ECS171: Machine Learning

L13: Decision trees and Principal Component Analysis

Instructor: Prof. Maike Sonnewald
TAs: Pu Sun & Devashree Kataria

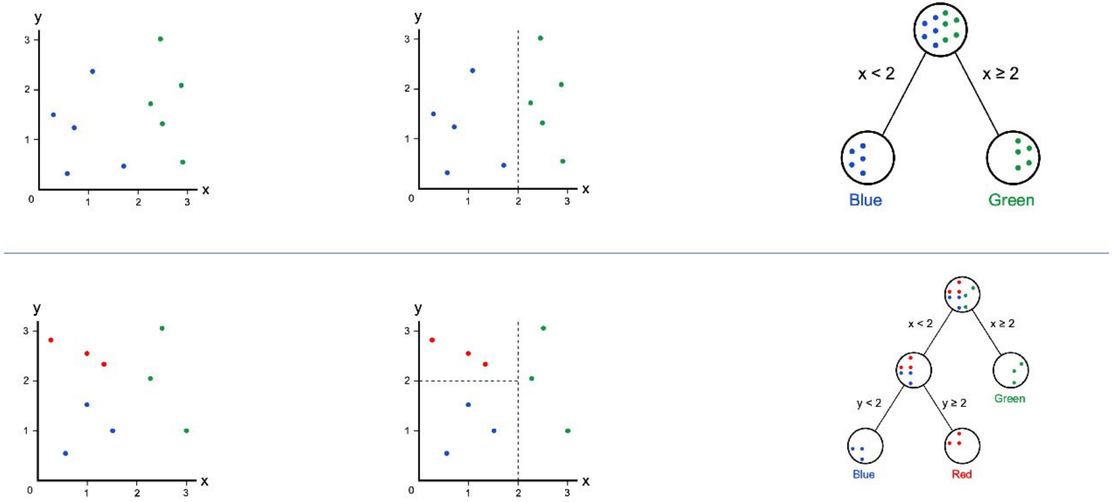
MOCO Amsterdam garden

Intended Learning Outcomes

- Describe how a Decision Tree is constructed and the relation to Bagging and Random Forests
- Describe a ‘Greedy Search’
- Describe and apply methods for component selection such as Entropy and Tree Optimisation such as Information Gain
- Describe what uses Principal Component Analysis has for data pre-processing and projection
- Apply the steps of a PCA on a simple example
    - Calculate the eigenvalues and eigenfunctions, percentage of variance and re-priented data
- Describe the importance of the assumption of linearity, and what that implies in terms of the preservation of global vs local structure in the data

Decision trees

- Partition the data to discern predictive patterns
- Key concepts:
    - Entropy
    - Information Gain
- Decision Trees are the foundation for several classical machine learning algorithms including:
    - Bagging
    - Random Forests
    - Boosted Decision trees



Decision trees

- Constructing a decision tree is recursive problem
  - Top-down approach
- Fast and accurate
- Invariant to scaling of inputs
  - Recall e.g. SVMs require the features be normalized prior to training the model
- Good for both classification and regression
- Constructed by divide and conquer, and a greedy search algorithm
- The core algorithm for building decision trees is called ID3 [by J. R. Quinlan]
- Decision trees are constructed using:
  - ‘Divide and Conquer’ algorithm recursively
  - Followed removing sections or ‘pruning’
- Pruning helps generalisation
  - Avoids overfitting

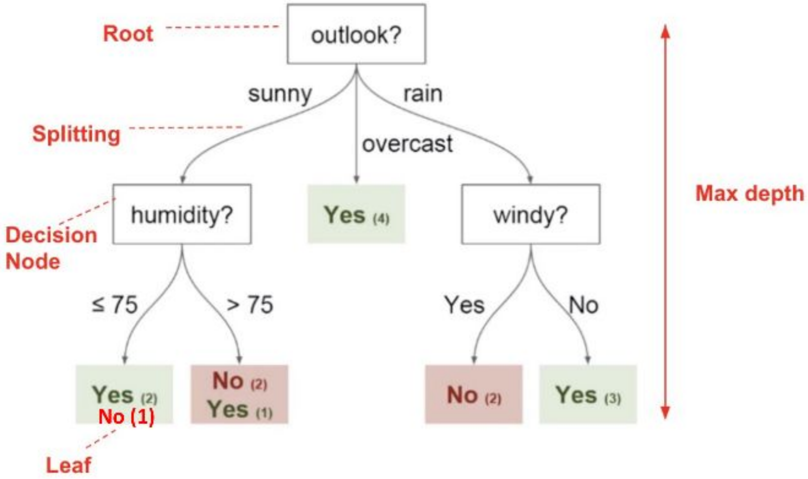
Construction of a Decision Tree

Overall approach:

1. Start with an attribute
2. Split into branches based on attribute values
3. Stop expanding the sub-branch once all samples in branch have the same class label
4. For every node at a branch, repeat step 1-3

- Challenges in constructing a decision tree:
    - Which attribute to select?
    - How to specify threshold for a numeric attribute?
    - How to keep the decision tree small?

Will we play a game?
Decision Tree diagram

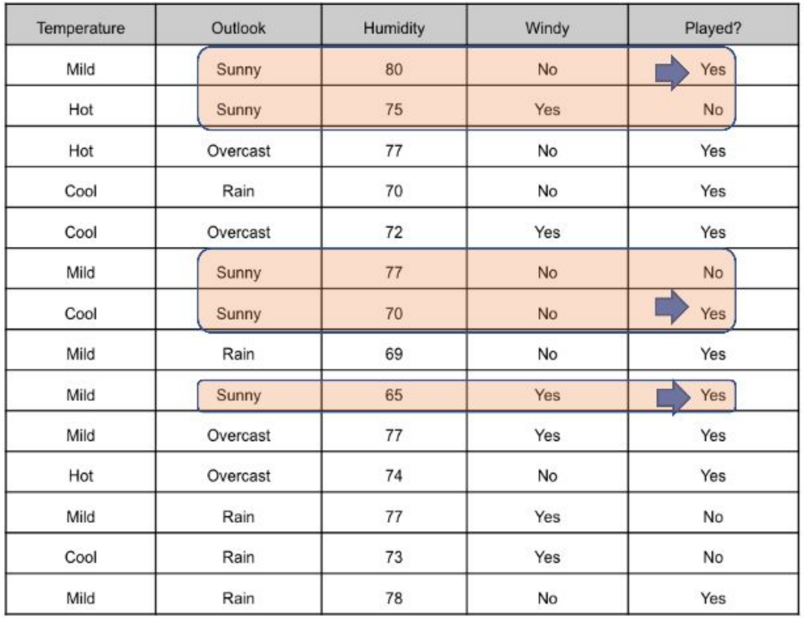


@Kdnuggets: Clare Liu, Fintech industry

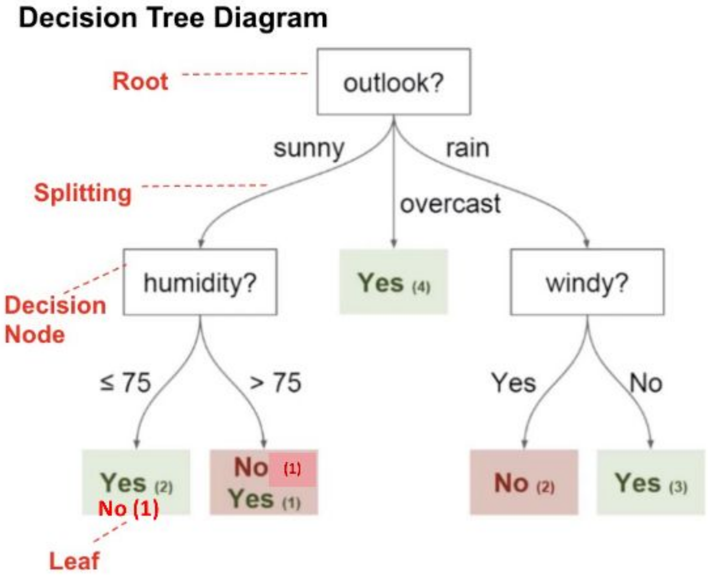
Recursive divide and conquer

Outlook='sunny' has 3 Yes and 2 No , so we can split

Dataset



Temperature	Outlook	Humidity	Windy	Played?
Mild	Sunny	80	No	Yes
Hot	Sunny	75	Yes	No
Hot	Overcast	77	No	Yes
Cool	Rain	70	No	Yes
Cool	Overcast	72	Yes	Yes
Mild	Sunny	77	No	No
Cool	Sunny	70	No	Yes
Mild	Rain	69	No	Yes
Mild	Sunny	65	Yes	Yes
Mild	Overcast	77	Yes	Yes
Hot	Overcast	74	No	Yes
Mild	Rain	77	Yes	No
Cool	Rain	73	Yes	No
Mild	Rain	78	No	Yes



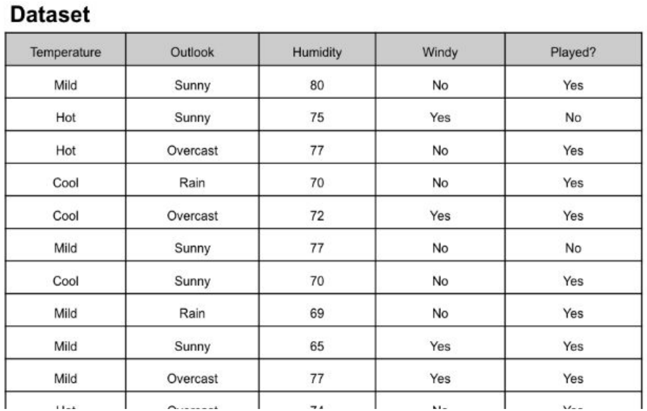
@Kdnuggets: Clare Liu, Fintech industry

Greedy Search

- Uses a measure of ‘purity’ for each node
- With this measure, the tree construction algorithm selects the attribute that generates the purest sub-nodes

Decision tree construction procedure:

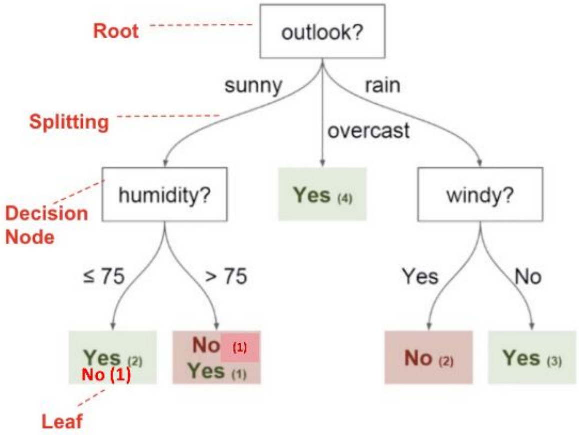
- Create a splitting rule
- Divide the data using the splitting rule
- Repeat
- Stop when final sets (leaves) are homogenous (i.e. same class)



Dataset

Temperature	Outlook	Humidity	Windy	Played?
Mild	Sunny	80	No	Yes
Hot	Sunny	75	Yes	No
Hot	Overcast	77	No	Yes
Cool	Rain	70	No	Yes
Cool	Overcast	72	Yes	Yes
Mild	Sunny	77	No	No
Cool	Sunny	70	No	Yes
Mild	Rain	69	No	Yes
Mild	Sunny	65	Yes	Yes
Mild	Overcast	77	Yes	Yes
Hot	Overcast	74	No	Yes

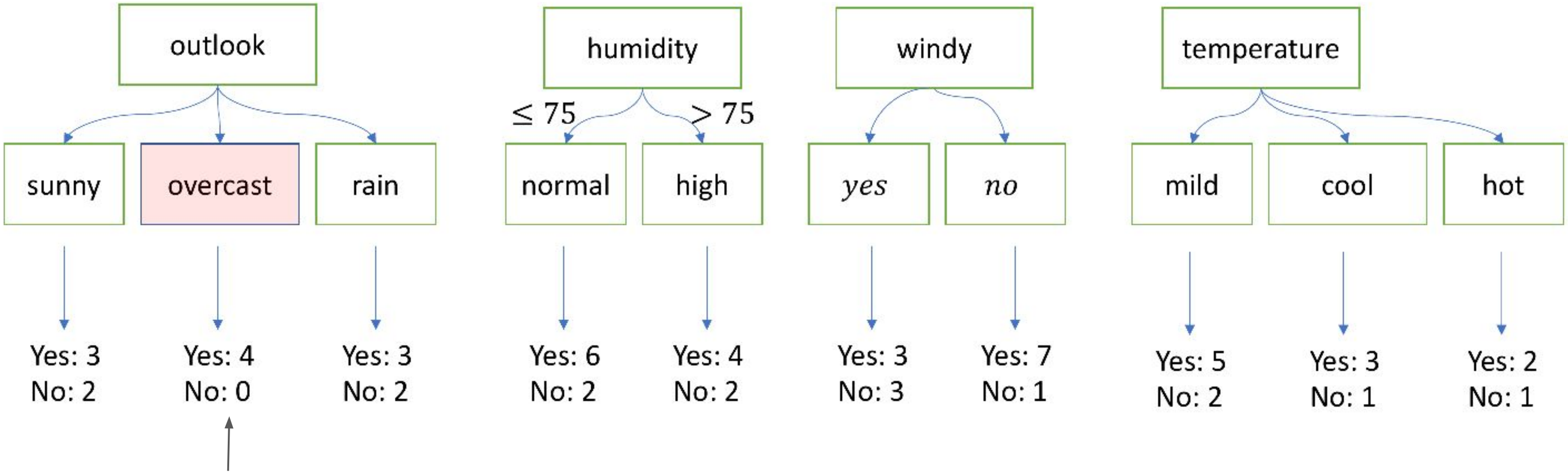
Decision Tree Diagram



@Kdnuggets: Clare Liu, Fintech industry

Attribute selection

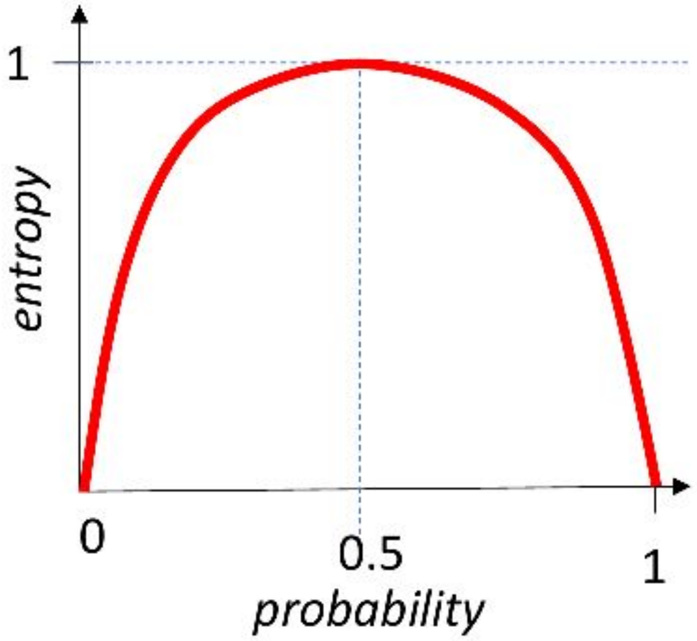
- Quantifying ‘purity’ is key to training
- We will cover concepts in the context of a ‘Greedy’ search: Entropy and Information



‘pure’

Entropy and Information are metrics for uncertainty

- Entropy is the measure of purity of a decision node in a decision tree
- Entropy is used to calculate the homogeneity of a decision node
    - Measured in units called ”bits”
    - Is ”0” if the sample in a decision node is homogeneous in terms of the class label
    - Is “1” if the sample is equally divided
- ‘Information Gain’: measure of the decrease in uncertainty obtained by splitting a dataset based on some *additional attribute*



Estimating Entropy

$$Entropy = -\sum_{i=1}^{c} p_i \log_2 p_i \quad ; s.t. \;\; c \in X$$

$X : dataset\ attributes$

$c_i : the\ i^{th}\ value\ for\ feature\ c$

$p_i : probability\ of\ the\ i^{th}\ value\ for\ feature\ c \quad (0 \leq p \leq 1)$

$0 \leq Entropy \leq 1$

Estimating Entropy

$$Entropy = -\sum_{i=1}^{c} p_i \log_2 p_i \quad ; s.t.\ \ c \in X$$

X : *dataset attributes*

c_i: *the* i^{th} *value for feature c*

p_i: *probability of the* i^{th} *value for feature c* $(0 \leq p \leq 1)$

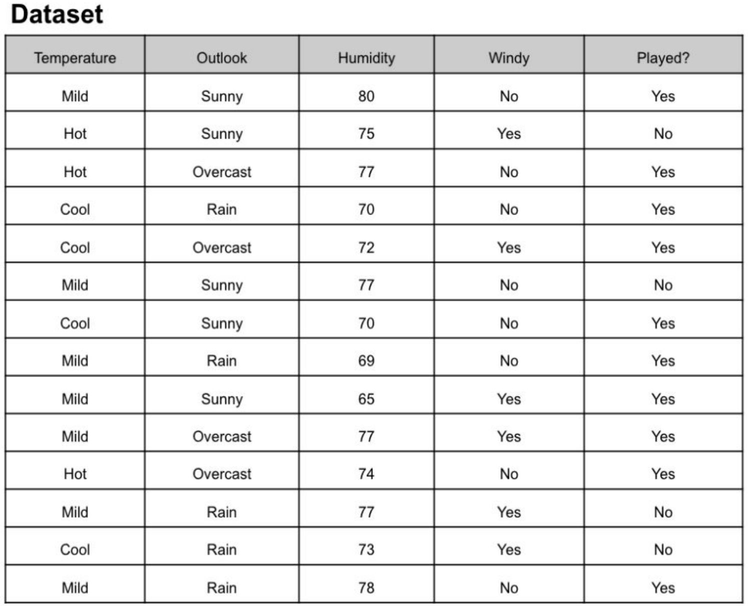
$0 \leq Entropy \leq 1$

- For the weather dataset,
- Dataset for variable “Played” contains:
  - 10 counts of ‘Yes’, and 4 counts of ‘No’
- Take i=1 is ‘Yes’, and i=2 is ‘No’
- Estimating the Entropy now becomes:

$$\begin{array}{l} \qquad\qquad\quad\ \ p_1 \qquad\quad p_2 \\ E(Played) = E(\ 4/14\ ,\ 10/14) = E(0.28\ ,\ 0.71) \end{array}$$

$$= -\ 0.28 \log_2 0.28\ \ -\ 0.71 \log_2 0.71$$

$$= 0.50 + 0.35 = 0.85 \text{ bits}$$



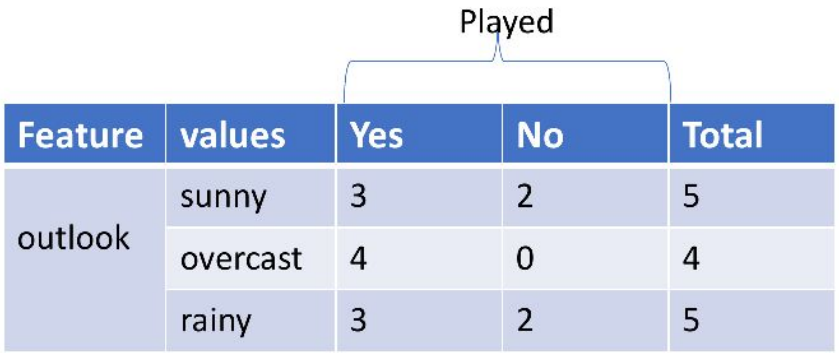
Dataset

Temperature	Outlook	Humidity	Windy	Played?
Mild	Sunny	80	No	Yes
Hot	Sunny	75	Yes	No
Hot	Overcast	77	No	Yes
Cool	Rain	70	No	Yes
Cool	Overcast	72	Yes	Yes
Mild	Sunny	77	No	No
Cool	Sunny	70	No	Yes
Mild	Rain	69	No	Yes
Mild	Sunny	65	Yes	Yes
Mild	Overcast	77	Yes	Yes
Hot	Overcast	74	No	Yes
Mild	Rain	77	Yes	No
Cool	Rain	73	Yes	No
Mild	Rain	78	No	Yes

@Kdnuggets: Clare Liu, Fintech industry

Measuring the Entropy of ‘Outlook’ variable

- Contingency table considering the dependency between “Played” and “Outlook”:

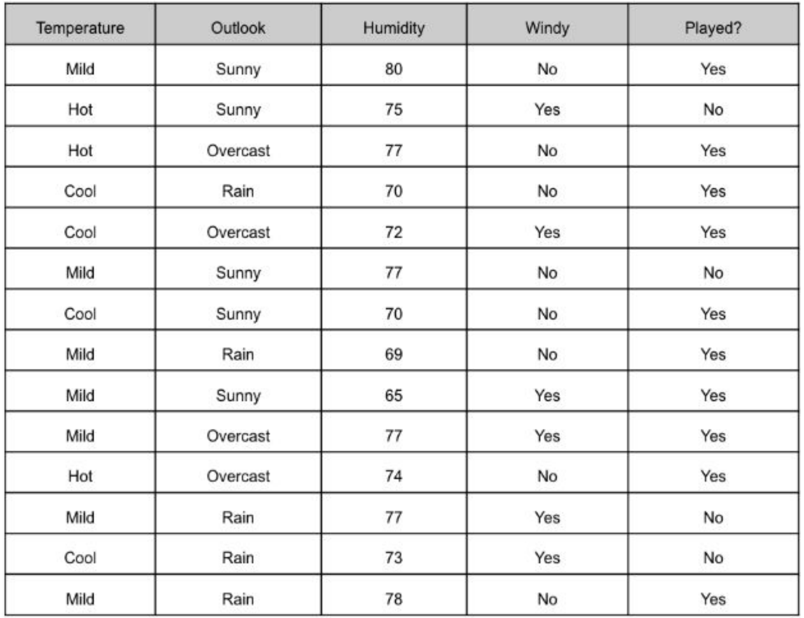


Played

Feature	values	Yes	No	Total
outlook	sunny	3	2	5
	overcast	4	0	4
	rainy	3	2	5

What is the entropy of “Outlook” wrt “Played” ?
E(Played, Outlook) = ?

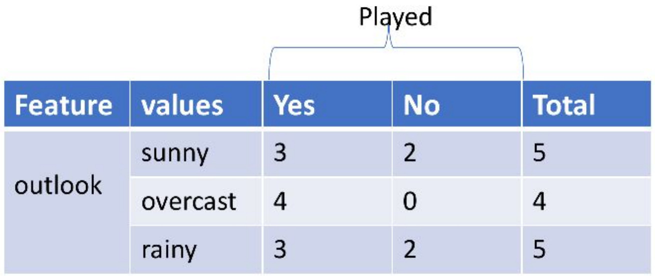
Dataset



Temperature	Outlook	Humidity	Windy	Played?
Mild	Sunny	80	No	Yes
Hot	Sunny	75	Yes	No
Hot	Overcast	77	No	Yes
Cool	Rain	70	No	Yes
Cool	Overcast	72	Yes	Yes
Mild	Sunny	77	No	No
Cool	Sunny	70	No	Yes
Mild	Rain	69	No	Yes
Mild	Sunny	65	Yes	Yes
Mild	Overcast	77	Yes	Yes
Hot	Overcast	74	No	Yes
Mild	Rain	77	Yes	No
Cool	Rain	73	Yes	No
Mild	Rain	78	No	Yes

Estimate the entropy of “Outlook” wrt “Played”: E(Played, Outlook)

- Here, E(Played, Outlook) is estimated looking at the various possibilities for ‘outlook’, carrying the ‘Played’:



		Played		
Feature	values	Yes	No	Total
outlook	sunny	3	2	5
	overcast	4	0	4
	rainy	3	2	5

$$E(\text{Played, Outlook}) = P(\text{sunny}) * E(3/5,\ 2/5) + P(\text{overcast}) * E(4/4,0) + P(\text{rainy}) * E(3/5,2/5)$$

$$= (5/14) * E(3/5,2/5) + (4/14) * E(4/4,0) + (5/14) * E(3/5,2/5)$$

- We see that:

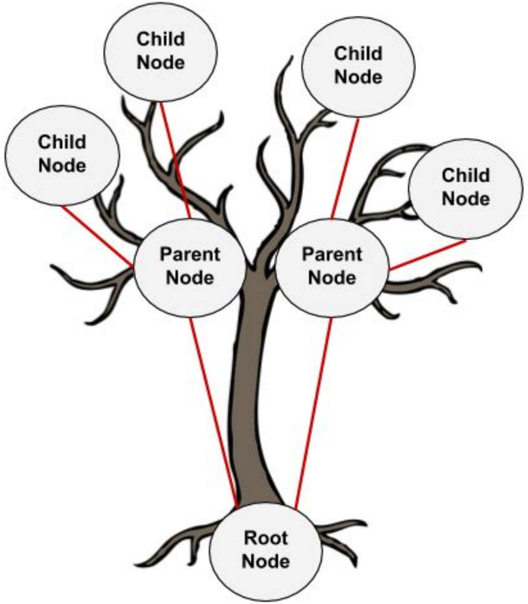
$$E(3/5,2/5) = E(2/5,3/5) = -\ (2/5) \log (2/5) – (3/5) \log(3/5) = 0.97 \text{ bits}$$
$$E(4/4,0) = 0.0 \text{ bits}$$

- And finally, that:

$$E(\text{Played, outlook}) = (5/14) * (0.97) + (4/14) * (0.0) + (5/14) * (0.97) = 0.693$$

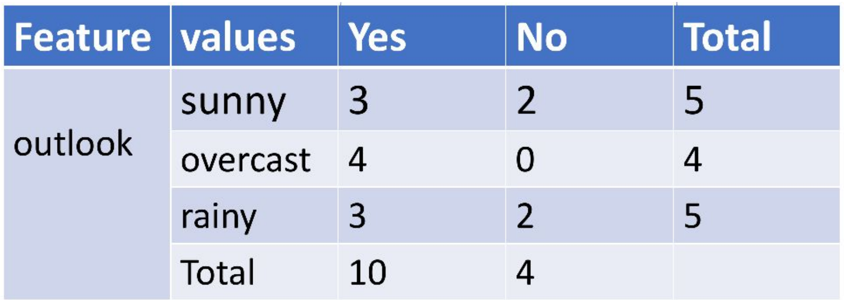
Decision Tree optimisation problem

- Minimize Entropy.
- Maximizing Information Gain (IG) minimized entropy
- This is an optimization problem



Entropy and Information Gain (IG)

- Information Gain (IG) is the informational value of creating a *branch* on the outlook attribute
- The feature with the highest information gain is selected as a decision node
- We calculate the IG for each attribute, and split on the attribute that gains us the most information



Feature	values	Yes	No	Total
outlook	sunny	3	2	5
	overcast	4	0	4
	rainy	3	2	5
	Total	10	4	

$$IG(outlook) = E(Played) - E(Played, outlook)$$

$$= E(4{,}10) - E(Played, outlook)$$

$$= 0.85 \ - 0.693 = 0.157 \text{ bits}$$

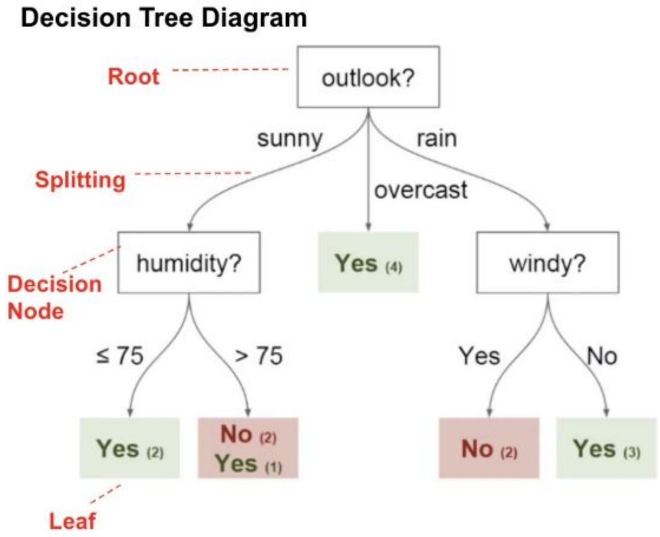
Generating Classification Rules

- For an input instance:
  - If the outlook = 'sunny' and humidity > 75 , then class="NO".
  - If the outlook='overcast' then class= "YES".
  - So forth.

- Unnecessary structure can 'clutter' the tree
- We 'prune' to eliminate unnecessary structure

Decision tree construction procedure:

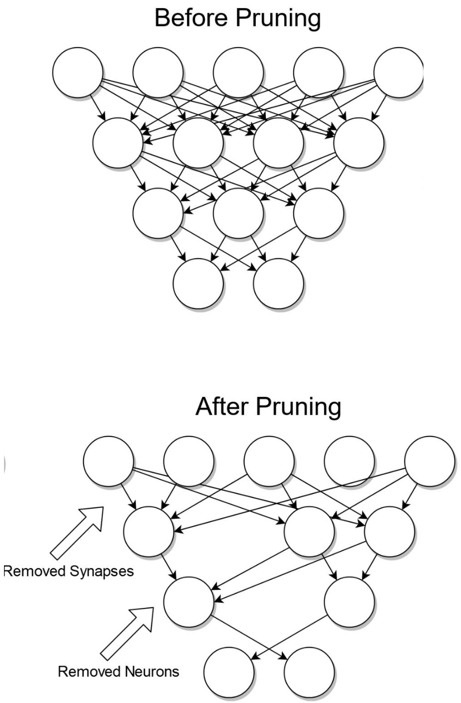
- Create a splitting rule
- Divide the data using the splitting rule
- Repeat
- Stop when final sets (leaves) are homogenous (i.e. same class)



@Kdnuggets: Clare Liu, Fintech industry

Selecting an optimal size tree

- Post pruning (backward pruning):
  - Build a large tree that fully describes the data
  - Very complex model with zero training error
  - Start from the leaves and prune back by erasing the rules with minimal (negative) impact to the error/performance, based on cross validation.
- Takes into account a combination of attributes.
- Pre pruning (during construction of trees)



Wikipedia

Post-pruning operations

1. Subtree replacement (primary pruning operation)

- Select some subtrees and replace them with single leaves
- It may decrease the accuracy of the training set. However, it may increase the accuracy on an independently chosen test set.

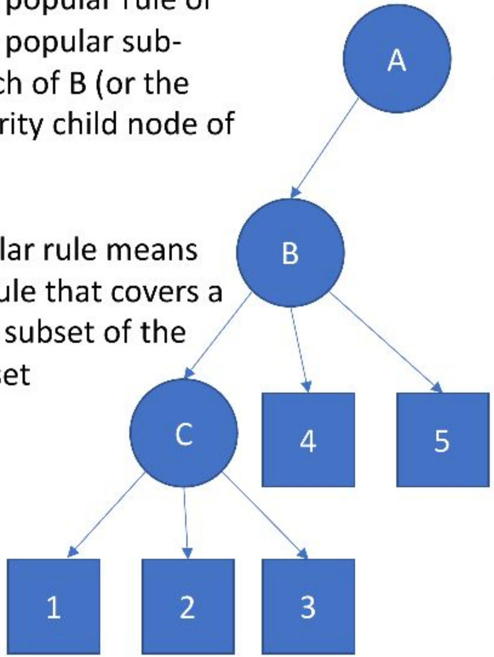
2. Subtree raising

- More complex than subtree replacement
- Used in C4.5 decision tree building system
- Time-consuming operation
- May require reclassification of the samples at some of the affected nodes

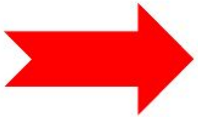
Subtree Raising Operation

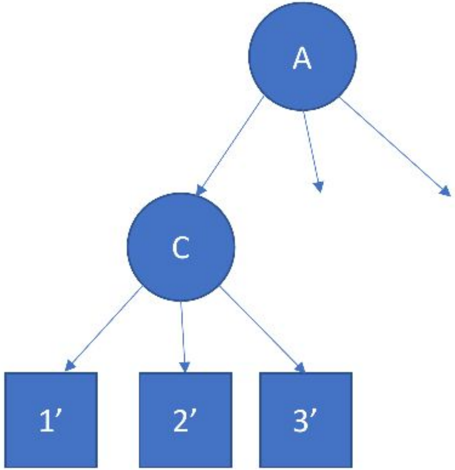
Assume that C is the most popular rule or most popular sub-branch of B (or the majority child node of B).

Popular rule means the rule that covers a large subset of the dataset



The entire subtree from node C downward is raised to replace the B subtree.





overhead : reclassify the samples at nodes 4 and 5

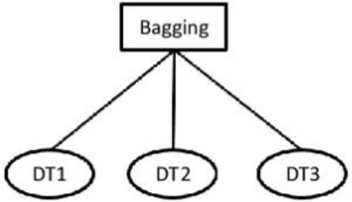
What if node '4' was the majority child node of B?

Bagging/Bagged Decision Trees (Bootstrap Aggregating)

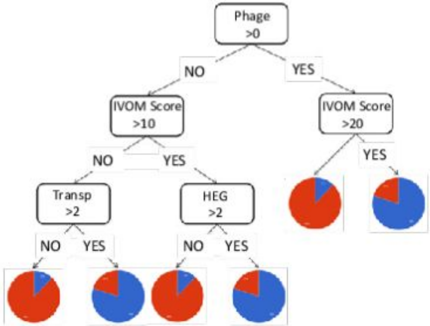
- Hyperparameter in Bagging: the ‘t’ number of trees
- Algorithm:

1. Sample with Replacement, n training observations from the dataset for the training set
2. Train a decision tree on the observations
3. Repeat 1) and 2) t times. This will result in t decision trees.
4. Aggregate the predictions from the t decision trees.
   - Either take the majority vote or take the average if the output of the decision trees are numerical values (such as predicting temperature, price, etc.)

- Very similar to Random Forest algorithm



(A). Bagging model



(B). Decision tree model

Che, Dongsheng & Chen, Bernard. (2014). An Accurate Genomic Island Prediction Method for Sequenced Bacterial and Archaeal Genomes. Journal of Proteomics & Bioinformatics. 07. 10.4172/jpb.1000322.

Random Forest

- Unlike Bagged DT, a Random Forest has a hyperparameter that controls the number of features to try when finding the best split (injects randomness)
- Feature bagging:
    - Given a dataset D with *p* features, try only a random subset of features given by size $\frac{p}{3}$ or $\sqrt{p}$
- Randomness makes individual subtrees more unique and reduces correlation between the trees
- Great overall performance.

Comparison: Random Forests and Decision Trees

- Random Forests
  - Prevent Overfitting
  - Can make computations slower
- Decision Trees
  - A Deep DT overfitts
  - Does not need additional overhead

Data exploration and a precursor to unsupervised learning

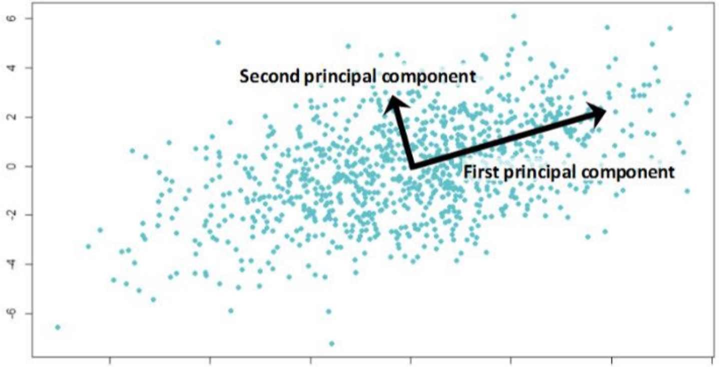
- Covered under pre-processing, we now revisit Principal Component Analysis
- Use examples:
  - Dimensionality reduction/data compression
  - Data visualization and Exploratory Data Analysis
  - Create uncorrelated features/variables that can be an input to a prediction model
  - Uncovering latent variables/themes/concepts
  - Noise reduction in the dataset

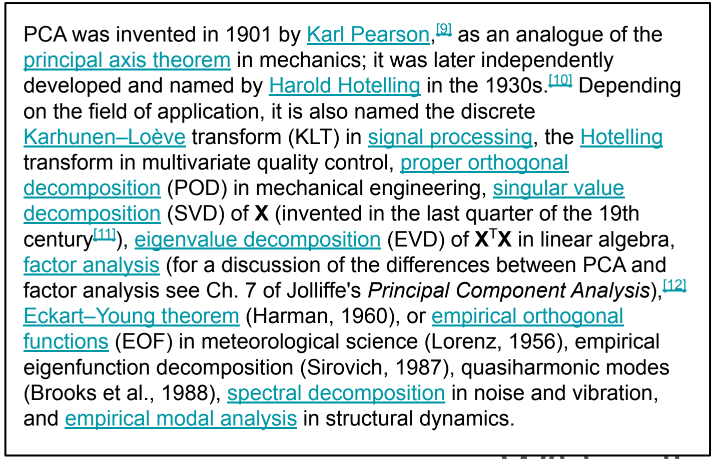
PCA example for preprocessing

- Principal Component Analysis (aka see below)
- Basic principle: Some axes of variability are more important than others
- PCA is a dimensionality reduction technique used to transform high-dimensional data into a lower-dimensional representation
- Preserving the most ‘important information’
- Transforms a set of possibly correlated variables into a set of linearly uncorrelated variables

- See “pca_example.ipynb” from L2

linkedin.com

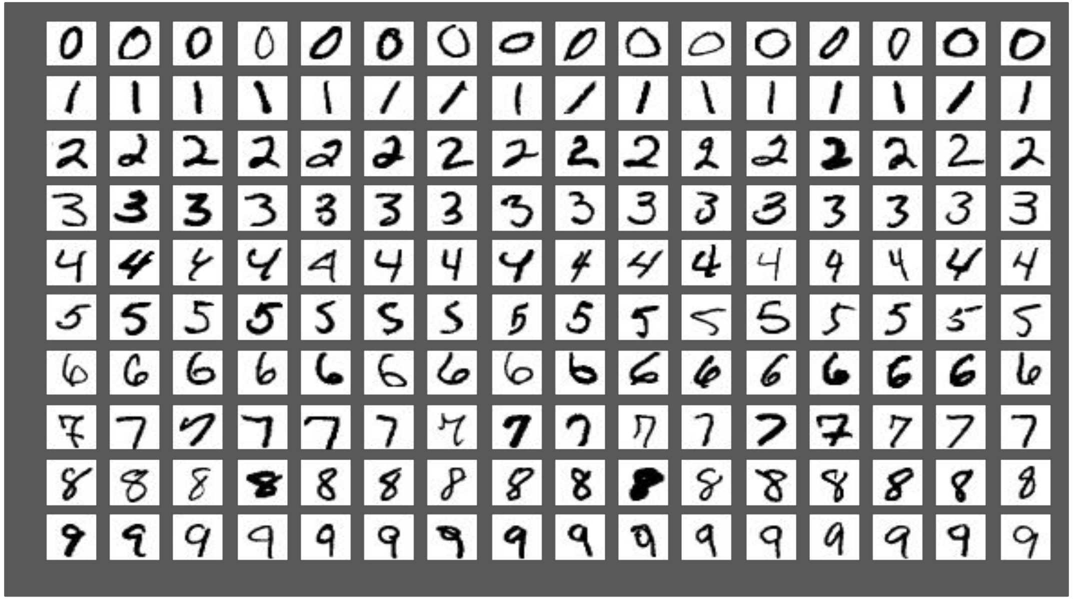



PCA was invented in 1901 by Karl Pearson,[9] as an analogue of the principal axis theorem in mechanics; it was later independently developed and named by Harold Hotelling in the 1930s.[10] Depending on the field of application, it is also named the discrete Karhunen–Loève transform (KLT) in signal processing, the Hotelling transform in multivariate quality control, proper orthogonal decomposition (POD) in mechanical engineering, singular value decomposition (SVD) of **X** (invented in the last quarter of the 19th century[11]), eigenvalue decomposition (EVD) of $\mathbf{X}^{T}\mathbf{X}$ in linear algebra, factor analysis (for a discussion of the differences between PCA and factor analysis see Ch. 7 of Jolliffe's *Principal Component Analysis*),[12] Eckart–Young theorem (Harman, 1960), or empirical orthogonal functions (EOF) in meteorological science (Lorenz, 1956), empirical eigenfunction decomposition (Sirovich, 1987), quasiharmonic modes (Brooks et al., 1988), spectral decomposition in noise and vibration, and empirical modal analysis in structural dynamics.

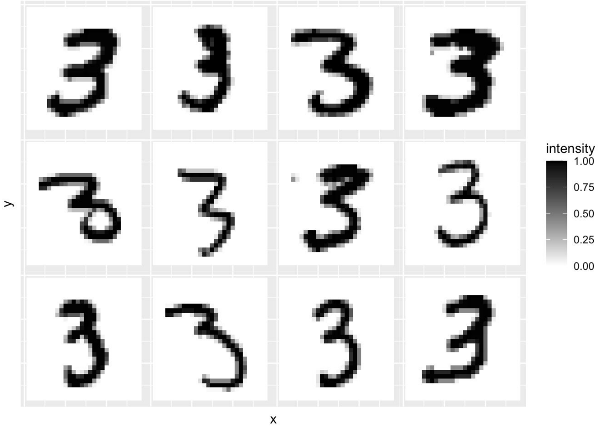
Wikipedia

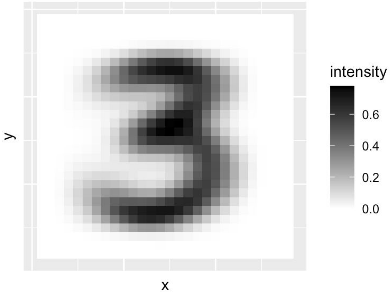
PCA on MNIST dataset

- The MNIST dataset has 784 dimensions for each example - 784 pixels
- Do we need all 784 dimensions?

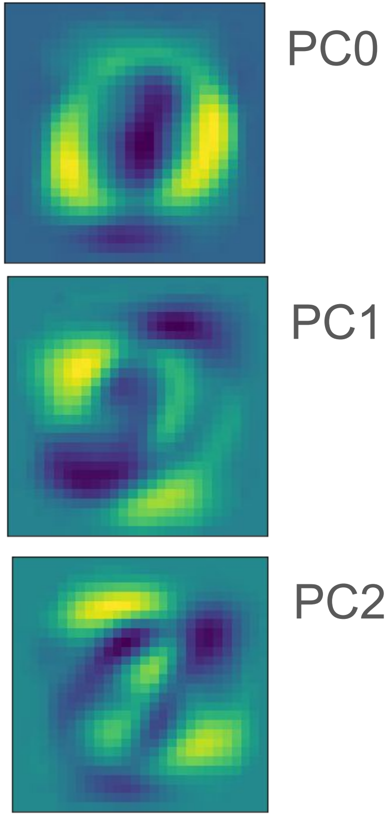


Principal Components don’t look intuitive





Average 3



Whole dataset

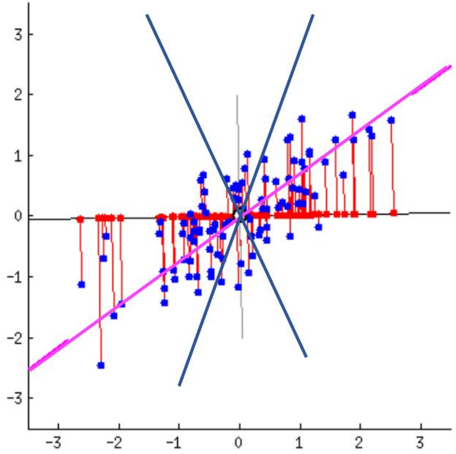
Does the dataset have a normal distribution?

Steps of a PCA calculation

Dataset D with size *n* and *m* features

1. Standardize: Z-score (z = (x - μ) / σ) so has μ=0 , σ=1
2. Compute covariance matrix (relationship and the dependency) ∑ or correlation matrix (strength and direction)
3. Compute eigenvalues (λ_i) and eigenvectors (e_i) with the covariance matrix ∑
4. Obtain the principal components: Pick the top x eigenvectors corresponding to the largest eigenvalues in descending order
5. Project the data into eigenvectors: Reorient (transform) the original data into the new coordinate system defined by the selected principal components

Put differently we maximize variance: Project to the line that minimizes variance or Squared Sum of deviations from the mean



Recall: eigenvalues (λ_i) and eigenvectors (e_i)

- Eigenvalues (λ_i) determine the importance of eigenvectors (e_i) (i.e., components)
- Indicate variance in data (spread)
- The direction of the principal components oriented towards the larger spread of the data

$$\underbrace{A}_{\text{Matrix}} \overbrace{e_i}^{Eigenvector} = \underbrace{\lambda}_{Eigenvalue} \overbrace{e_i}^{Eigenvector}$$

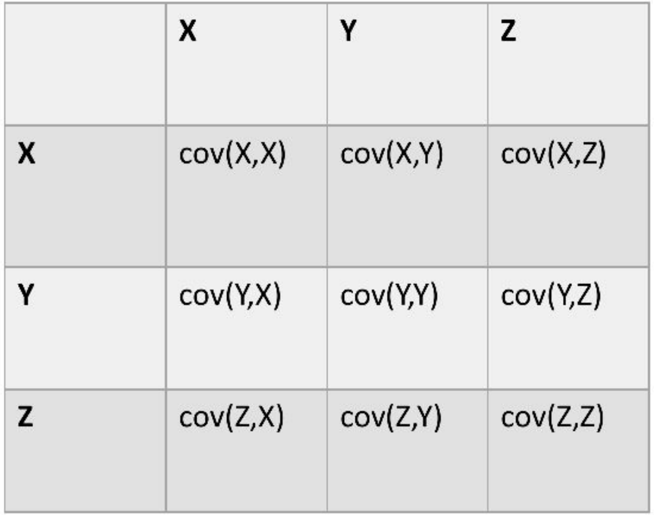
$$Ae_i = \lambda e_i$$

$$Ae_i - \lambda I e_i = 0$$

$$(A - \lambda I)e_i = 0$$

$$det(A - \lambda I) = 0$$

Covariance Matrix ∑



	X	Y	Z
X	cov(X,X)	cov(X,Y)	cov(X,Z)
Y	cov(Y,X)	cov(Y,Y)	cov(Y,Z)
Z	cov(Z,X)	cov(Z,Y)	cov(Z,Z)

$$\sum\nolimits_{dxd}$$

$$cov(X,Y) = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{N}$$

Using ∑ Find Eigenvalues

- Solve the following given I is the identity matrix:

$$\det(\Sigma - \lambda I) = 0$$

- Example:

$$\Sigma = \begin{bmatrix} 2 & 0.8 \\ 0.8 & 0.6 \end{bmatrix}, I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \Rightarrow \lambda I = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$$

- Filling in from above we have:

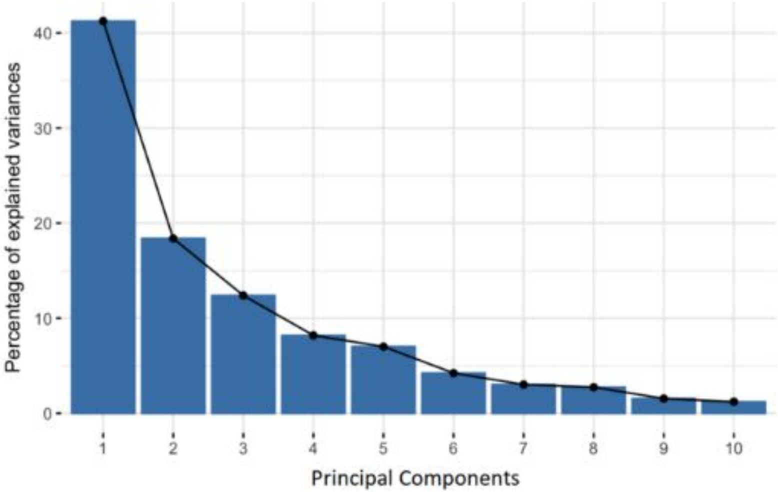
$$\Sigma - \lambda I = \begin{bmatrix} 2 - \lambda & 0.8 \\ 0.8 & 0.6 - \lambda \end{bmatrix}$$

- Using the determinant:
- $\det(\Sigma - \lambda I) = 0 : (2 - \lambda)(0.6 - \lambda) - 0.64 = 0$
- This leaves us with the eigenvalues λ_1 and λ_2:

$$\lambda^2 - 2.6\lambda + 0.56 = 0 \Rightarrow \lambda = \{\lambda_1, \lambda_2\} = \{2.36 , 0.23\}$$

Percentage of variance in the Principal Components

$$Percentage\ of\ Variance = \frac{\lambda_i}{\sum_{i=1}^{d} \lambda_i}$$

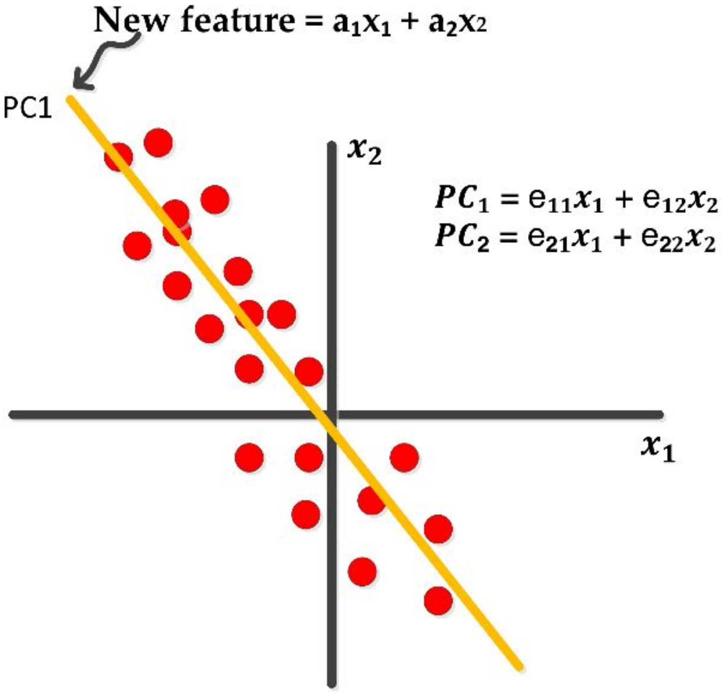


Transformed features

- PCA provides the linear Transformation of the original features which will produce new features called principal components that are uncorrelated (perpendicular or orthogonal)

$$
\begin{aligned}
PC_1 &= e_{11}x_1 + e_{12}x_2 + \ldots + e_{1n}x_n \\
PC_2 &= e_{21}x_1 + e_{22}x_2 + \ldots + e_{2n}x_n \\
&\ldots \\
PC_k &= e_{k1}x_1 + e_{k2}x_2 + \ldots + e_{kn}x_n
\end{aligned}
$$

- The transformation is done by calculating eigenvalues and eigenvectors


New feature = $a_1x_1 + a_2x_2$
PC1
x_2
$PC_1 = e_{11}x_1 + e_{12}x_2$
$PC_2 = e_{21}x_1 + e_{22}x_2$
x_1

Find Eigenvectors

- Solve: $$\Sigma * e_i = \lambda_i * e_i$$
- Example, given $\lambda_1$and ∑:

$$\lambda_1 = 2.36, \Sigma = \begin{bmatrix} 2 & 0.8 \\ 0.8 & 0.6 \end{bmatrix}$$

- We now have:

$$\begin{bmatrix} 2 & 0.8 \\ 0.8 & 0.6 \end{bmatrix} * e_1 = 2.36 * e_1 \Rightarrow \begin{bmatrix} 2 & 0.8 \\ 0.8 & 0.6 \end{bmatrix} * \begin{bmatrix} e_{1,1} \\ e_{1,2} \end{bmatrix} = 2.36 * \begin{bmatrix} e_{1,1} \\ e_{1,2} \end{bmatrix}$$

- For e_1 leading to:

$$e_{1,1} = 2.2\, e_{1,2} \Rightarrow \text{if } e_{1,2} = 1, \text{ then } e_{1,1} = 2.2 \Rightarrow e_1 = \begin{bmatrix} 2.2 \\ 1 \end{bmatrix}$$

- Eigenvectors have to be unit length to avoid multiple solutions (i.e., $|| e_i ||=1$)
- We divide e_1 by its length:

$$e_1 = \begin{bmatrix} 0.91 \\ 0.41 \end{bmatrix}, \ || e_1 ||=1$$

- Now do the same for e_2 etc…

Re-Orienting the Data using Principal Components

- Sort by which eigenvalue is largest, from before:

$$\lambda_1 > \lambda_2\text{: } 2.36 > 0.23$$

- If we only use one principal component:

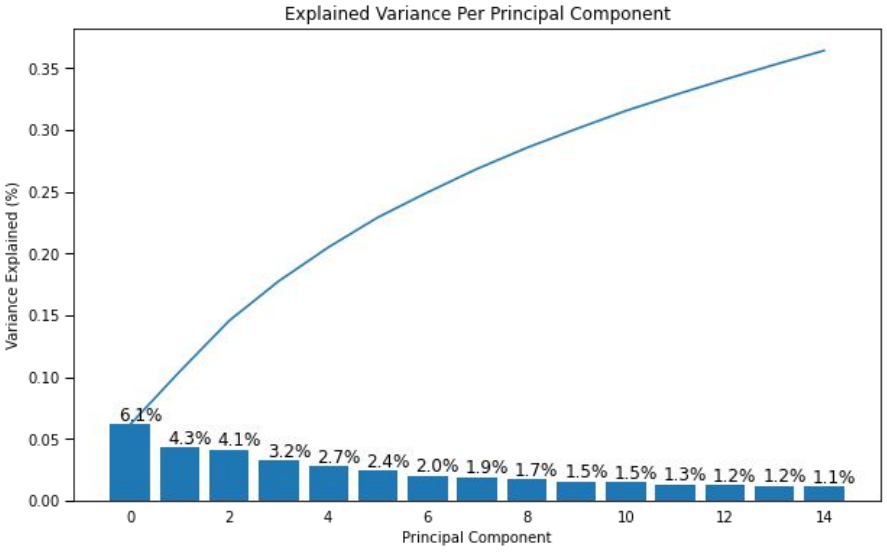
$$e_1 = \begin{bmatrix} e_{11} \\ e_{12} \end{bmatrix} = \begin{bmatrix} 0.91 \\ 0.41 \end{bmatrix}$$

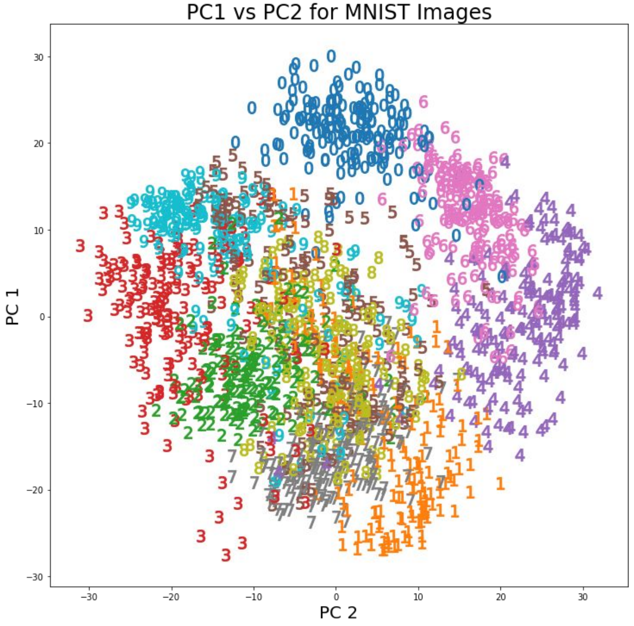
- The transformed dataset becomes:

$$\text{Transformed dataset} = standardized\ D * Feature\ Vector^T$$

The feature vector is the selected PC(s). In this case, it is e_1.

Adding more Principle components allows us to capture more of the variance



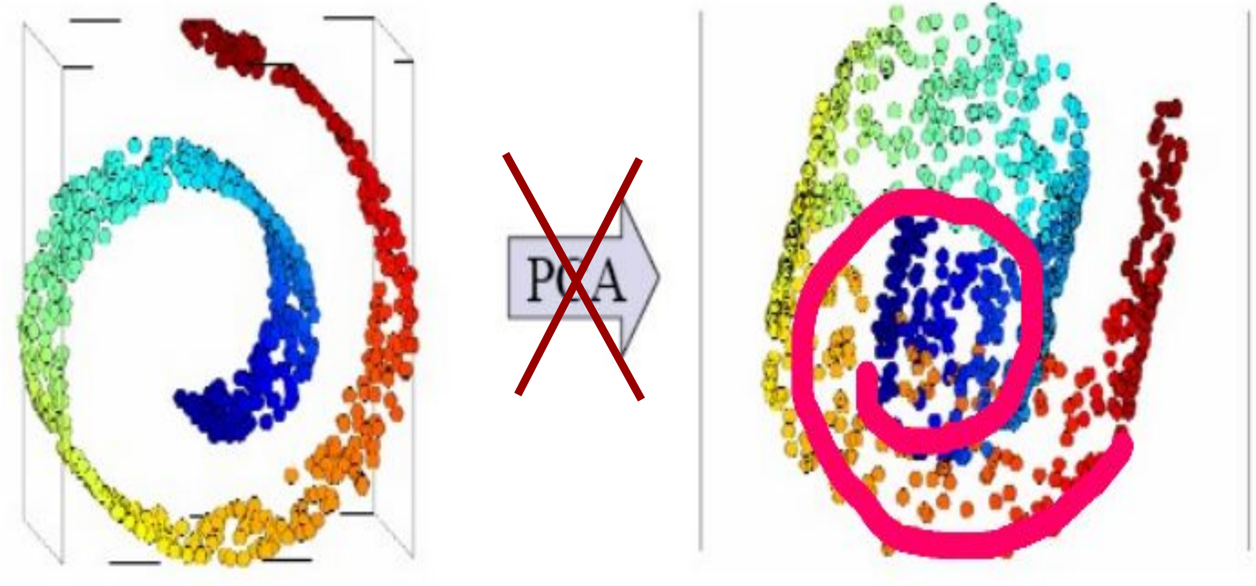


For PCA to work perfectly data must have a normal distribution

Assumptions of PCA

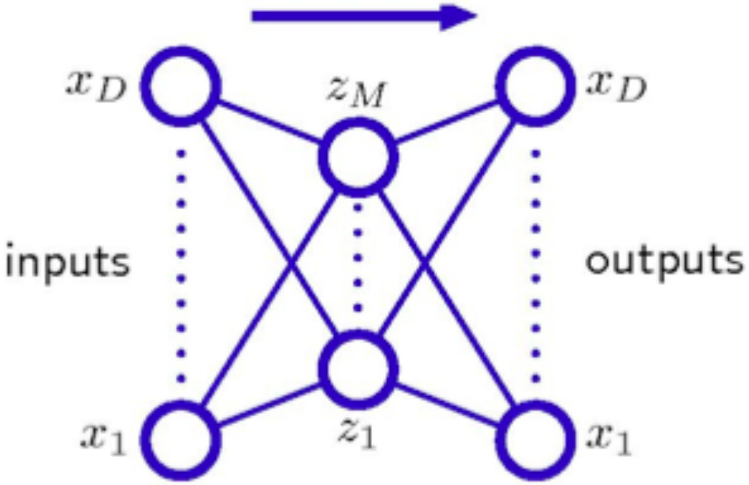
- PCA is a linear dimension reduction technique. However, it can lead to poor results when dealing with non-linear structures.
- PCA must be performed with standardized data (mean =0, Variance = 1).
- Just one PCA plot should be generated by considering the maximum variances within the attributes. *It preserves large pairwise distances.*
  - It doesn’t work well when the pairwise distances are small
- PCA *can not preserve local distances* in data.
  - Only global distance (large distances) are preserved.
- PCA transform the data into a new set of orthogonal components, ensuring that the first component aligns to the maximum variance in the dataset, and subsequent components align to the next maximum orthogonal variance, and so on.

Nonlinearity and PCA



PCA relationship to Neural Networks

- PCA is closely related to a particular form of neural network
- An autoencoder is a neural network whose outputs are its own inputs
- The goal is to minimize reconstruction error


x_D
z_M
x_D
inputs
outputs
x_1
z_1
x_1