DMI

Unsupervised Learning:
Dimensionality reduction

Data Mining and Data integration in Biomedicine
Master in Bioinformatics

Janet Piñero
Medbioinformatics Solutions SL
2025-2026

Outline

DMI

What is dimensionality reduction

Feature selection methods

PCA

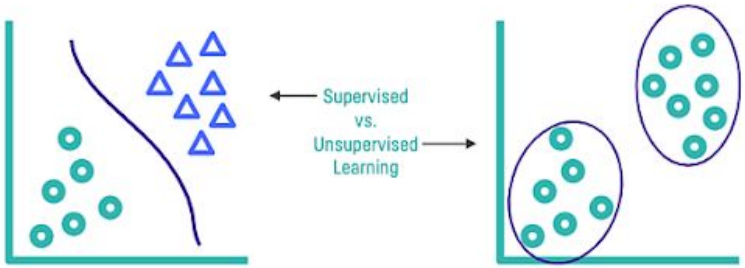
Other techniques for dimensionality reduction

Hands on session

Summary

Bibliography

Unsupervised vs supervised learning algorithms



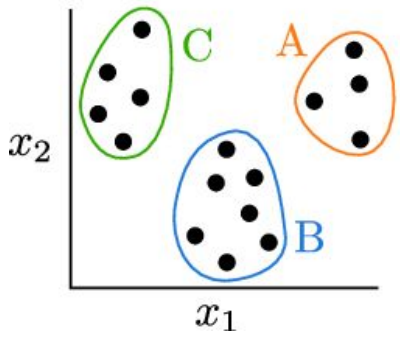
"the algorithm is taught by example" The algorithm is trained using data that is "labeled." It means some data is already tagged with the correct answer.

"The algorithm is made to learn by itself." The algorithm uses data that is not labeled. The machine must be able to classify the data without any prior information about the data.

Unsupervised Learning

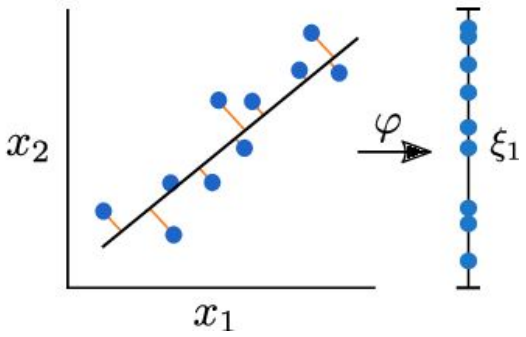
DMI

Clustering



the goal is to find homogeneous subgroups within the data
the grouping is based on distance between observations.

Dimensionality Reduction



The goal is to identify patterns in the data, to reduce the initial number of features or variables to a smaller set of variables, while retaining variation

Dimensionality Reduction

DMI

- Dimensionality reduction refers to techniques that reduce the number of input variables in a dataset.
- Data reduction goal is to better manage, understand and visualize data while at the same time it reduces memory usage and the time needed for the execution of data mining and machine learning algorithms.
- More input features often make a predictive modeling task more challenging to model, more generally referred to as the curse of dimensionality.

Applying dimensionality reduction techniques to a data set creates a data set with the same number of observations but less variables, whilst maintaining the general "flavor" of the data set.

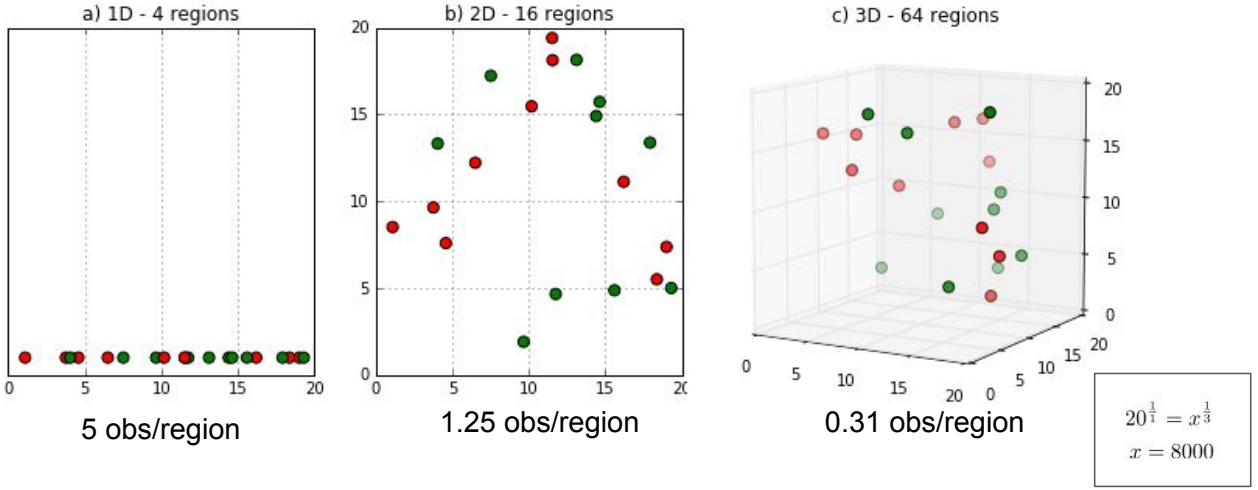
Dimensionality Reduction

DMI

Curse of dimensionality

- When dimensionality increases, data becomes increasingly **sparse**
- Density and distance between points, which is critical to clustering, outlier analysis, becomes less meaningful
- The possible combinations of subspaces will grow exponentially
- The more features we have, the more number of samples we will need to have all combinations of feature values well represented in our sample.

The curse of dimensionality

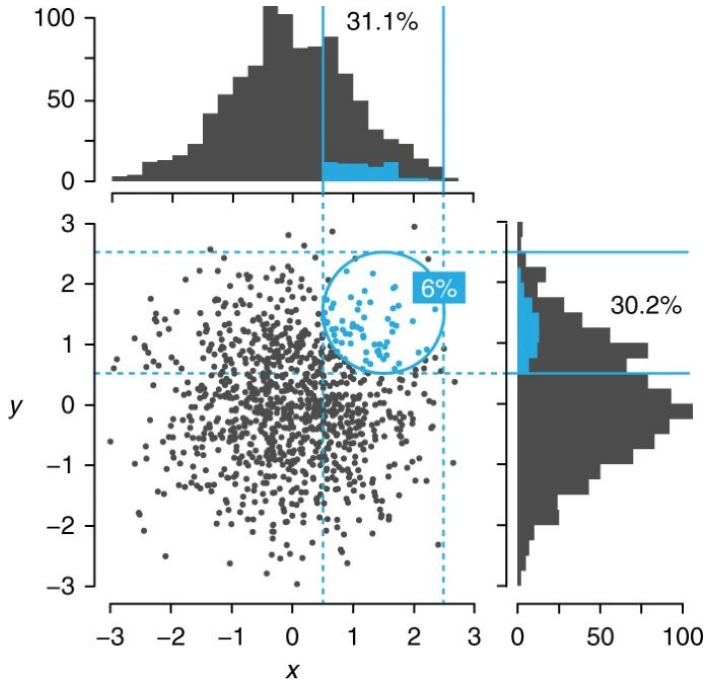


The curse of dimensionality – as the number of dimensions increases, the number of regions grows exponentially

The curse of dimensionality

Among 1,000 (x, y) points in which both x and y are normally distributed with a mean of 0 and s.d. σ = 1, only 6% fall within σ of (x, y) = (1.5, 1.5) (blue circle). However, when the data are projected into a lower dimension—shown by histograms—about 30% of the points (all bins within blue solid lines) are within σ of 1.5. Blue bins in histograms correspond to the blue points.

https://doi.org/10.1038/s41592-018-0019-x

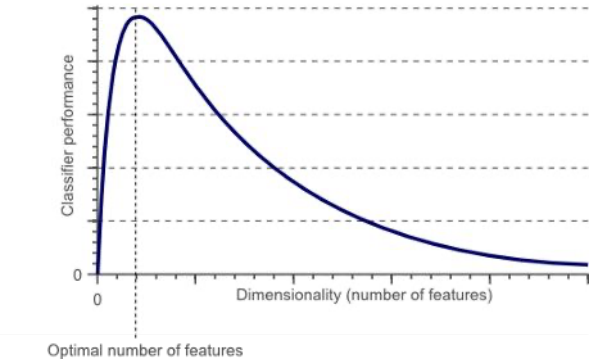


Dimensionality Reduction

DMI

Curse of dimensionality

- As the number of features increases, the model becomes more complex.
- The more the number of features, the more the chances of overfitting.



A machine learning model that is trained on a large number of features, gets increasingly dependent on the data it was trained on and in turn overfitted, resulting in poor performance on real data, beating its purpose.

Dimensionality Reduction

DMI

Advantages of dimensionality reduction

- Avoid the curse of dimensionality
- Decrease collinearity
- Help eliminate irrelevant features and reduce noise
- Reduce time and space required in data mining, also for EDA
- Allow easier visualization
- This becomes more important when the number of features are very large.
- You need not to use every feature at your disposal for creating an algorithm.
- You can assist your algorithm by feeding in only those features that are really important.
- It enables the machine learning algorithm to train faster, reduces the complexity and makes it easier to interpret.
- It improves the accuracy of a model if the right subset is chosen.
- It reduces overfitting.

Dimensionality Reduction Techniques

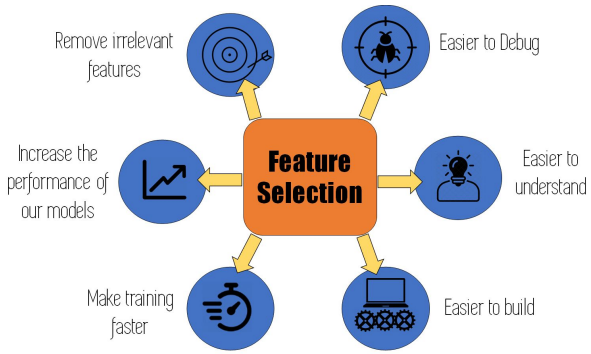
DMI

- **Feature selection**: Find a subset of the original variables (or features, attributes). Primarily focused on removing **non-informative** or **redundant** predictors from the model.
    - Advantages: simple to implement, makes our data set small, including only variables in which we are interested, reduces overfitting and training time.
    - Disadvantages: we might lose some information from the variables which we dropped.
- **Feature extraction**: Transform the data in the high-dimensional space to a space of fewer dimensions
    - Advantages: preserves the most structure of your original data set.
    - Disadvantage - the newly created variables loose their interpretation.

Feature selection

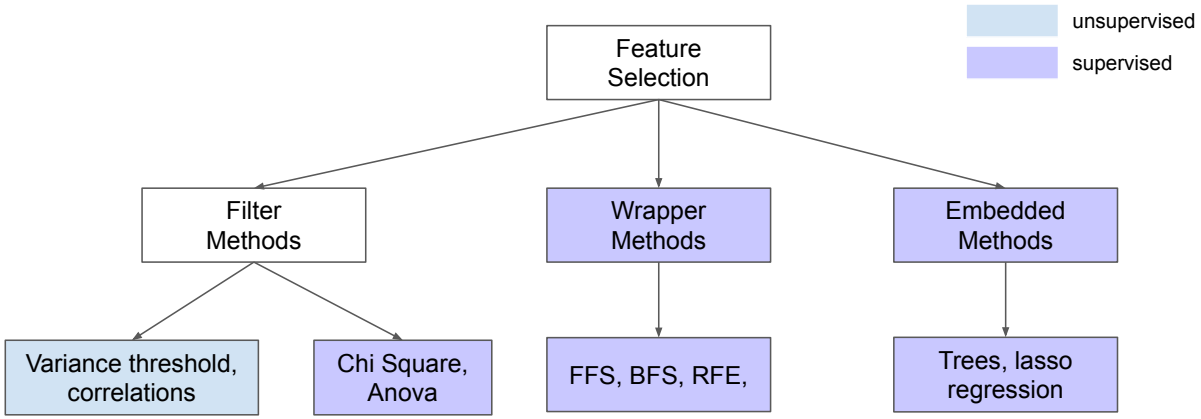
DMI

- Can be done manually or programmatically
  - Manually
    - domain-specific, requires proper domain knowledge
    - Heatmaps that show the correlation between features is a good idea
  - Programmatically
    - Filtering-based methods
    - Wrapper-based methods
    - Embedded methods



Types of Feature Selection Methods

DMI



Types of Supervised Feature Selection Methods

DMI

Filter Methods

- Compare each of the predictors against the outcome variable, and calculate a metric of how much the outcome varies with the predictor.
- Generally used as a preprocessing step
- Independent of any machine learning algorithm
- Can be univariate (Chi-Squared test) or multivariate (correlation-based feature selection)
- Common filter methods are correlation metrics (Pearson, Spearman, Distance), Chi-Squared test, Anova, Fisher's Score

Types of Supervised Feature Selection Methods

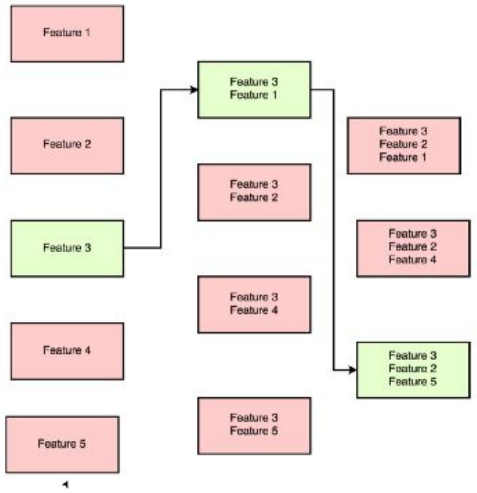
DMI

Wrapper Methods

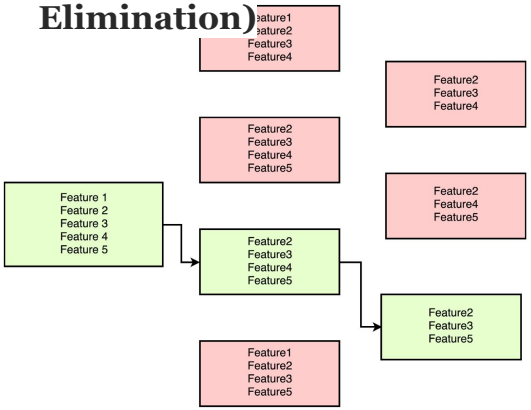
- Rather than using a single statistic to estimate feature importance, models are iteratively trained with different combinations of predictor variables. Eventually, the combination of predictors that gives the best performing model is chosen.
- The selection criterion of feature subset is determined by the performance of the classification model, which is strongly dependent on the specific classification method used.
- Deterministic wrapper feature selection methods either start with no features (forward-selection) or with all features included in the model (backward-selection) and iteratively refine the set of chosen features according to some model quality measures.
- Examples: Sequential forward selection, Recursive Feature Elimination, Forward Selection, Backward elimination

Feature selection methods: wrapper

Forward Stepwise selection



Backward Stepwise Selection (~Recursive Feature Elimination)



Recursive Feature Elimination

DMI

In caret,

```
set.seed(7) # ensure the results are repeatable
library(mlbench) # load the libraries
library(caret)
data(PimaIndiansDiabetes) # load the data
# define the control using a random forest selection function
control <- rfeControl(functions=rfFuncs, method="cv", number=10)
# run the RFE algorithm
results <- rfe(x = PimaIndiansDiabetes[,1:8],
                    Y = PimaIndiansDiabetes[,9],
                    sizes=c(1:8),
                     rfeControl=control)
print(results) # summarize the results
predictors(results) # list the chosen features
plot(results, type=c("g", "o")) # plot the results
```

- x, a matrix or data frame of predictor variables
- y, a vector (numeric or factor) of outcomes
- sizes, a integer vector for the specific subset sizes that should be tested (which need not to include ncol(x))
- rfeControl, a list of options that can be used to specify the model and the methods for prediction, ranking etc.

Types of Supervised Feature Selection Methods

DMI

Embedded Methods

- Feature selection is built into the classifier, feature selection and model tuning are done jointly. algorithms that have their own built-in feature selection methods
- The feature subset generation and evaluation is embedded as an integral part of the classification algorithm.
- Examples: Decision trees, and random forests, lasso and ridge regression, etc

Advantages of Embedded Methods:

- They take into consideration the interaction of features like wrapper methods do.
- They are faster like filter methods.
- They are more accurate than filter methods.
- They find the feature subset for the algorithm being trained.
- They are much less prone to overfitting.

Feature extraction

DMI

Create new attributes (features) that can capture the important information in a data set more effectively than the original ones

Three general methodologies

- Attribute extraction (natural language processing)
- Domain-specific
- Mapping data to new space (dimensionality reduction)

E.g., Fourier transformation, wavelet transformation, manifold approaches (not covered)

Dimensionality Reduction Methods

DMI

Linear methods ():

- **PCA** (Principal Component Analysis): unsupervised method that projects the data onto directions of maximum variance. The resulting new features, called principal components, are linear combinations of the original features, ordered by the amount of variance they explain. (Pearson, 1901).
- **LDA** (Linear Discriminant Analysis): supervised method that projects data to maximize class separability. It places examples from the same class close together while maximizing the distance between different classes (Fisher, 1936).

Non-linear Dimensionality Reduction Methods

- **MDS (Multidimensional scaling)** : A technique used for analyzing similarity or dissimilarity of data as distances in a geometric spaces. It projects data to a lower dimension such that data points that are close to each other (in terms if Euclidean distance) in the higher dimension are close in the lower dimension as well (Torgerson, 1952?).
- **t-SNE** (t-distributed Stochastic Neighbor Embedding): Computes the probability that pairs of data points in the high-dimensional space are related and then chooses a low-dimensional embedding which produce a similar distribution (van der Maaten, 2008).
- **UMAP** (Uniform Manifold Approximation and Projection): constructs a high-dimensional graph of the data and optimizes a lower-dimensional representation to preserve the local and global data structure (McInnes, Healy, and Melville J, 2018)

What is Principal Component Analysis

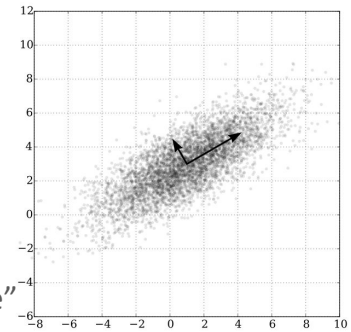
- Principal component analysis (PCA) is a mathematical algorithm that reduces the dimensionality of the data while retaining most of the variation in the data set.
- It accomplishes this reduction by identifying directions, called principal components, along which the variation in the data is maximal.
- By using a few components, each sample can be represented by relatively few numbers instead of by values for thousands of variables.
- It uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.
- Method: Find the eigenvectors of the covariance matrix, and these eigenvectors define the new space
- The original data are projected onto a much smaller space, resulting in dimensionality reduction
- Samples can then be plotted, making it possible to visually assess similarities and differences between samples and determine whether samples can be grouped.

https://www.nature.com/articles/nbt0308-303

PCA

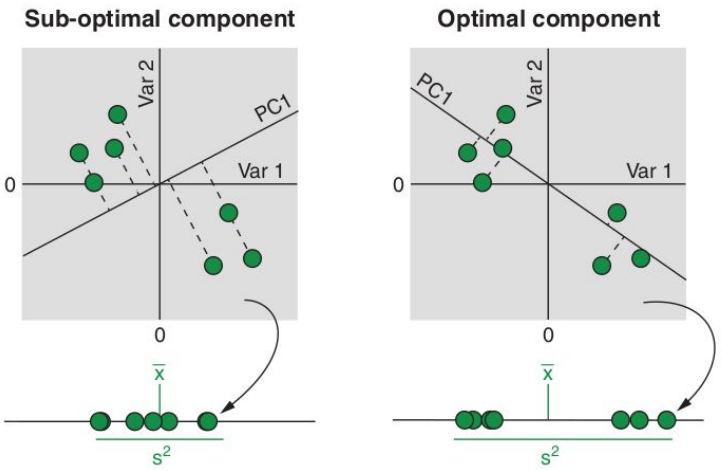
DMI

Given *N* data vectors from *n*-dimensions, find $k \leq n$ orthogonal vectors (*principal components*) best used to represent data



- Normalize input data: Each attribute falls within the same range
- Compute *k* orthonormal (unit) vectors, i.e., *principal components*
- Each input data (vector) is a linear combination of the *k* principal component vectors
- The principal components are sorted in order of decreasing "significance" or strength
- Since the components are sorted, the size of the data can be reduced by eliminating the *weak components*, i.e., those with low variance (i.e., using the strongest principal components, to reconstruct a good approximation of the original data)
- Works for numeric data only!

PCA: Two-Dimensional Representation



- It determines the direction of highest variability in the data. The larger the variability captured in first component, the larger the information captured by component.

- First principal component is a linear combination of original predictor variables which captures the maximum variance in the data set.

-The variance of the data along a particular principal component is called its Eigenvalue

- The first principal component results in a line which is closest to the data i.e. it minimizes the sum of squared distance between a data point and the line.

- No other component can have variability higher than first principal component.

PCA in R

DMI

We will use `prcomp` function for PCA. The `prcomp` provides five output variables

```
model = prcomp(data, scale=TRUE)
model$sdev
model$rotation
model$center
model$scale
model$x
```

- sdev – This defines the standard deviation of projected points on PC1, PC2, PC3 and PC3. As expected, the standard deviation of projected point is in decreasing order from PC1 to PC4.
- rotation – This defines the principal components axis. Here there are p principal components as there are p input features (loadings).
- Center/Scale – These are mean and standard deviation of input features in original feature space (without any transformation).
- The principal components **scores** are stored in the x list item of our results.

PCA in R

DMI

```
model = prcomp(input, scale=TRUE)
model$sdev
model$rotation
model$center
model$scale
```

- The *center* and *scale* components correspond to the means and standard deviations of the variables that were used for scaling prior to implementing PCA.
- The *rotation* matrix provides the principal component loadings; each column of `pca_result$rotation` contains the corresponding principal component loading vector.
- There are in general $\min(n-1,p)$ informative principal components in a data set with n observations and p variables.
- the principal components scores from our results as these are stored in the *x* list item of our results.
- The *sdev* is the standard deviation of each principal component.

PCA using prcomp: example Iris data

DMI

```
pca_result <- prcomp(iris[,-5] , scale = TRUE)
summary(pca_result)
```

```
> iris[1:5, ]
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
```

```
names(pca_result)
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

pca_result$center

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
5.843333	3.057333	3.758000	1.199333

Means of the variables that were used for scaling prior to implementing PCA.

pca_result$scale

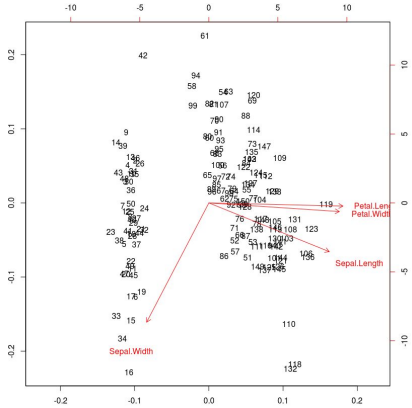
Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
0.8280661	0.4358663	1.7652982	0.7622377

standard deviations of the variables that were used for scaling prior to implementing PCA.

PCA using prcomp: example Iris data

The *rotation* matrix provides the principal component **loadings**; each column of `pca_result$rotation` contains the corresponding principal component loading vector.

```
pca_result$rotation
##                PC1        PC2        PC3        PC4
## Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
## Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
## Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
## Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
```



It tells us how much each variable contributes to each principal component.

For example, Sepal.Width contributes little to PC1 but makes up much of PC2.

Often it is helpful to plot the rotation matrix as arrows using the function biplot

The Sign in PCs is arbitrary

PCA using prcomp: example Iris data

DMI

observations

```
The rotated data are available as pca$x:
head(pca$x)
##             PC1        PC2         PC3          PC4
## [1,] -2.257141 -0.4784238  0.12727962  0.024087508
## [2,] -2.074013  0.6718827  0.23382552  0.102662845
## [3,] -2.356335  0.3407664 -0.04405390  0.028282305
## [4,] -2.291707  0.5953999 -0.09098530 -0.065735340
## [5,] -2.381863 -0.6446757 -0.01568565 -0.035802870
## [6,] -2.068701 -1.4842053 -0.02687825  0.006586116
```

the original observations projected onto the principal components

```
pca_result$sdev
[1] 1.7083611 0.9560494 0.3830886 0.1439265
```

As we can see, these data don't tell us to which species which observation belongs

PCA using prcomp: example Iris data

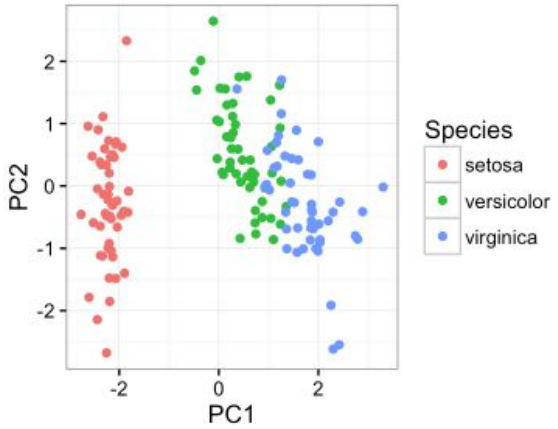
DMI

```
add species information back into PCA data
pca_data <- data.frame(pca$x, Species=iris$Species)
  head(pca_data)
```

```
##        PC1        PC2        PC3         PC4 Species
## 1 -2.257141 -0.4784238  0.12727962  0.024087508  setosa
## 2 -2.074013  0.6718827  0.23382552  0.102662845  setosa
## 3 -2.356335  0.3407664 -0.04405390  0.028282305  setosa
## 4 -2.291707  0.5953999 -0.09098530 -0.065735340  setosa
## 5 -2.381863 -0.6446757 -0.01568565 -0.035802870  setosa
  ## 6 -2.068701 -1.4842053 -0.02687825  0.006586116  setosa
```

Now we can plot as usual:

```
ggplot(pca_data, aes(x=PC1, y=PC2, color=Species)) + geom_point()
```



PCA using prcomp: example Iris data

Finally, we want to look at the percent variance explained.

The prcomp() function gives us standard deviations (stored in pca$sdev). To convert them into percent variance explained, we square them and then divide by the sum over all squared standard deviations:

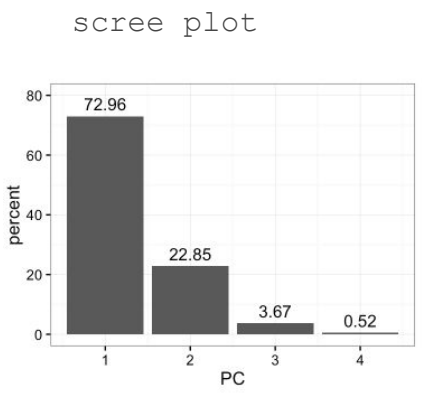
```
percent <- 100*pca$sdev^2/sum(pca$sdev^2)
percent
```

```
## [1] 72.9624454 22.8507618  3.6689219  0.5178709
```

The PC1 explains 73% of the variance, the second 23%, the third 4%,the last 0.5%.

We can visualize these results nicely in a bar plot:

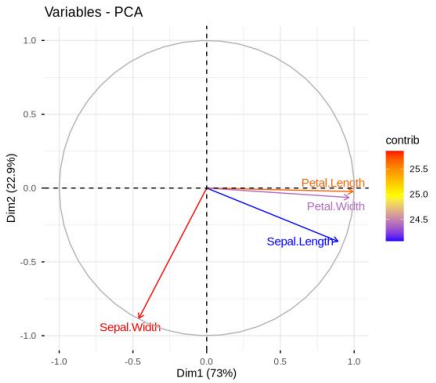
```
perc_data <- data.frame(percent=percent, PC=1:length(percent))
ggplot(perc_data, aes(x=PC, y=percent)) +
 geom_bar(stat="identity") +
 geom_text(aes(label=round(percent, 2)), size=4, vjust=-.5) +
  ylim(0, 80)
```



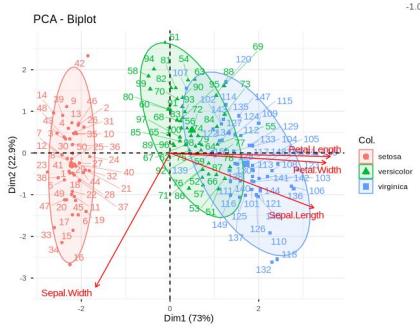
PCA using prcomp: example Iris data

```
library(factoextra)

fviz_pca_var( pca_result,
 col.var = "contrib",
 gradient.cols = c("blue", "yellow", "red"),
   repel = TRUE )
```




```
fviz_pca_biplot(
  pca_result,
  col.var = "red",
  col.ind = iris$Species,
  addEllipses = TRUE,
  repel = TRUE)
```



PCA

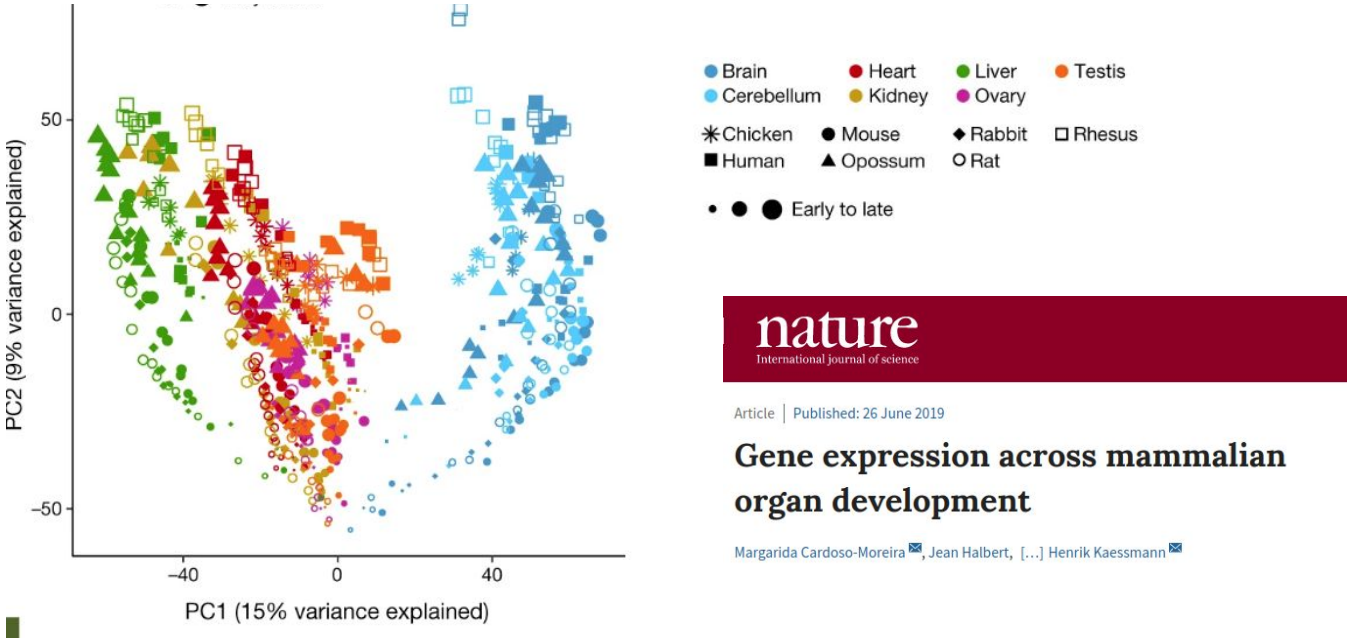
DMI

Some important point to note before using PCA:

- As PCA tries to find the linear combination of data and if the data in the dataset has non-linear relation then PCA will not work efficiently.
- Data should be normalized before performing PCA. PCA is sensitive to scaling of data as higher variance data will drive the principal component.
- PCA does not handle missing values
- Beware of categorical data: Categories are generally represented as factors, which are encoded as integer levels, and might give the impression that a distance between levels is a relevant measure (which it is not, unless the factors are ordered). Drop them or encode categories as binary dummy variables.

PCA

PCA based on gene expression for 7,696 1:1 orthologues across all species



nature
International journal of science

Article | Published: 26 June 2019

Gene expression across mammalian organ development

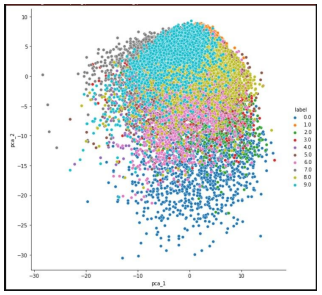
Margarida Cardoso-Moreira, Jean Halbert, […] Henrik Kaessmann

t-SNE & UMAP

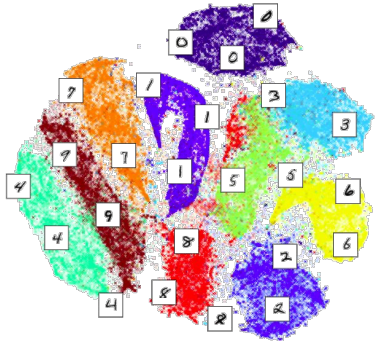
DMI

High dimensionality has always been a curse - it is extremely hard to make sense of, and requires a lot of work and domain knowledge to boil down to few dimensions without losing a lot of information.

PCA has long reigned the linear case, but two new(er) non-linear and powerful candidates are around: t-SNE and UMAP.



PCA



t-SNE

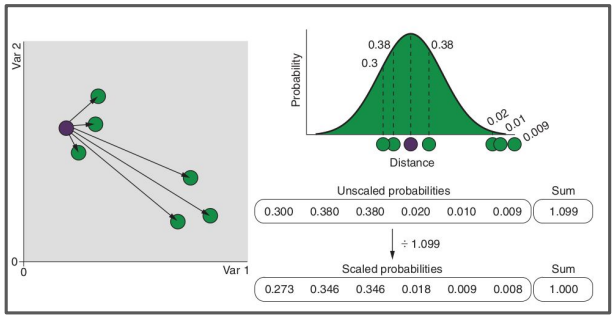
UMAP

t-SNE: t-Distributed Stochastic Neighbor Embedding

DMI

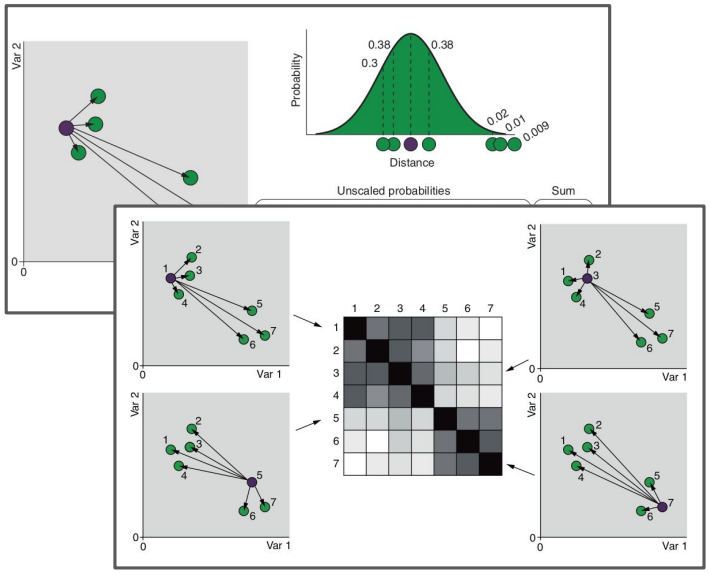
- t-SNE is a **non-linear** dimensionality reduction algorithm used for exploring high-dimensional data. It maps multi-dimensional data to two or more dimensions suitable for human observation.
- t-SNE is the dimensionality reduction which maps data in a higher dimensional space to that of a lower dimensional space just like PCA but uses a similarity measure like Euclidean distance to learn about discrepancies between pairs of data. While doing this, the local structures of data are preserved which is not the case of PCA.
- t-SNE often achieves better resolution than PCA

t-SNE: t-Distributed Stochastic Neighbor Embedding



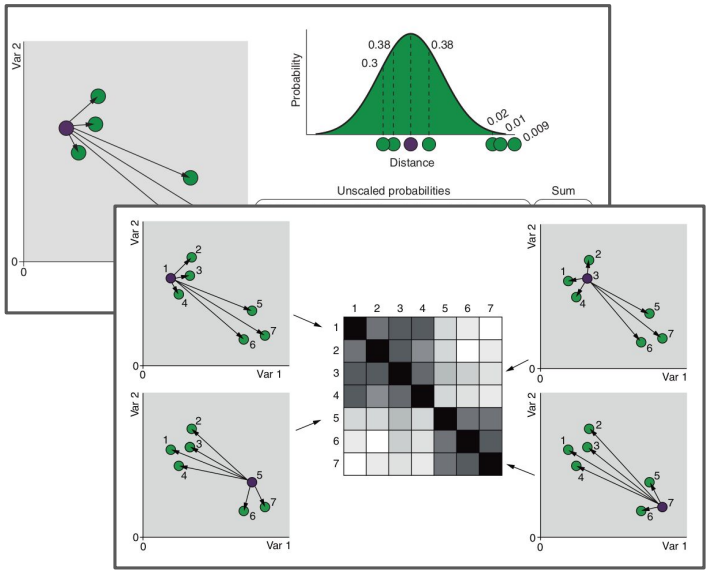
t-SNE measures the distance from each case to every other case, converted into a probability by fitting a normal distribution over the current case. These probabilities are scaled by dividing them by their sum, so that they add to 1.

t-SNE: t-Distributed Stochastic Neighbor Embedding



The scaled probabilities for each case are stored as a matrix of values. This is visualized here as a heatmap: the closer two cases are, the darker the box is that represents their distance in the heatmap.

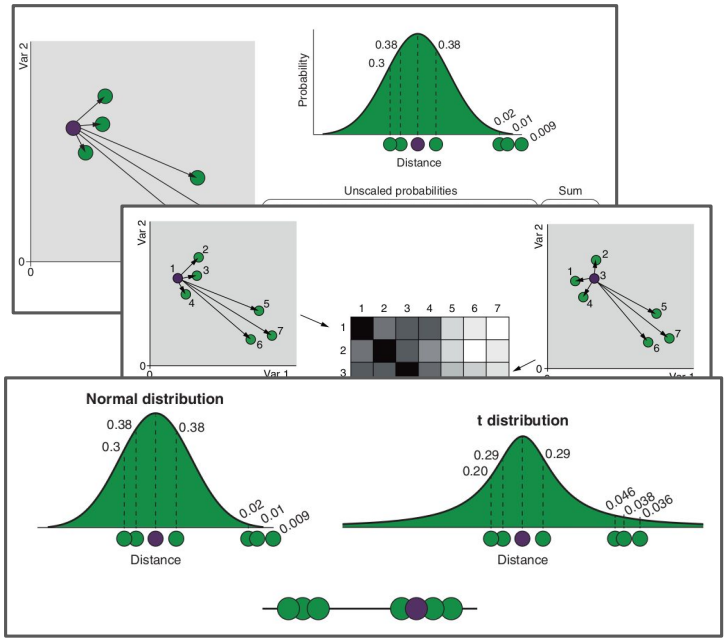
t-SNE: t-Distributed Stochastic Neighbor Embedding



The scaled probabilities for each case are stored as a matrix of values. This is visualized here as a heatmap: the closer two cases are, the darker the box is that represents their distance in the heatmap.

The next step in the t-SNE algorithm is to randomize the cases along (usually) two new axes (this is where the "stochastic" bit of the name comes from).

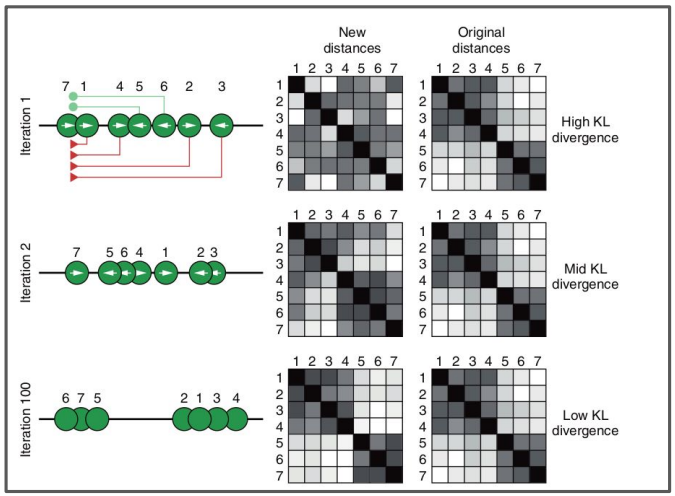
t-SNE: t-Distributed Stochastic Neighbor Embedding



When converting distances in the lower-dimensional representation into probabilities, t-SNE fits a Student's t distribution over the current case instead of a normal distribution.

The Student's t distribution has longer tails, meaning dissimilar cases are pushed further away to achieve the same probability as in the high-dimensional representation.

t-SNE: t-Distributed Stochastic Neighbor Embedding



Cases are randomly initialized over the new axes (one axis is shown here).
The probability matrix is computed for this axis, and the cases are shuffled around to make this matrix resemble the original, high-dimensional matrix by minimizing the Kullback-Leibler (KL) divergence.
During shuffling, cases are attracted toward cases that are similar to them (lines with circles) and repulsed away from cases that are dissimilar (lines with triangles).

t-SNE: t-Distributed Stochastic Neighbor Embedding

DMI

- t-SNE finds patterns in the data by identifying observed clusters based on similarity of data points with multiple features.
- It is not a clustering algorithm
- it maps the multidimensional data to a lower dimensional space, the input features are no longer identifiable. Thus you cannot make any inference based only on the output of t-SNE. So essentially it is mainly a data exploration and visualization technique.
- Because t-SNE is able to provide a 2D or 3D visual representation of high-dimensional data that preserves the original structure, we can use it during initial data exploration as visual check to see if there is some ‘order’ or some ‘pattern’ in the dataset. It can aid our intuition about what we think we know about the domain we are working in

t-SNE: t-Distributed Stochastic Neighbor Embedding

DMI

The perplexity parameter is crucial for t-SNE to work correctly – this parameter determines how the local and global aspects of the data are balanced.
A perplexity value between 30 and 50 is recommended.

According to the official documentation, perplexity is related to the importance of neighbors: "It is comparable with the number of nearest neighbors k that is employed in many manifold learners."
"Typical values for the perplexity range between 5 and 50"

t-SNE: t-Distributed Stochastic Neighbor Embedding

DMI

```
library(Rtsne)

tsne <- Rtsne(train[,-1], dims = 2, perplexity=30, verbose=TRUE, max_iter = 500)
```

dims - number of dimensions the data must be reduced to

perplexity—Says (loosely) how to balance attention between local and global aspects of your data. **Controls the width of the distributions used to convert distances into probabilities.** High values place more focus on global structure, whereas small values place more focus on local structure. Typical values lie in the range 5 to 50. The default value is 30.

max_iter—The maximum iterations allowed before computation stops. This will depend on your computational budget, but it's important to have enough iterations to reach convergence. The default value is 1,000.

t-SNE: t-Distributed Stochastic Neighbor Embedding

DMI

- t-SNE does not scale well for rapidly increasing sample sizes
- t-SNE does not preserve global data structure, meaning that only within cluster distances are meaningful while between cluster similarities are not guaranteed, therefore it is widely acknowledged that clustering on t-SNE is not a very good idea.
- t-SNE can practically only embed into 2 or 3 dimensions, i.e. only for visualization purposes, so it is hard to use t-SNE as a general dimension reduction technique in order to produce e.g. 10 or 50 components.

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

UMAP: uniform manifold approximation and projection

- ✓ nonlinear dimension-reduction algorithm published in 2018
- ✓ considerably faster than t-SNE
- ✓ preserves both local and global structure
- ✓ Deterministic algorithm
- ✓ UMAP constructs a high dimensional graph representation of the data then optimizes a low-dimensional graph to be as structurally similar as possible.
- ✓ New data can be projected

For more, see https://pair-code.github.io/understanding-umap/

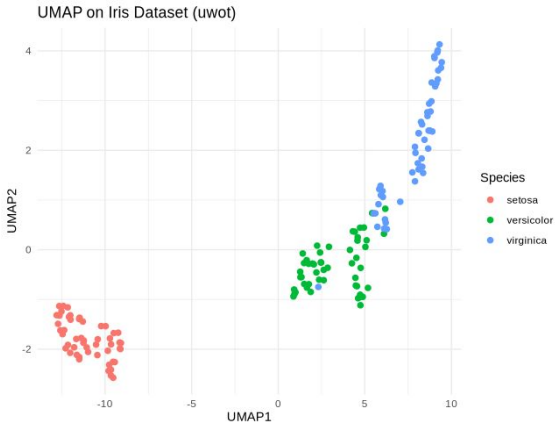
UMAP: uniform manifold approximation and projection

```
library(ggplot2)
library(uwot)
data(iris)
X <- iris[, 1:4]      # numeric features
y <- iris$Species
set.seed(42)

umap_result <- umap(   X,  n_neighbors = 15,
   min_dist = 0.1,  n_components = 2 )

umap_df <- data.frame(
    UMAP1 = umap_result[,1],
    UMAP2 = umap_result[,2],
    Species = y )

ggplot(umap_df, aes(x = UMAP1, y = UMAP2, color = Species)) +
    geom_point(size = 2) +
    theme_minimal() +   labs(title = "UMAP on Iris Dataset (uwot)")
```



The two most commonly used parameters: n_neighbors and min_dist, which are effectively used to control the balance between local and global structure in the final projection.

Strengths and weaknesses of t-SNE and UMAP

DMI

strengths

- ✓ They can learn nonlinear patterns in the data.
- ✓ They tend to separate clusters of cases better than PCA.
- ✓ UMAP can make predictions on new data.
- ✓ UMAP is computationally inexpensive.
- ✓ UMAP preserves both local and global distances.

https://plotly.com/r/t-sne-and-umap-projections/

Strengths and weaknesses of t-SNE and UMAP

DMI

weaknesses

- ✓ The new axes of t-SNE and UMAP are not directly interpretable in terms of the original variables.
- ✓ t-SNE cannot make predictions on new data (different result each time).
- ✓ t-SNE is computationally expensive.
- ✓ t-SNE doesn’t necessarily preserve global structure.
- ✓ They cannot handle categorical variables natively.

On your own: Ten quick tips for effective dimensionality reduction

Tip 1: Choose an appropriate method

Tip 2: Preprocess continuous and count input data

Tip 3: Handle categorical input data appropriately

Tip 4: Use embedding methods for reducing similarity and dissimilarity input data

Tip 5: Consciously decide on the number of dimensions to retain

Tip 6: Apply the correct aspect ratio for your visualizations

Tip 7: Understand the meaning of the new dimensions

Tip 8: Find the hidden signal

Tip 9: Take advantage of multidomain data

Tip 10: Check the robustness of your results and quantify uncertainties

https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1006907

Some final remarks

DMI

Such tools “can act as either a blessing or a curse in understanding the geometric and neighborhood structures of datasets,” … In some instances, t-SNE can create spurious clusters that mislead researchers. Both t-SNE and UMAP “perform beautifully in preserving local structure but struggle to preserve global structure.”

They offer little of value to a paper,...and the output from these tools is analytically intractable. “I sometimes call them an artistic rendition of the data,” he says of t-SNE and UMAP plots. They lack confidence measures and indications of how much uncertainty rests inside their visually neat data clusters.

For single-cell RNA sequencing (scRNA-seq) data, t-SNE and UMAP are often used to reduce data to two dimensions to enable plotting in papers and on slides. To highlight clusters, t-SNE and UMAP are preferred over PCA because high-dimensional datapoints that are close become “really close in the two final dimensions.” That leaves room to separate groups out. PCA is used first because it accelerates t-SNE and UMAP, which can be very slow with 20,000 dimensions, he says. Thus, many scRNA-seq analysis pipelines first reduce data dimensions with PCA to compress dimensions, say, to between 30 and 100. Then t-SNE or UMAP are run.

https://www.nature.com/articles/s41592-024-02301-x

Hands-on session

DMI

Go to this page and accept the assignment (if you have not done it yet!)

Some ideas for the hands-on:

http://uc-r.github.io/pca

http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical-guide/112-pca-principal-component-analysis-essentials/

Check these videos to know more about tSNE and UMAP

Deadline to submit the assignment: 15/03/2026

Bibliography

A Review of Feature Selection Methods for Machine Learning-Based Disease Risk Prediction

Dimension Reduction PCA, tSNE, UMAP

Chapter 12.2 Principal Components Analysis from An Introduction to Statistical Learning