geodesic distance** instead of Euclidean distance when looking for each point's **neighbours**.

Neighbourhoods can be selected with k NN or with a fixed ε .

These neighbourhood relations are represented by a **graph** \mathcal{G} in which each **observation** is connected to its neighbours *via* **edges** with weight $d_x(i, j)$ between neighbours.

The **geodesic distances** $d_{\mathcal{M}}(i, j)$ between all pairs of points on the manifold \mathcal{M} are then estimated in the second step.

Unified Framework

The preceding algorithms can all be rewritten in the kernel PCA framework:

- **LLE** – if λ_{\max} is the **largest** eigenvalue of $\mathbf{L} = (\mathbf{I} - \mathbf{Z})^\top(\mathbf{I} - \mathbf{Z})$, then $K_{\text{LLE}} = \lambda_{\max}\mathbf{I} - \mathbf{L}$;
- **LE** – same, but with $\mathbf{L} = \mathbf{D} - \mathbf{W}$, then the corresponding K_{LE} is related to **commute times of diffusion on the underlying graph**, and
- **Isomap** – with element-wise **squared geodesic distance matrix** \mathcal{D}^2 ,

$$K_{\text{Isomap}} = (-2n^2)^{-1} (n\mathbf{I} - \mathbf{1}_{n \times n}) \mathcal{D}^2 (n\mathbf{I} - \mathbf{1}_{n \times n}).$$

Note that this kernel is not always p.s.d.

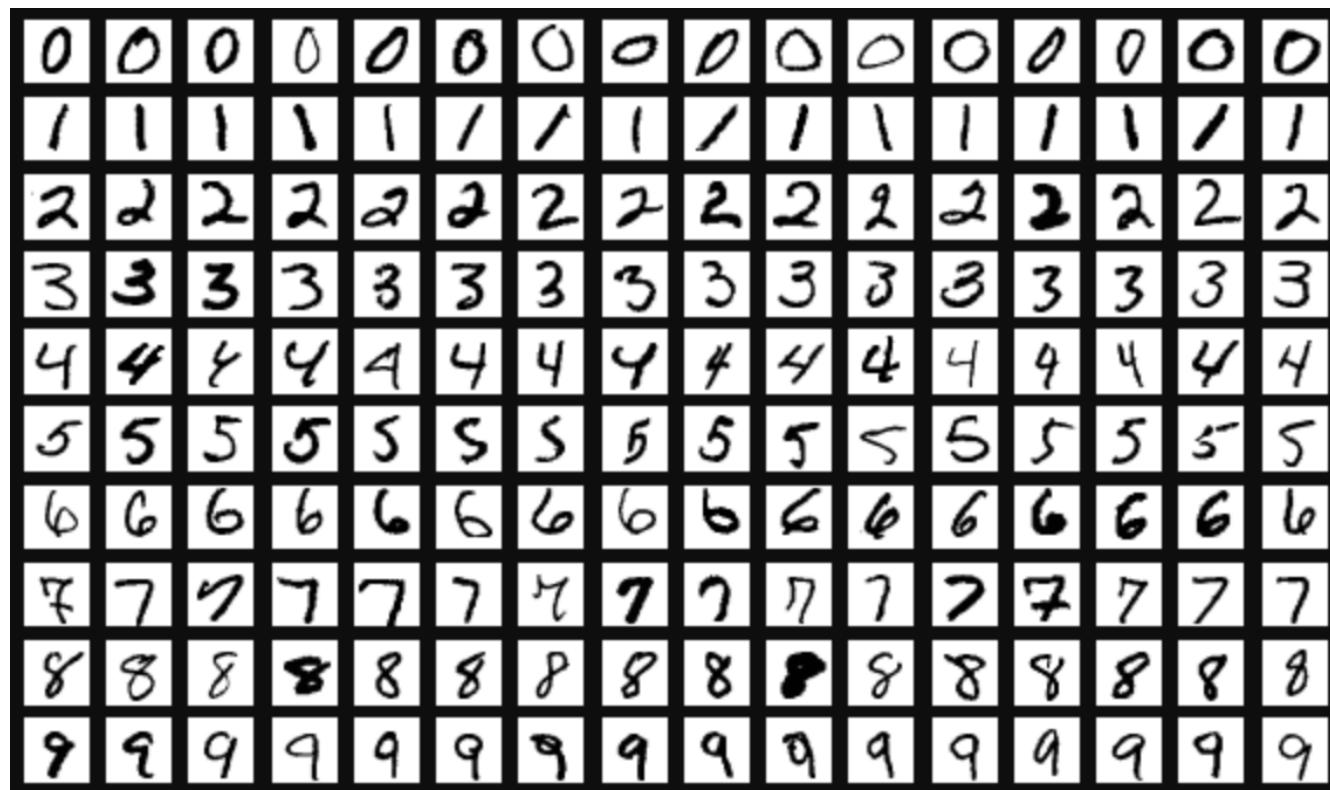
MNIST Example

The methods of this section are used to learn manifolds for the MNIST dataset, a database of handwritten digits.

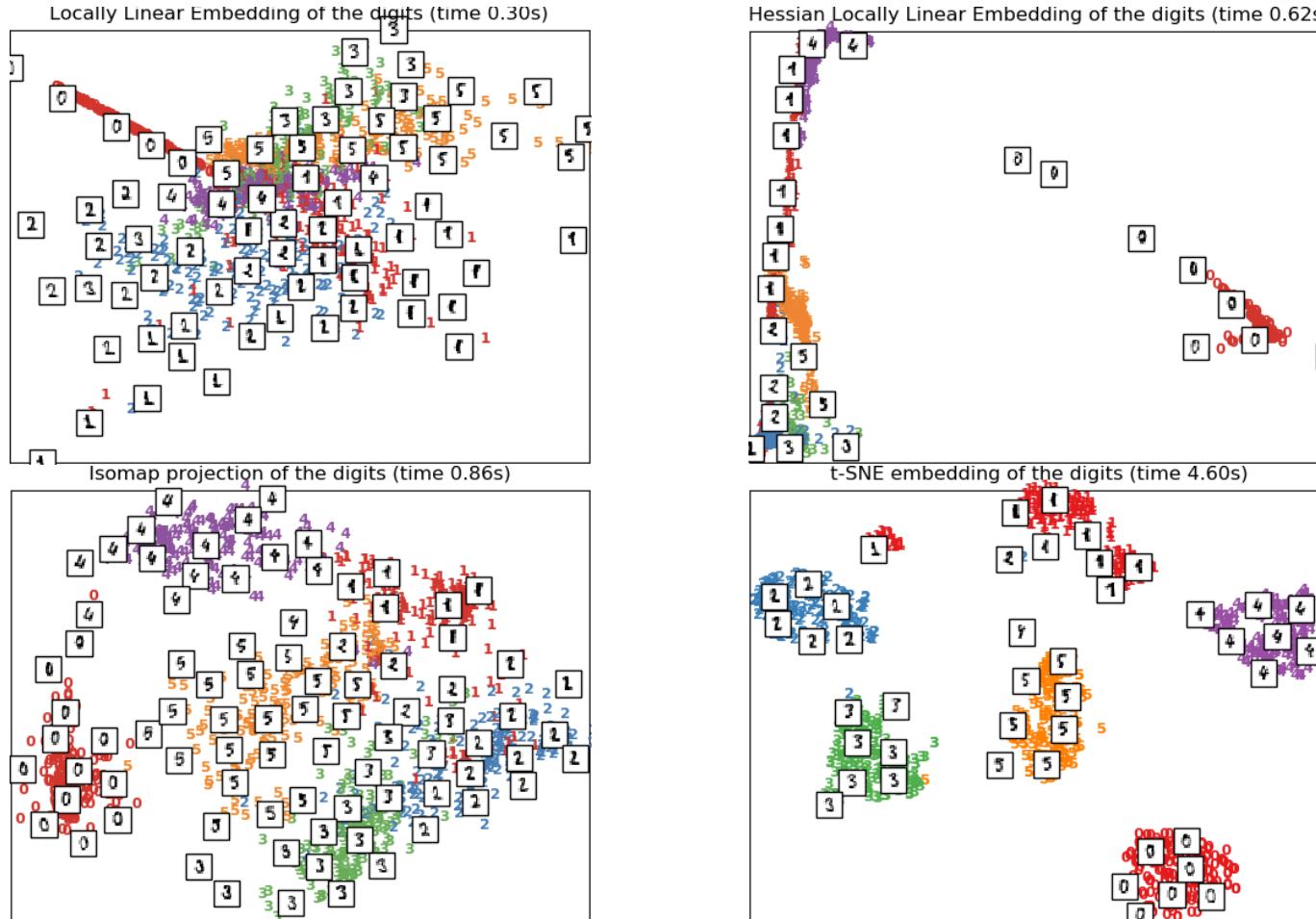
The results for 4 of those are shown on p.37.

The analysis of optimal manifold learning methods is **subjective**, as it depends on the outcome AND on the computing cost and run time.

Naïvely, one would expect to see the coordinates in the reduced manifold congregate in **10 (or more) distinct groups**; in that regard, t –SNE seems to perform admirably on MNIST.



Sample from the MNIST dataset (LeCun, Cortes, Burges).



Manifold learning on digits 0 – 5: LLE, Hessian LLE, Isomap, t -SNE.

5.2 – Feature Selection

Dimension reduction methods can be used to learn **low-dimensional manifolds** for high-dimensional data.

If the resulting loss in information content can be kept small (**not always possible**), this can help to mitigate the impact of the CoD.

Non-technical challenge: the manifold coordinates are not usually **interpretable** in the context of the original dataset.

Example: consider a dataset with 4 features ($X_1 = \text{Age}$, $X_2 = \text{Height}$, $X_3 = \text{Weight}$, and $X_4 = \text{Gender}$ (0, 1), say).

It is straightforward to **justify** a data-driven decision based on the rule

$$X_1 = \text{Age} > 25,$$

for example, but not as easy for a rule such as

$$Y_2 = 3(\text{Age} - \overline{\text{Age}}) - (\text{Height} - \overline{\text{Height}}) + 4 \cdot \text{Gender} > 7$$

(even if there is nothing wrong with the rule from a **technical perspective**).

Datasets often contain **irrelevant** and/or **redundant** features; **identifying** and **removing** these variables is a common data processing task.

Motivations:

- modeling tools do not handle redundant variables well, due to **variance inflation** or similar issues,
- attempts to **overcome the CoD** or to avoid situations where the number of variables p is **larger** than the number of observations n .

The main goal of **feature selection** is to remove (not **transform** nor **project**) attributes that add noise and reduce model performance: we seek to retain a subset of the most **relevant features**, in order to create simpler models, decrease training time, and reduce **overfitting**.

This requires a **target value** to predict, against which we can evaluate features for relevance.

Feature selection methods fall in one of three families:

- **filter methods** focus on the relevance of each feature individually, applying a **ranking metric** to each of them; the variables that do not meet a **preset benchmark** on the ranking or the ranking metric value are removed from the model building process; different metrics/thresholds might retain **different** relevant features;
- **wrapper methods** focus on the usefulness of each feature to the task (classification/regression/etc.), but do not consider features individually; they evaluate and compare the performance of different **combinations of features** in order to select the best-performing subset of features;
- **embedded methods** are a combination of both, using **implicit metrics** to evaluate the performance of various subsets.

Feature selection methods can also be categorized:

- **unsupervised methods**, which determine features' importance only through their values (with potential feature interactions), and
- **supervised methods**, which evaluate features' importance in relationship with the **target feature**.

Wrapper methods are typically **supervised**.

Unsupervised filter methods search for **noisy features** and include the removal of **constant** variables, of **ID-like** variables, or features with **low variability**.

5.2.1 – Filter Methods

Filter methods evaluate features without resorting to **classification** or **regression** algorithms; these methods can either be

- **univariate**, where each feature is ranked independently, or
- **multivariate**, where features are considered jointly.

Filter criteria are chosen based on which metrics suit the data/problem.

The selected criterion is used to assign a **score/rank** to the features; those for which it lies beyond a pre-selected threshold τ are deemed **relevant** and are retained.

Advantages:

- computationally **efficient**;
- tend to be **robust against overfitting**

Common methods:

- **Pearson correlation coefficient**;
- **information gain** (or mutual information), and
- **relief**.

Let Y be the **target variable**, and X_1, \dots, X_p be the **predictors**.

Pearson Correlation Coefficient

Pearson's correlation coefficient (PCC) quantifies the linear relationship between two **numerical** variables.

The PCC between a predictor X_i and the target Y is

$$\rho_i = \frac{\text{Cov}(X_i, Y)}{\sigma_{X_i} \sigma_Y}.$$

Features for which

- $|\rho_i|$ is **large** (near 1) are **linearly correlated** with Y ;
- those for which $|\rho_i| \approx 0$ are **not linearly correlated** with Y .

We might decide to **retain only** the features X_i for which $|\rho_i| > \tau$, for a given $0 < \tau < 1$.

We might also decide to **rank** the features according to

$$|\rho_{i_1}| \geq |\rho_{i_2}| \geq \cdots \geq |\rho_{i_p}|$$

and **retain only** the first d features, for a given d .

The PCC ρ_i is only defined if both X_i and Y are **numerical**; there are alternatives for **categorical** and for **mixed** X_i and Y .

⚠ The correlation between a predictor X_i and the target Y could be **strong**, but **not linear**; as the PCC cannot capture such relationships, it is likely that the Pearson filter would **fail to retain** the predictor.

Mutual Information

Information gain (IG) is an **entropy-based** method that measures the dependence between features by quantifying the amount of **mutual information** between them:

$$\text{IG}(X_i; Y) = H(X_i) - H(X_i | Y),$$

where $H(X_i)$ is the **marginal entropy** of X_i and $H(X_i | Y)$ is the **conditional entropy** of X_i given Y , and

$$H(X_i) = E_{X_i}[-\log p(X_i)], \quad H(X_i | Y) = E_{(X_i, Y)}[-\log p(X_i | Y)]$$

where $p(X_i)$ and $p(X_i | Y)$ are the PDFs of X_i and $X_i | Y$, respectively.

Example: let Y represent the **salary** of an individual (continuous), X_1 their **hair colour** (categorical), X_2 their **age** (continuous), X_3 their **height** (continuous), and X_4 their self-reported **gender** (categorical), in a dataset from the 2000s.

Summary statistics for a sample of 2144 individuals are shown on p. 49.

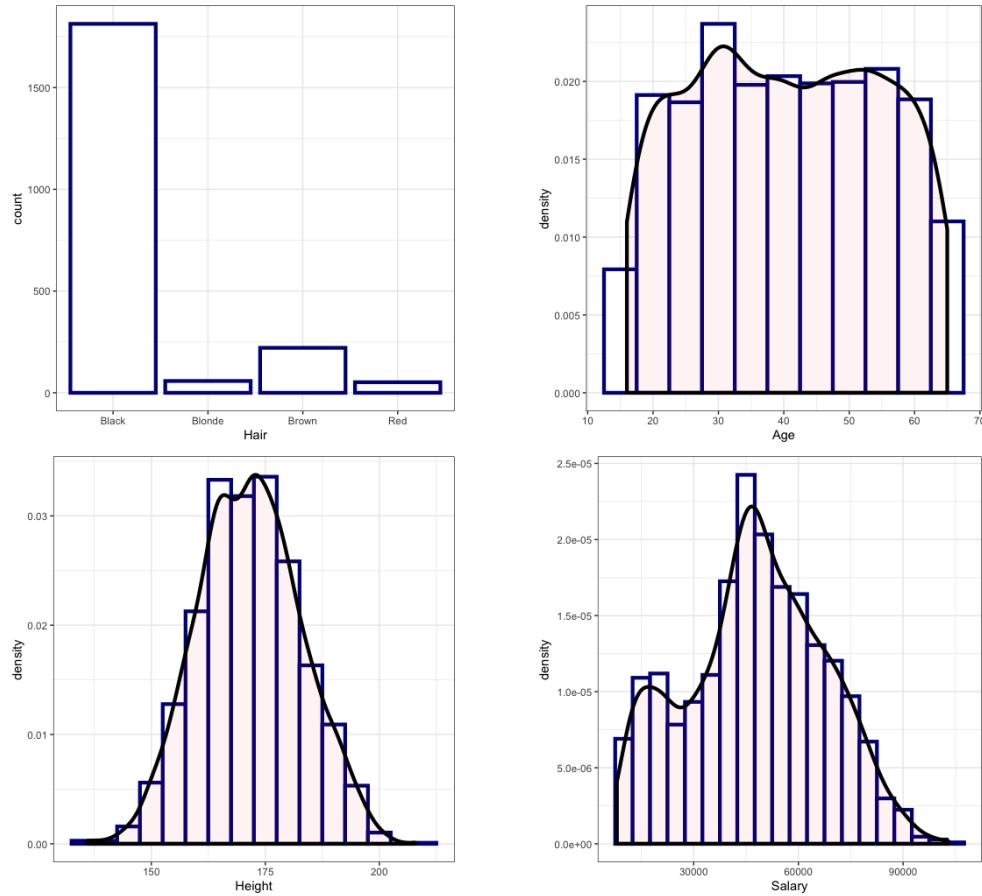
In a general population, the distribution of **salaries**, say, is likely to be fairly **haphazard**. It might be hard to explain why, specifically, it has the shape that it does (see p. 50).

It could be perhaps be explained by knowing the relationship between the **salary** and the other variables.

It is this idea that forms the basis of **mutual information feature selection**.

Hair		Gender		Summary	Age	Height	Salary	Salary Deciles	Black	Blonde	Brown	Red	Total										
Black	1813	Female	1083	MIN	16	136	8000	1	193	7	25	8	233										
Blonde	58	Male	1061	Q1	29	164	34000	2	180	9	25	8	222										
Brown	221		2144	MED	40	172	48000	3	180	4	30	6	220										
Red	52			Q3	53	179	61250	4	187	4	14	3	208										
	2144			MAX	65	208	103000	5	182	8	26	8	224										
				MEAN	40.4	171.6	47674.4	6	185	5	18	1	209										
				STDEV	14.2	11.2	19710.3	7	183	5	18	5	211										
				SKEW	0.01	0.06	-0.04	8	184	6	19	8	217										
								9	168	8	22	5	203										
								10	171	2	24		197										
								Total	1813	58	221	52	2144										
Salary Deciles		Age Deciles										Height Deciles											
Salary Deciles	1	2	3	4	5	6	7	8	9	10	Total	Salary Deciles	1	2	3	4	5	6	7	8	9	10	Total
1	203	30									233	1	28	22	21	24	25	17	27	21	21	27	233
2	40	133	39	1	1	5	1			2	222	2	19	26	22	18	30	19	23	22	22	21	222
3		26	73	21	24	15	11	4	14	32	220	3	34	38	34	26	39	12	15	9	6	7	220
4		6	26	50	27	17	13	14	24	31	208	4	33	32	31	28	18	16	15	20	8	7	208
5		5	31	44	31	18	16	25	26	28	224	5	34	29	19	25	32	18	21	20	14	12	224
6		3	21	39	34	27	16	21	33	15	209	6	33	25	26	14	33	13	22	15	13	15	209
7			7	40	33	36	23	31	19	22	211	7	20	24	24	17	24	16	16	19	30	21	211
8			2	30	38	36	28	33	20	30	217	8	12	15	15	22	32	16	18	36	20	31	217
9				2	19	54	30	39	36	23	203	9	6	13	12	17	27	9	24	28	34	33	203
10					2	12	41	32	57	47	197	10	1	4	10	15	18	13	30	35	31	40	197
Total	243	203	199	229	219	249	170	224	219	189	2144	Total	220	228	214	206	278	149	211	225	199	214	2144
												Total	1083	1061	2144								

Summary statistics for the salary dataset; two-way tables use decile data.



Univariate distributions (**hair colour, age, height, salary**).

If the theoretical distributions are known, the **entropy integrals** can be computed/approximated directly.

Gender and **hair colour** can be modeled using **multinomial** distributions, but there is more uncertainty related to the numerical variables.

$$H(X_1) = - \sum_{\text{colour}} p(\text{colour}) \log p(\text{colour})$$

$$H(X_2) = - \int p(\text{age}) \log p(\text{age}) d\text{age}$$

$$H(X_3) = - \int p(\text{height}) \log p(\text{height}) d\text{height}$$

$$H(X_4) = - \sum_{\text{gender}} p(\text{gender}) \log p(\text{gender})$$

$$H(X_1 \mid Y) = - \int p(Y) \left\{ \sum_{\text{colour}} p(\text{colour} \mid Y) \log p(\text{colour} \mid Y) \right\} dY$$

$$H(X_2 \mid Y) = - \iint p(Y) p(\text{age} \mid Y) \log p(\text{age} \mid Y) d\text{age} dY$$

$$H(X_3 \mid Y) = - \iint p(Y) p(\text{ht} \mid Y) \log p(\text{ht} \mid Y) d\text{ht} dY$$

$$H(X_4 \mid Y) = - \int p(Y) \left\{ \sum_{\text{gender}} p(\text{gender} \mid Y) \log p(\text{gender} \mid Y) \right\} dY$$

Potential approach: recode the continuous variables as **decile variables** taking values $\{1, \dots, 10\}$ according to which **decile** of the original variable the observation falls. The integrals can then be replaced by sums:

$$H(X_1) = - \sum_{\text{colour}} p(\text{colour}) \log p(\text{colour})$$

$$H(X_2) \approx - \sum_{k=1}^{10} p(\text{age}_d = k) \log p(\text{age}_d = k)$$

$$H(X_3) \approx - \sum_{k=1}^{10} p(\text{height}_d = k) \log p(\text{height}_d = k)$$

$$H(X_4) = - \sum_{\text{gender}} p(\text{gender}) \log p(\text{gender})$$

$$H(X_1 \mid Y) \approx - \sum_{j=1}^{10} p(Y_d = j) \sum_{c \in \text{colour}} p(c \mid Y_d = j) \log p(c \mid Y_d = j)$$

$$H(X_2 \mid Y) \approx - \sum_{j=1}^{10} p(Y_d = j) \sum_{k=1}^{10} p(a_d = k \mid Y_d = j) \log p(a_d = k \mid Y_d = j)$$

$$H(X_3 \mid Y) \approx - \sum_{j=1}^{10} p(Y_d = j) \sum_{k=1}^{10} p(h_d = k \mid Y_d = j) \log p(h_d = k \mid Y_d = j)$$

$$H(X_4 \mid Y) \approx - \sum_{j=1}^{10} p(Y_d = j) \sum_{g \in \text{gender}} p(g \mid Y_d = j) \log p(g \mid Y_d = j)$$

X	$H(X)$	$H(X Y)$	$IG(X;Y)$	Ratio
Hair	0.24	0.24	0.00	0.00
Age	1.00	0.74	0.26	0.26
Height	1.00	0.96	0.03	0.03
Gender	0.30	0.22	0.08	0.26

Mutual information obtained about each predictor after observing the target response Y (**salary**).

The percentage decrease in entropy after having observed Y is shown in the column “**Ratio**.”

Raw IG numbers would seem to suggest that **gender** only has a small link to **salary**; the **Ratio** numbers suggest that this could be due to the way the **age** and **height** levels have been categorized (**deciles**).

Relief

Relief scores (numerical) features based on the identification of feature value differences between **nearest-neighbour instance pairs**.

If there is a feature value difference in a **neighbouring instance pair**:

- of the **same class** (as given by Y), the relief score **decreases**;
- in **different classes**, the relief score **increases**.

More specifically, let $D = \{(\mathbf{x}_n, y_n)\}_{n=1}^N \subset \mathbb{R}^p \times \{\pm 1\}$ be a dataset where \mathbf{x}_n is the n -th data sample and y_n is its corresponding class label.

For each feature i and observation n , two values are selected:

- the **near hit** $H(x_{n,i})$ is the value of X_i which is **nearest to** $x_{n,i}$ among all instances in the **same** class as \mathbf{x}_n ;
- the **near miss** $M(x_{n,i})$ is the value of X_i which is **nearest to** $x_{n,i}$ among all instances in the **opposite** class as \mathbf{x}_n .

The **relief score** of the i^{th} feature is

$$S_i^d = \frac{1}{N} \sum_{n=1}^N \{d(x_{n,i}, M(x_{n,i})) - d(x_{n,i}, H(x_{n,i}))\},$$

for some pre-selected distance $d : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}_0^+$.

A feature for which **near-hits** tend to be nearer than **near-misses** has

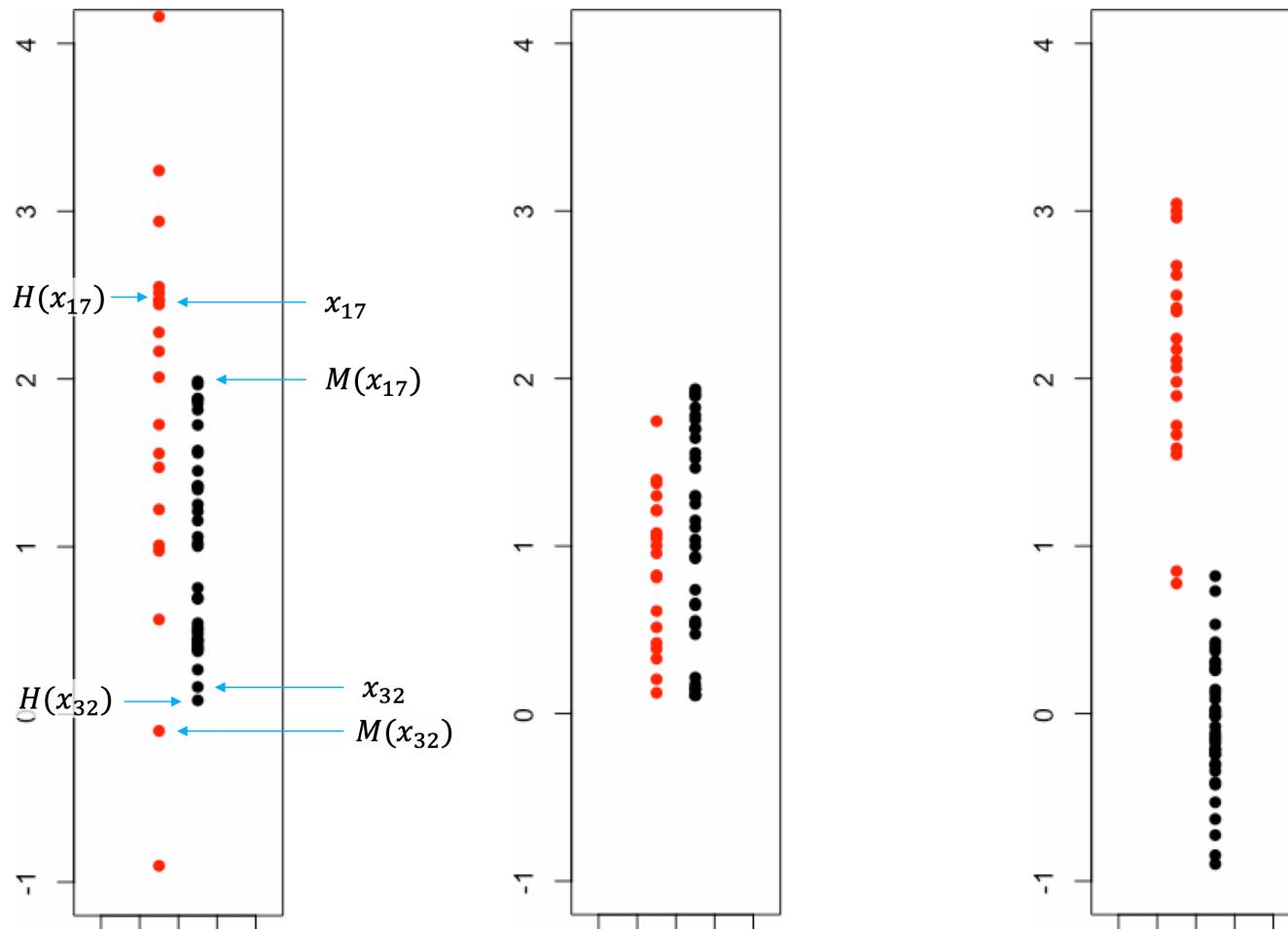
$$d(x_{n,i}, M(x_{n,i})) > d(x_{n,i}, H(x_{n,i})),$$

on average; S_i^d should be **larger** than one for which the opposite holds.

Features are **relevant** when their relief score is greater than some threshold τ .

Relief is **noise-tolerant** and **robust to interactions**, but its effectiveness decreases with **small** dataset size.

Variants exist to accommodate potential feature interactions of multi-class problems and other issues.



Other Filter Methods (Non-Exhaustive)

- Other correlation metrics (**Kendall**, **point-biserial correlation**, etc.)
- Other entropy-based metrics (**gain ratio**, **symmetric uncertainty**, etc.)
- Other relief-type algorithms (**ReliefF**, **Relieved-F**, etc.)
- **ANOVA**
- **Fisher Score**
- **Gini Index**

5.2.2 – Wrapper Methods

Wrapper methods evaluate the quality of **subsets of features** for predicting the target output under a selected predictive algorithm and select the optimal combination (for a given training set and algorithm).

In contrast to filter methods, wrapping methods are **integrated directly** into the classification or clustering process.

Such methods treats feature selection as a **search problem** in which different subsets of features are explored.

This process is **computationally expensive**: the size of the search space increases **exponentially** with the number of predictors.

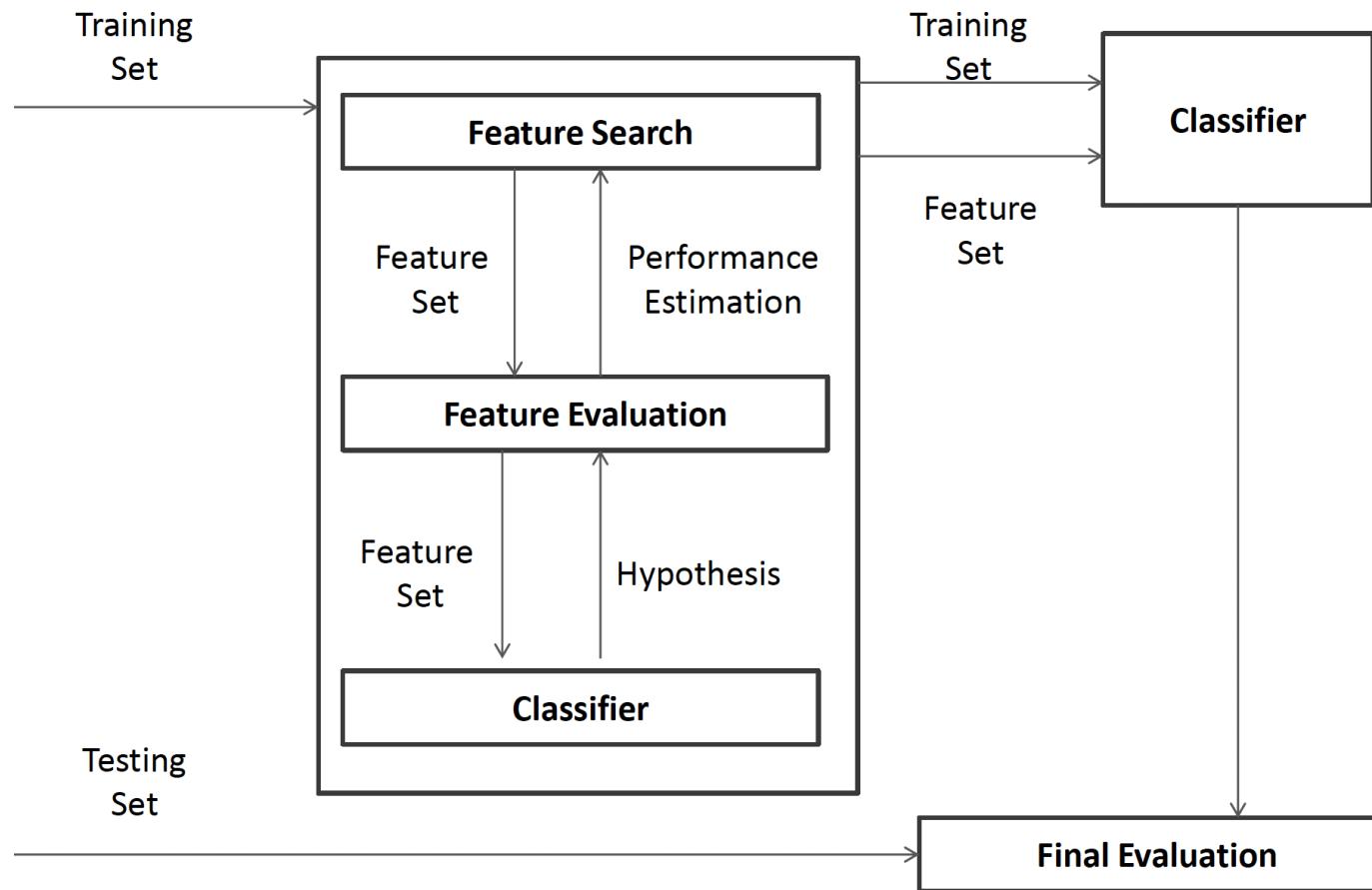
Wrapper methods iterate over the following steps, until an “**optimal**” set of features is identified:

- select a **feature subset**, and
- evaluate its **performance**.

The search ends when the desired quality is reached (R_a^2 , etc.).

Various search methods provide approximate solutions to the **optimal feature subset problem**: hill-climbing, best-first, genetic algorithms, etc.

Wrapper methods are **not as efficient** as filter methods and **not as robust** against overfitting; but are very effective at **improving the model’s performance** due to their attempt to **minimize** the error rate.



Feature selection process for classification wrapper methods [Aggarwal].

5.2.3 – Subset Selection Methods

Stepwise selection is a form of **Occam's Razor**: at each step, a new feature is considered for inclusion or removal from the current features set based on some criterion (*F-test*, *t-test*, etc.).

Greedy search methods have proven to be **robust against overfitting** and among the **least computationally expensive** wrapper methods:

- **Backward elimination** begins with the full set of features and sequentially eliminates the least relevant ones until further removals increase the error rate of the predictive model above some threshold.
- **Forward selection** begins with an empty set of features and progressively adds relevant features until some threshold is met.

In both cases, **model performance** should be tested using **cross-validation** – see Chapter 2 for more information on this approach to performance evaluation.

Stepwise Selection Methods Limitations:

- the tests are **biased**, since they are all based on the same data;
- R_a^2 only takes into account the # of features in the **final fit**, and not the **df** that have been used in the **entire model**;
- if **cross-validation** is used, stepwise selection has to be repeated for **each sub-model**, but that is rarely done, and
- it's a classic example of ***p*-value hacking**.

5.2.4 – Regularization Methods

An interesting hybrid is provided by the **least absolute shrinkage and selection operator (LASSO)** and its variants, which we discussed before.

For a **loss** function $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}(\mathbf{W}))$, where $\hat{\mathbf{y}}(\mathbf{W})$ are the (depending on the parameters \mathbf{W}), and a vector

$$\mathbf{R}(\mathbf{W}) = (R_1(\mathbf{W}), \dots, R_k(\mathbf{W})),$$

\mathbf{W}^* solves the problem

$$\mathbf{W}^* = \arg \min_{\mathbf{W}} \{\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}(\mathbf{W})) + \lambda^\top \mathbf{R}(\mathbf{W})\},$$

which can be solved numerically, assuming nice properties on \mathcal{L} and \mathbf{R} .

5.2.5 – SL and UL Methods

Feature selection methods are usually categorised as **filter**, **wrapper**, or **embedded (regularization)**, but also as **supervised** or **unsupervised** methods.

Such methods are **supervised** if the **labels** are incorporated into the feature reduction process; otherwise they are **unsupervised**.

In **UL** methods, feature selection is carried out based only on the **characteristics of the attributes**, without any reference to **labels** or a **target** variable.

For **clustering problems**, **SL** feature selection methods are **contraindicated**.

What Method Should Be Used?

It depends on a number of factors:

- required **processing time** and **size of dataset**;
- acceptable **level of uncertainty** for task;
- **past successes**, etc.

Suggestion: always try **multiple methods**. If they agree on a **core set of features**, there is some **model-independent** support for the **relevance** of the **core set** to the task at hand.

5.3 – PCA Regression and Partial LS

More traditional dimension reduction techniques include:

- **principal component analysis** (numerical data)
- **independent component analysis** (numerical data)
- **factor analysis** (numerical data)
- **multiple correspondence analysis** (categorical data)

Idea: project multi-dimensional datasets onto **low-dimensional** but **high-information** spaces (**Manifold Hyp**) prior to fitting an appropriate model.

Some information is **necessarily lost** in the process, but the drain can usually be **kept under control** and the gains made by **working with smaller datasets** can offset the **loss of completeness**.

For $m = 1, \dots, M \leq p$, let $z_m = \vec{X}_j^\top \boldsymbol{\phi}$ be linear combinations of the **original predictors** $\{X_1, \dots, X_p\}$.

If we are fitting $y = f(\mathbf{x}) = \mathbb{E}[Y | \vec{X} = \mathbf{x}]$ using OLS, we can also fit

$$y_i = \theta_0 + \mathbf{z}_i^\top \boldsymbol{\theta} + \varepsilon_i,$$

$i = 1, \dots, N$, also using OLS.

If the constants $\phi_{m,j}$ are selected **wisely**, then transforming the variables can yield a model that **outperforms** OLS regression – the predictions might be **better** than those obtained by fitting $y_i = \beta_0 + \mathbf{x}_i^\top \boldsymbol{\beta} + \delta_i$, $i = 1, \dots, N$.

By definition, $\theta_0 = \beta_0$ and

$$\begin{aligned}\mathbf{z}_i^\top \boldsymbol{\theta} &= \sum_{m=1}^M \theta_m z_{i,m} = \sum_{m=1}^M \theta_m \mathbf{x}_i^\top \boldsymbol{\phi} = \sum_{m=1}^M \theta_m \sum_{j=1}^p \phi_{m,j} x_{i,j} \\ &= \sum_{j=1}^p \sum_{m=1}^M \theta_m \phi_{m,j} x_{i,j} = \sum_{j=1}^p \beta_j x_{i,j} = \mathbf{x}_i^\top \boldsymbol{\beta},\end{aligned}$$

where $\beta_j = \sum_{m=1}^M \theta_m \phi_{m,j}$, i.e., reduced regression is a **special case** of the original OLS model, with **constrained coefficients** β_j .

These **constraints** can help with the **bias-variance trade-off** (if $p \gg N$, picking $M \ll p$ can **reduce the variance** of the fitted coefficients).

Challenge: how do we select $\phi_{m,j}$? We will consider two approaches.

5.3.1 – Principal Components Analysis

Principal component analysis (PCA) can be used to find the combinations of variables along which the data points are **most spread out**.

Geometrically, the procedure fits the “best” p -ellipsoid to a **centered** representation of the data.

The ellipsoid axes are the **principal components** of the data.

Small axes are components along which the variance is “**small**”; removing these components may lead to only a “**small**” loss of information.

But there are scenarios where it could be those “**small**” axes that are more interesting – such as in the “**pancake stack**” problem.

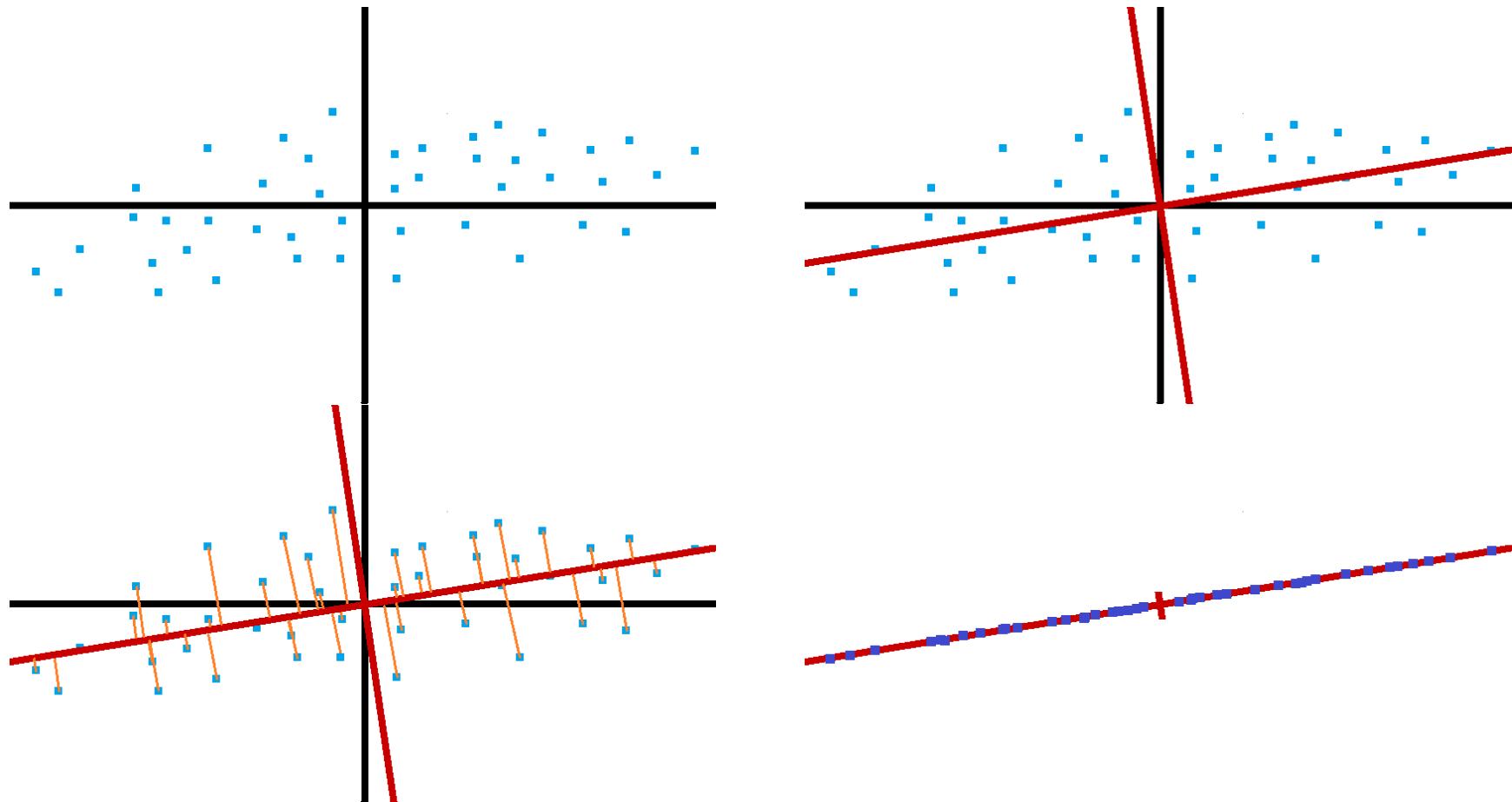


Illustration of PCA on an artificial 2D dataset. The red axes provide the best elliptic fit. The projection on the major axis leads to dimension reduction but a loss of information.

PCA Procedure:

1. **centre** and “**scale**” the data to obtain a matrix \mathbf{X} ;
2. compute the data’s “**covariance matrix**” $\mathbf{K} = \mathbf{X}^\top \mathbf{X}$;
3. compute \mathbf{K} ’s **eigenvalues** Λ (ordered **diagonal matrix**), and its **orthonormal eigenvectors** matrix \mathbf{W} ;
4. each **eigenvector** \mathbf{w} (also known as a **loading**) represents an **axis**, whose variance is given by the associated **eigenvalue** λ .

Note that $\mathbf{K} \geq 0 \implies \Lambda \geq 0$.

The **first principal component** PC_1 is the **eigenvector** \mathbf{w}_1 of \mathbf{K} associated to its **largest eigenvalue** λ_1 , and the **variance** of the data along \mathbf{w}_1 is **proportional** to λ_1 .

The **second principal component** PC_2 is the **eigenvector** \mathbf{w}_2 of \mathbf{K} associated to its **second largest eigenvalue** $\lambda_2 \leq \lambda_1$, and the **variance** of the data along \mathbf{w}_2 is **proportional** to λ_2 , and so on.

Final Result: $r = \text{rank}(\mathbf{X})$ **orthonormal** principal components

$$\text{PC}_1, \dots, \text{PC}_r.$$

If some of the **eigenvalues** are 0, then $r < p$, and *vice-versa* \implies data is embedded in an **r -dimensional subspace** in the first place.

PCA can provide an avenue for dimension reduction by “**removing**” components with **small eigenvalues**.

The **proportion of the spread in the data** which can be explained by each PC can be placed in a **scree plot (eigenvalues against ordered component indices)**, and we **retain** the ordered PCs:

- for which the **eigenvalue** is **above** some threshold (say, **25%**);
- for which the **cumulative proportion** of the spread falls **below** some threshold (say **95%**), or
- prior to a **kink** in the **scree plot**.

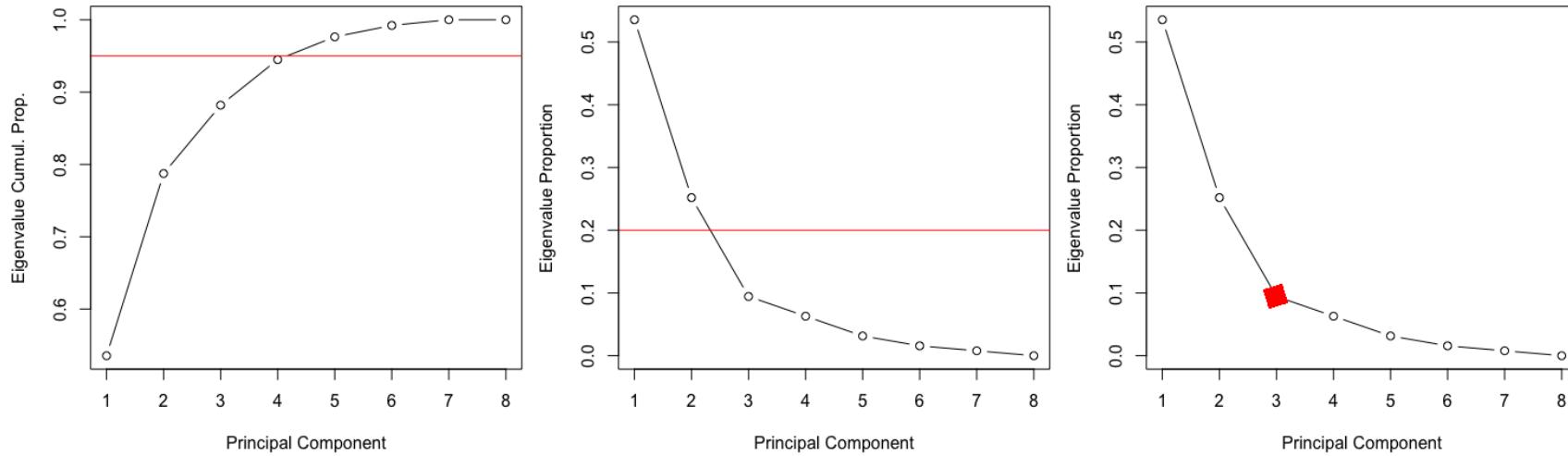
Example: consider an $8D$ dataset for which the ordered PCA eigenvalues are

PC	1	2	3	4	5	6	7	8
Var	17	8	3	2	1	0.5	0.25	0
Prop	54	25	9	6	3	2	1	0
Cumul	54	79	88	94	98	99	100	100

If only the PCs that explain up to 95% of the **cumulative variance** are retained, the original dataset reduces to a $4D$ subset.

If only the PCs that **individually explain** more than 25% of the variance are retained, the original dataset reduces to a $2D$ subset.

If only the PCs that lead into the **first kink** in the scree plot are retained, the original dataset reduces to a $3D$ subset.

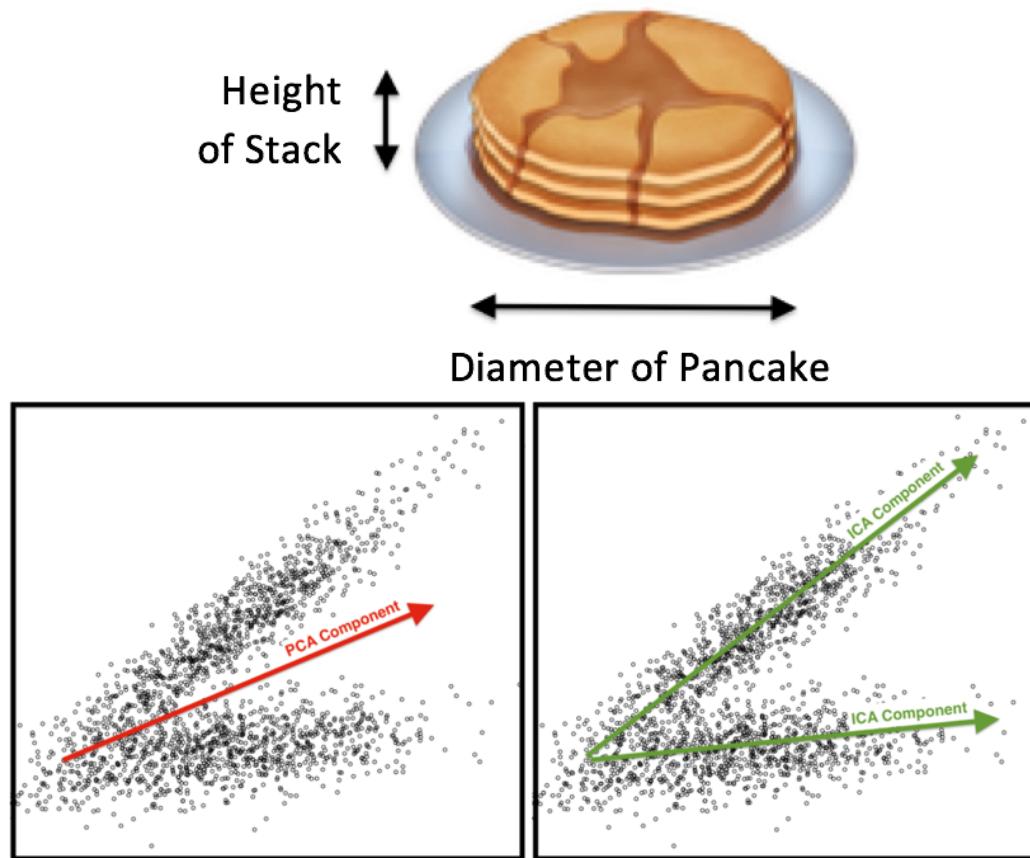


The proportion of the variance explained by each (ordered) component is shown in the 3 charts.

The **cumulative proportion** method is shown in the first image, the **individual threshold** method in the second, and the **kink** method in the third.

PCA Limitations:

- dependent on **scaling**, and so **not unique**;
- interpreting the PCs requires **domain expertise**;
- (quite) sensitive to **outliers**;
- analysis goals **not always aligned** with the PCs, and
- data assumptions **not always met** – does it always make sense that important data structures and data spread be linked (see **counting pancakes** problem), or that the PCs be **orthogonal**?



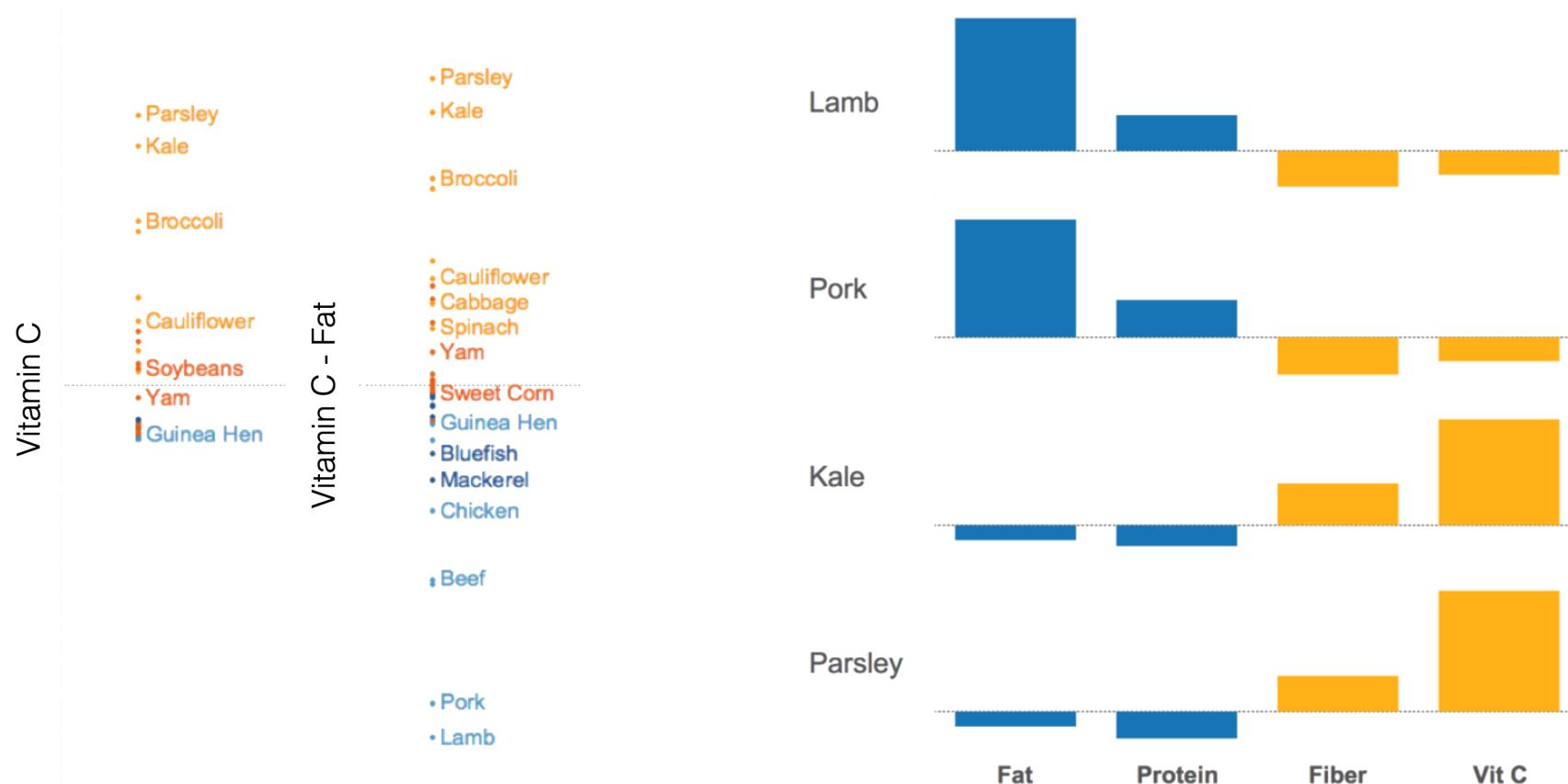
Example: USDA Data

What is the best way to differentiate food items? By vitamin content, fat, or protein level? A bit of each?

Vitamin C is present in various levels in fruit and vegetables, but not in meats. It separates vegetables from meats, and specific vegetables from one another (to some extent), but the meats are clumped together (see Vitamin C chart on the next slide).

The situation is reversed for Fat levels, so a combination of Vitamin C and Fat separates vegetables from meats, while it spreads them within their own groups (see Vitamin C – Fat chart on the next slide).

[algobeans.com]



The presence of various nutrients seems to be correlated among food items.

In a sample consisting of Lamb, Pork, Kale, and Parsley, both **Fat** and **Protein** levels seem **in step**, as do **Fiber** and **Vitamin C** (see previous slide).

(In a larger dataset, the correlations between Fat and Protein, and between Fiber and Vitamin C, are $r = 0.56$ and $r = 0.57$, respectively.)

The **loadings matrix** **W** for the **full** nutrition dataset is

	PC1	PC2	PC3	PC4
Fat	-0.45	0.66	0.58	0.18
Protein	-0.55	0.21	-0.46	-0.67
Fiber	0.55	0.19	0.43	-0.69
Vitamin C	0.44	0.70	-0.52	0.22

The first principal component PC_1 is thus the linear combination

$$\text{PC}_1 = -0.45 \times \text{Fat} - 0.55 \times \text{Protein} + 0.55 \times \text{Fiber} + 0.44 \times \text{Vitamin C}.$$

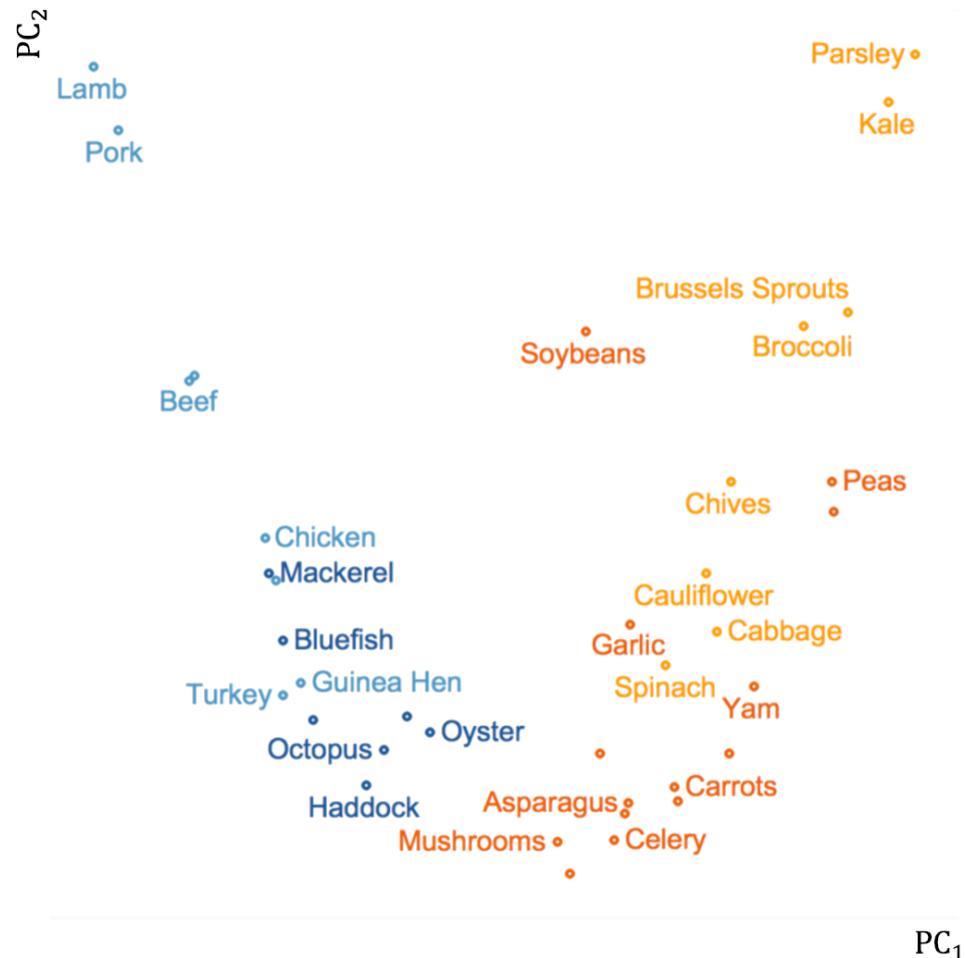
Note that

$$|-0.45|^2 + |-0.55|^2 + |0.55|^2 + |0.45|^2 = 1$$

and that PC_1 pairs **Fat** with **Protein**, and **Fiber** with **Vitamin C**, with slightly more emphasis put on **Protein** and **Fiber**.

PC_2 pairs **Fat** with **Vitamin C (strongly)**, and Protein with Fiber (**weakly**). Note that $\text{PC}_1^\top \text{PC}_2 = 0$. What about PC_3 and PC_4 ?

The corresponding eigenvalues have relative weights 43%, 23%, 20%, and 14%, then PC_1 and PC_2 explain about 66% of the spread in the data.



PC_1 differentiates meats from vegetables; PC_2 differentiates **sub-categories** within meats (using **Fat**) and vegetables (using **Vitamin C**):

- meats are concentrated on the **left (low** PC_1 values);
- vegetables are concentrated on the **right (high** PC_1 values);
- “seafood meats” are concentrated at the **bottom (low** PC_2 values);
- “non-leafy vegetables” are concentrated at the **bottom (low** PC_2 values).

There are other methods to find the **principal manifolds** of a dataset (**UMAP, self-organizing maps, auto-encoders, curvilinear component analysis, manifold sculpting, kernel PCA, SVD decomposition**, etc.).

5.3.2 – PCA Formalism

Because \mathbf{K} is **positive semi-definite** ($\mathbf{K} \geq 0$), its eigenvalues $\lambda_i = s_i^2 \geq 0$; they can be ordered in a **decreasing sequence**

$$\Lambda = \text{diag}(\lambda_1, \dots, \lambda_p), \quad \text{where } \lambda_1 \geq \dots \geq \lambda_p \geq 0 \quad \text{and} \quad \mathbf{W} = [\mathbf{w}_1 \mid \dots \mid \mathbf{w}_p].$$

If $k = \text{rank}(\mathbf{X})$, there are $p - k$ “**empty**” PC ($\lambda = 0$) and k “**regular**” PC ($\lambda \neq 0$). We write $\mathbf{W}^* = [\mathbf{w}_1 \mid \dots \mid \mathbf{w}_k]$ and $\Lambda^* = \text{diag}(\lambda_1, \dots, \lambda_k)$. If $p - k \neq 0$, then the **eigenvalue decomposition** of \mathbf{K} is

$$\mathbf{K} = [\mathbf{W}^* \quad \mathbf{0}] \begin{bmatrix} \Lambda^* & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} (\mathbf{W}^*)^\top \\ \mathbf{0} \end{bmatrix} = \mathbf{W} \Lambda \mathbf{W}^\top.$$

if \mathbf{X} is of **full rank**, then $\mathbf{W}^* = \mathbf{W}$, $\Lambda^* = \Lambda$, and $\mathbf{K} = \mathbf{W} \Lambda \mathbf{W}^\top$ (still).

The eigenvectors of \mathbf{K} (the \mathbf{w}_j) are the **singular vectors** of \mathbf{X} : there exist $\mathbf{U}_{n \times n}$ and $\Sigma_{n \times p}$ such that

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{W}^\top, \quad \text{where } \mathbf{U} = [\mathbf{U}^* \mid \mathbf{0}], \quad \Sigma = \begin{bmatrix} \text{diag}(s_i) \\ \mathbf{0} \end{bmatrix}.$$

If \mathbf{X} is of **full rank**, then \mathbf{W} is **orthonormal (rotation matrix)**. As $\mathbf{W}^{-1} = \mathbf{W}^\top$, we have $\mathbf{XW} = \mathbf{U}\Sigma$, the **PC decomposition** of \mathbf{X} :

$$\mathbf{T}_{n \times p} = \mathbf{XW}, \quad [\mathbf{t}_1 \mid \cdots \mid \mathbf{t}_p] = [\mathbf{x}_1 \mid \cdots \mid \mathbf{x}_n]^\top [\mathbf{w}_1 \mid \cdots \mid \mathbf{w}_p].$$

PC_1 is the **loading** \mathbf{w}_1 ($\|\mathbf{w}_1\|_2 = 1$) which **maximizes the variance** of \mathbf{t}_1 :

$$\mathbf{w}_1 = \arg \max_{\|\mathbf{w}\|_2=1} \{\text{Var}(\mathbf{t}_1)\} = \arg \max_{\|\mathbf{w}\|_2=1} \left\{ \frac{1}{n-1} \sum_{i=1}^n (t_{1,i} - \bar{t}_1)^2 \right\}.$$

Since

$$\overline{t_1} = \frac{1}{n} \sum_{j=1}^n \mathbb{E}[\mathbf{x}_j^\top \mathbf{w}_1] = \frac{1}{n} \sum_{j=1}^n \mathbb{E} \left[\sum_{i=1}^p x_{j,i} w_{i,1} \right] = \frac{1}{n} \sum_{j=1}^n \sum_{i=1}^p w_{i,1} \underbrace{\mathbb{E}[x_{j,i}]}_{=0} = 0,$$

then $\text{Var}(t_1) = \frac{1}{n-1} (t_{1,1}^2 + \cdots + t_{n,1}^2)$ and the problem is equivalent to

$$\mathbf{w}_1 = \arg \max_{\|\mathbf{w}\|_2=1} \{t_{1,1}^2 + \cdots + t_{n,1}^2\},$$

By construction, $t_{i,1}^2 = (\mathbf{x}_i^\top \mathbf{w}_1)^2$ for all i , so

$$t_{1,1}^2 + \cdots + t_{n,1}^2 = (\mathbf{x}_1^\top \mathbf{w}_1)^2 + \cdots + (\mathbf{x}_n^\top \mathbf{w}_1)^2 = \|\mathbf{X}\mathbf{w}_1\|_2^2 = \mathbf{w}_1^\top \mathbf{X}^\top \mathbf{X} \mathbf{w}_1.$$

Hence,

$$\mathbf{w}_1 = \arg \max_{\|\mathbf{w}\|_2=1} \{\mathbf{w} \mathbf{X}^\top \mathbf{X} \mathbf{w}\} = \arg \max_{\|\mathbf{w}\|_2=1} \{\mathbf{w}^\top \mathbf{K} \mathbf{w}\};$$

this is equivalent to finding the **maximizer** of $F(\mathbf{w}) = \mathbf{w}^\top \mathbf{K} \mathbf{w}$ subject to

$$G(\mathbf{w}) = 1 - \mathbf{w}^\top \mathbf{w} = 0.$$

We solve the problem using the method of **Lagrange multipliers**; any optimizer \mathbf{w}^* must be either:

1. a **critical point** of F , or
2. a solution of $\nabla F(\mathbf{w}) + \lambda \nabla G(\mathbf{w}) = \mathbf{0}$, $\lambda \neq 0$.

But $\nabla F(\mathbf{w}) = 2\mathbf{K}\mathbf{w}$ and $\nabla G(\mathbf{w}) = -2\mathbf{w}$; either $\mathbf{w}^* \in \ker(\mathbf{K})$ (case 1) or $2\mathbf{K}\mathbf{w}^* - 2\lambda^*\mathbf{w}^* = \mathbf{0}$ (case 2); so, either

$$\mathbf{K}\mathbf{w}^* = \mathbf{0} \quad \text{or} \quad (\mathbf{K} - \lambda^* I)\mathbf{w}^* = \mathbf{0}, \quad \lambda^* \neq 0.$$

Either way, $\lambda^* \geq 0$ is an **eigenvalue** of K , with associated **eigenvector** \mathbf{w}^* . There are **at most** p **distinct** pairs $\{(\lambda_j, \mathbf{w}_j)\}_{j=1}^p$; for each of them,

$$\mathbf{w}_j^\top \mathbf{K} \mathbf{w}_j = \mathbf{w}_j^\top \lambda_j \mathbf{w}_j = \lambda_j \mathbf{w}_j^\top \mathbf{w}_j = \lambda_j,$$

since $\mathbf{w}_j^\top \mathbf{w}_j = 1$. As such,

$$\arg \max_{\|\mathbf{w}\|_2=1} \{\text{Var}(\mathbf{t}_1)\} = \arg \max_{\|\mathbf{w}\|_2=1} \{\lambda_j\} = \mathbf{w}_1 = \text{PC}_1,$$

since $\lambda_1 \geq \lambda \geq 0$ for all **eigenvalues** λ of \mathbf{K} .

For $j = 2, \dots, p$, a similar argument shows that \mathbf{w}_j is the direction along which the **variance is the j th highest**, assuming that \mathbf{w}_j is **orthonormal** to all the preceding \mathbf{w}_ℓ , $\ell = 1, \dots, j-1$, and that the **variance is proportional** to λ_j .

But that is exactly the case due to the nature of (\mathbf{W}, Λ) as the **eigenvalue decomposition** of \mathbf{K} .

This process can be repeated **at most p times**, yielding $0 \leq r \leq p$ **non-trivial** principal components $\text{PC}_1, \dots, \text{PC}_r$, where $r = \text{rank}(\mathbf{X})$.

Thus, we see that the **rotation matrix** \mathbf{W} that **maximizes** the variance **sequentially** in the columns of $\mathbf{T} = \mathbf{X}\mathbf{W}$ is the matrix of **eigenvectors** \mathbf{W} of $\mathbf{K} = \mathbf{X}^\top \mathbf{X}$.

$\text{PC} \implies$ finds the “best” rotation of the axes to “fit” the data.

5.3.3 – PCA Regression

We return to the original problem of determining the coefficients $\phi_{m,j}$ in the **reduced LS**.

Assume that M **principal components** $\{Z_1, \dots, Z_M\}$ are retained, where

$$Z_i = \mathbf{w}_i^\top (X_1, \dots, X_p),$$

and the **eigenvectors** \mathbf{w}_i have the order of the **corresponding eigenvalues** ($\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p \geq 0$):

- PC_1 is the **normalized (centered and scaled)** linear combination of variables with the **largest** variance;

- PC_1 is the **normalized** linear combination of variables with the **largest** variance, subject to having **no correlation** with all preceding **principal components** (PC_1);
- ...
- PC_M is the **normalized** linear combination of variables with the **largest** variance, subject to having **no correlation** with all preceding **principal components** ($\text{PC}_1, \dots, \text{PC}_{M-1}$).

In an “ideal” world, the regression function $f(\mathbf{x}) = \mathbb{E}[Y \mid \vec{X} = \mathbf{x}]$ is **well-approximated** by the function

$$g(\mathbf{z}) = \mathbb{E}[Y \mid \vec{Z} = \mathbf{z}].$$

In other words, if $\mathbf{z} = \mathbf{W}\mathbf{x}$, we hope that

$$\hat{y}_z = g(\mathbf{z}) = \gamma_0 + \gamma_1 z_1 + \cdots + \gamma_M z_M$$

compares “**reasonably well**” to

$$\hat{y}_x = f(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p.$$

Main challenge: determining the **optimal** M . If M is **too small**, the model may have **high** bias and **low** variance (**underfitting**); if M is **too large**, not only do we not achieve much in the way of **dimension reduction**, but the model may have **low** bias and **high** variance (**overfitting**).

Any method that allows for the estimation of the **test error** (i.e., **cross-validation**) could be used to select M ; we have seen other approaches.

5.3.4 – Partial Least Squares

In **PC regression (PCR)**, the **directions** (linear combinations) that “best” represent the predictors $\{X_1, \dots, X_p\}$ are determined in an **unsupervised** manner: the response Y plays no role in determining the PC.

There is **no guarantee** that the directions that “best” explain the predictors are also the “best” directions to use to **predict** the response (i.e., pancakes).

The framework for **partial least squares** is the same as that for PCR, except that the directions Z_i are selected **both** to **describe** the predictors and to **relate** to the response Y .

We start by **normalizing** (**centering** and **scaling**) Tr (design matrix \mathbf{X}).

The **1st direction** Z_1 is computed using the **OLS coefficient** estimates of

$$Y_i = \phi_{0,j}^1 + \phi_{1,j} X_{i,j} + \gamma_i, \quad j = 1, \dots, p, \quad i = 1, \dots, N.$$

Each $\phi_{1,j}$ is **proportional** to $\rho_{X_j, Y}$ and the direction $Z_1 = \phi_1^\top \vec{X}$ places the **largest weights** on the predictors that are **most strongly linked** to Y .

Next, we run an OLS regression of Y using Z_1 as a predictor:

$$Y_i = \psi_0 + \psi_1 Z_{1,i} + \varepsilon_{i;(1)}, \quad i = 1, \dots, N;$$

let $\varepsilon_{i;(1)} = Y_i - \psi_0 - \psi_1 Z_{1,i}$ be the part of the data not “explained” by Z_1 .

The **2nd direction** Z_2 is computed using the **OLS coefficient** estimates of

$$\varepsilon_{i;(1)} = \phi_{0,j}^2 + \phi_{2,j} X_{i,j} + \gamma_i, \quad j = 1, \dots, p, \quad i = 1, \dots, N.$$

Each $\phi_{2,j}$ is **proportional** to $\rho_{X_j, \varepsilon}$ and the direction

$$Z_2 = \sum_{j=1}^p \phi_{2,j} X_j = \boldsymbol{\phi}_2^\top \vec{X}$$

places the **largest** weights on the predictors that are **most strongly linked** to $\varepsilon^{(1)}$ (i.e., the component that **does not explain** Z_1).

The process continues in the same way, building directions Z_3, \dots, Z_p that are **strongly linked**, in sequence, to the **preceding residuals**.

As the chain starts with the response Y , the directions do take into account both the **related response** and the **predictor structure**.

The problem of selecting M is tackled as it is in PCR.

The Rest of the DS/ML Picture

- Data preparation
- Data visualization
- Data engineering
- Anomaly detection and outlier analysis
- Webscraping and automated data collection
- Data streams
- Text mining and natural language processing
- Deep learning
- Reinforcement learning
- Bayesian data analysis
- Big Data and parallel processing
- Recommender systems
- Network data analysis
- Image mining and computer vision
- Causality modeling
- Time series analysis
- Categorical data analysis
- Sampling methods
- Design of experiments

References

G.James, D.Witten, T.Hastie, and R.Tibshirani, An Introduction to Statistical Learning: With Applications in R. Springer, 2014.

“An interactive visualization to teach about the curse of dimensionality,” simplystatistics.org.

J.B.Tenenbaum, V.de Silva, and J.C.Langford, “A Global Geometric Framework for Nonlinear Dimensionality Reduction,” Science, vol. 290, no. 5500, p. 2319, 2000.

S.T.Roweis and L.K.Saul, “Nonlinear Dimensionality Reduction by Locally Linear Embedding,” Science, vol. 290, no. 5500, pp. 2323–2326, 2000.

Wikipedia, “T-distributed stochastic neighbor embedding.”

scikit-learn.org, “Manifold learning.”

Y.LeCun, C.Cortes, and C.J.C.Burges, “The MNIST database of handwritten digits.”

scikit-learn.org, “Manifold learning on handwritten digits.”

C.C.Aggarwal and C.K.Reddy, Eds., Data Clustering: Algorithms and Applications. CRC Press, 2014.

C.C.Aggarwal, Data Mining: The Textbook. Cham: Springer, 2015.

C.C.Aggarwal, Ed., Data Classification: Algorithms and Applications. CRC Press, 2015.

Wikipedia, “Pearson correlation coefficient.”

Wikipedia, “Point biserial correlation coefficient.”

Wikipedia, “Eta-squared.”

learntech, “Chi-squared test for nominal (categorical) data.” UWE Bristol.

Wikipedia, “Mutual information.”

Y.Sun and D.Wu, “A RELIEF based feature extraction algorithm,” Apr. 2008, pp. 188–195.

F.E.Harrell, Regression Modeling Strategies: With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis. Springer International Publishing, 2015.

“Stopping stepwise: Why stepwise selection is bad and what you should use instead,” towardsdatascience.com, 2018.

J.Leskovec, A.Rajamaran, and J.D.Ullman, Mining of Massive Datasets. Cambridge Press, 2014.

Y.Goldberg, Neural Network Methods for Natural Language Processing. Morgan, 2017.

Z.A.Zhao and H.Liu, Spectral Feature Selection for Data Mining. CRC Press, 2011.

T.Zhang and R.K.Ando, “Analysis of spectral kernel design based semi-supervised learning,” in Advances in neural information processing systems, 2005, vol. 18.

A.J.Smola and R.Kondor, “Kernels and regularization on graphs,” in Learning Theory and Kernel Machines, 2003, pp. 144–158.

T.Gowers, J.Barrow-Green, and I.Leader, Eds., The Princeton Companion to Mathematics. Princeton University Press, 2008.

L.McInnes, J.Healy, and J.Melville, “UMAP: Uniform manifold approximation and projection for dimension reduction,” arXiv preprint, 2018.