

MAT 4376/5314E

Techniques of Data Analysis

Module 4

Feature Selection and Dimension Reduction

P.Boily (uOttawa)
with O.Leduc, A.Macfie, A.Maheshwari, M.Pelletier

Fall 2020

Feature Selection and Dimension Reduction

Outline

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. The challenges presented by high data dimensionality must be addressed in order to achieve insightful and interpretable analytical results.

Scenario – NHL Game and Data Reduction (p.3)

4.1 – Dimension Reduction (p.9)

- The Curse of Dimensionality (p.11)
- Principal Component Analysis (p.21)
- The Manifold Hypothesis (p.36)

4.2 – Feature Selection (p.64)

- Filter Methods (p.69)
- Wrapper Methods (p.87)
- Subset Selection Methods (p.90)
- Regularization (Embedded) Methods (p.92)
- Supervised and Unsupervised Methods (p.98)

4.3 – Advanced Topics (p.100)

Course Notes + Examples (with Code) + References:

Leduc, O., Macfie, A., Maheshwari, A., Pelletier, M., Boily, P. [2020],
Feature Selection and Data Reduction, Data Science Report Series.

Scenario – NHL Game and Data Reduction

Consider the NHL game that took place between the Ottawa Senators and the Toronto Maple Leafs on February 18, 2017.

1st Approximation: a hockey game is a series of sequential and non-overlapping “events” involving two teams of skaters.

What does it mean to extract useful insights from such a series of events?

Most complete raw understanding of a game might belong to its active and passive participants: players, referees, coaches, general managers, official scorer and time-keeper, etc.

People who attended the game in person, watched it on TV/Internet, or listened to it on the radio presumably also have a lot of the facts at their disposal, with **possible contamination by commentators and analysts**.

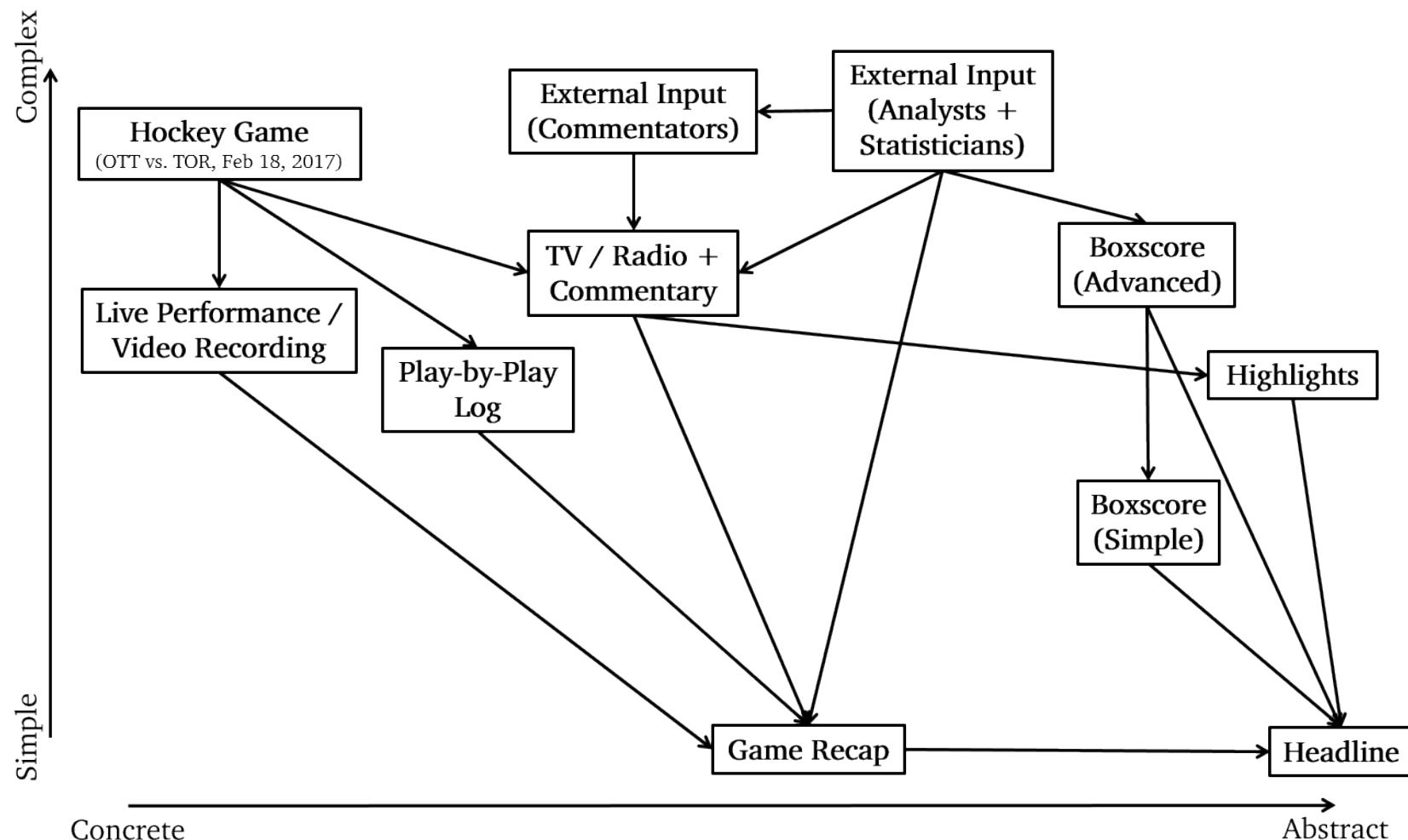
How could information about the game best be relayed to people who did not play/catch the game?

There are many ways to do so, depending on the intended **level of abstraction** and on the **target audience**.

Some of these methods might yield different (even contradictory) insights.

Who is right?

Does it even make sense to ask the question?



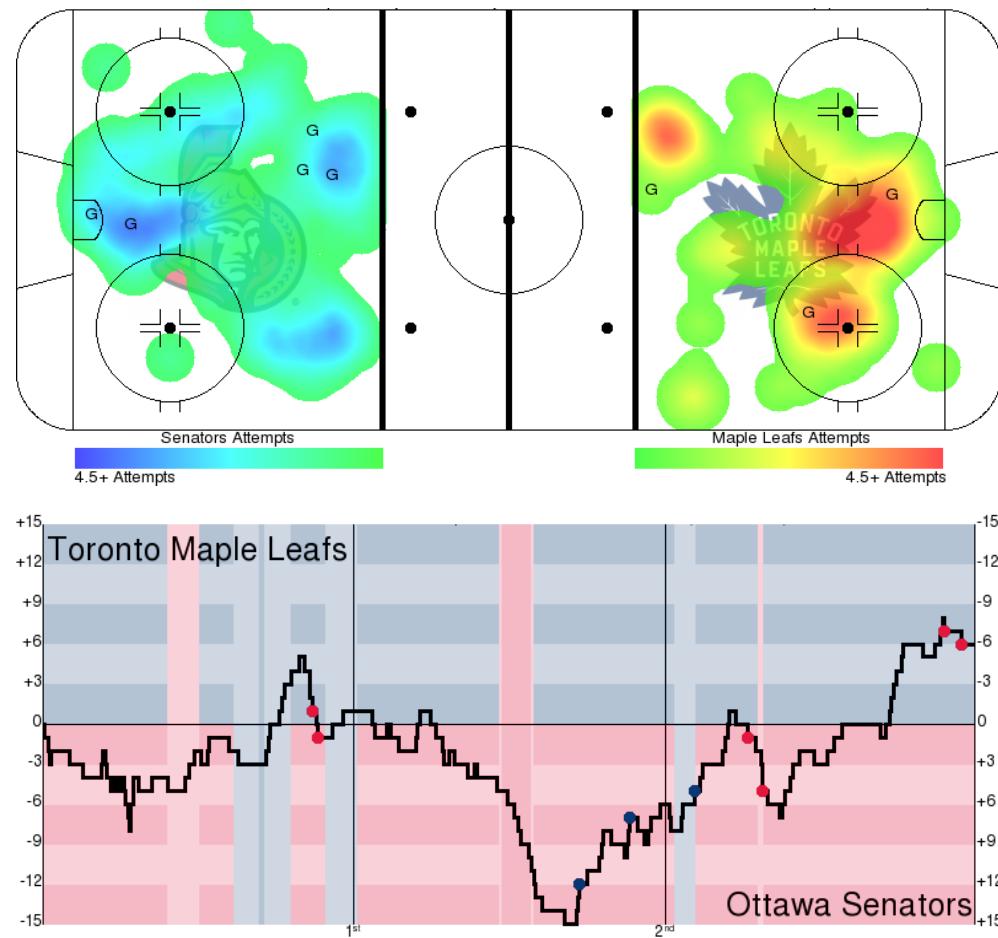
Schematic diagram of data reduction – professional hockey game.

Ottawa				
Senators				
31-19-6		6		
	1 2 3 T			
OTT	2 0 4 6			★ Stone (Senators - RW): Goals: 1, Assists: 4
TOR	0 2 1 3			★★ Brassard (Senators - C): Goals: 2, Assists: 1
				★★★ Nylander (Maple Leafs - RW): Goals: 1, Assists: 1

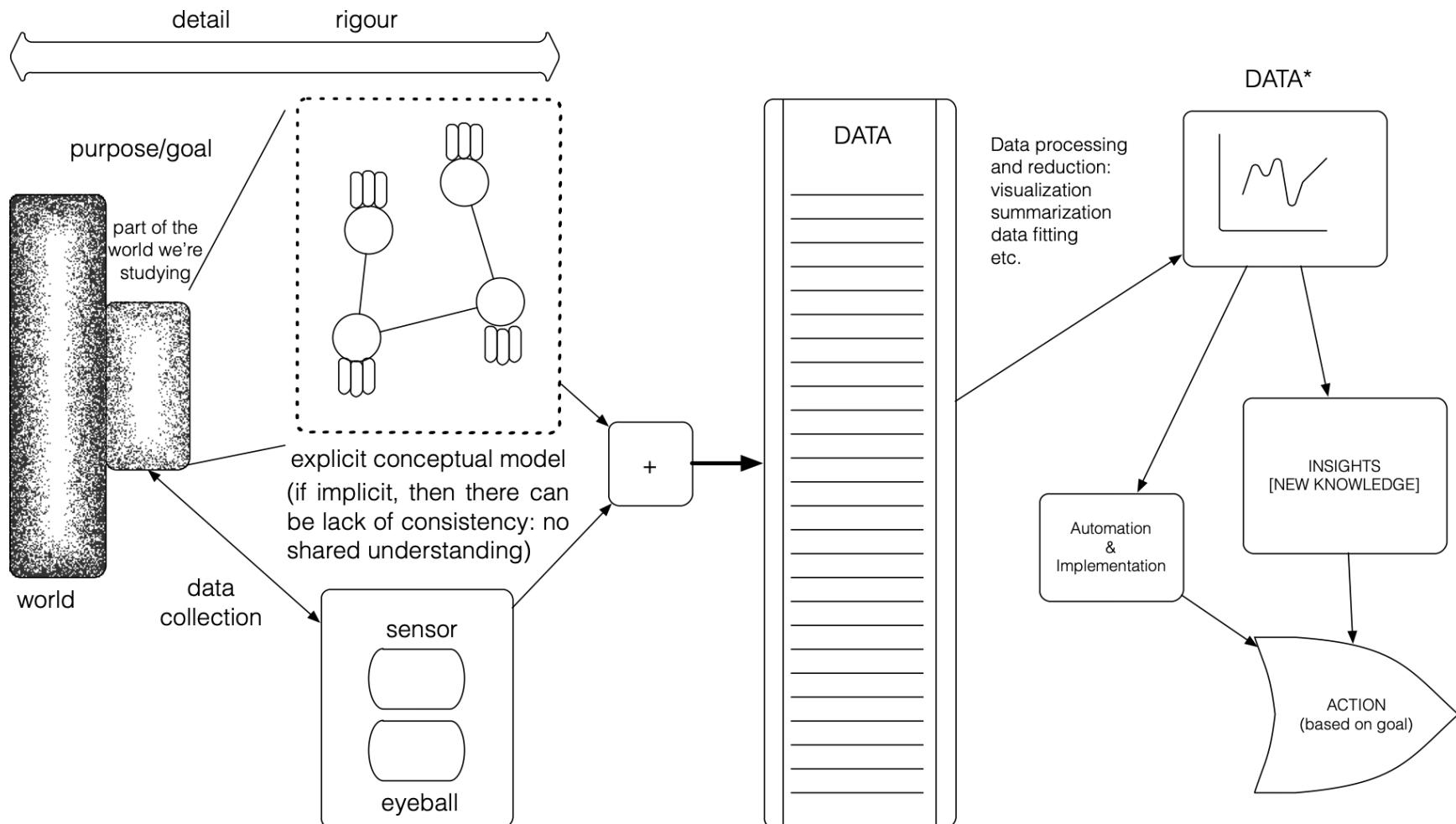
Boxscore, Ottawa Senators @ Toronto Maple Leafs, 18-02-2017 (espn.com)

**Sens rally after blowing lead,
beat Leafs, gain on Habs.**

Headline, Associated Press, 18-02-2017



Unblocked shot heatmap and Corsi gameflow chart (Natural Stat Trick).



Schematic diagram of data reduction – general problem (Schellinck).

4.1 – 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.

4.1.1 – The 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 training set, increasing k in a k NN model, say, will yield enough data 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 doesn't, local models become global models (global models are not affected).

Manifestations of CoD

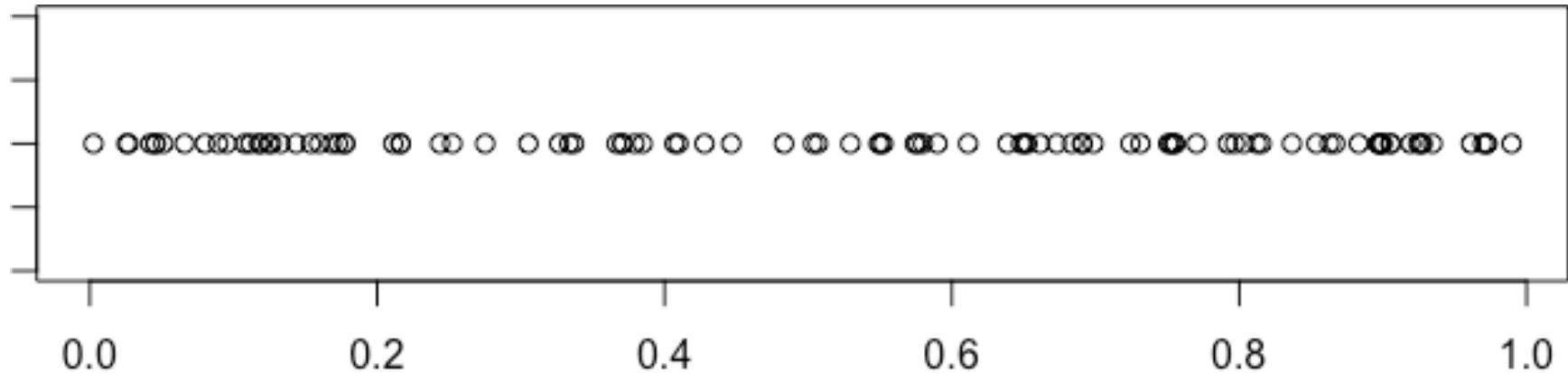
Let $x_i \sim U^1(0, 1)$ be i.i.d. for $i = 1, \dots, N$. For any $z \in [0, 1]$ and $\varepsilon > 0$ such that

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

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

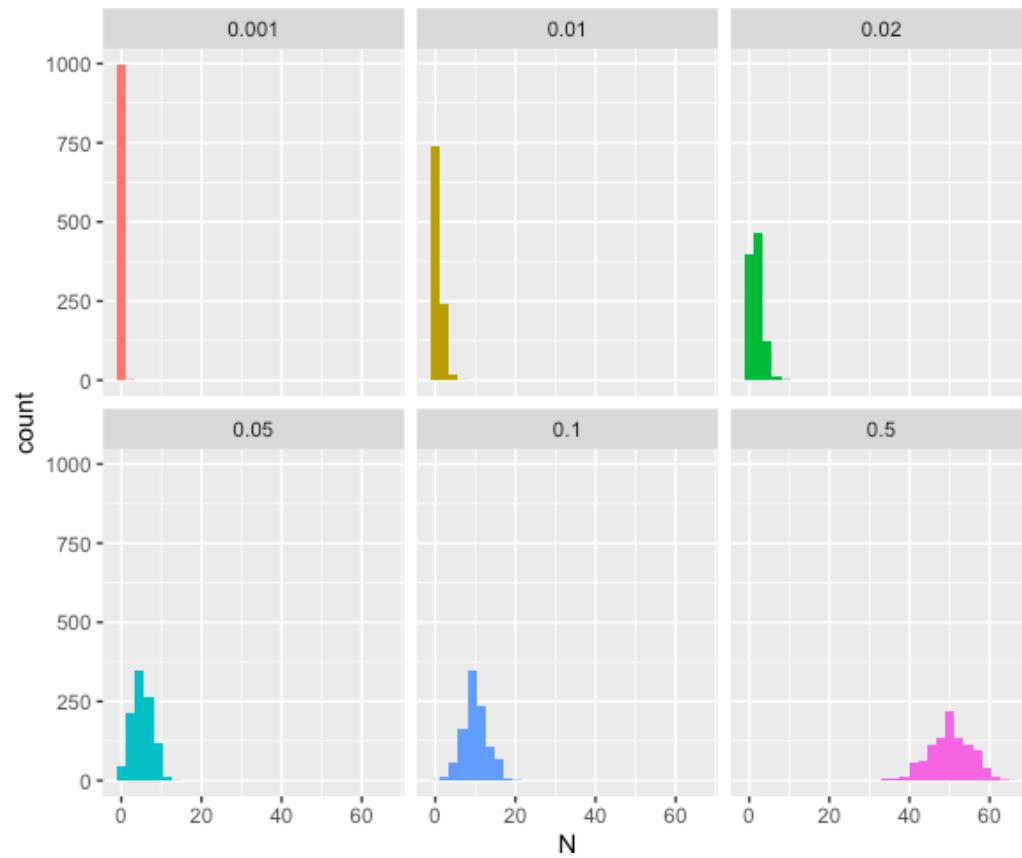
$$|I_1(z; \varepsilon) \cap \{x_i\}_{i=1}^N| \approx \varepsilon \cdot N.$$

In other words, an ε_∞ -ball subset of $[0, 1]^1$ contains about ε 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
N	0	1	2	3	5	39
$E[N]$	0.1	1	2	5	10	50



Repeating the experiment
1000 times yields:

ϵ	\bar{N}	$E[N]$
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

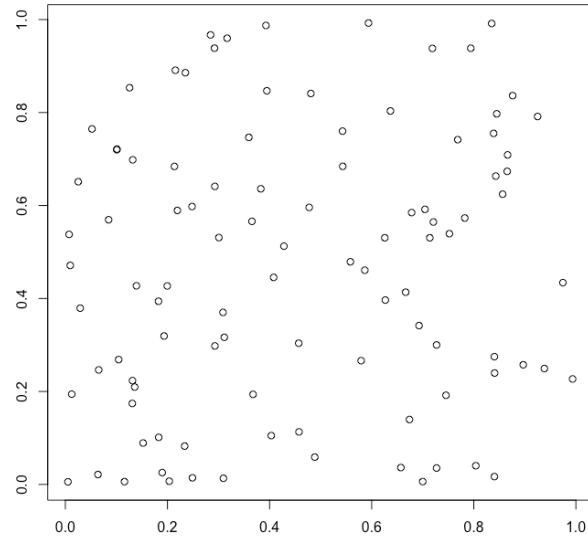
Now, let $\mathbf{x}_i \sim U^2(0, 1)$ be i.i.d. for $i = 1, \dots, N$. For any $\mathbf{z} \in [0, 1]^2$ and $\varepsilon > 0$ such that

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

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

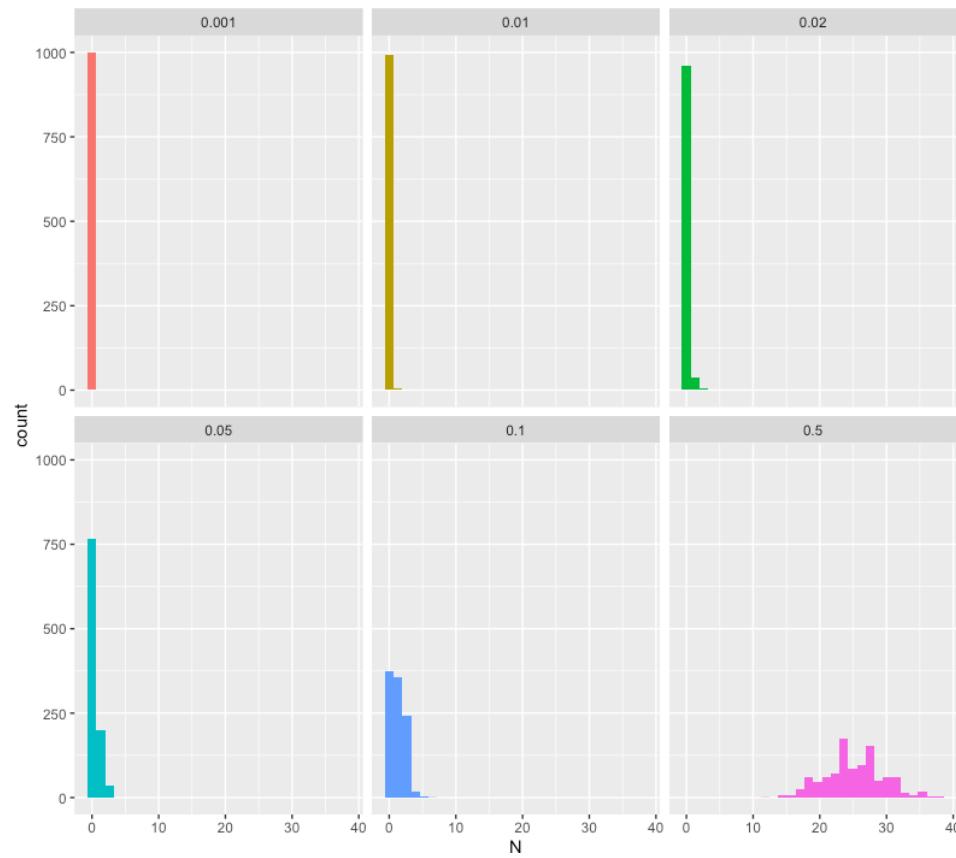
$$|I_1(\mathbf{z}; \varepsilon) \cap \{\mathbf{x}_i\}_{i=1}^N| \approx \varepsilon^2 \cdot N.$$

In other words, an ε_∞ -ball subset of $[0, 1]^2$ contains approximately ε^2 of the observations in $\{\mathbf{x}_i\}_{i=1}^N \subseteq \mathbb{R}^2$, on average.



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
N	0	0	0	0	0	25
$E[N]$	0.0001	0.01	0.04	0.25	1	25



Repeating the experiment
1000 times yields:

ε	\bar{N}	$E[N]$
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 increases, 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!** (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**.

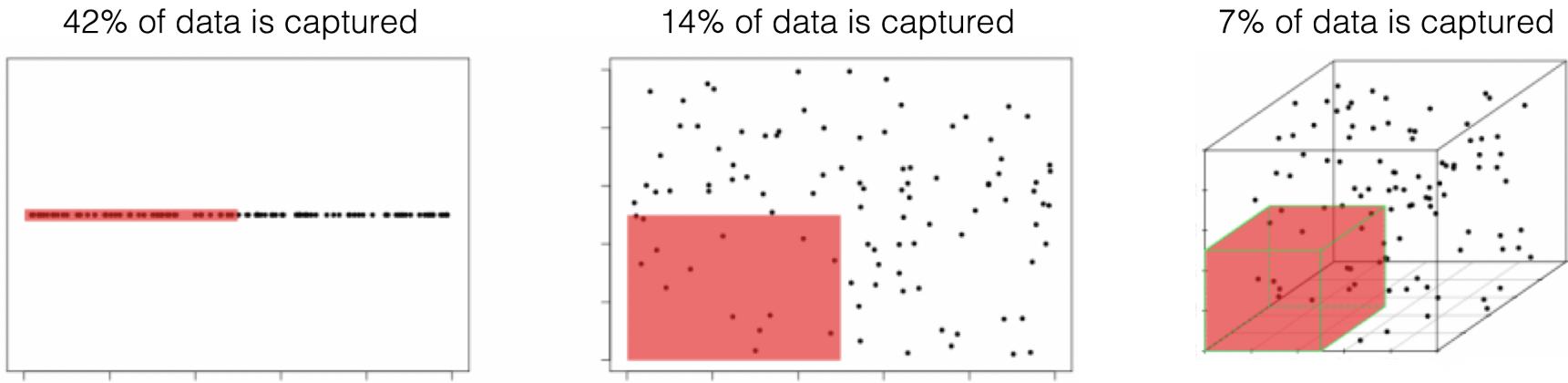


Illustration of the CoD; $N = 100$ observations are uniformly distributed on the unit hypercube $[0, 1]^d$, $d = 1, 2, 3$.

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

The percentage of captured datapoints is seen to decrease with an increase in d (from simplystatistics.com).

4.1.2 – Principal Component 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 can lead to a “small” loss of information.

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

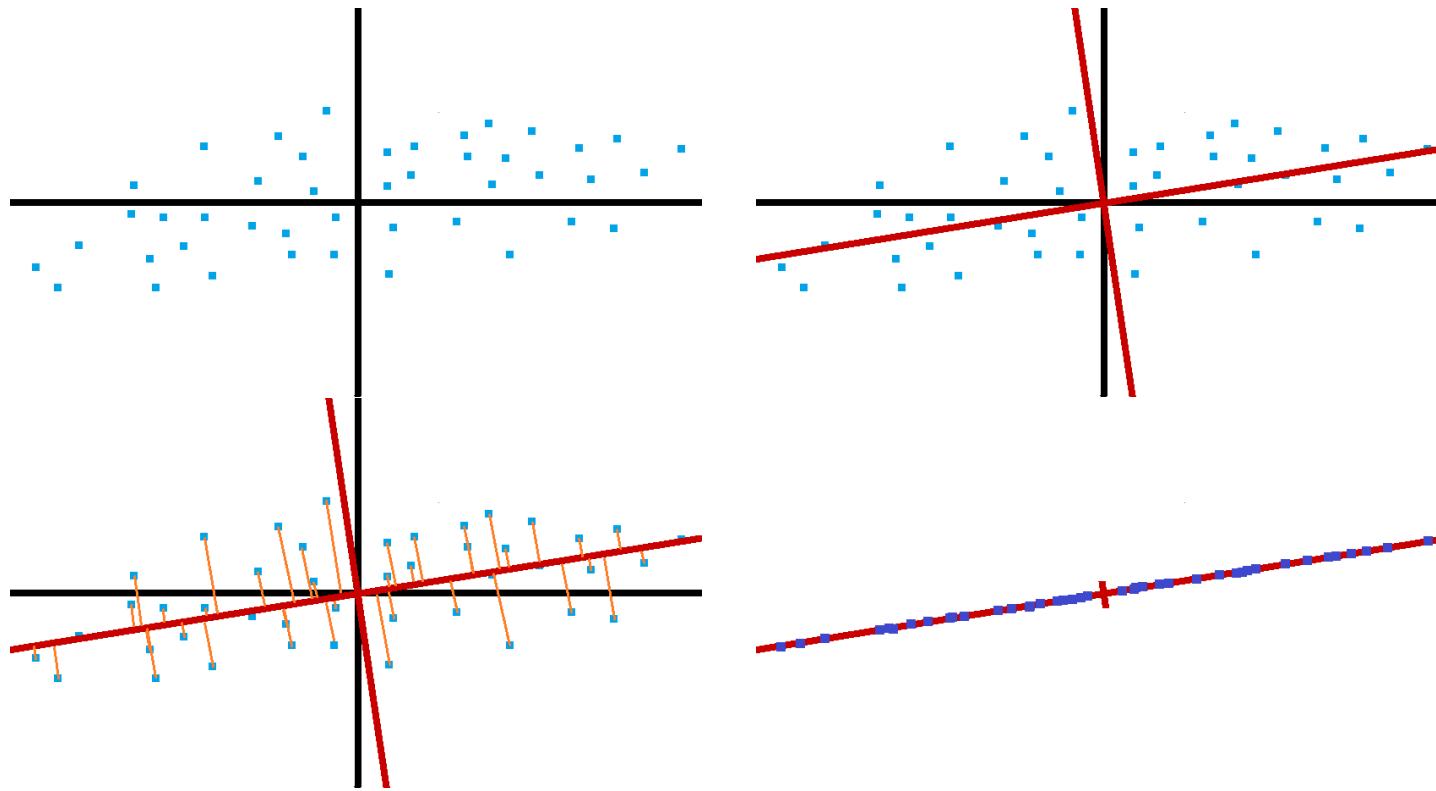


Illustration of PCA on an artificial 2D dataset. The red axes provide the best elliptic fit. Removing the minor axis by projecting the points on the major axis leads to dimension reduction and a (small) loss of information.

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 **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, $r < p$, and *vice-versa* \implies data is embedded in a 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 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.

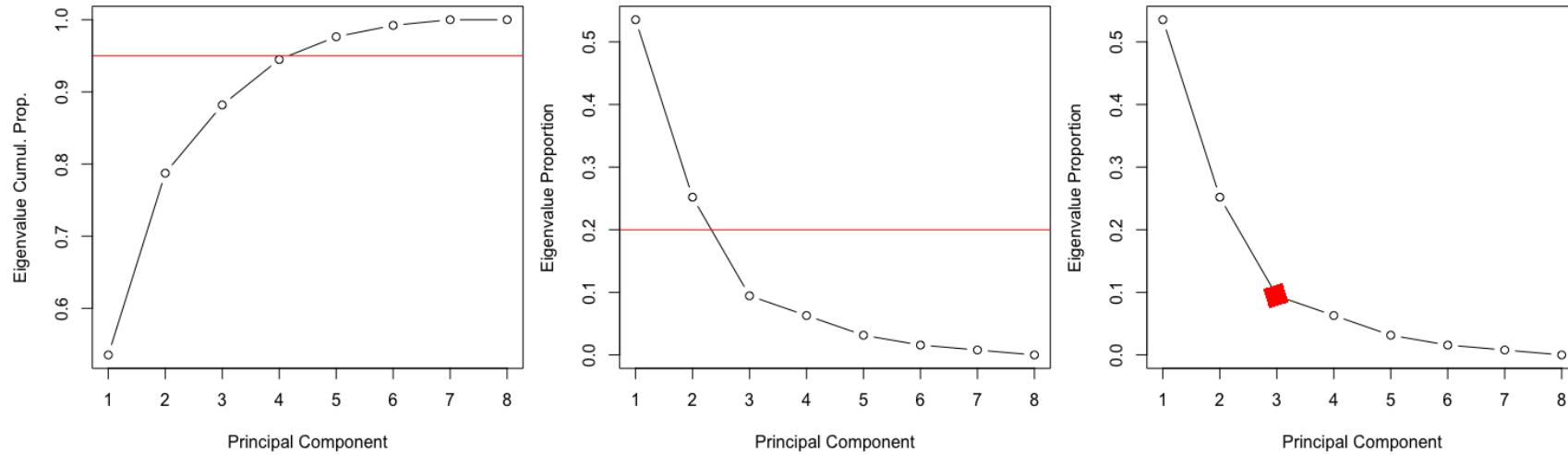
Example: consider an $8D$ dataset for which the ordered PCA eigenvalues are

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.

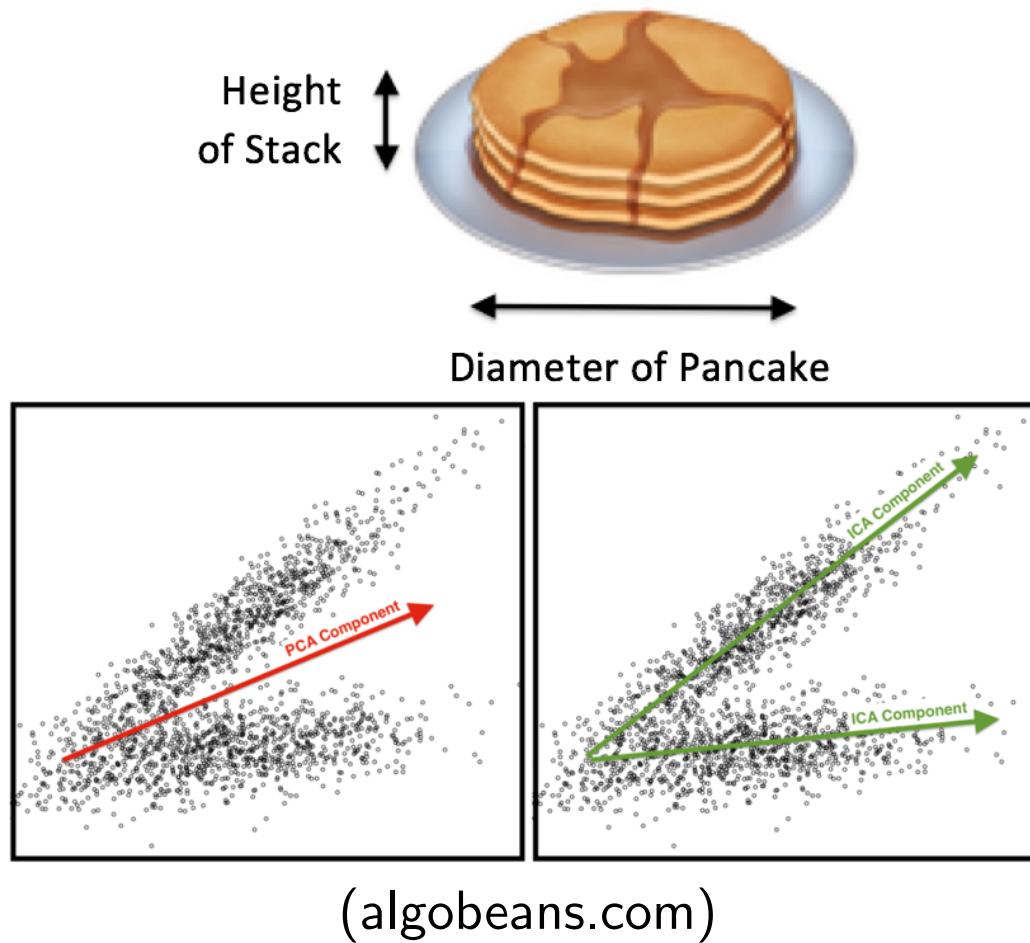


The proportion of the variance explained by each (ordered) component is shown in the first 3 charts; the cumulative proportion is shown in the last chart.

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 require 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**?



PCA Example (Ng, Soo; 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**.

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 two leftmost charts on p.31.



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

In the (small) sample consisting of Lamb, Pork, Kale, and Parsley, both Fat and Protein levels seem in step, as do Fiber and Vitamin C (see p.31).

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 \mathbf{W} for the 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 the linear combination

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

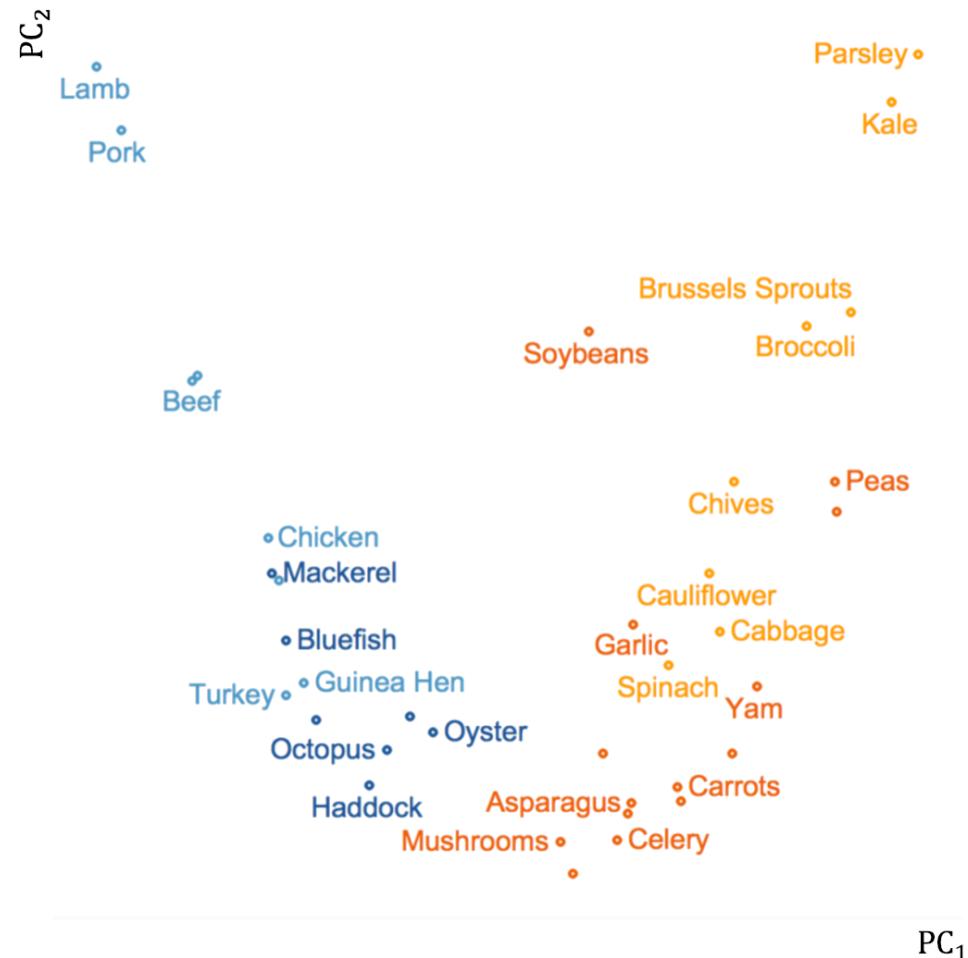
Note that

$$|-0.45|^2 + |-0.55|^2 + |0.55|^2 + |0.44|^2 = 1$$

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, and Protein with Fiber, but weakly. Note that $\text{PC}_1^\top \text{PC}_2 = 0$. What about PC_3 and PC_4 ?

If 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, including UMAP, self-organizing maps, auto-encoders, curvilinear component analysis, manifold sculpting and kernel PCA.

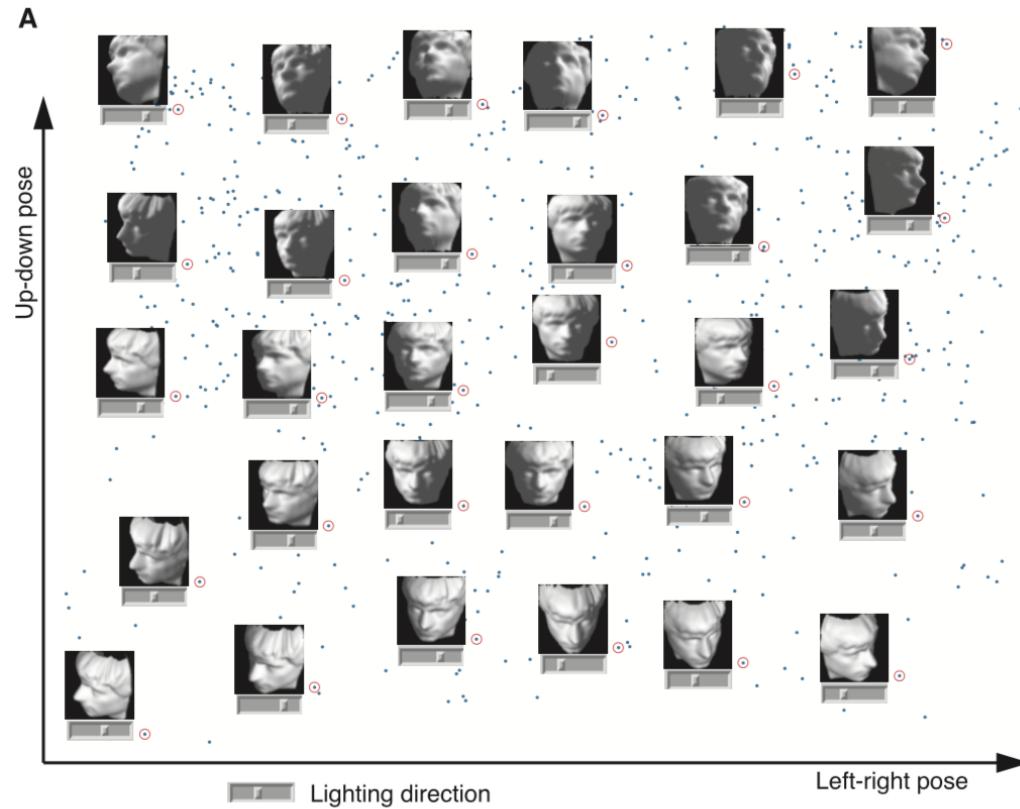
4.1.3 – The Manifold Hypothesis

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.

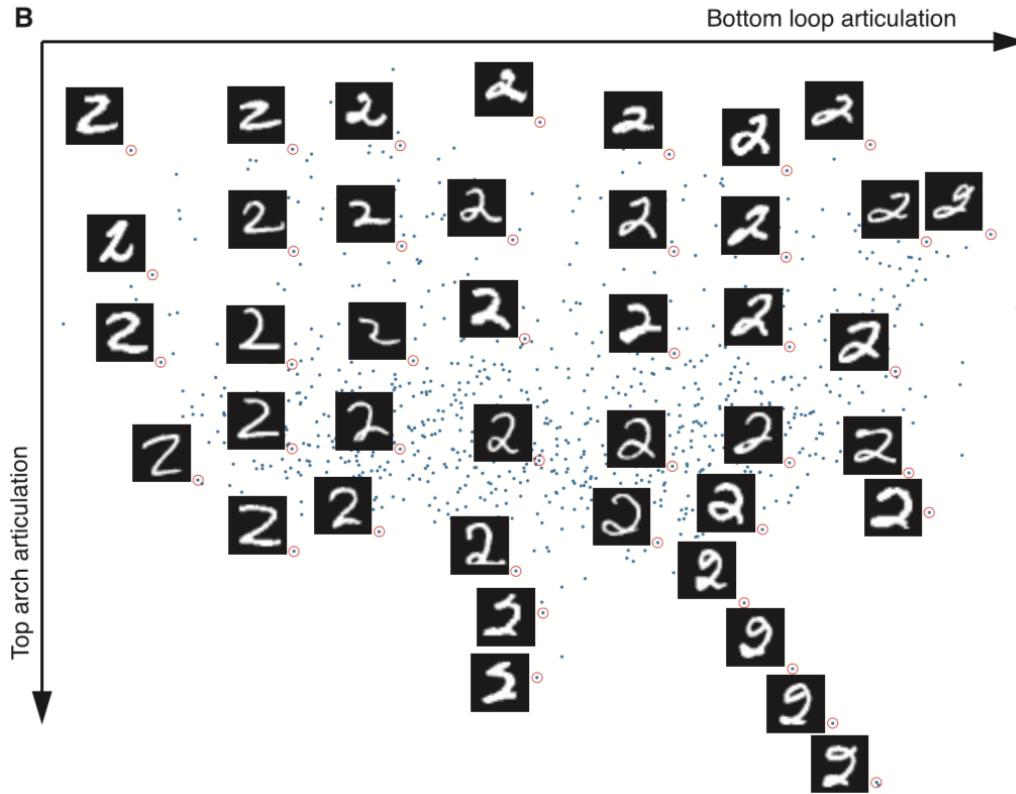
Example: multiple photographs of a $3D$ object taken from different positions (but at the same distance) can be represented by 2 d.f.: a horizontal angle and a vertical angle.



Plots showing degrees of freedom manifolds for images of faces – 3D object (Tenenbaum, Silva, Langford).

Another example: set of hand-written drawings of the digit “2”. Each of these drawings can also be represented using a small number of d.f.:

- the ratio of the length of the lowest horizontal line to the height of the hand-written drawing;
- the ratio of the length of the arch in the curve at the top to the smallest horizontal distance from the end point of the arch to the main vertical curve;
- the rotation of the digit as a whole with respect to some baseline orientation, etc.



Plots showing degrees of freedom manifolds for images of handwritten digits (Tenenbaum, Silva, Langford).

Dimensionality reduction and manifold learning are often used:

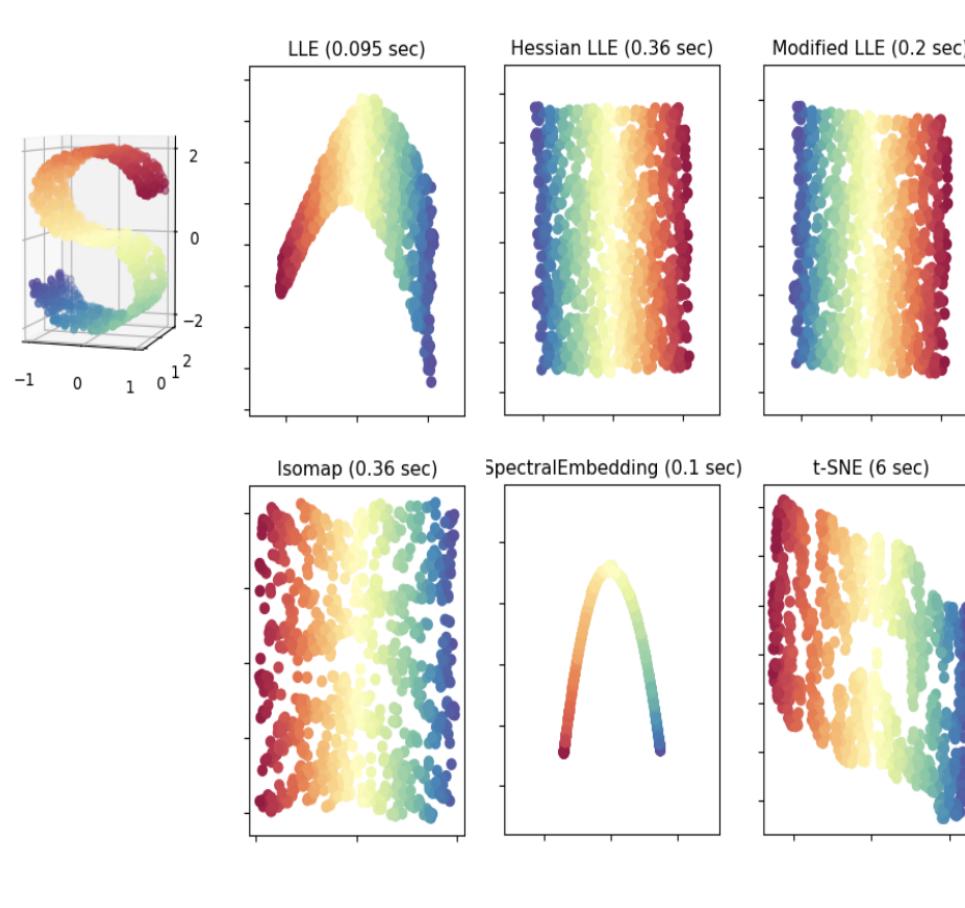
- to **reduce the overall dimensionality** of the data while trying to **preserve its variance**;
- to **display** high-dimensional datasets, or
- to **reduce the processing time** of supervised learning algorithms by lowering the dimensionality of the processed data.

Example: PCA 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.

In this section, the focus is on non-linear dimensionality reduction methods, most of which are a variant of **kernel PCA**:

- LLE
- Laplacian eigenmap
- isomap
- semidefinite embedding,
- t -SNE.

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



Comparison of manifold learning methods on an artificial dataset (Cayton).

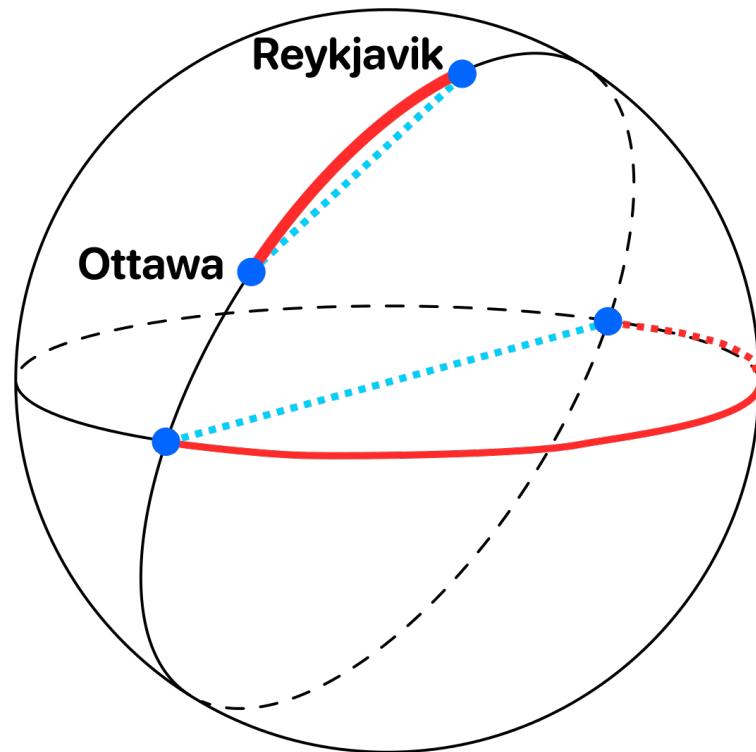
Kernel Principal Component Analysis

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

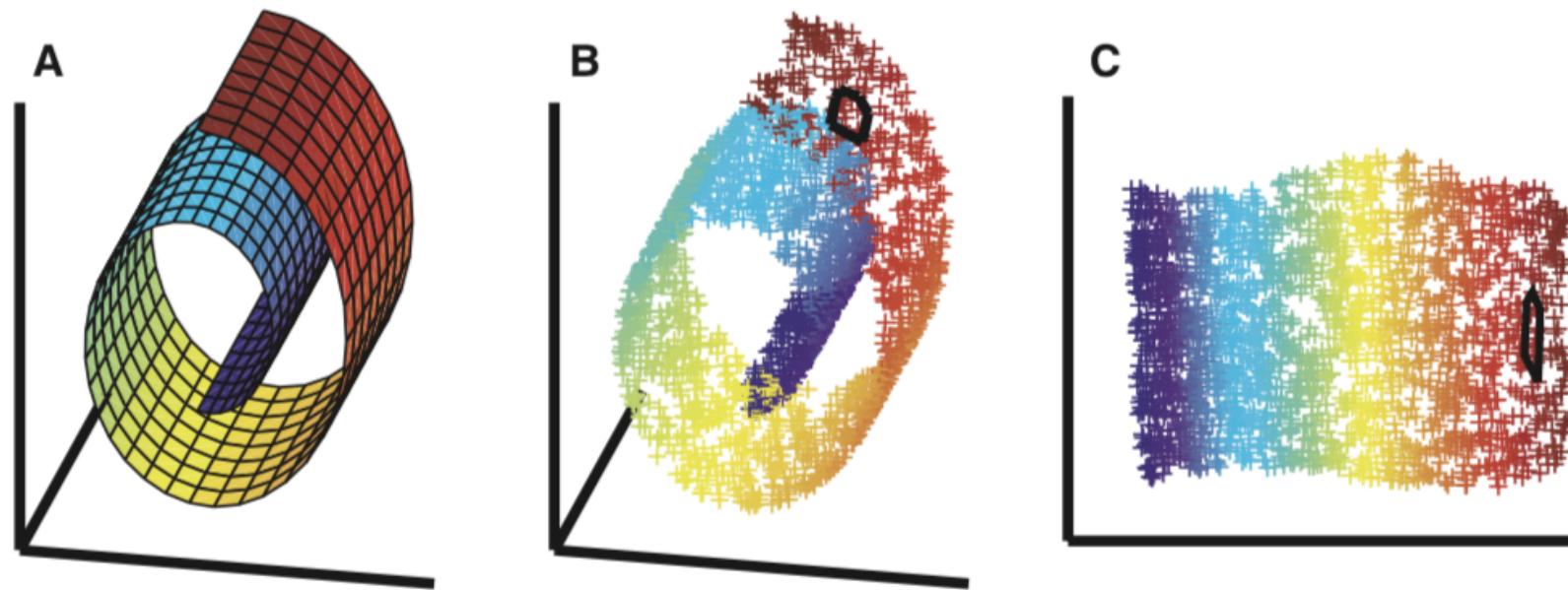
This is partly because PCA relies on Euclidean distance as opposed to **geodesic distance**: the distance between two points *along* the manifold **if it was first unrolled** (see p.45).

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

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

(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 :

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

over the set of $m \times q$ matrices 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 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 ;
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 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} , but 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

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

where $L = (I - \mathbf{Z})^\top (I - \mathbf{Z})$ 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 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 L .

Laplacian Eigenmaps

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

As with LLE, 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:

- 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 their definition of L .

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.

Semidefinite Embedding

Semidefinite embeddings (SDE) involve learning the kernel

$$K(\mathbf{x}, \mathbf{z}) = \Phi(\mathbf{x})^\top \Phi(\mathbf{z})$$

from the data before applying the kernel PCA transformation Φ (**semidefinite programming**).

The distances and angles between observations and their neighbours are preserved under transformations by Φ :

$$\|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\|^2 = \|\mathbf{x}_i - \mathbf{x}_j\|^2,$$

for all $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^n$.

In terms of the kernel matrix, this constraint can be written as

$$K(\mathbf{x}_i, \mathbf{x}_i) - 2K(\mathbf{x}_i, \mathbf{x}_j) + K(\mathbf{x}_j, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|^2,$$

for all $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^n$.

By adding an objective function to maximize $\text{Tr}(K)$, that is, the variance of the observations in the learned feature space, SDE constructs a semidefinite program for learning the **kernel matrix**

$$K = (K_{i,j})_{i,j=1}^N = (K(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1}^N,$$

from which kernel PCA can proceed.

Unified Framework

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

- **LLE** – if λ_{\max} is the largest eigenvalue of $L = (I - \mathbf{Z})^\top(I - \mathbf{Z})$, then $K_{\text{LLE}} = \lambda_{\max}I - L$;
- **LE** – same, but with $L = D - W$, 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 ,

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

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

t -SNE

Some of the new manifold learning techniques do not fit neatly in the kernel PCA framework: Uniform Manifold Approximation and Projection (see Report) and the **T -distributed stochastic neighbour embedding** (t -SNE).

For a dataset $\{\mathbf{x}_i\}_{i=1}^N \subseteq \mathbb{R}^n$, the latter involves calculating probabilities

$$p_{i,j} = \frac{1}{2N} \left\{ \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)} + \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_j^2)}{\sum_{k \neq j} \exp(-\|\mathbf{x}_j - \mathbf{x}_k\|^2/2\sigma_j^2)} \right\},$$

which are proportional to the similarity of points in \mathbb{R}^n for all i, j (p_{ii} is set to 0 for all i).

The first component in the similarity metric measures how likely it is that \mathbf{x}_i would choose \mathbf{x}_j as its neighbour $\sim N(\mathbf{x}_i, \sigma_i^2)$. The bandwidths σ_i are selected to be smaller in denser data areas.

The lower-d manifold $\{\mathbf{y}_i\}_{i=1}^N \subseteq \mathcal{M} \subseteq \mathbb{R}^d$ is selected to preserve the similarities $p_{i,j}$ as much as possible, by building the (reduced) probabilities

$$q_{i,j} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq i} (1 + \|\mathbf{y}_i - \mathbf{y}_k\|^2)^{-1}}$$

for all i, j (note the asymmetry) and minimizing the **Kullback-Leibler divergence** of Q from P over possible coordinates $\{\mathbf{y}_i\}_{i=1}^N$:

$$\text{KL}(P||Q) = \sum_{i \neq j} p_{i,j} \log \frac{p_{i,j}}{q_{i,j}}.$$

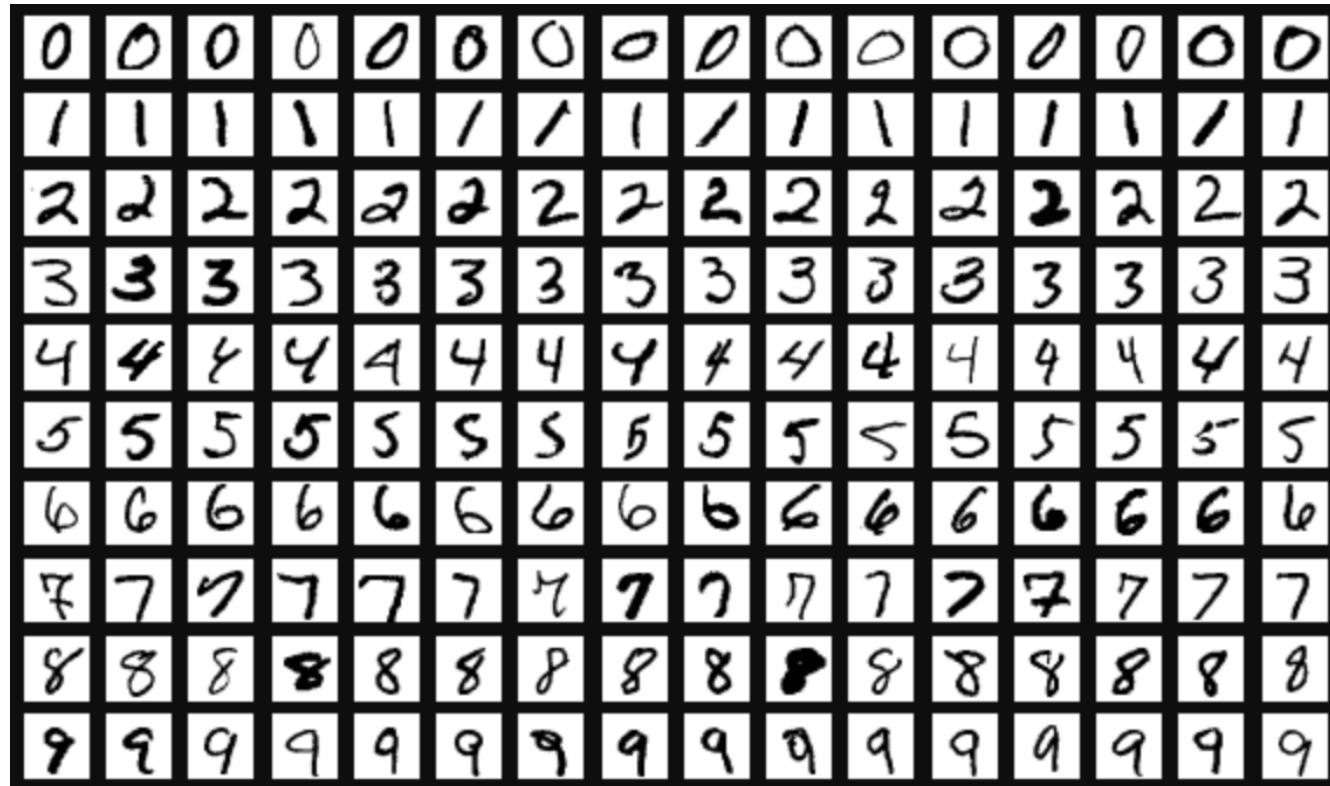
MNIST Example

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

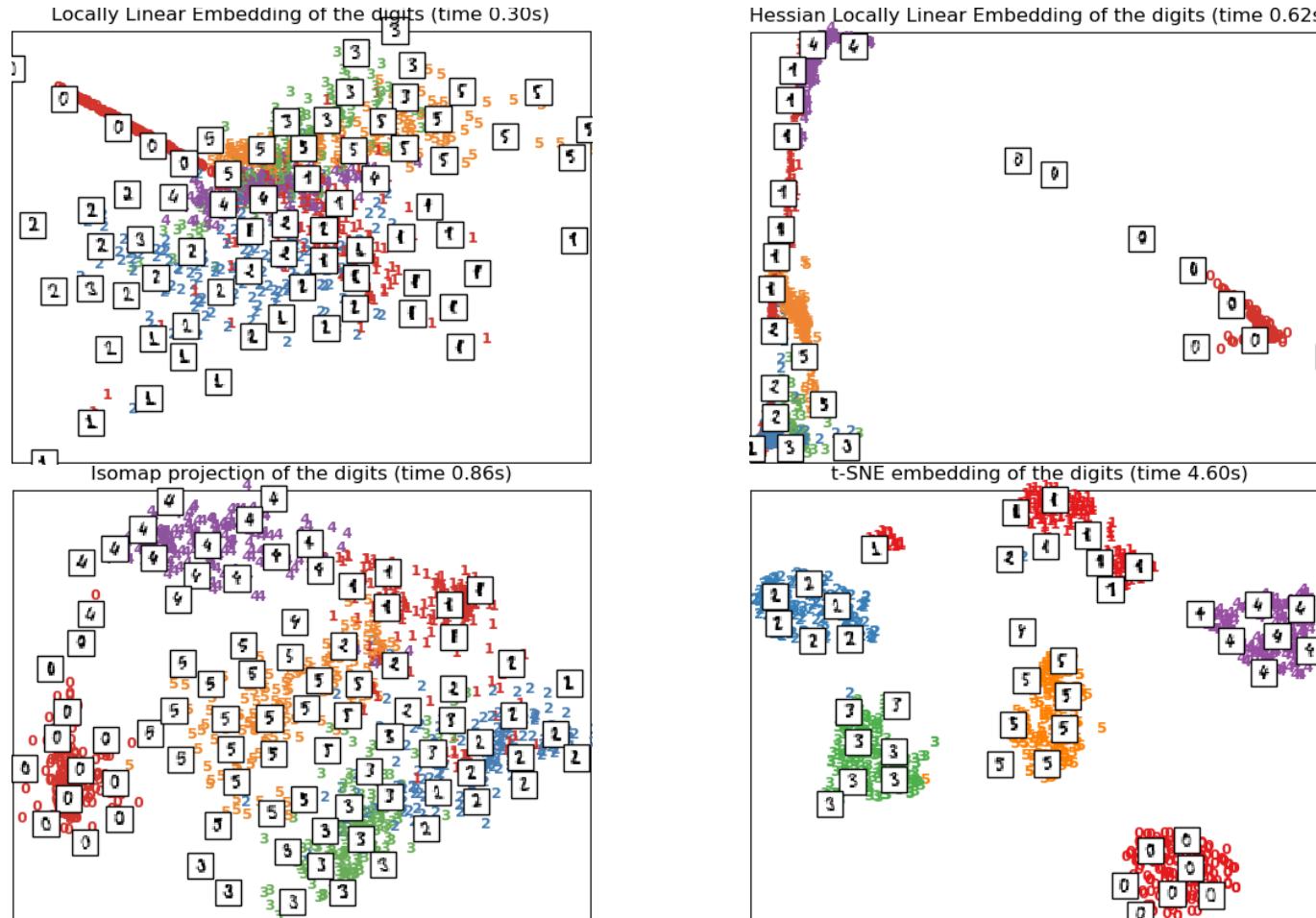
The results for 4 of those are shown on p.63.

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.

4.2 – 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

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

(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 is larger than the number of observations.

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.

4.2.1 – Filter Methods

Filter methods evaluate features without resorting to the use of 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

The **Pearson correlation coefficient** (PCC) quantifies the linear relationship between two continuous 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 only retain those 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 only retain 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 this 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).

Summary statistics for a sample of 2144 individuals are shown on p. 75.

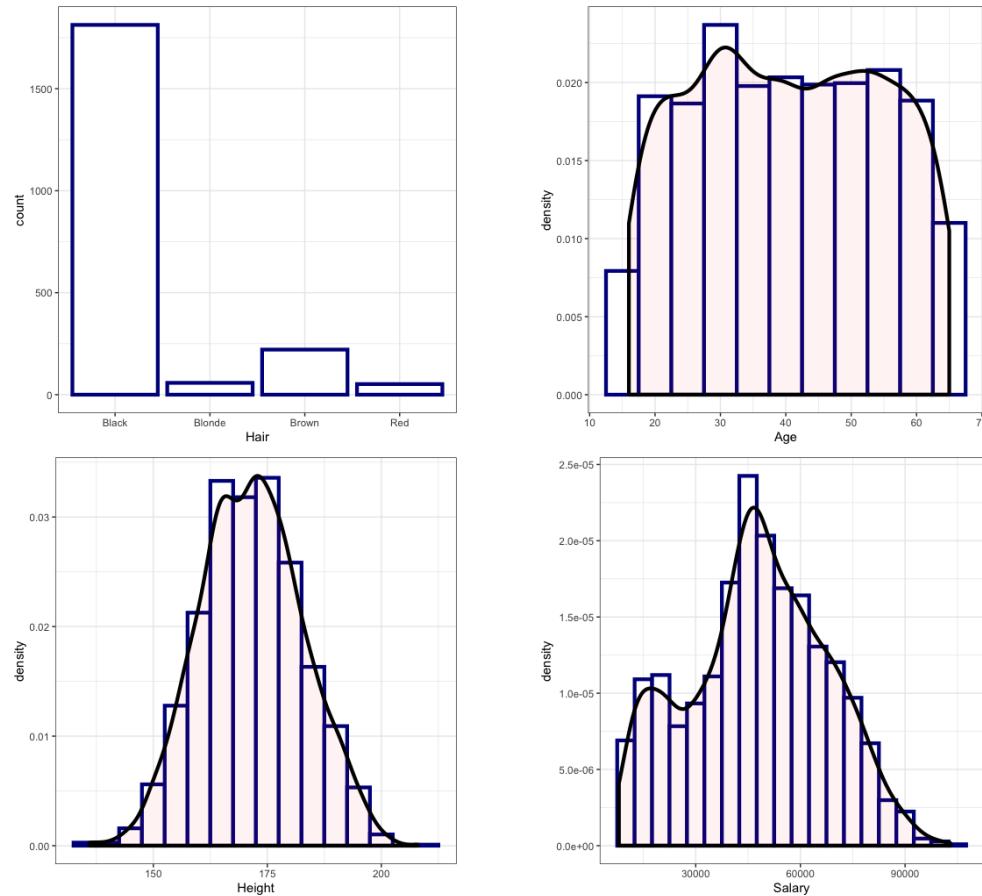
In a general population, the distribution of salaries, say, is likely to be fairly haphazard. It might be hard to explain why, specifically, it has the shape that it does (see p. 76).

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	Hair				Total		
		Female	Male	MIN	16	136	8000	Deciles	Black	Blonde	Brown	Red			
Black	1813	Female	1083	Q1	29	164	34000	1	193	7	25	8	233		
Blonde	58	Male	1061	MED	40	172	48000	2	180	9	25	8	222		
Brown	221		2144	Q3	53	179	61250	3	180	4	30	6	220		
Red	52			MAX	65	208	103000	4	187	4	14	3	208		
	2144			MEAN	40.4	171.6	47674.4	5	182	8	26	8	224		
				STDEV	14.2	11.2	19710.3	6	185	5	18	1	209		
				SKEW	0.01	0.06	-0.04	7	183	5	18	5	211		
								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										Gender			
Salary Deciles	1	2	3	4	5	6	7	8	9	10	Total	Deciles	Female	Male	Total
1	203	30									233	1	144	89	233
2	40	133	39	1	1	5	1				222	2	117	105	222
3		26	73	21	24	15	11	4	14	32	220	3	198	22	220
4		6	26	50	27	17	13	14	24	31	208	4	169	39	208
5		5	31	44	31	18	16	25	26	28	224	5	158	66	224
6		3	21	39	34	27	16	21	33	15	209	6	128	81	209
7		7	40	33	36	23	31	19	22	211		7	95	116	211
8		2	30	38	36	28	33	20	30	217		8	54	163	217
9			2	19	54	30	39	36	23	203		9	18	185	203
10			2	12	41	32	57	47	6	197		10	2	195	197
Total	243	203	199	229	219	249	170	224	219	189	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.

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

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

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

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

$$\begin{aligned} H(X_1|Y) &= - \int p(Y) \left\{ \sum_{\text{colour}} p(\text{colour}|Y) \log p(\text{colour}|Y) \right\} dY \\ H(X_2|Y) &= - \iint p(Y) p(\text{age}|Y) \log p(\text{age}|Y) d\text{age} dY \\ H(X_3|Y) &= - \iint p(Y) \int p(\text{ht}|Y) \log p(\text{ht}|Y) d\text{ht} dY \\ H(X_4|Y) &= - \int p(Y) \left\{ \sum_{\text{gender}} p(\text{gender---Y}) \log p(\text{gender---Y}) \right\} dY \end{aligned}$$

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:

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

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

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

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

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

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

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

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

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 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.

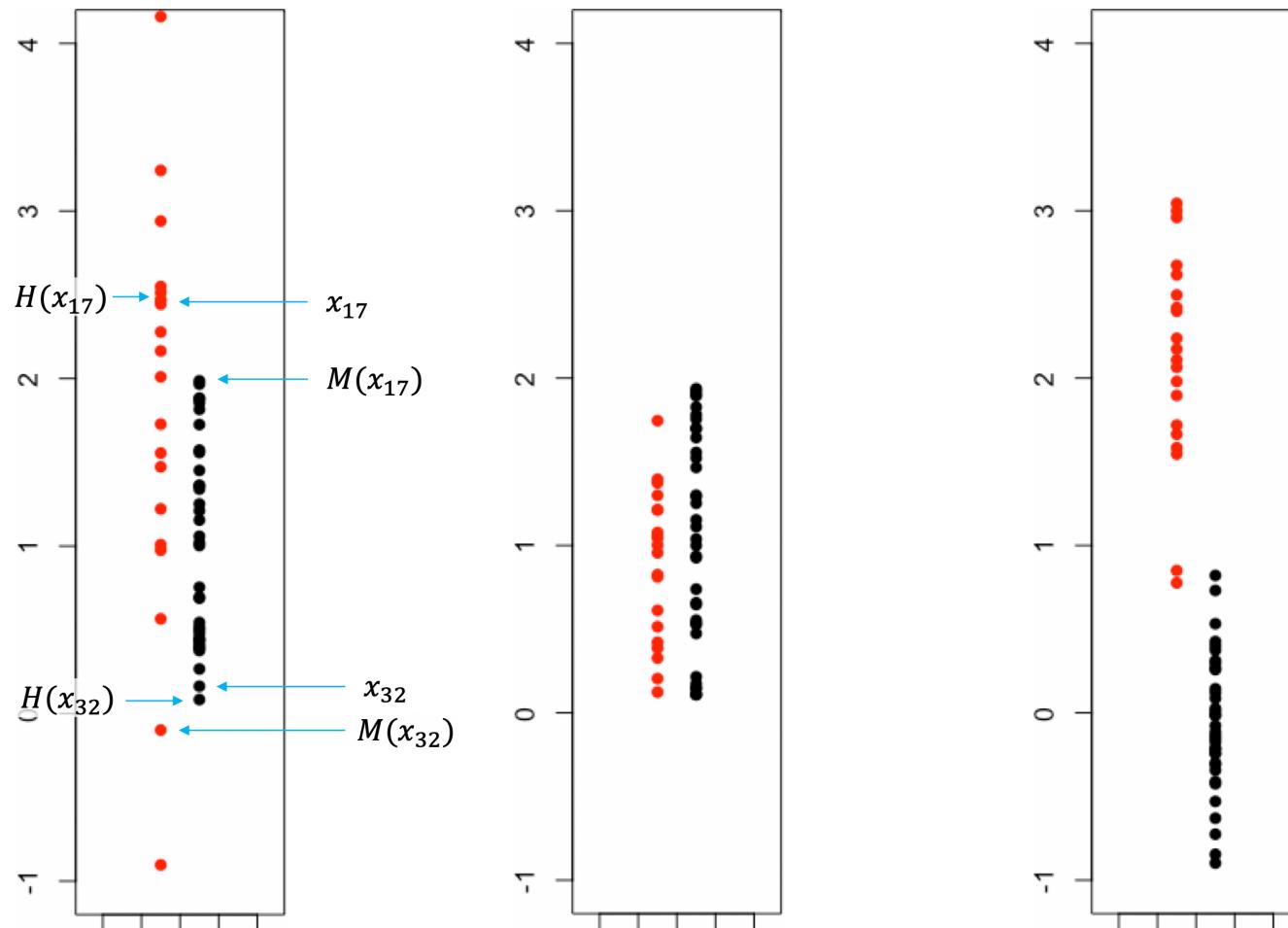
For each feature i and observation n , two values are selected:

- 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 a threshold τ .

Relief is noise-tolerant and robust to interactions; its effectiveness decreases with small N .

There are variants to accommodate potential feature interactions of multi-class problems.

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

4.2.2 – 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.

Wrapper 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.

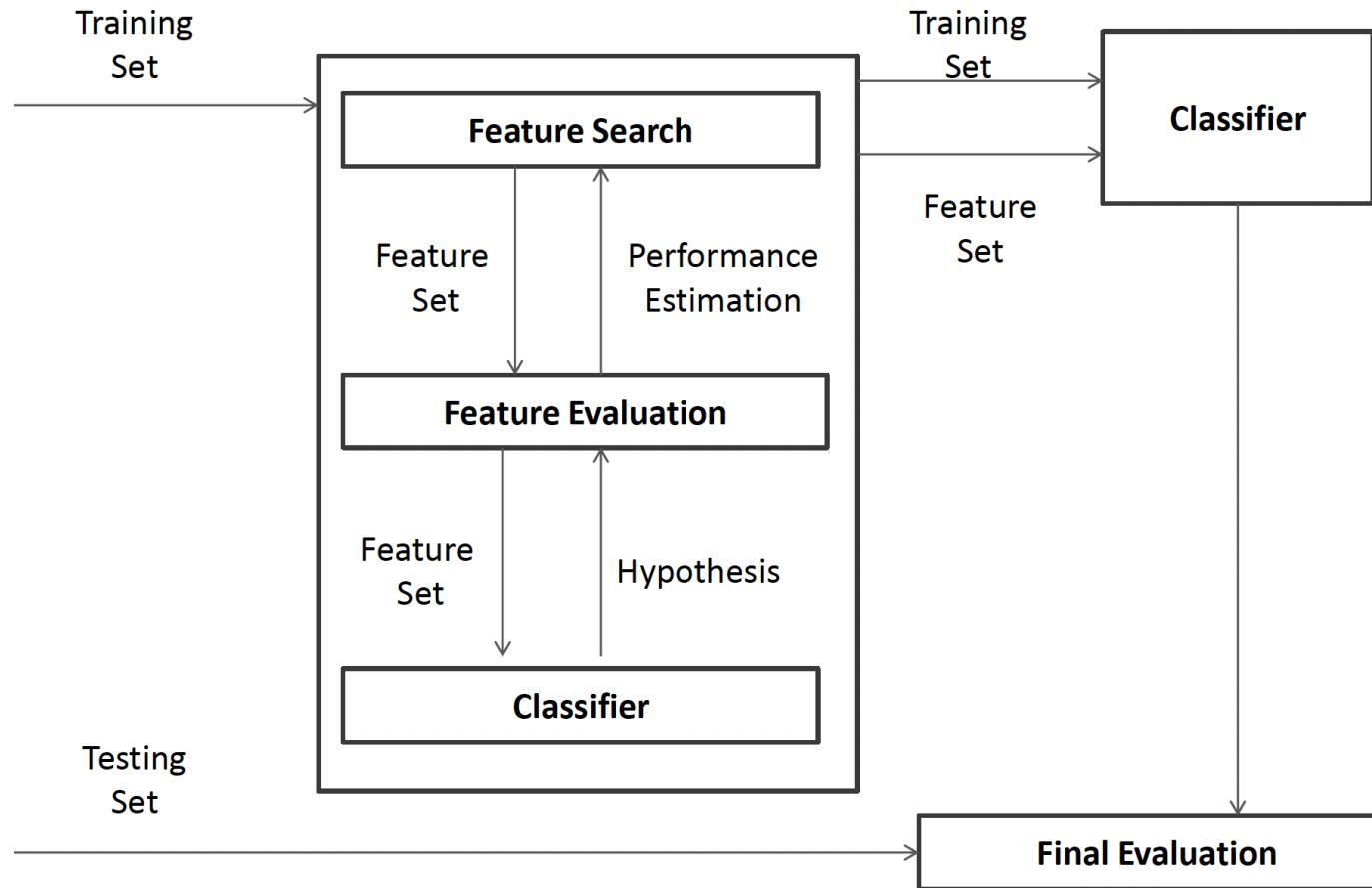
Wrapper methods iterate over the following steps, until an “optimal” set of features is identified:

- select a feature subset, and
- evaluate the performance of the selected feature subset.

The search ends when the desired quality is reached (adjusted R^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).

4.2.3 – 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** – more information on this very important approach to performance evaluation is available in *ISLR* (James, et al).

Stepwise Selection Methods Limitations:

- the tests are biased, since they are all based on the same data;
- the adjusted R^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 not usually done, and
- it's a classic example of *p*-hacking.

4.2.4 – Regularization (Embedded) Methods

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

Let $\mathbf{X} \in \mathbb{M}_{N,p}$ be the **centered** and **scaled** training matrix and let \mathbf{y} be the target output vector; the j^{th} ordinary least square (OLS) coefficient is

$$\hat{\beta}_{\text{LS},j} = [(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}]_j.$$

Set a threshold $\lambda > 0$, whose actually value depends on the training dataset (in practice, good values can be determined by cross-validation).

By construction, $\hat{\beta}_{\text{LS}}$ is the exact solution to the OLS problem

$$\hat{\beta}_{\text{LS}} = \arg_{\beta} \min \{ \| \mathbf{y} - \mathbf{X}\beta \|_2^2 \}.$$

There is **no restriction** on the values taken by the coefficients $\hat{\beta}_{\text{LS},j}$;

$|\hat{\beta}_{\text{LS},j}|$ large $\implies X_j$ **plays an important role** in predicting Y .

This observation forms the basis of a series of useful OLS variants.

Ridge regression (RR) provides a way to **regularize** the OLS regression coefficients, by penalizing solutions with large coefficient magnitudes.

If, in spite of this, the magnitude of a specific coefficient is “large,” then it must have **great relevance** in predicting the target variable.

The problem consists in solving a modified version of the OLS scenario:

$$\hat{\beta}_{\text{RR}} = \arg_{\beta} \min \{ \| \mathbf{y} - \mathbf{X}\beta \|_2^2 + N\lambda \|\beta\|_2^2 \}.$$

Usually, solving the RR problem requires the use of numerical methods and of cross-validation to determine the optimal λ .

For **orthonormal covariates** ($\mathbf{X}^\top \mathbf{X} = I_p$), however, the **ridge coefficients** can be expressed in terms of the OLS coefficients:

$$\hat{\beta}_{\text{RR},j} = \frac{\hat{\beta}_{\text{LS},j}}{1 + N\lambda}.$$

Regression with best subset selection (BS) uses a different penalty term, which effectively sets some of the coefficients to 0, which could be used to select the features with non-zero coefficients.

The problem consists in solving a modified version of the OLS scenario:

$$\hat{\beta}_{\text{BS}} = \arg_{\beta} \min \{ \| \mathbf{y} - \mathbf{X}\beta \|_2^2 + N\lambda \|\beta\|_0 \}, \quad \|\beta\|_0 = \sum_j \text{sgn}(|\beta_j|).$$

For orthonormal covariates, the **best subset** coefficients can be expressed in terms of the OLS coefficients:

$$\hat{\beta}_{\text{BS},j} = \begin{cases} 0 & \text{if } |\hat{\beta}_{\text{LS},j}| < \sqrt{N\lambda} \\ \hat{\beta}_{\text{LS},j} & \text{if } |\hat{\beta}_{\text{LS},j}| \geq \sqrt{N\lambda} \end{cases}$$

For the **LASSO** problem

$$\hat{\beta}_{\text{BS}} = \arg_{\beta} \min \{ \| \mathbf{y} - \mathbf{X}\beta \|_2^2 + N\lambda \|\beta\|_1 \},$$

the penalty effectively yields coefficients combining the properties of RR and BS, usually selecting no more than one feature per group of highly correlated variables.

For orthonormal covariates, the LASSO coefficients can be expressed in term of the OLS coefficients:

$$\hat{\beta}_{\text{L},j} = \hat{\beta}_{\text{LS},j} \cdot \max \left(0, 1 - \frac{N\lambda}{|\hat{\beta}_{\text{LS},j}|} \right).$$

Other penalty functions provides various extensions: elastic nets; group, fused and adaptive lassos; bridge regression, etc.

Regularization can be achieved for general models as well.

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

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

\mathbf{W}^* solves the **general regularization** 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} .

4.2.5 – Supervised and Unsupervised Methods

Feature selection methods are usually categorised as filter, wrapper, or embedded, but they can also be categorised as **supervised** or **unsupervised** methods.

Feature selection methods are supervised if the labels are incorporated into the feature reduction process, otherwise they are unsupervised.

In unsupervised 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**, supervised 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 multiple feature selection methods agree on a core set of features, that provides some model-independent support for the relevance of that set of features to the prediction task at hand.

4.3 – Advanced Topics

When used appropriately, the approaches to feature selection and dimension reduction methods presented in the last two sections provide a **solid toolkit** to help mitigate the effects of the curse of dimensionality.

These are, for the most part, rather straightforward.

However, an increase in conceptual complexity can lead to insights that are out of reach by more direct approaches.

In the accompanying report, we discuss 3 additional methods that are decidedly more sophisticated, from a mathematical perspective: **singular value decomposition**, **spectral feature selection**, and **uniform manifold approximation and projection**.