UNIT IV DIMENSIONALITY REDUCTION AND GRAPHICAL MODELS

DIMENSIONALITY REDUCTION

In machine learning we are having too many factors on which the final classification is done. These factors are basically, known as variables. The higher the number of features, the harder it gets to visualize the training set and then work on it. Sometimes, most of these features are correlated, and hence redundant. This is where dimensionality reduction algorithms come into play.

Motivation

- When we deal with real problems and real data we often deal with high dimensional data that can go up to millions.
- In original high dimensional structure, data represents itself. Although, sometimes we need to reduce its dimensionality.
- We need to reduce the dimensionality that needs to associate with visualizations. Although, that is not always the case.

Components of Dimensionality Reduction

a. Feature selection

In this, we need to find a subset of the original set of variables. Also, need a subset which we use to model the problem. It usually involves three ways:

1. Filter
2. Wrapper
3. Embedded

b. Feature Extraction

We use this, to reduce the data in a high dimensional space to a lower dimension space, i.e. a space with lesser no. of dimensions.

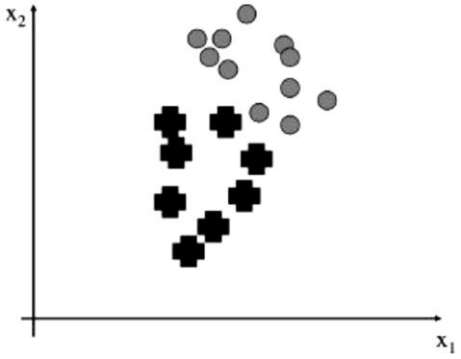
Dimensionality Reduction Methods

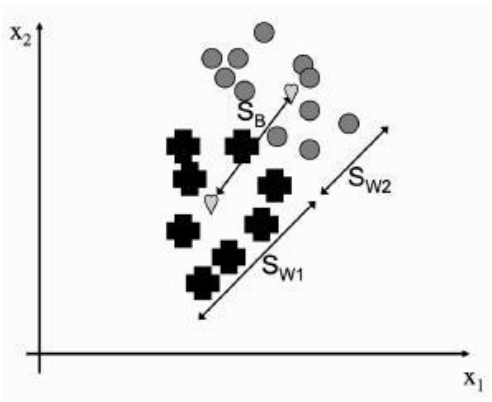
1. Principal Component Analysis (PCA)
2. Linear Discriminant Analysis (LDA)
3. Generalized Discriminant Analysis (GDA)

Dimensionality reduction may be both linear or non-linear, depending upon the method used.

LINEAR DISCRIMINANT ANALYSIS

- Linear Discriminant Analysis or Discriminant Function Analysis is a dimensionality reduction technique which is commonly used for the supervised classification problems.
- It is used for modeling differences in groups i.e. separating two or more classes.
- It is used to project the features in higher dimension space into a lower dimension space.
- Figure 6.2 shows a simple two-dimensional dataset consisting of two classes.
- Compute the means of the two classes in the data, μ1 and μ2, the mean of the entire dataset (μ), and the covariance of each class.
- The principal insight of LDA is that the covariance matrix can tell us about the scatter within a dataset, which is the amount of spread that there is within the data.





- The way to find this scatter is to multiply the covariance by the *pc*, the probability of the class (that is, the number of datapoints there are in that class divided by the total number).
- Adding the values of this for all of the classes gives us a measure of the within-class scatter of the dataset:

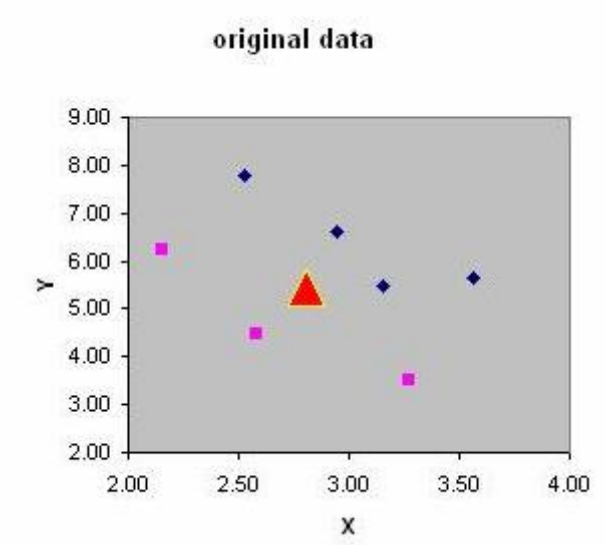
$$S_W = \sum_{\text{classes } c} \sum_{j \in c} p_c(\mathrm{x}_j - \boldsymbol{\mu}_c)(\mathrm{x}_j - \boldsymbol{\mu}_c)^T.$$

$$S_B = \sum_{\text{classes } c} (\boldsymbol{\mu}_c - \boldsymbol{\mu})(\boldsymbol{\mu}_c - \boldsymbol{\mu})^T.$$

Example

Chip Quality Result Data

Curvature	Diameter	Quality Control Result
2.95	6.63	Passed
2.53	7.79	Passed
3.57	5.65	Passed
3.16	5.47	Passed
2.58	4.46	Not Passed
2.16	6.22	Not Passed
3.27	3.52	Not Passed



$\mathbf{x}$ = features (or independent variables) of all data. Each row (denoted by k) represents one object; each column stands for one feature.

$\mathbf{y}$ = group of the object (or dependent variable) of all data. Each row represents one object and it has only one column.

In our example, $\mathbf{x} = \begin{bmatrix} 2.95 & 6.63 \\ 2.53 & 7.79 \\ 3.57 & 5.65 \\ 3.16 & 5.47 \\ 2.58 & 4.46 \\ 2.16 & 6.22 \\ 3.27 & 3.52 \end{bmatrix}$ and $\mathbf{y} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 2 \\ 2 \\ 2 \end{bmatrix}$

$\mathbf{x}_k$ = data of row k. For example, $\mathbf{x}_3 = [3.57 \quad 5.65]$, $\mathbf{x}_7 = [3.27 \quad 3.52]$

g = number of groups in $\mathbf{y}$. In our example, $g = 2$

$\mathbf{x}_i$ = features data for group i. Each row represents one object; each column stands for one feature. We separate $\mathbf{x}$ into several groups based on the number of category in $\mathbf{y}$.

x1	x2	Class
4	1	1
2	4	1
2	3	1
3	6	1
4	4	1
9	10	2
6	8	2
9	5	2
8	7	2
10	8	2

x1	x2
4	1
2	4
2	3
3	6
4	4

x1	x2
9	10
6	8
9	5
8	7
10	8

***Step*1:**

$\boldsymbol{\mu 1 = [3, 3.6]}$ **for class1**

$\boldsymbol{\mu 2 = [8.4, 7.6]}$ **for class2**

$\boldsymbol{\mu = [5.7,\ 5.6]}$ **for overall data**

***Step*2:**

Mean Corrected Data = $\mathbf{x_i}$ - μi

Mean Corrected Data

1	-2.6
-1	0.4
-1	-0.6
0	2.4
1	0.4
0.6	2.4
-2.4	0.4
0.6	-2.6
-0.4	-0.6
1.6	0.4

Step*3: *Calculating within class Scatter Matrix

$S_w = S_{w1} + S_{w2}$

$S_{w1} = \sum(x1 - \mu 1)(x1 - \mu 2)T$

$S_{w2} = \sum(x2 - \mu 2)(x1 - \mu 2)T$

Mean Corrected Data	
X1o	X2o
1	-2.6
-1	0.4
-1	-0.6
0	2.4
1	0.4
0.6	2.4
-2.4	0.4
0.6	-2.6
-0.4	-0.6
1.6	0.4

$x1 - \mu1$

$x2 - \mu2$

Covariance group 1

$(x1 - \mu1)$

1	-1	-1	0	1
-2.6	0.4	-0.6	2.4	0.4

$(x1 - \mu1)^T$

1	-2.6
-1	0.4
-1	-0.6
0	2.4
1	0.4

Covariance group 2

0.6	-2.4	0.6	-0.4	1.6
2.4	0.4	-2.6	-0.6	0.4

$(x2 - \mu2)^T$

0.6	2.4
-2.4	0.4
0.6	-2.6
-0.4	-0.6
1.6	0.4

$(x2 - \mu2)$

1st Row 1st Column

= (1*1) + (-1*-1) + (-1*-1)+ (0*0) +(1*1)

= 4

Sw1=

4/5	**-2/5**
-2/5	**13.20/5**

1st Row 2nd Column

= (1*-2.6) + (-1*0.4) + (-1*-0.6)+ (0*2.4) +(1*0.4)

= -2

Sw1=

0.8	**-0.4**
-0.4	**2.64**

2nd Row 1st Column

= (-2.6*1) + (0.4*-1) + (-0.6*-1)+ (2.4*0)+(0.4*1)

= -2

Sw2=

1.84	**-0.04**
-0.04	**2.64**

2nd Row 2nd Column

= (-2.6*-2.6) + (0.4*0.4) + (-0.6*-0.6)+ (2.4*2.4)+(0.4*0.4)

= 13.20

$$S_w = S_{w1} + S_{w2}$$

Sw1=

0.8	-0.4
-0.4	2.64

Sw2=

1.84	-0.04
-0.04	2.64

Sw=

2.64	-0.44
-0.44	5.28

Step4: **Calculate b/w class Scatter matrix**

$\mathbf{S_B} = (\mu 1 - \mu 2)(\mu 1 - \mu 2)T$

$\boldsymbol{\mu 1} = [\mathbf{3}, \mathbf{3.6}] \quad \boldsymbol{\mu 2} = [\mathbf{8.4}, \mathbf{7.6}]$

$$S_B \begin{pmatrix} 29.16 & 21.6 \\ 21.65 & 16.00 \end{pmatrix}$$

Step5: Eigen vector &values for $M = Sw^{-1} S_B$

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \rightarrow \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

$$Sw^{-1} = \frac{1}{13.74} \begin{pmatrix} 5.28 & 0.44 \\ 0.44 & 2.64 \end{pmatrix} \qquad Sw = \begin{pmatrix} 2.64 & -0.44 \\ -0.44 & 5.28 \end{pmatrix}$$

$$Sw^{-1} = \begin{pmatrix} 0.384 & 0.032 \\ 0.032 & 0.192 \end{pmatrix} \qquad M = \begin{pmatrix} 0.384 & 0.032 \\ 0.032 & 0.192 \end{pmatrix} \begin{pmatrix} 29.16 & 21.6 \\ 21.65 & 16.00 \end{pmatrix}$$

$$M = \begin{pmatrix} 11.88 & 8.806 \\ 5.08 & 3.76 \end{pmatrix}$$

Step5: Eigen vector &values for $M = Sw^{-1} S_B$

$Z = W^T X \leftarrow$ *adjusted data*

$W = [0.91 \\ 0.39]$

$W^T = [0.91\ 0.39] * X$

2D

Mean Corrected Data	
X1o	X2o
1	-2.6
-1	0.4
-1	-0.6
0	2.4
1	0.4
0.6	2.4
-2.4	0.4
0.6	-2.6
-0.4	-0.6
1.6	0.4

$=$ 1D

PRINCIPAL COMPONENT ANALYSIS

Need For Principal Component Analysis (PCA)

- Machine Learning in general works wonders when the dataset provided for training the machine is large and concise. Usually having a good amount of data lets us build a better predictive model since we have more data to train the machine with. However, using a large data set has its own pitfalls. The biggest pitfall is the curse of dimensionality.
- It turns out that in large dimensional datasets, there might be lots of inconsistencies in the features or lots of redundant features in the dataset, which will only increase the computation time and make data processing and EDA more convoluted.
- To get rid of the curse of dimensionality, a process called dimensionality reduction was introduced. Dimensionality reduction techniques can be used to filter only a limited number of significant features needed for training and this is where PCA comes in.

What Is Principal Component Analysis (PCA)?

- Principal components analysis (PCA) is a dimensionality reduction technique that enables you to identify correlations and patterns in a data set so that it can be transformed into a data set of significantly lower dimension without loss of any important information.
- The main idea behind PCA is to figure out patterns and correlations among various features in the data set. On finding a strong correlation between different variables, a final decision is made about reducing the dimensions of the data in such a way that the significant data is still retained.
- Such a process is very essential in solving complex data-driven problems that involve the use of high-dimensional data sets. PCA can be achieved via a series of steps. Let’s discuss the whole end-to-end process.

Step By Step Computation Of PCA

The below steps need to be followed to perform dimensionality reduction using PCA:

1. Standardization of the data
2. Computing the covariance matrix
3. Calculating the eigenvectors and eigenvalues
4. Computing the Principal Components
5. Reducing the dimensions of the data set

Example:

X_1^*	X_2^*
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2.0	1.6
1.0	1.1
1.5	1.6
1.1	0.9

Sample size n = 10

Variables p = 2

Step 1: Standardization of the data

If you're familiar with data analysis and processing, you know that missing out on standardization will probably result in a biased outcome. Standardization is all about scaling your data in such a way that all the variables and their values lie within a similar range.

Therefore, standardizing the data into a comparable range is very important. Standardization is carried out by subtracting each value in the data from the mean and dividing it by the overall deviation in the data set.

It can be calculated like so:

$$X_1^* = \frac{2.5+0.5+2.2.....+1.1}{10}$$

$$X_1^* = 1.81$$

$$X_2^* = \frac{2.4+0.7+2.9.....+0.9}{10}$$

$$X_2^* = 1.91$$

Subtract the mean from the corresponding data component to recentre the dataset.

X_1^*	X_2^*	$X_1 = X_1^* - \overline{X_1^*}$	$X_2 = X_2^* - \overline{X_2^*}$
2.5	2.4	0.69	0.49
0.5	0.7	-1.31	-1.21
2.2	2.9	0.39	0.99
1.9	2.2	0.09	0.29
3.1	3.0	1.29	1.09
2.3	2.7	0.49	0.79
2.0	1.6	0.19	-0.31
1.0	1.1	-0.81	-0.81
1.5	1.6	-0.31	-0.31
1.1	0.9	-0.71	-1.01

Step 2: Computing the covariance matrix

As mentioned earlier, PCA helps to identify the correlation and dependencies among the features in a data set. A covariance matrix expresses the correlation between the different variables in the data set. It is essential to identify heavily dependent variables because they contain biased and redundant information which reduces the overall performance of the model.

Mathematically, a covariance matrix is a p × p matrix, where p represents the dimensions of the data set. Each entry in the matrix represents the covariance of the corresponding variables.

Consider a case where we have a 2-Dimensional data set with variables a and b, the covariance matrix is a 2×2 matrix as shown below:

$$\begin{bmatrix} Cov(a, a) & Cov(a, b) \\ Cov(b, a) & Cov(b, b) \end{bmatrix}$$

In the above matrix:

Cov(a, a) represents the covariance of a variable with itself, which is nothing but the variance of the variable 'a'

Cov(a, b) represents the covariance of the variable 'a' with respect to the variable 'b'. And since covariance is commutative, Cov(a, b) = Cov(b, a)

Here are the key takeaways from the covariance matrix:

The covariance value denotes how co-dependent two variables are with respect to each other If the covariance value is negative, it denotes the respective variables are indirectly proportional to each other. A positive covariance denotes that the respective variables are directly proportional to each other

$$C = \frac{1}{N-1}(X - 1\dot{X})(X - 1\dot{\bar{X}})$$

$$= \frac{1}{N-1}\dot{X}X$$

$$\dot{X} = transpose\ of\ X$$

$$C = \frac{1}{10-1}\begin{pmatrix} 0.69 & -1.31\ldots\ldots \\ 0.49 & -1.21\ldots\ldots \end{pmatrix} 2\ X\ 10\ \begin{pmatrix} 0.69 & 0.49 \\ -1.31 & -1.21. \end{pmatrix} 10\ X\ 2$$

$$C = \frac{1}{10-1}\begin{pmatrix} 0.69 & -1.31\ldots\ldots \\ 0.49 & -1.21\ldots\ldots \end{pmatrix} 2\ X\ 10\ \begin{pmatrix} 0.69 & 0.49 \\ -1.31 & -1.21. \end{pmatrix} 10\ X\ 2$$

Step 3: Calculating the Eigenvectors and Eigenvalues

Eigenvectors and eigenvalues are the mathematical constructs that must be computed from the covariance matrix in order to determine the principal components of the data set.

- **λ is an eigen value for a matrix M if it is a solution of the characteristic equation |M – λI| = 0.**

$$\lambda I = \lambda = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$$

$$A - \lambda I = \begin{bmatrix} 0.6166 & 0.6154 \\ 0.6154 & 0.7166 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$$

$$= \begin{bmatrix} 0.6166 - \lambda & 0.6154 \\ 0.6154 & 0.7166 - \lambda \end{bmatrix}$$

$$= \det \begin{bmatrix} 0.6166 - \lambda & 0.6154 \\ 0.6154 & 0.7166 - \lambda \end{bmatrix}$$

$$= (0.61 - \lambda)(0.71 - \lambda) - (0.61)(0.6154)$$

$$= 0.441 - 0.661\lambda - 0.7166\lambda + \lambda^2 - 0.37$$

$$= \lambda^2 - 1.33\lambda + 0.0631$$

Here we use the quadratic formula to get the value of λ1 and λ2.

$\lambda 1 = 1.28$ and $\lambda 2 = 0.0492$

These values are the eigenvalue

$$\text{Now} \begin{bmatrix} 0.6166 - \lambda & 0.6154 \\ 0.6154 & 0.7166 - \lambda \end{bmatrix}$$

$\lambda 1 = 1.28$

$$\text{So} \begin{bmatrix} 0.6166 - 1.28 & 0.6154 \\ 0.6154 & 0.7166 - 1.28 \end{bmatrix}$$

$$B = \begin{bmatrix} -0.6634 & 0.6154 \\ 0.6154 & -0.5634 \end{bmatrix}$$

Solve $B\overline{X} = 0$

$$\begin{bmatrix} -0.6634 & 0.6154 \\ 0.6154 & -0.5634 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

x1= 0.678 and x2=0.735

$$\text{Eigen Vector} \begin{bmatrix} 0.678 \\ 0.735 \end{bmatrix}$$

Similarly for $\lambda 2 = 0.0492$

$$\text{So Eigen Vector} \begin{bmatrix} 0.735 \\ -0.678 \end{bmatrix}$$

So getting the value of both Eigen Values and Eigen Vectors

$\lambda 1 = 1.28$ and $\lambda 2 = 0.0492$

Variable X1 X2	Eigen Vector 1 0.678 0,735	Eigen Vector 2 0.735 -0.678
Eigen Value	$\lambda 1 = 1.28$	$\lambda 1 = 0.49$
% total Variance	$\frac{1.28}{1.33} = 96\%$ PC 1	$\frac{0.0490}{1.33} = 3.7\%$ PC 2

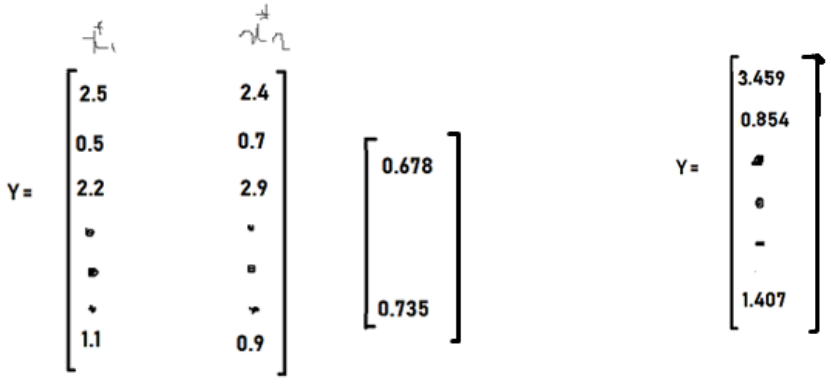
Step 4: Computing the Principal Components

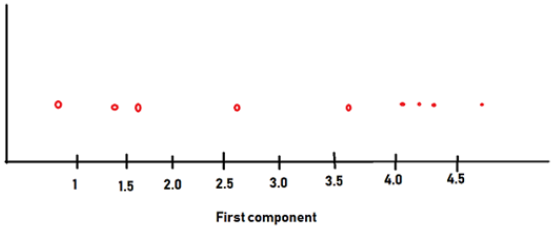
- Once we have computed the Eigenvectors and eigenvalues, all we have to do is order them in the descending order, where the eigenvector with the highest eigenvalue is the most significant and thus forms the first principal component. The principal components of lesser significances can thus be removed in order to reduce the dimensions of the data.
- The final step in computing the Principal Components is to form a matrix known as the feature matrix that contains all the significant data variables that possess maximum information about the data.

Step 5: Reducing the dimensions of the data set

The last step in performing PCA is to re-arrange the original data with the final principal components which represent the maximum and the most significant information of the data set. In order to replace the original data axis with the newly formed Principal Components, you simply multiply the transpose of the original data set by the transpose of the obtained feature vector.

Reducing the dimensions of the data set $Y = XV$





FACTOR ANALYSIS

Factor Analysis is a method for modeling observed variables, and their covariance structure, in terms of a smaller number of underlying unobservable (latent) "factors." The factors typically are viewed as broad concepts or ideas that may describe an observed phenomenon. For example, a basic desire of obtaining a certain social level might explain most consumption behavior. These unobserved factors are more interesting to the social scientist than the observed quantitative measurements. Factor analysis is generally an exploratory/descriptive method that requires many subjective judgments. It is a widely used tool and often controversial because the models, methods, and subjectivity are so flexible that debates about interpretations can occur.

•Data reduction tool

•Removes redundancy or duplication from a set of correlated variables

•Represents correlated variables with a smaller set of "derived "variables.

•Factors are formed that are relatively independent of one another.

•Two types of "variables":

–latent variables: factors

–observed variables

1. Identification of Underlying Factors:

–clusters variables into homogeneous sets

–creates new variables (i.e. factors)

–allows us to gain insight to categories

2. Screening of Variables:

–identifies groupings to allow us to select one variable to represent many

–useful in regression (recall collinearity)

3. Summary:

–Allows us to describe many variables using a few factors

4. Sampling of variables:

–helps select small group of variables of representative variables from larger set

5. Clustering of objects:

–Helps us to put objects (people) into categories depending on their factor scores

Orthogonal One Factor Model

Classical Test Theory Idea:

Ideal: $X_1 = F + e_1$ $var(e_j) = var(e_k)$, $j \neq k$

$X_2 = F + e_2$

...

$X_p = F + e_p$

Reality: $X_1 = \lambda_1 F + e_1$ $var(e_j) \neq var(e_k)$, $j \neq k$

$X_2 = \lambda_2 F + e_2$

...

$X_p = \lambda_p F + e_p$

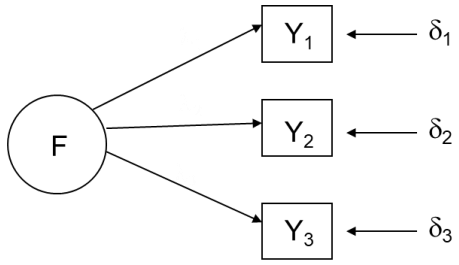
(unequal "sensitivity" to change in factor)Key Concepts

- F is latent (i.e. unobserved, underlying) variable called a *factor*.
- X's are the observed variables.
- e_j is measurement error for X_j.
- λ_j is the "loading" on factor F for X_j.
- X's are *standardized* prior to beginning a factor analysis, i.e. converted to z-scores.
- F is also standardized, that is the standard deviation of F is 1 and the mean is 0.

Math associated with the ONE FACTOR model

- λ_j^2 is also called the "communality" of X_j in the one factor case
  (Standard notation for communality: h_j^2)
- For standardized X_j , $Corr(F, X_j) = \lambda_j$
- For standardized variables, the percentage of variability in X_j explained by F is λ_j^2.
- Interpretation of λ_j:
  - standardized regression coefficient (regression)
  - path coefficient (path analysis)
  - factor loading (factor analysis)
- $Corr(X_j , X_k) = \lambda_j\lambda_k$
- Note that the correlation between X_j and X_k is <u>completely</u> determined by the common factor F.
- Factor loadings (λ_j) are equivalent to correlation between factors and variables when only a SINGLE common factor is involved.

$$Y_1 = \lambda_1 F + \delta_1$$
$$Y_2 = \lambda_2 F + \delta_2$$
$$Y_3 = \lambda_3 F + \delta_3$$



Given all variables in standardized form, i.e. $var(Y_i)=var(F)=1$

- <u>Factor loadings</u>: λ_i

  $\lambda_i = corr(Y_i,F)$
- <u>Communality of Y_i</u>: h_i^2

  $h_i^2 = \lambda_i^2 = [corr(Y_i,F)]^2$

  =% variance of Y_i explained by F

- Uniqueness of Y_i: $1-h_i^2$

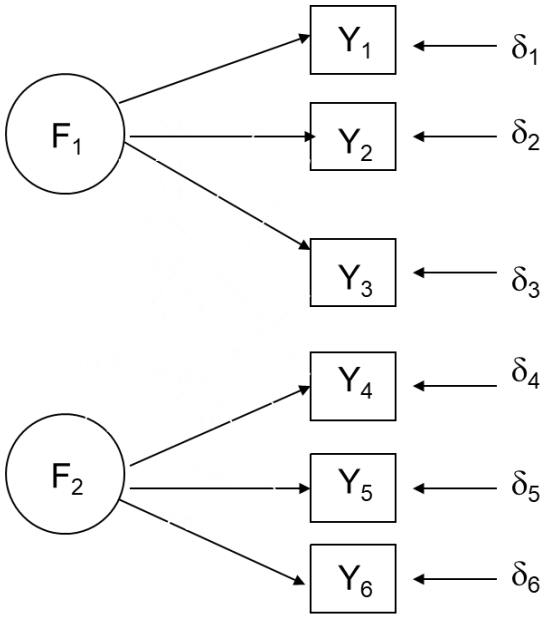
  = residual variance of Y_i

- Degree of factorial determination:

  $=\Sigma \lambda_i^2/n$, where n=# observed variables Y

Two-Common Factor Model (Orthogonal):

Model Specification



$$Y_1 = \lambda_{11}F_1 + \lambda_{12}F_2 + \delta_1$$
$$Y_2 = \lambda_{21}F_1 + \lambda_{22}F_2 + \delta_2$$
$$Y_3 = \lambda_{31}F_1 + \lambda_{32}F_2 + \delta_3$$
$$Y_4 = \lambda_{41}F_1 + \lambda_{42}F_2 + \delta_4$$
$$Y_5 = \lambda_{51}F_1 + \lambda_{52}F_2 + \delta_5$$
$$Y_6 = \lambda_{61}F_1 + \lambda_{62}F_2 + \delta_6$$

Matrix Notation

with n variables and m factors

$\mathbf{Y_{nx1} = \Lambda_{nxm}F_{mx1} + \delta_{nx1}}$

$$\begin{bmatrix} Y_1 \\ \vdots \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} \lambda_{11} & \cdots & \cdots & \lambda_{1m} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \lambda_{n1} & \cdots & \cdots & \lambda_{nm} \end{bmatrix}_{n\times m} \begin{bmatrix} F_1 \\ \vdots \\ F_m \end{bmatrix}_{m\times 1} + \begin{bmatrix} \delta_1 \\ \vdots \\ \vdots \\ \delta_n \end{bmatrix}_{n\times 1}$$

Given all variables in standardized form, i.e. $var(Y_i)=var(F_i)=1$;

AND **orthogonal** factors, i.e. $cov(F_1,F_2)=0$

Factor loadings: λ_{ij}

$\lambda_{ij} = corr(Y_i,F_j)$

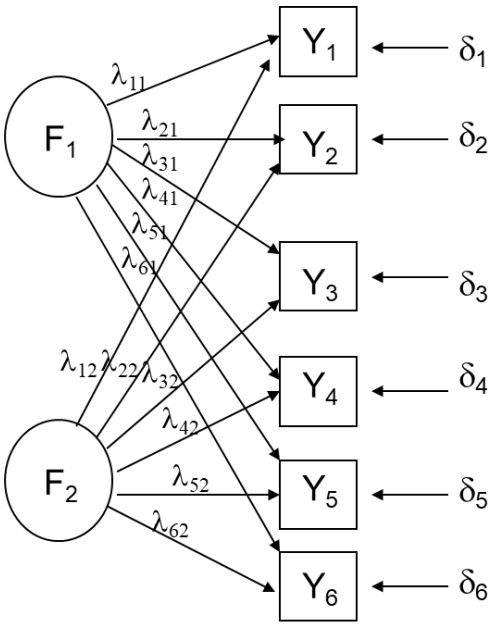
Communality of Y_i: h_i^2

$h_i^2 = \lambda_{i1}^2 + \lambda_{i2}^2$=% variance of Y_i explained by F_1 AND F_2

Uniqueness of Y_i: $1-h_i^2$

Degree of factorial determination:

$=\Sigma \lambda_{ij}^2/n$, n=# observed variables Y



1. Steps in Exploratory Factor Analysis

- Collect and explore data: choose relevant variables.
- Extract initial factors (via principal components)
- Choose number of factors to retain
- Choose estimation method, estimate model
- Rotate and interpret
- (a) Decide if changes need to be made (e.g. drop item(s), include item(s))
- (b) repeat (4)-(5)
- Construct scales and use in further analysis

2. Confirmatory Factor Analysis (CFA)

- Takes factor analysis a step further.
- We can "test" or "confirm" or "implement" a "highly constrained a priori structure that meets conditions of model identification"
- But be careful, a model can never be confirmed!!
- CFA model is constructed in advance
- number of latent variables ("factors") is pre-set by analyst (not part of the modeling usually)
- Whether latent variable influences observed is specified
- Measurement errors may correlate
- Difference between CFA and the usual SEM:
    - SEM assumes causally interrelated latent variables
    - CFA assumes interrelated latent variables (i.e. exogenous)

Identifiability Rules for CFA

Three-indicator rule (<u>sufficient, not necessary</u>)

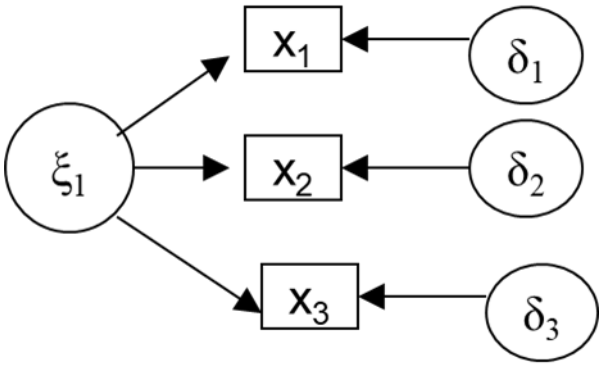
1) at least one factor
2) at least three indicators per factor
3) one non-zero element per row of Λ

(translation: each x only is pointed at by one LV)

1) non-correlated errors (Θ is diagonal)

(translation: no double-headed arrows between the δ's)

[Note: no condition about correlation of factors (no restrictions on Φ).]

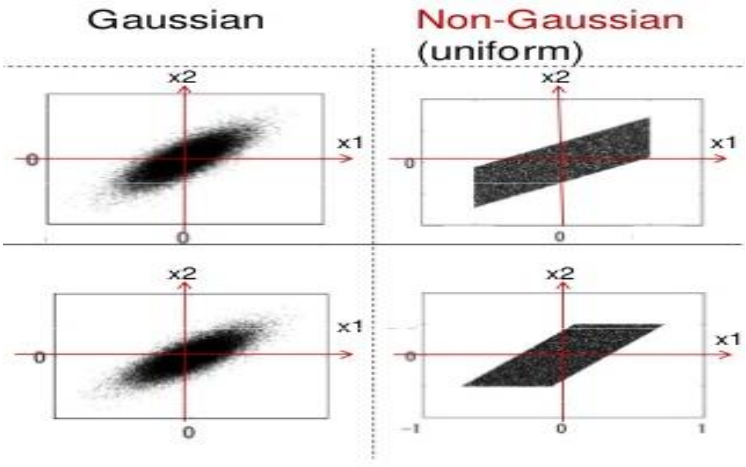


INDEPENDENT COMPONENT ANALYSIS

Independent Components Analysis

What is ICA?

"Independent component analysis (ICA) is a method for finding underlying factors or components from multivariate (multi-dimensional) statistical data. What distinguishes ICA from other methods is that it looks for components that are both statistically independent, and nonGaussian."



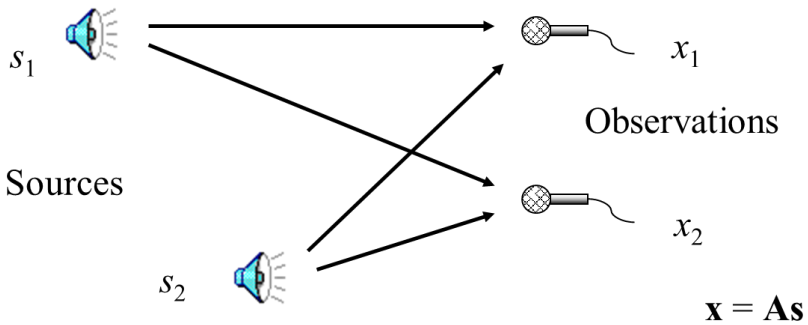
- Blind Signal Separation (BSS) or Independent Component Analysis (ICA) is the identification & separation of mixtures of sources with little prior information.
- Applications include:
    - Audio Processing
    - Medical data
    - Finance
    - Array processing (beamforming)
    - Coding
- Most applications where Factor Analysis and PCA is currently used.
- While PCA seeks directions that represents data best in a Σ|x0 - x|2 sense, ICA seeks such directions that are most independent from each other.
- Often used on Time Series separation of Multiple Targets

ICA mathematical approach

- "Given a set of observations of random variables $x_1(t)$, $x_2(t)$…$x_n(t)$, where t is the time or sample index, assume that they are generated as a linear mixture of independent components:
- $\boldsymbol{y=Wx}$, where $\boldsymbol{W}$ is some unknown matrix. Independent component analysis now consists of estimating both the matrix $\boldsymbol{W}$ and the $y_i(t)$, when we only observe the $x_i(t)$."

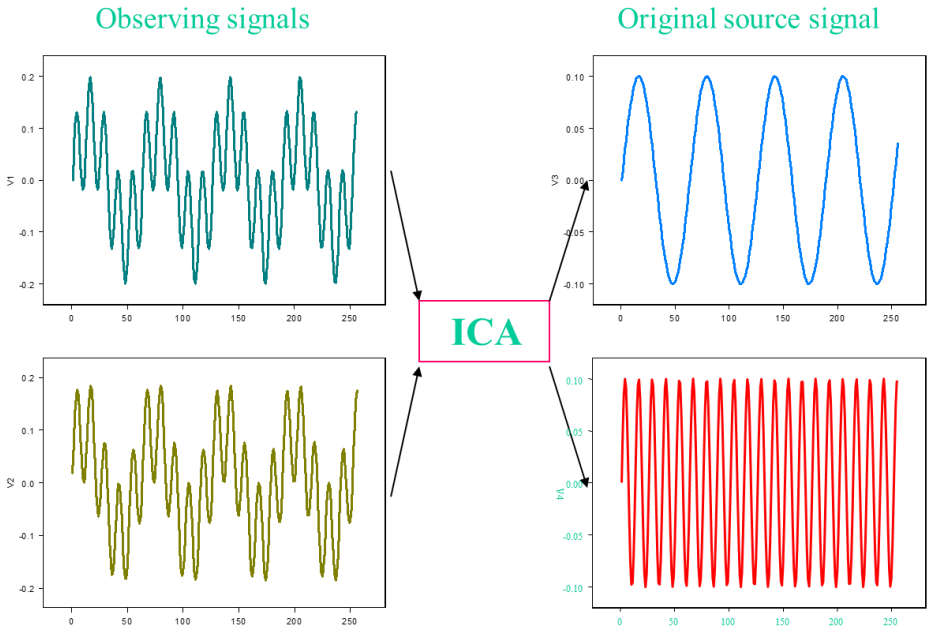
The simple "Cocktail Party" Problem

Mixing matrix A



n sources, m=n observations

Classical ICA (fast ICA) estimation



ICA Model (Noise Free)

- Use statistical "latent variables" system
- Random variable s_k instead of time signal
- $x_j = a_{j1}s_1 + a_{j2}s_2 + .. + a_{jn}s_n$, for all j

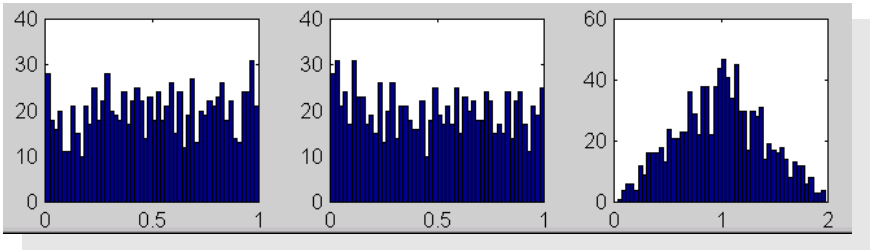
x = As

- IC's **s** are latent variables & are unknown AND Mixing matrix **A** is also unknown
- Task: estimate **A** and **s** using only the observeable random vector **x**
- Lets assume that no. of IC's = no of observable mixtures

  and **A** is square and invertible
- So after estimating A, we can compute $W=A^{-1}$ and hence

s = Wx = A^{-1}x

ICA Principal (Non-Gaussian is Independent)

- Key to estimating A is non-gaussianity
- The distribution of a sum of independent random variables tends toward a Gaussian distribution. (By CLT)
- 

  $f(s_1)$ $f(s_2)$ $f(x_1) = f(s_1 + s_2)$

- Where ***w*** is one of the rows of matrix ***W.***
- y is a linear combination of s_i, with weights given by z_i.
- Since sum of two indep r.v. is more gaussian than individual r.v., so $z^T s$ is more gaussian than either of s_i. AND becomes least gaussian when its equal to one of s_i.
- So we could take ***w*** as a vector which <u>maximizes the non-gaussianity</u> of $\boldsymbol{w^T x}$.
- Such a ***w*** would correspond to a ***z*** with only one non zero comp. So we get back the s_i.

Computing the pre-processing steps for ICA

0) Centring = make the signals centred in zero

$x_i \leftarrow x_i - E[x_i]$ for each i

1) Sphering = make the signals uncorrelated. I.e. apply a transform **V** to **x** such that Cov(**Vx**)=**I**

   // where Cov(**y**)=E[**yy**^{T}] denotes covariance matrix **V**=E[**xx**^{T}]$^{-1/2}$
   x$\leftarrow$**Vx** // for all t (indexes t dropped here)

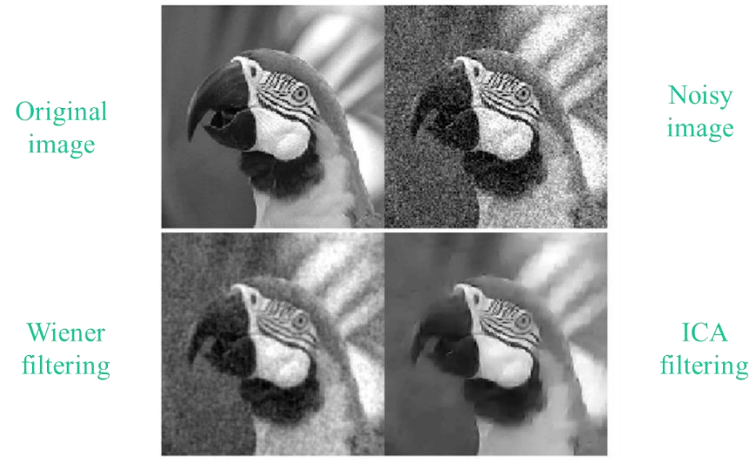
   // bold lowercase refers to column vector; bold upper to matrix

Scope: to make the remaining computations simpler. It is known that independent variables must be uncorrelated – so this can be fulfilled before proceeding to the full ICA

Application domains of ICA

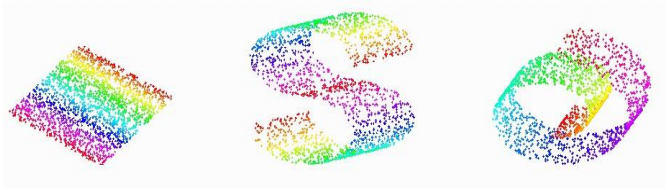
- Blind source separation (Bell&Sejnowski, Te won Lee, Girolami, Hyvarinen, etc.)
- Image denoising (Hyvarinen)
- Medical signal processing – fMRI, ECG, EEG (Mackeig)
- Modelling of the hippocampus and visual cortex (Lorincz, Hyvarinen)
- Feature extraction, face recognition (Marni Bartlett)
- Compression, redundancy reduction
- Watermarking (D Lowe)
- Clustering (Girolami, Kolenda)
- Time series analysis (Back, Valpola)
- Topic extraction (Kolenda, Bingham, Kaban)

Image denoising



LOCALLY LINEAR EMBEDDING

- Dimensionality reduction can be done through
    1. **feature selection**: only keeps the most relevant variables from the original dataset.
    2. **dimensionality reduction**: finds the smaller set of new variables, containing basically the same information as the original variables.
- Dimensionality reduction can also be categorized into:
- linear dimensionality reduction (e.g. PCA, SVD)
- non-linear dimensionality reduction (e.g. autoencoders, kernel PCA and others).
- **Methods :**
    - Isomaps
    - Locally Embedding Space (LLE)
    - Eigenmaps
- Manifolds : Definition: A manifold is a topological space that locally resembles Euclidean space **near each point**.



- Three examples of manifolds
- All three are two-dimensional data embedded in 3D
    - Linear, “S”-shape, “Swiss roll”
- For all three examples, we would like to recover:
    - Their two-dimensional representation
    - “Consistent” coordinates of the data in the 2D

Manifolds and local distances

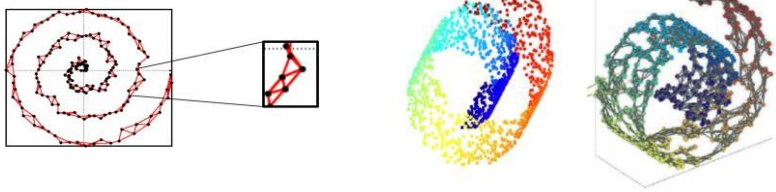
- In general, the distances induced by data (manifolds)

may not be Euclidean. That is the global distances don't respect the geometry.

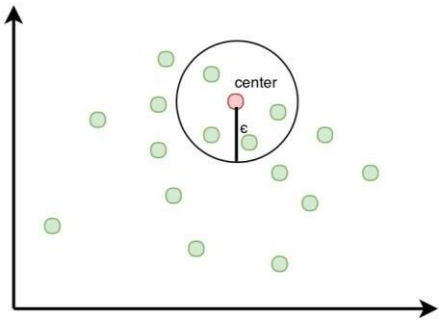
- Local distances can be still approximated with Euclidean distances
- **Idea for the dimensionality reduction:**
  - Define global distances/similarity in terms of local distances/similarity
  - Use these to define a low-dimensional embedding (low-dimensional representation) of the data
- **The idea can be implemented with the help of the Neighborhood graph**

Neighborhood Graph

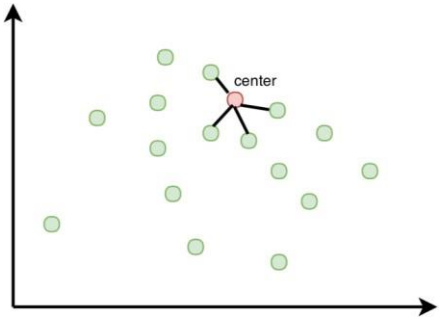
- A neighborhood graph
  - Vertices = data points
  - Edges and their weights reflect local similarity or local distances
  - Only points close to each other (neighbors) are connected



Approach 1: select and connect all points within the ε neighborhood



Approach 2: pick and connect k nearest neighbor points



- **Distance-based neighborhood graph:**
  - **Edge weight: Euclidean distance between two data points – *d* (x_i,x_j)**
- **Similarity-based neighborhood graph:**
  - **Edge weight:**

- Simple: $W_{ij} = 1$ if connected, 0 otherwise
- Kernel: $W_{i,j} = exp\left(-\frac{\|x_i - x_j\|^2}{t}\right)$ if connected, 0 otherwise

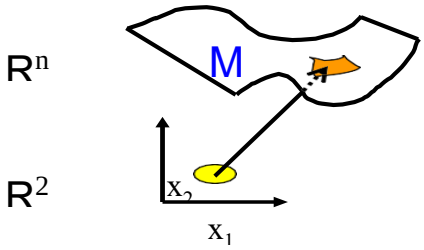
Three methods to define lower dimensional embedding of data instances:

- **Isomaps**
- **Locally Embedding Space (LLE)**
- **Eigenmaps**

- **All of these rely on the neighborhood graph connecting only data instances close to each other**

Locally Linear Embedding (LLE)

- Manifold Characteristics/Key Assumption
  - We expect each data point and its neighbors to lie on or close to a locally linear patch
  - But, how to combine all local patches together?

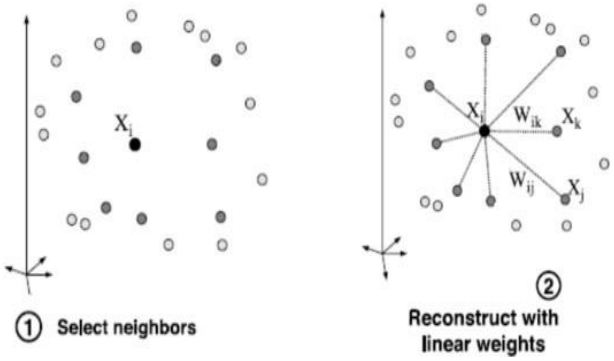


ssume that manifold is approximately "linear" when viewed locally (in a small neighborhood)

- A good projection should preserve this local geometric property as much as possible

LLE algorithm

- **Step 1:** Select neighbors for each data instance xi
- **Step 2:** Each data instance is written as a convex combination of its neighbors. Weights of the convex combination 'reconstruct' each point from its neighbors.

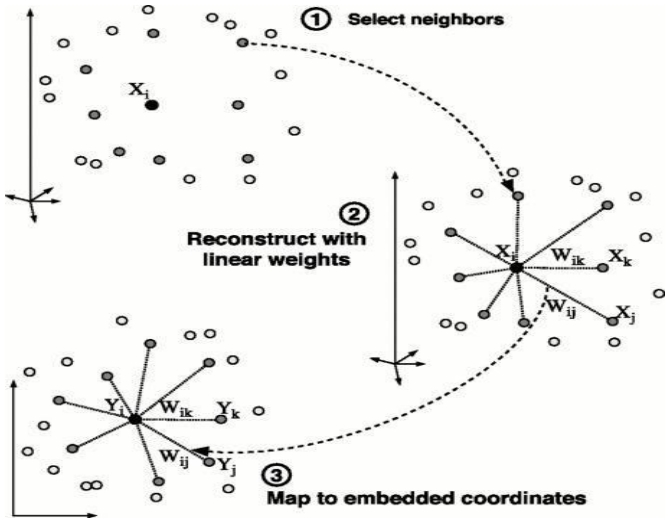


- **Step 3:** The weights chosen aim to minimize the reconstruction error.

$$\min \; X_i - \sum W_{ij} X_j$$

- **Step 4:** Map R^n to a low- dimensional embedding R^k

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}^{R^n} \rightarrow \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}^{R^k}$$



ISOMAP

Algorithm

- **Step 1:** Construct the neighborhood graph G with edge weights corresponding to local distances
- **Step 2:** compute the shortest distance between all pairs of points:

  – The shortest distance can be calculated via Floyd or Dijkstra algorithm.

.

$$\Delta := \begin{pmatrix} \delta_{1,1} & \delta_{1,2} & \dots & \delta_{1,l} \\ \delta_{2,1} & \delta_{2,2} & \dots & \delta_{2,l} \\ \vdots & \vdots & & \vdots \\ \delta_{l,1} & \delta_{l,2} & \dots & \delta_{l,l} \end{pmatrix}$$

- **Step 3:** Construct the k-dimensional embedding

  – Use classical MDS to find a k-dimensional embedding

$$\min \sum (\quad z - z \quad - \delta$$

All $z_1, z_2, \dots z_l$ are k-dimensional

- **Advantages:**
  - Non-linear dimensionality reduction
  - Non-iterative polynomial time algorithm
  - Guarantee of globally optimality:
    - For intrinsically Euclidean manifolds, a guarantee of asymptotic convergence to the true structure
    - The ability to discover manifolds of arbitrary dimensionality
- **Disadvantage:**
  - Sensitive to noise
  - Few free parameters

LEAST SQUARES OPTIMIZATION

Optimization is at the core of Machine Learning.

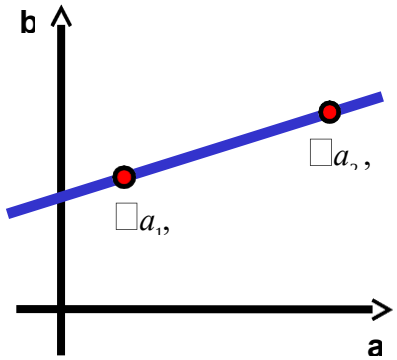
- Optimization, in very strict terms, is the process of finding the values for which your Cost Function gives a minimum value. For any Optimization problem with respect to Machine Learning, there can be either a numerical approach or an analytical approach.
- The numerical problems are Deterministic, meaning that they have a closed form solution which doesn't change.
- Hence it is also called time invariant problems. These closed form solutions are solvable analytically. But these are not optimization problems.
- Optimization comes in when you have words like 'Min' or 'max' of a function f(x) — the Objective Function or the Cost Function.
- This Objective Function could define anything with respect to the problem you are optimizing. It could be costs for a company, Losses for another or even revenue etc.
- This function will be optimal at a specific point X*. This X* is the optimal point.
- Hence your optimization problem could be — Find X* for which f(x) is minimum/maximum. This can also be written as argmin(f(x)) — argument where the function f(x) is minimum (or argmax(f(x) conversely).

Simple Example

- Example

– Line equation passing **two points** on the a-b plane

Unique solution



Line equation: $xa + yb + 1 = 0$

We know:

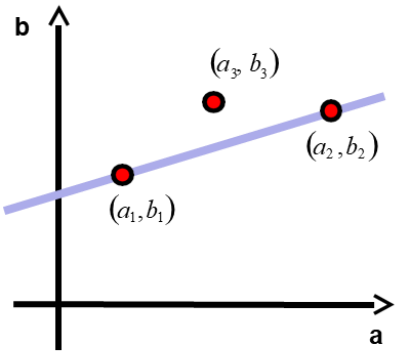
$$xa_1 + yb_1 + 1 = 0$$
$$xa_2 + yb_2 + 1 = 0$$

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{a_1 b_2 - a_2 b_1}\begin{bmatrix} b_2 & -b_1 \\ -a_2 & a_1 \end{bmatrix}\begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

- Line equation passing **three points** on the a-b plane

No unique solution



Line equation: $xa + yb + 1 = 0$

We know:

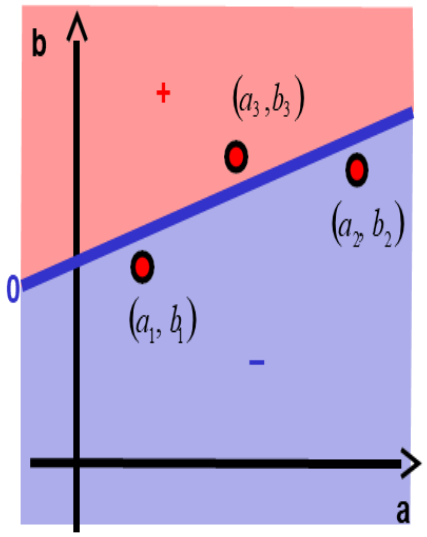
$$xa_1 + yb_1 + 1 = 0$$
$$xa_2 + yb_2 + 1 = 0$$
$$xa_3 + yb_3 + 1 = 0$$

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

- No exact solution,
- But we can compute the approximated solution.

- Passing all points would be impossible,
- Find the line that **minimizes distances from all points**

Objective Function



Point on the line: $xa + yb + 1 = 0$

Point out of the line:

$$xa + yb + 1 > 0,$$
$$xa + yb + 1 < 0$$

Objective function:

$$O(x, y) = \sum_{i=1}^{3} (xa_i + yb_i + 1)^2$$

- Problem description
  - Find the line coefficients (x, y) that minimize a sum of squared distances between the line and given points.
  - Mathematically,

$$\text{minimize} \sum_{i=1}^{3} (xa_i + yb_i + 1)^2$$

  - More compact description

$$(x_o, y_o) = \text{argmin}_{x,y} \sum_{i=1}^{3} (xa_i + yb_i + 1)^2$$

- Solution of the example

$$O(x,y)=\sum_{i=1}^{3}(xa_i+yb_i+1)^2$$

– The objective function has a parabolic shape

 The objective function would be minimized at the **zero gradient of the function**

$$\frac{\partial O}{\partial x}=2\sum_{i=1}^{3}a_i(xa_i+yb_i+1)=2\sum_{i=1}^{3}(xa_i^2+ya_ib_i+a_i)=0$$
$$\frac{\partial O}{\partial y}=2\sum_{i=1}^{3}b_i(xa_i+yb_i+1)=2\sum_{i=1}^{3}(xa_ib_i+yb_i^2+b_i)=0$$

MARKOV CHAIN MONTE CARLO METHODS

MCMC plays significant role in **statistics, econometrics, physics and computing science**.

- **Sampling from high-dimensional, complicated distributions**
- **Bayesian inference and learning**
  - *Marginalization*

$$p(x)=\int_Z p(x,z)dz$$

  - *Normalization*

$$p(x|y)=\frac{p(y|x)p(x)}{\int_X p(y|x)p(x)dx}$$

  - *Expectation.*

$$\mathbb{E}_{p(x)}(f(x))=\int_X f(x)p(x)\,dx$$

- **Global optimization**

$$\arg\max_x f(x)$$

The Monte Carlo principle

- **Our goal** is to estimate the following integral:

$$I(f) = \int_{\mathcal{X}} f(x)p(x)dx$$

- **The idea** of Monte Carlo simulation is to draw an i.i.d. set of samples

  $\{x^{(i)}\}$ from a target density $p(x)$ defined on a high-dim. space X.

  Let $x^{(1)}, \ldots, x^{(N)}$ be i.i.d samples from density p

Estimator:

$$I_N(f) = \frac{1}{N} \sum_{i=1}^{N} f(x^{(i)})$$

- Monte Carlo methods need sample from distribution *p(x)*.
- When *p(x)* has standard form, e.g. uniform or Gaussian, it is straightforward to sample from it using easily available routines.
- However, when this is not the case, we need to introduce more sophisticated sampling techniques. ⇒ MCMC sampling

SAMPLING

- The process of drawing a number of individual cases from a larger population
- A way to learn about a larger population by obtaining information from a subset of a larger population
- Example
  - Presidential polls are based upon samples of the population that might vote in an election
    - Why Sample?
      - Improve data quality
      - Obtain in-depth information about each subject rather than superficial data on all

Types of Probability Sampling

- Simple Random Sampling
    - A probability sample in which every member of a study population has been given an equal chance of selection
        - One way to draw a simple random sample, is to put all possibilities on paper, cut them up, and then draw a sample from a hat
        - Research Randomizer (http://randomizer.org)
- Systematic Sampling
    - A probability sampling procedure that involves selecting every *k*th element from a list of population elements, after the first element has been randomly selected
    - Example
        - Divide the total number of elements by the number you want in your sample 24/6 = 4
        - Randomly select a number between 1 and 4 and then select every 4^{th} element from that number
- Stratified Sampling
    - A probability sampling procedure that involves dividing the population in groups or strata defined by the presence of certain characteristics and then random sampling from each stratum
    - Example
        - If you had a population that was 10% women and you want a sample that is also 10% women

    Steps to draw a stratified random sample

        1. Group the study population into strata or into groups that share a given characteristic
        2. Enumerate each group separately
        3. Randomly sample within each strata

- Cluster Sampling
  - ✓ A probability sampling procedure that involves randomly selecting clusters of elements from a population and subsequently selecting every element in each selected cluster for inclusion in the sample
  - ✓ Cluster sampling is an option if data collection involves visits to sites that are far apart
- Multistage Sampling
  - ✓ A probability sampling procedure that involves several stages, such as randomly selecting clusters from a population, then randomly selecting elements from each of the clusters

PROPOSAL DISTRIBUTION

***Sample from distribution p*(*x*) that is only known up to a proportionality constant**

For example,

$p(x) \propto 0.3 \exp(-0.2x^2) + 0.7 \exp(-0.2(x-10)^2)$

Suppose that

- ❑ $p(x)$ is known up to a proportionality constant

  $p(x) \propto 0.3 \exp(-0.2x^2) + 0.7 \exp(-0.2(x-10)^2)$
- ❑ It is easy to sample from $q(x)$ that satisfies $p(x) \leq M\, q(x)$, $M < \infty$

Two techniques are:

1. Rejection sampling

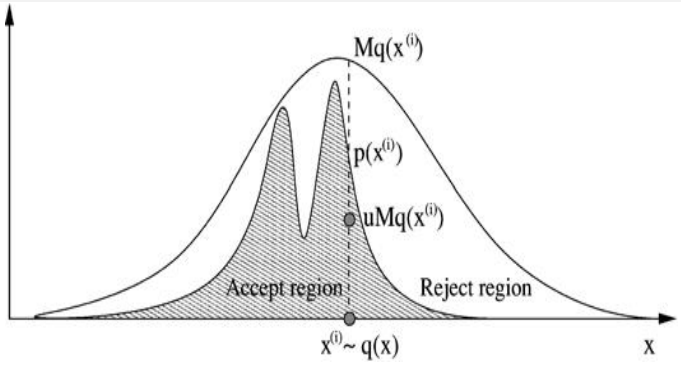
Theorem

- The accepted $x^{(i)}$ can be shown to be sampled with probability $p(x)$ (Robert & Casella, 1999, p. 49).

Severe limitations:

- It is not always possible to bound p(x)/q(x) with a reasonable constant M over the whole space X.
- If M is too large, the acceptance probability is too small.
- In high dimensional spaces it can be exponentially slow to sample points. (The points usually will be rejected)

1) $i = 1$
2) Repat until $i = N$
   a) Sample: $x^{(i)} \sim q(x)$ and $u \sim U_{[0,1]}$
   b) If $u < \frac{p(x^{(i)})}{Mq(x^{(i)})}$ then accept $x^{(i)}$ and $i = i + 1$,
   otherwise reject $x^{(i)}$.



2. Importance sampling

***Goal: Sample from distribution p*(x) that is only known up to a proportionality constant**

- ✓ Importance sampling is an alternative "classical" solution that goes back to the 1940's.
- ✓ Let us introduce, again, an arbitrary importance proposal distribution
- ✓ *q(x)* such that its support includes the support of *p(x)*.
- ✓ Then we can rewrite *I(f)* as follows:

$$\begin{aligned} I(f) &= \int f(x)p(x)dx \\ &= \int f(x)\frac{p(x)}{q(x)}q(x)dx \\ &= \int f(x)w(x)q(x)dx \end{aligned}$$

$$\begin{aligned} I(f) &= \int f(x)p(x)dx \\ &= \int f(x)\frac{p(x)}{q(x)}q(x)dx \\ &= \int f(x)w(x)q(x)dx \end{aligned} \qquad w(x) = \frac{p(x)}{q(x)}$$

Consequently,

⋆ if one can draw N i.i.d. $x^{(i)}$ $i = 1, \ldots, N$ from $q(x)$,

⋆ and evaluate $w(x^{(i)})$, then

$\Rightarrow$ possible Monte Carlo estimate of $I(f)$ is

$$\hat{I}_N(f) = \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)})w(x^{(i)})$$

- ❑ The optimal proposal is not very useful in the sense that it is not easy to sample from

$$q^*(x) = \frac{|f(x)|p(x)}{\int |f(x)|p(x)dx}$$

- High sampling efficiency is achieved when we focus on sampling from p(x) in the important regions where |f (x)|p(x) is relatively large; hence the name ***importance sampling***
- Importance sampling estimates can be **super-efficient**:

For a given function $f(x)$, it is possible to find a distribution $q(x)$ that yields an estimate with a lower variance than when using $q(x)= p(x)$!

- In high dimensions it is not efficient either…

MARKOV CHAIN

- Assume that the state space is finite:

$$\mathcal{X} = \{1, \ldots, k\}.$$

- 1-Step state transition matrix:

$$T_{ij} = P(X_{t+1} = j | X_t = i)$$

Lemma: The state transition matrix is stochastic:

$$\sum_j T_{ij} = 1 \ \forall i$$

t-Step state transition matrix:

$$Q_{ij} \doteq P(X_{k+t} = j | X_k = i)$$

Lemma:

$$P(X_{k+t} = j | X_k = i) = Q_{ij} = [T^t]_{ij}, \ \forall (k, i, j)$$

Markov chain:

$$P(X_{t+1} | X_t, \ldots, X_1) = P(X_{t+1} | X_t)$$

Homogen Markov chain:

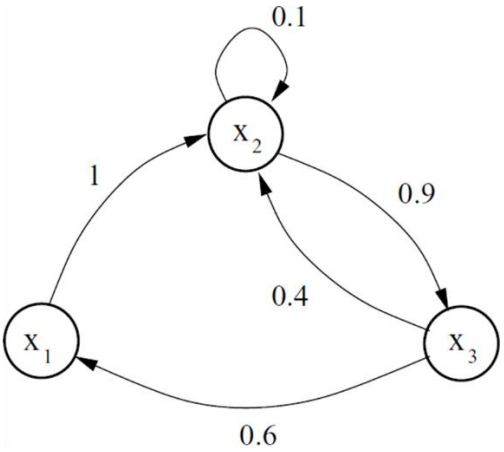
$$P(X_{t+1} | X_t) \text{ is invariant for all } t.$$

Markov Chains Example

Markov chain with three states ($s = 3$)

$$T = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0.1 & 0.9 \\ 0.6 & 0.4 & 0 \end{bmatrix}$$

Transition matrix



Transition graph

Our goal is to **find conditions** under which the Markov chain converges to a unique limit distribution (independently from its Starting state distribution)

Observation:

If this limiting distribution exists, it has to be the stationary distribution.

Definition

Irreducibility:

For each pairs of states *(i,j)*, there is a positive probability, starting in state *i*, that the process will ever enter state *j*.

= The matrix *T* cannot be reduced to separate smaller matrices

= Transition graph is connected.

Let {1,2,…,m} be the state space of a Markov chain that we can simulate.

It is possible to get to any state from any state.

Let $q(i,j) = p(j|i)$

Let $\{X_0, X_1, \ldots X_n, \ldots\}$ Markov chain be defined as follows:

The Hastings-

$$\Pr(X_n = j \,|X_{n-1} = i) =$$

1., from state i go to state j with prob. $q(i,j)$

2., $\begin{cases} \text{with prob } 1-\alpha(i,j) \text{ go back to state } i, \\ \text{with prob } \alpha(i,j) \text{ stay in state } j. \end{cases}$

No rejection: we use all X_1, X_2,... X_n, ... Metropolis Algorithm

Let $b_1, \ldots, b_m > 0$, and $B = \sum_{j=1}^{m} b_j$

Assume that m is so big, that it is difficult to calculate B.

Our goal:

Generate samples from the following **discrete** distribution:

We don't know *B*!

The main idea is to construct a time-reversible Markov chain with $(\pi_1,\ldots,\pi_m)$ limit distributions

Later we will discuss what to do when the distribution is continuous

1) Let Q be a Markov chain with $q(i,j) = P(j|i)$ state transition probabilites. Assume that we can sample from $q(i,j) = P(j|i)$.

2) Let $1 \leq k \leq m$ arbitrary, $n = 0$, and $X_0 = k$.

3) Sample X^* according to $P(X^* = j) = q(X_n, j)$, $j = 1, \ldots, m$ distribution.
(Go from X_n to state j using Markov chain Q)

4) Let $u \sim U_{[0,1]}$ (With prob $\alpha(i,j)$ stay in $X^* = j$)

5) If $u < \dfrac{b(X)q(X, X_n)}{b(X_n)q(X_n, X)} \Rightarrow X_{n+1} = X^*$
else $\Rightarrow X_{n+1} = X_n$ (Otherwise go back)

6) $n = n + 1$

7) Back to 3

Gibbs Sampling: Pseudo Code

1. We are in $\mathbf{x} = (x_1, \ldots, x_n) \in A$

2. Draw a random state $i \in \{1, \ldots, n\}$ with prob. $1/n$.

3. Sample x from $x \sim P(X_i = x | X_j = x_j, \forall j \neq i)$.

4. Let $\mathbf{y} = (x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_n)$

5. If

$(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_n) \in \mathbf{A} \Rightarrow x_i = x$, accept this new state
$(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_n) \notin \mathbf{A} \Rightarrow x_i$ stays in the old x_i

GRAPHICAL MODELS

- Graphical models are combination of probability theory and graph theory. They provide a natural tool for dealing with two problems that occur throughout applied mathematics and engineering -- uncertainty and complexity -- and in particular they are playing an increasingly important role in the design and analysis of machine learning algorithms. Fundamental to the idea of a graphical model is the notion of modularity -- a complex system is built by combining simpler parts. Probability theory provides the glue whereby the parts are combined, ensuring that the system as a whole is consistent, and providing ways to interface models to data.
- The graph theoretic side of graphical models provides both an intuitively appealing interface by which humans can model highly-interacting sets of variables as well as a data structure that lends itself naturally to the design of efficient general-purpose algorithms.
- Many of the classical multivariate probabalistic systems studied in fields such as statistics, systems engineering, information theory, pattern recognition and statistical mechanics are special cases of the general graphical model formalism -- examples include mixture models, factor analysis, hidden Markov models, Kalman filters and Ising models.
- The graphical model framework provides a way to view all of these systems as instances of a common underlying formalism.
- This view has many advantages -- in particular, specialized techniques that have been developed in one field can be transferred between research communities and exploited more widely. Moreover, the graphical model formalism provides a natural framework for the design of new systems.

Representation

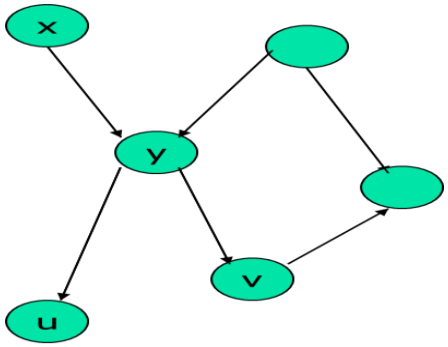
Graphical representation of probabilistic relationship between a set of random variables.

Variables are represented by nodes.

  - Binary events
  - Discrete variables
  - Continuous variables

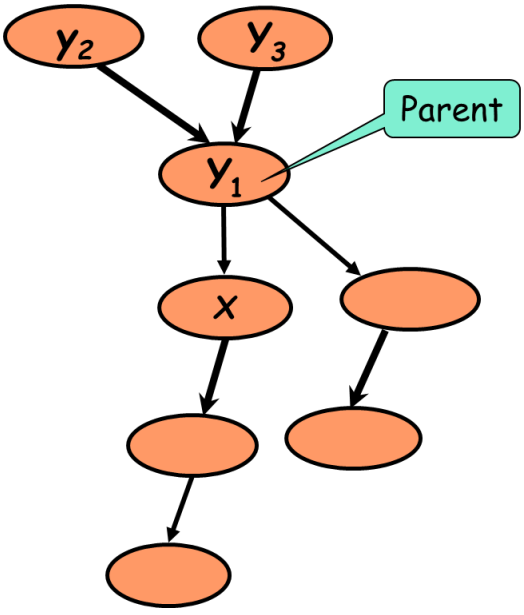
- Conditional (in)dependency is represented by (missing) edges.

- Directed Graphical Model: (Bayesian network)
- Undirected Graphical Model: (Markov Random Field)
- Combined: chain graph



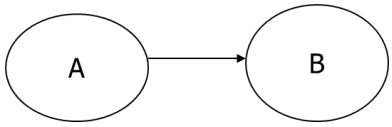
BAYESIAN NETWORKS

- Directed acyclic graphs (DAG).
- Directed edge means causal dependencies.
- For each variable X and parents pa(X) exists a conditional probability
- --P(X|pa(X))
- Joint distribution



Simple Case:

- That means: the value of B depends on A
- Dependency is described by the conditional probability P(B|A)
- Knowledge about A: prior probability P(A)
- Thus computation of joint probability of A and B : P(A,B)=P(B|A)P(A)



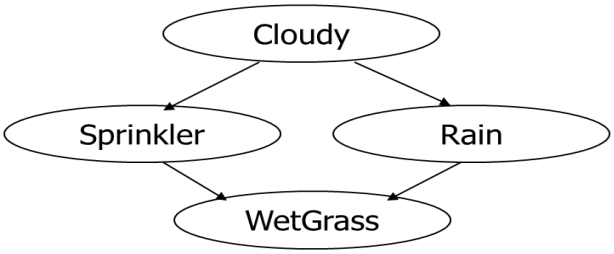
- From the joint probability, we can derive all other probabilities:
  - Marginalization: (sum rule)

$$P(A) = \sum_{B} P(A,B) \qquad P(B) = \sum_{A} P(A,B)$$

  - Conditional probabilities: (Bayesian Rule)

$$P(A \mid B) = \frac{P(A,B)}{P(B)} \quad P(B \mid A) = \frac{P(A,B)}{P(A)}$$

Simple Example



$$P(W = T) = \sum_{c,s,r} P(C = c, S = s, R = r, W = T)$$

$$P(S = T \mid W = T) = \frac{P(S = T, W = T)}{P(W = T)}$$

$$= \frac{\sum_{c,r} P(C = c, S = T, R = r, W = T)}{P(W = T)}$$

Bayesian Network

- Variables: $U = \{X_1, X_2, ...X_n\}$
- The joint probability of P(U) is given by

$$P(U) = P(X_{1,}...X_n) = P(X_n \mid X_{n-1},...X_1)P(X_{n-1} \mid X_{n-2},...X_1)...P(X_2 \mid X_1)P(X_1)$$

- If the variables are binary,

we need $O(2^n)$ parameters to describe P

- Key idea: use properties of independence.

1. Independent Random Variables
   - X is independent of Y iif for all values x,y $P(X = x \mid Y = y) = P(X = x)$
   - If X and Y are independent then

$$P(X_{,}Y) = P(X \mid Y)P(Y) = P(X)P(Y)$$

$$P(U) = P(X_{1,}...X_n) = P(X_1)P(X_2)...P(X_n)$$

   - Unfortunately, most of random variables of interest are not independent of each other

2. Conditional Independence

- A more suitable notion is that of conditional independence.
- X and Y are conditional independent given Z iff

  P(X=x|Y=y,Z=z)=P(X=x|Z=z) for all values x,y,z

- notion: I(X,Y|Z)
- P(X,Y,Z)=P(X|Y,Z)P(Y|Z)P(Z)=P(X|Z)P(Y|Z)P(Z)

3. Factored representation of joint probability

- Variables: $U = \{X_1, X_2, ...X_n\}$

- The joint probability of P(U) is given by

$$P(U) = P(X_{1,}...X_n) = P(X_n \mid X_{n-1},...X_1)...P(X_2 \mid X_1)P(X_1)$$

$$= \prod_{i=1}^{n} P(X_i \mid pa(X_i))$$

- the joint probability is product of all conditional probabilities

Serial Connection

- ***Calculate as before:***

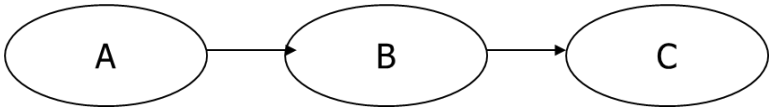
  --P(A,B)=P(B\|A)P(A)

  --P(A,B,C)=P(C\|A,B)P(A,B)

  =P(C\|B)P(B\|A)P(A)

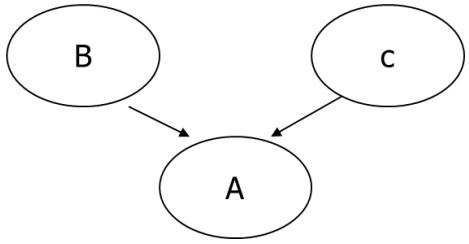
- ***I(C,A\|B).***

$$(P(U) = \prod_{i=1}^{n} P(X_i \mid pa(X_i)))$$



Converging Connection

$$(P(U) = \prod_{i=1}^{n} P(X_i \mid pa(X_i)))$$



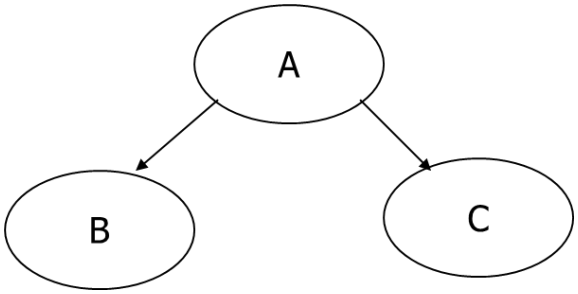
- Value of A depends on B and C:

  P(A|B,C)

- P(A,B,C)=P(A|B,C)P(B)P(C)

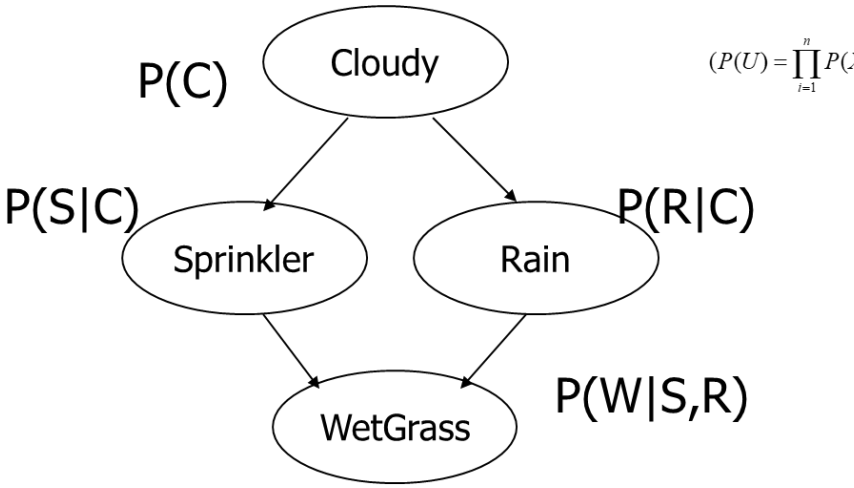
Diverging Connection

$$(P(U) = \prod_{i=1}^{n} P(X_i \mid pa(X_i)))$$



- B and C depend on A: P(B|A) and P(C|A)
- P(A,B,C)=P(B|A)P(C|A)P(A)
- I(B,C|A)

Wetgrass Example:



$$(P(U) = \prod_{i=1}^{n} P(X_i \mid pa(X_i)))$$

P(C,S,R,W)=P(W|S,R)P(R|C)P(S|C)P(C)

MARKOV RANDOM FIELDS

- Bayesian networks are a class of models that can compactly represent many interesting probability distributions. However, we have seen in the previous chapter that some distributions may have independence assumptions that cannot be perfectly represented by the structure of a Bayesian network.
- In such cases, unless we want to introduce false independencies among the variables of our model, we must fall back to a less compact representation (which can be viewed as a graph with additional, unnecessary edges). This leads to extra, unnecessary parameters in the model, and makes it more difficult to learn these parameters and to make predictions.
- There exists, however, another technique for compactly representing and visualizing a probability distribution that is based on the language of undirected graphs. This class of models (known as Markov Random Fields or MRFs) can compactly represent independence assumptions that directed models cannot represent.

Formal definition

A Markov Random Field (MRF) is a probability distribution pp over variables x1,…,xn defined by an *undirected* graph GG in which nodes correspond to variables xi. The probability p has the form

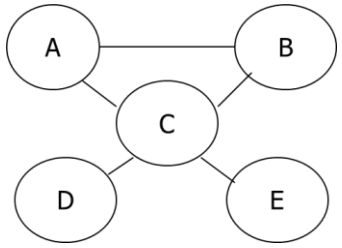
$$p(x_1, \ldots, x_n) = \frac{1}{Z} \prod_{c \in C} \phi_c(x_c),$$

where C denotes the set of *cliques* (i.e., fully connected subgraphs) of G, and each *factor* ϕ_c is a non-negative function over the variables in a clique. The *partition function*

$$Z = \sum_{x_1, \ldots, x_n} \prod_{c \in C} \phi_c(x_c)$$

Thus, given a graph G , our probability distribution may contain factors whose scope is any clique in G, which can be a single node, an edge, a triangle, etc. Note that we do not need to specify a factor for each clique. In our above example, we defined a factor over each edge (which is a clique of two nodes). However, we chose not to specify any unary factors, i.e., cliques over single nodes.

- Links represent symmetrical probabilistic dependencies
- Direct link between A and B: conditional dependency.
- Weakness of MRF: inability to represent induced dependencies.



- Global Markov property: x is independent of Y given Z iff all paths between X and Y are blocked by Z. (here: A is independent of E, given C)
- Local Markov property: X is independent of all other nodes given its neighbors. (here: A is independent of D and E, given C and B

More generally, MRFs have several advantages over directed models:

- They can be applied to a wider range of problems in which there is no natural directionality associated with variable dependencies.
- Undirected graphs can succinctly express certain dependencies that Bayesian nets cannot easily describe (although the converse is also true)

They also possess several important drawbacks:

- Computing the normalization constant Z requires summing over a potentially exponential number of assignments. We will see that in the general case, this will be NP-hard; thus many undirected models will be intractable and will require approximation techniques.
- Undirected models may be difficult to interpret.
- It is much easier to generate data from a Bayesian network, which is important in some applications.

HIDDEN MARKOV MODELS

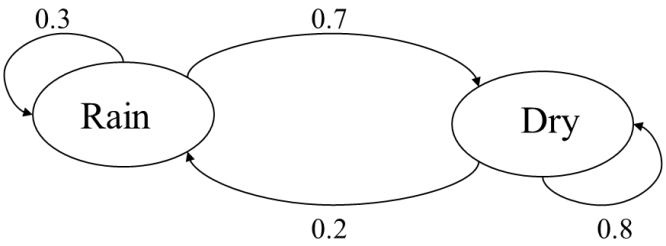
- Set of states: $\{s_1, s_2, \ldots, s_N\}$
- Process moves from one state to another generating a sequence of states : $s_{i1}, s_{i2}, \ldots, s_{ik}, \ldots$

- Markov chain property: probability of each subsequent state depends only on what was the previous state:

$$P(s_{ik} \mid s_{i1}, s_{i2}, \ldots, s_{ik-1}) = P(s_{ik} \mid s_{ik-1})$$

- To define Markov model, the following probabilities have to be specified: transition probabilities $a_{ij} = P(s_i \mid s_j)$ and initial probabilities $\pi_i = P(s_i)$

Example of Markov Model



- Two states : 'Rain' and 'Dry'.
- Transition probabilities: P('Rain'|'Rain')=0.3 , P('Dry'|'Rain')=0.7 , P('Rain'|'Dry')=0.2, P('Dry'|'Dry')=0.8
- Initial probabilities: say P('Rain')=0.4 , P('Dry')=0.6 .

Calculation of sequence probability

- By Markov chain property, probability of state sequence can be found by the formula:

$$P(s_{i1}, s_{i2}, \ldots, s_{ik}) = P(s_{ik} \mid s_{i1}, s_{i2}, \ldots, s_{ik-1}) P(s_{i1}, s_{i2}, \ldots, s_{ik-1})$$
$$= P(s_{ik} \mid s_{ik-1}) P(s_{i1}, s_{i2}, \ldots, s_{ik-1}) = \ldots$$
$$= P(s_{ik} \mid s_{ik-1}) P(s_{ik-1} \mid s_{ik-2}) \ldots P(s_{i2} \mid s_{i1}) P(s_{i1})$$

- Suppose we want to calculate a probability of a sequence of states in our example, {'Dry','Dry','Rain',Rain'}.

  P({'Dry','Dry','Rain',Rain'}) =

  P('Rain'|'Rain') P('Rain'|'Dry') P('Dry'|'Dry') P('Dry')=

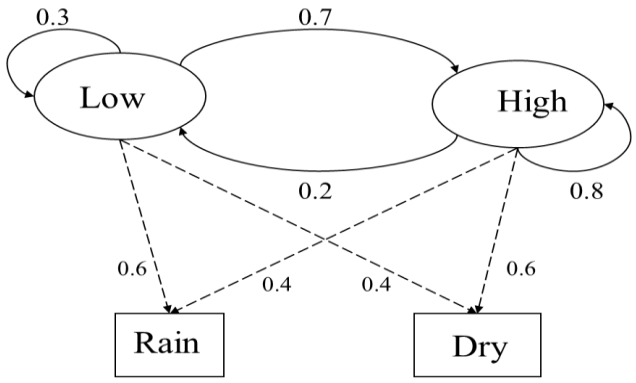
  = 0.3*0.2*0.8*0.6

Hidden Markov models.

- Set of states: $\{s_1, s_2, \ldots, s_N\}$
- Process moves from one state to another generating a sequence of states : $s_{i1}, s_{i2}, \ldots, s_{ik}, \ldots$
- Markov chain property: probability of each subsequent state depends only on what was the previous state:

  $P(s_{ik} \mid s_{i1}, s_{i2}, \ldots, s_{ik-1}) = P(s_{ik} \mid s_{ik-1})$
- States are not visible, but each state randomly generates one of M observations (or visible states) $\{v_1, v_2, \ldots, v_M\}$
- To define hidden Markov model, the following probabilities have to be specified: matrix of transition probabilities $A=(a_{ij})$, $a_{ij}= P(s_i \mid s_j)$, matrix of observation probabilities $B=(b_i(v_m))$, $b_i(v_m) = P(v_m \mid s_i)$ and a vector of initial probabilities $\pi=(\pi_i)$, $\pi_i = P(s_i)$. Model is represented by $M=(A, B, \pi)$.

Example of Hidden Markov Model



- Two states : ‘Low’ and ‘High’ atmospheric pressure.
- Two observations : ‘Rain’ and ‘Dry’.
- Transition probabilities: P(‘Low’|‘Low’)=0.3 , P(‘High’|‘Low’)=0.7 , P(‘Low’|‘High’)=0.2, P(‘High’|‘High’)=0.8
- Observation probabilities : P(‘Rain’|‘Low’)=0.6 , P(‘Dry’|‘Low’)=0.4 , P(‘Rain’|‘High’)=0.4 , P(‘Dry’|‘High’)=0.3 .
- Initial probabilities: say P(‘Low’)=0.4 , P(‘High’)=0.6 .

Calculation of observation sequence probability

- Suppose we want to calculate a probability of a sequence of observations in our example, {'Dry','Rain'}.
- Consider all possible hidden state sequences:

  P({'Dry','Rain'}) = P({'Dry','Rain'} , {'Low','Low'}) + P({'Dry','Rain'} , {'Low','High'}) + P({'Dry','Rain'} , {'High','Low'}) + P({'Dry','Rain'} , {'High','High'})

  where first term is :

  P({'Dry','Rain'} , {'Low','Low'})=

  P({'Dry','Rain'} | {'Low','Low'}) P({'Low','Low'}) =

  P('Dry'|'Low')P('Rain'|'Low') P('Low')P('Low'|'Low)

  = 0.4*0.4*0.6*0.4*0.3

Main issues using HMMs :

Evaluation problem. Given the HMM $M=(A, B, \pi)$ and the observation sequence $O=o_1 o_2 ... o_K$, calculate the probability that model M has generated sequence O .

- **Decoding problem.** Given the HMM $M=(A, B, \pi)$ and the observation sequence $O=o_1 o_2 ... o_K$, calculate the most likely sequence of hidden states s_i that produced this observation sequence O.
- **Learning problem.** Given some training observation sequences $O=o_1 o_2 ... o_K$ and general structure of HMM (numbers of hidden and visible states), determine HMM parameters $M=(A, B, \pi)$ that best fit training data.

  $O=o_1...o_K$ denotes a sequence of observations $o_k \in \{v_1,...,v_M\}$.

TRACKING METHODS.

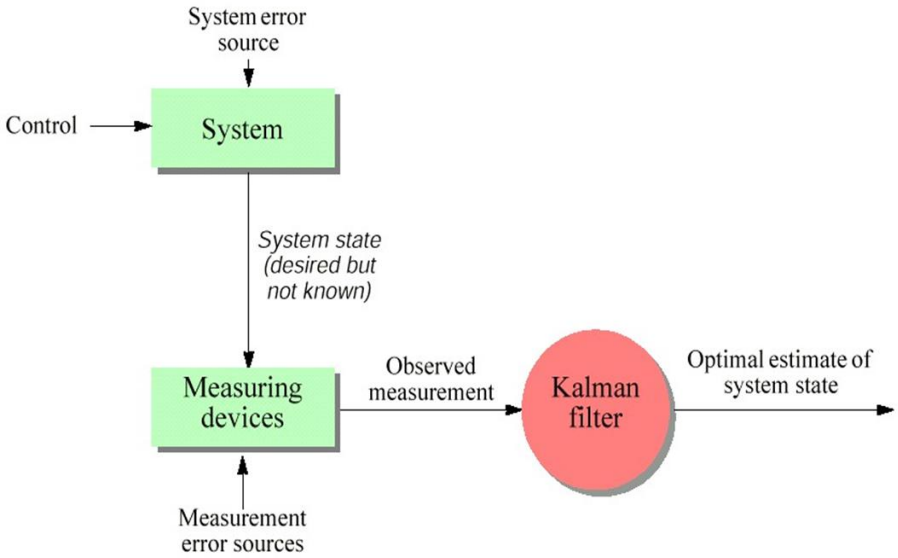
- There is a transition model that tells us how states change from one to another, and an observation model (also called the sensor model here) that tells us how states lead to observations.

- We write the process as a stochastic difference equation in x, which has n dimensions:
- $\mathbf{x}t+1 = \mathbf{A}\mathbf{x}t + \mathbf{B}\mathbf{u}t + \mathbf{w}t,.$
- *where At is an n×n matrix that represents the non-driven part of the underlying process,*
- *B is an n × l matrix that represents the driving force, and u is the l-dimensional driving*
- *force. w is the process noise, which is assumed to be zero mean with standard deviation Q.*
- The observations that we make are *m*-dimensional:

  $\mathbf{y}t = \mathbf{H}\mathbf{x}t + \mathbf{v}t,$

The Kalman Filter Algorithm

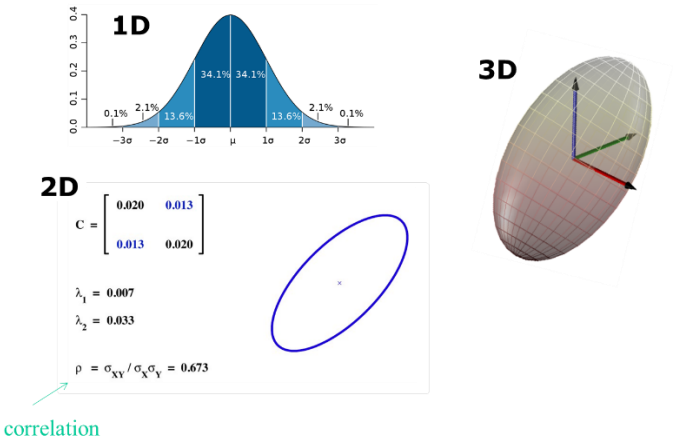
- Bayes filter with Gaussians
- Developed in the late 1950's
- Most relevant Bayes filter variant in practice
- Applications range from economics, wheather forecasting, satellite navigation to robotics and many more.
- The Kalman filter "algorithm" is a couple of matrix multiplications!



The Kalman Filter Algorithm

- Given an initial estimate $\mathbf{x}(0)$
- For each timestep:
  - **predict the next step**
    - predict state as $\hat{\mathbf{x}}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t$
    - predict covariance as $\hat{\mathbf{\Sigma}}_{t+1} = \mathbf{A}\mathbf{\Sigma}_t\mathbf{A}^T + \mathbf{Q}$
  - **update the estimate**
    - compute the error in the estimate, $\epsilon = \mathbf{y}_{t+1} - \mathbf{H}\mathbf{A}\mathbf{x}_{t+1}$
    - compute the Kalman gain using Equation (16.27)
    - update the state using Equation (16.28)
    - update the covariance using Equation (16.29)

Gaussians



Properties of Gaussians

- Univariate

$$\left.\begin{array}{l} X \sim N(\mu, \sigma^2) \\ Y = aX + b \end{array}\right\} \Rightarrow \quad Y \sim N(a\mu + b, a^2\sigma^2)$$

$$\left.\begin{array}{l} X_1 \sim N(\mu_1, \sigma_1^2) \\ X_2 \sim N(\mu_2, \sigma_2^2) \end{array}\right\} \Rightarrow p(X_1) \cdot p(X_2) \sim N\left(\frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2}\mu_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\mu_2, \quad \frac{1}{\sigma_1^{-2} + \sigma_2^{-2}} \right)$$

- Multivariate

$$\left.\begin{array}{l} X \sim N(\mu, \Sigma) \\ Y = AX + B \end{array}\right\} \Rightarrow \quad Y \sim N(A\mu + B, A\Sigma A^T)$$

$$\left.\begin{array}{l} X_1 \sim N(\mu_1, \Sigma_1) \\ X_2 \sim N(\mu_2, \Sigma_2) \end{array}\right\} \Rightarrow p(X_1) \cdot p(X_2) \sim N\left(\frac{\Sigma_2}{\Sigma_1 + \Sigma_2}\mu_1 + \frac{\Sigma_1}{\Sigma_1 + \Sigma_2}\mu_2, \quad \frac{1}{\Sigma_1^{-1} + \Sigma_2^{-1}} \right)$$

- We stay in the "Gaussian world" as long as we start with Gaussians and perform only linear transformations

PART-A

1. Define dimensionality reduction

In machine learning and statistics, dimensionality reduction or dimension reduction is the process of reducing the number of random variables under consideration via obtaining a set of principal variables. It can be divided into feature selection and feature extraction.

2. List the different ways to do dimensionality reduction

- Feature selection
- Feature derivation
- Clustering

3. Define feature selection.

Feature selection means looking through the features that are available and seeing whether or not they are actually useful, i.e correlated to the output variables.

4. Define feature derivation

Feature derivation means deriving new features from the old ones, generally by applying transforms to the dataset that simply change the axes 9coordinate system) of the graph by moving and rotating them, which can be written simply as a matrix that we apply to the data.

5. What is Linear Discriminant Analysis (LDA)?

Linear Discriminant Analysis (LDA) is most commonly used as dimensionality reduction technique in the pre-processing step for pattern-classification and machine learning applications. The goal is to project a dataset onto a lower-dimensional space with good class-separability in order avoid overfitting (“curse of dimensionality”) and also reduce computational costs.

6. What is the idea of principal component analysis?

The idea of principal component analysis is that it is a direction in the data with the largest variation. The algorithm first centres the data by subtracting off the mean, and then chooses the direction with the largest variation and places an axis in that direction, and then looks at the variation that remains and finds another axis that it orthogonal to the first and covers as much of the remaining variation as possible. It then iterates this until it has run out of possible axes.

7. What is factor analysis?

The idea of factor analysis is to ask whether the data that is observed can be explained by a smaller number of uncorrelated factors ot latent variables.

8. What is Box-Muller transform?

The Box–Muller transform, by George Edward Pelham Box and Mervin Edgar Muller is a pseudo-random number sampling method for generating pairs of independent, standard, normally distributed (zero expectation, unit variance) random numbers, given a source of uniformly distributed random numbers.

9. What are the two forms that the Box–Muller transform is commonly expressed?

The Box–Muller transform is commonly expressed in two forms. The basic form as given by Box and Muller takes two samples from the uniform distribution on the interval [0, 1] and maps them to two standard, normally distributed samples. The polar form takes two samples from a different interval, [−1, +1], and maps them to two normally distributed samples without the use of sine or cosine functions.

10. What is chain?

In probabilistic term, a chain is a sequence of possible states, where the probability of being in state s at time t is a function of the previous state.

11. What is Markov chain?

Markov chain is a chain with the Markov property. That is the probability at time t depends only on the state at t-1.

12. What is graphical model?

A graphical model or probabilistic graphical model (PGM) is a probabilistic model for which a graph expresses the conditional dependence structure between random variables. They are commonly used in probability theory, statistics—particularly Bayesian statistics—and machine learning.

13. What is Metropolis–Hastings Algorithm?

The Metropolis–Hastings algorithm is a Markov chain Monte Carlo (MCMC) method for obtaining a sequence of random samples from a probability distribution for which direct sampling is difficult. This sequence can be used to approximate the distribution (e.g., to generate a histogram), or to compute an integral (such as an expected value). Metropolis–Hastings and other MCMC algorithms are generally used for sampling from multi-dimensional distributions, especially when the number of dimensions is high

14. What is Gibbs sampling?

Gibbs sampling is one MCMC technique suitable for the task. The idea in Gibbs sampling is to generate posterior samples by sweeping through each variable (or block of variables) to **sample** from its conditional distribution with the remaining variables fixed to their current values.

15. What is Transition Probabilities?

The one-step transition probability is the probability of transitioning from one state to another in a single step. The **Markov** chain is said to be time homogeneous if the transition probabilities from one state to another are independent of time index.

16. What is Markov chain Monte Carlo?

Markov chain Monte Carlo (MCMC) methods are a class of algorithms for sampling from a probability distribution based on constructing a Markov chain that has the desired distribution as

its equilibrium distribution. The state of the chain after a number of steps is then used as a sample of the desired distribution. The quality of the sample improves as a function of the number of steps.

17. What are the two types of graphical models?

Generally, probabilistic graphical models use a graph-based representation as the foundation for encoding a complete distribution over a multi-dimensional space and a graph that is a compact or factorized representation of a set of independences that hold in the specific distribution. Two branches of graphical representations of distributions are commonly used, namely, Bayesian networks and Markov random fields. Both families encompass the properties of factorization and independences, but they differ in the set of independences they can encode and the factorization of the distribution that they induce

18. What is Markov Random Fields.?

A Markov Random Field (MRF) is a graphical model of a joint probability distribution. It consists of an undirected graph in which the nodes represent random variables

19. What is hidden Markov model?

Hidden Markov Model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (i.e. *hidden*) states. The hidden Markov model can be represented as the simplest dynamic Bayesian network.

20. What are the two methods of tracking?

- Kalman filter
- Particle filter

21. What is Kalman filter?

A Kalman filter is an optimal estimator- ie infers parameters of interest from indirect, inaccurate and uncertain observations. It is recursive so that new measurements can be processed as they arrive. (cf batch processing where all data must be present).

22. What is particle filters?

Particle filters or Sequential Monte Carlo (SMC) methods are a set of genetic, Monte Carlo algorithms used to solve filtering problems arising in signal processing and Bayesian statistical inference. The filtering problem consists of estimating the internal states in dynamical systems when partial observations are made, and random perturbations are present in the sensors as well as in the dynamical system.

PART-B

1. Explain the three different ways to do dimensionality reduction in detail.
2. Explain Linear Discriminant analysis in detail.
3. Explain principal component analysis algorithm in detail.
4. Explain Factor analysis in detail.
5. Explain sampling in MCMC.
6. Explain MCMC in detail.
7. Explain the proposal distribution in detail.
8. Explain Metropolis-Hastings in detail.
9 Explain Bayesian network in detail.
10. Explain Markov random field in detail.
11. Explain HMM in detail.
12. Explain Kalman filter algorithm in detail.