Pattern Recognition

Department of Computer Engineering
Razi University

Dr. Abdolhossein Fathi

References

- **Textbook:**
  - **Theodoridis S., K. Koutroumbas , "*Pattern Recognition*", Academic Press, 2008.**
  - **Webb A.R., K.D. Copsey, "*Statistical Pattern Recognition*", Wiley, 3rd Edition, 2011.**
  - **C. M. Bishop, "*Pattern Recognition and Machine Learning*", Springer, 2006.**
  - **K. Fukunaga, "*Introduction to Statistical Pattern Recognition*", Academic Press, 1990.**
  - **A. K. Jain, R. C. Dubes, "*Algorithms for Clustering Data*", Prentice Hall, 1988.**

Evaluation

- **Score Details:**
  - **Presentation** **15%**
  - **Implementation Project** **15%**
  - **Exercises** **10%**
  - **Final Exam** **60%**

- **Your Presentations are in New Applications, Models or Concepts of pattern recognition (Each of yours selects one new paper (only from ISI Journals with P.R. scope)).**
- **Your Projects are implementation and/or enhancement of the method in your papers (Enhancement has additional impact) .**

Contents

- Introduction to Pattern Recognition
- Review of Basic Mathematics
  - Linear Algebra
  - Statistics
  - Probability Theory
- Feature Extraction and Dimensionality Reduction
  - Feature Extraction
  - Feature Selection
  - Feature Reduction
- Classifiers and Decision Functions
  - Parametric Models
  - Non-parametric Methods
  - Probabilistic Graphical Models
  - Non-Bayesian Classifiers
- Clustering Methods

Introduction to Pattern Recognition

Pattern Recognition

"The real power of human thinking is based on recognizing patterns. The better computers get at pattern recognition, the more humanlike they will become." ***Ray Kurzweil, NY Times, Nov 24, 2003***

"The problem of searching for patterns in data is a fundamental one and has a long and successful history." *Bishop*

Pattern Recognition

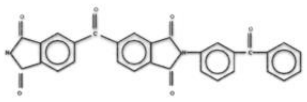
- ***Pattern recognition*** **is the scientific discipline whose goal is the classification of *objects* into a number of categories or *classes of patterns*.**

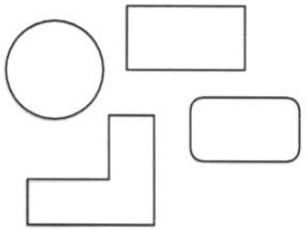
- **Depending on the application, these objects can be images or signal waveforms or any type of measurements that need to be classified.**

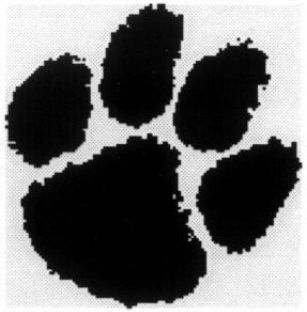
What is a Pattern?

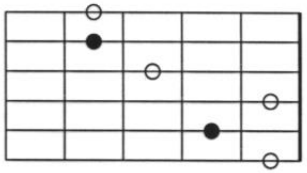
Razi University

"A pattern is the opposite of a chaos; it is an entity vaguely defined, that could be given a name or lable."





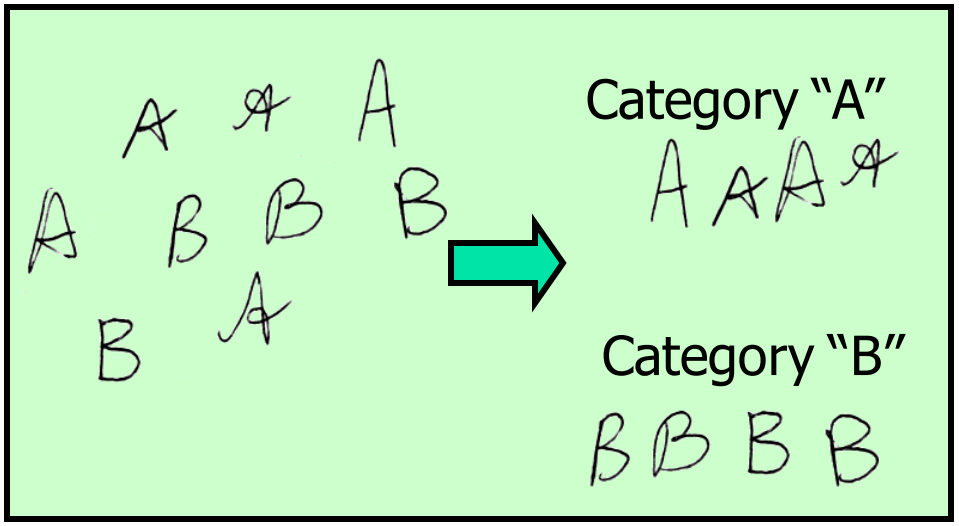




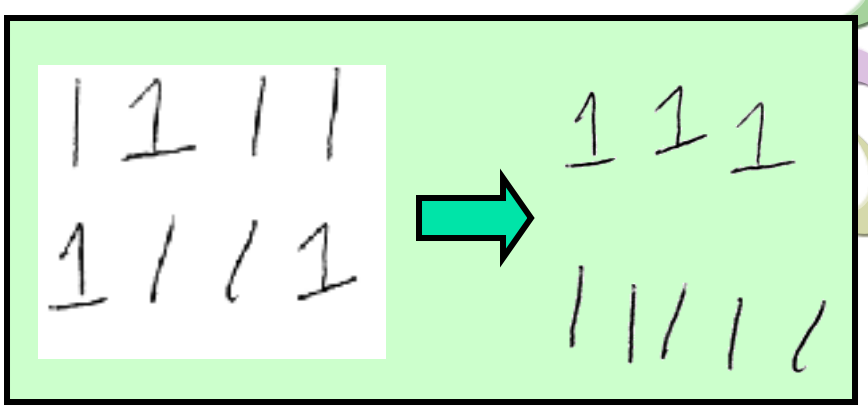
Recognition

Identification of a pattern as a member of a category we already know, or we are familiar with

- **Classification** (known categories)
- **Clustering** (learning categories)



Classification



Clustering

Pattern Recognition

- Given an input pattern, make a decision about the "category" or "class" of the pattern.
- Pattern recognition is needed in designing almost all automated systems.
- Other related disciplines: data mining, machine learning, computer vision, neural networks, statistical decision theory.
- This course will present "domain independent" techniques to solve P.R. problems and discuss their relative strengths and weaknesses.

Pattern Class

- A collection of similar (not necessarily identical) objects
- A class is defined by class samples (paradigms, exemplars, prototypes, training/learning samples)
- In almost all applications we have two challenges:
  - Intra-class variability
  - Inter-class similarity
- How do we define similarity?

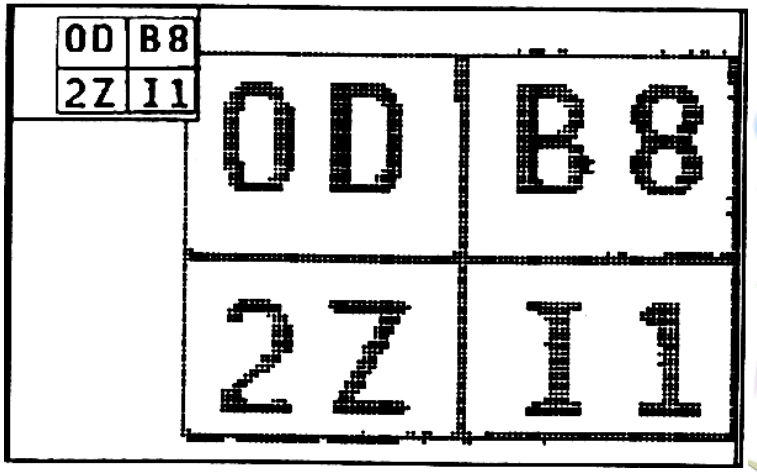
Intra-class Variability

The letter "T" in different typefaces

Same face under different expression, pose, illumination

Inter-class Similarity

Identical twins



Characters that look similar

Pattern Class Model

- A mathematical or statistical model (description) for each class (population);

  or other models: syntactic/structural, template

- Class description (class-conditional density) that is learned from samples

- Given a pattern, choose the best-fitting model for it; assign the pattern to the class associated with the best-fitting model

Pattern Recognition Applications

Problem	Input	Output
Speech recognition	Speech waveforms	Spoken words, speaker identity
Non-destructive testing	Ultrasound, eddy current, acoustic emission waveforms	Presence/absence of flaw, type of flaw
Detection and diagnosis of disease	EKG, EEG waveforms	Types of cardiac conditions, classes of brain conditions
Natural resource identification	Multispectral images	Terrain forms, vegetation cover
Aerial reconnaissance	Visual, infrared, radar images	Tanks, airfields
Character recognition (page readers, zip code, license plate)	scanned image	Alphanumeric characters

Pattern Recognition Applications

Problem	Input	Output
Identification and counting of cells	Slides of blood samples, micro-sections of tissues	Type of cells
Inspection (PC boards, IC masks, textiles)	Scanned image (visible, infrared)	Acceptable/unacceptable
Manufacturing	3-D images (structured light, laser, stereo)	Identify objects, pose, assembly
Web search	Key words specified by a user	Text relevant to the user
Fingerprint identification	Input image from fingerprint sensors	Owner of the fingerprint, fingerprint classes
Online handwriting retrieval	Query word written by a user	Occurrence of the word in the database

Pattern Recognition in Practice

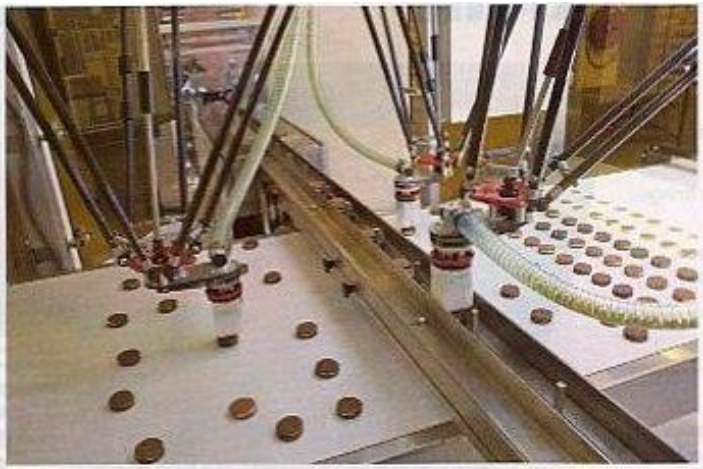
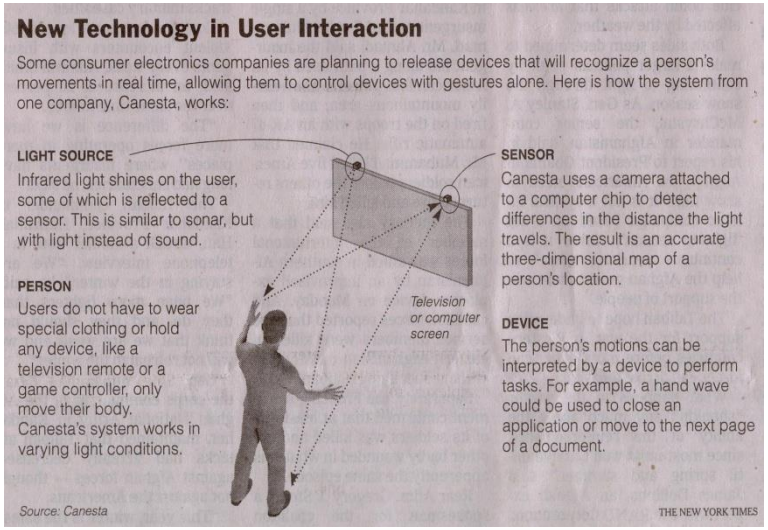


FIGURE 2. Single-camera solutions, such as this cookie packing system from Bosch Packaging Technology, use a geometric pattern-matching algorithm to determine the shape and location of cookies and then stack them in the appropriate section of the tray.

Vision System Design, Nov 2009

FIGURE 3. To automate the process of orange picking, Vision Robotics has proposed the development of a stereo camera-based system that will use multiple cameras placed at the end of multiaxis arms to create a virtual 3-D image of the entire orange tree.

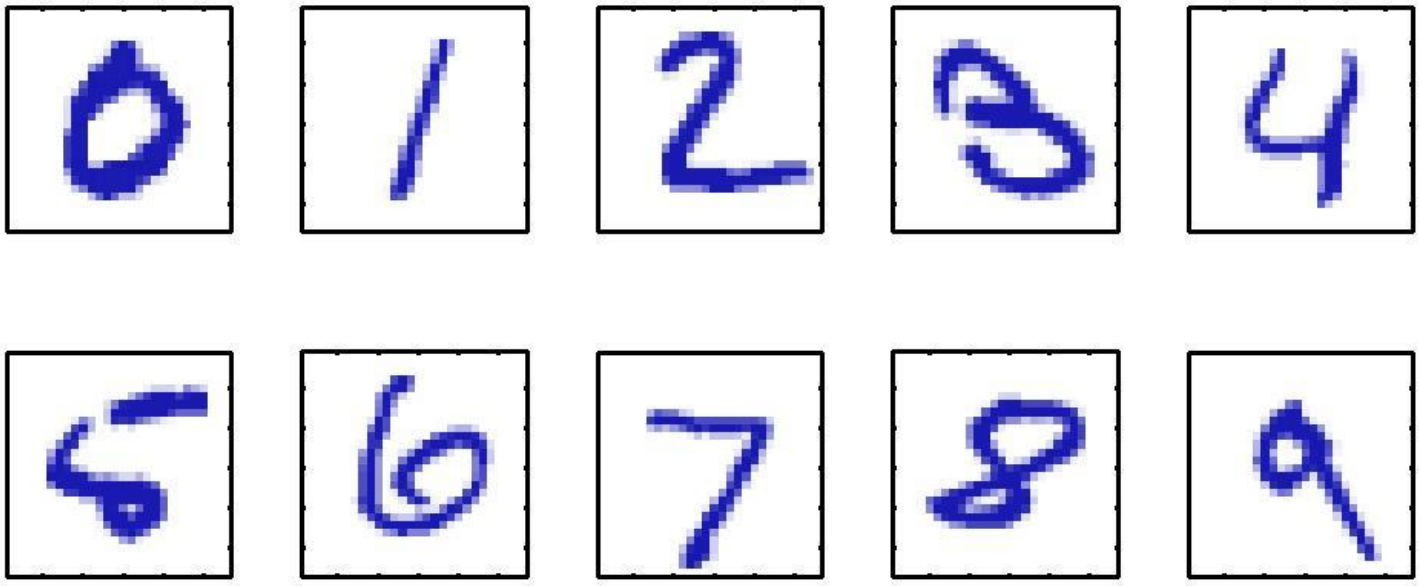


NY Times, Jan 12, 2010

Kinect for Xbox 360

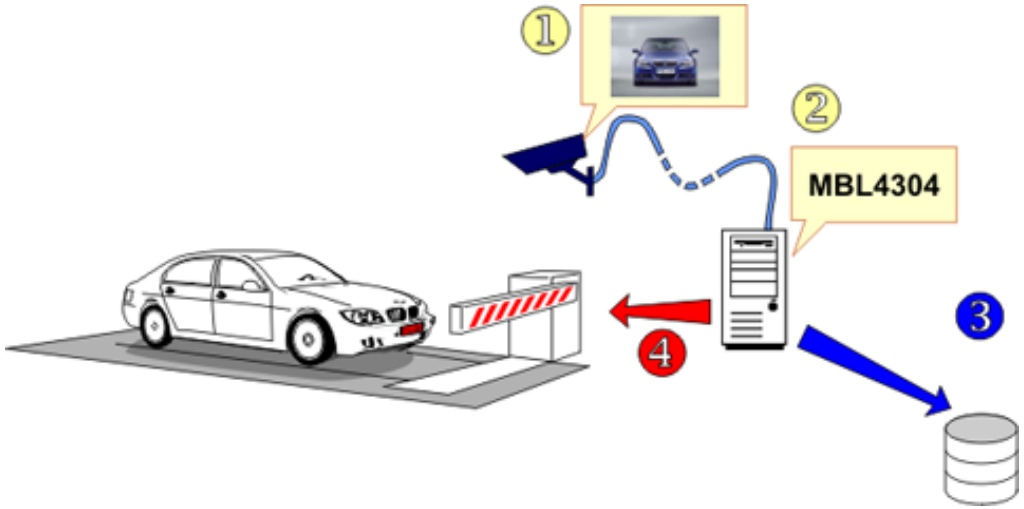
Digit Recognition

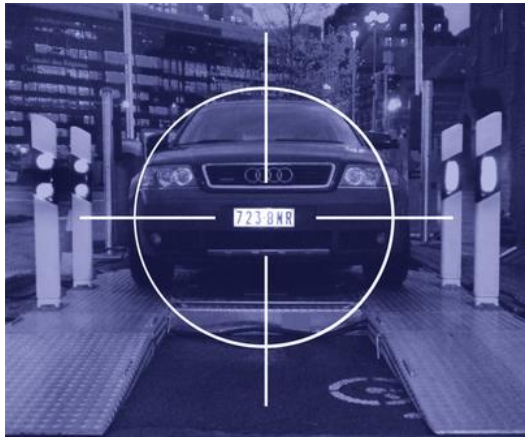
- **Machine printed**
- **Handwritten**
  - **Constrained**
  - **Unconstrained**



License Plate Reading System

- **Automatically detect and read the license plates on cars**

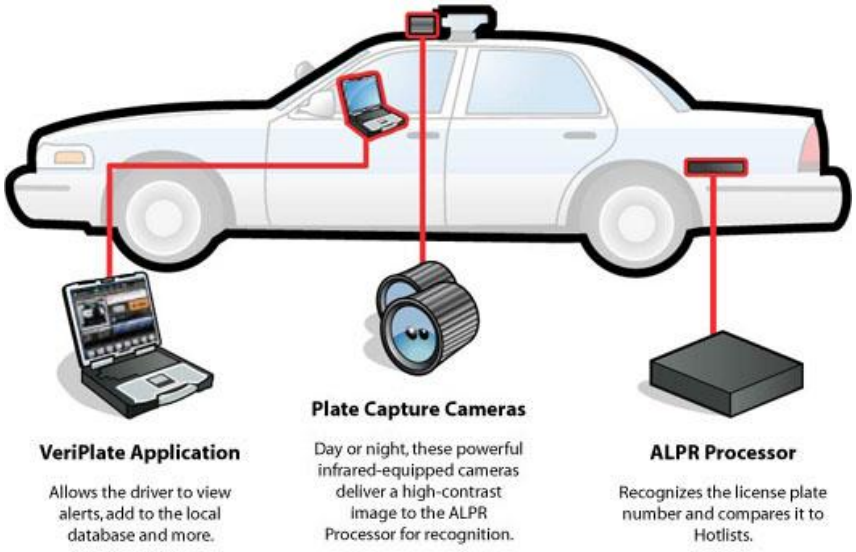




- **Modules: (i) acquisition, (ii) enhancement, (iii) segmentation, character recognition**
- **Should work in real time**

Applications

- Electronic toll collection
- Pay-per-use roads
- Homeland security & border control
- Finding stolen cars, unpaid tickets
- Automatic reporting of traffic violations



Steps

- Plate localization: isolate the plate in image
- Plate orientation and sizing; compensate for the skew of the plate and normalize the size
- Normalization: adjust image brightness & contrast
- Character segmentation: find individual characters
- Optical character recognition
- Syntactical/Geometrical analysis: check characters and their positions against country-specific rules

Challenges-I

- **Poor image resolution: plate is too far away or low-resolution camera**
- **Blurry images, particularly motion blur**
- **Poor lighting/low contrast: overexposure, reflection or shadows**
- **Viewpoint variation and occlusion (quite often dirt on the plate)**

Blurry image (left), lighting and viewpoint problem (middle) & occlusion (right)

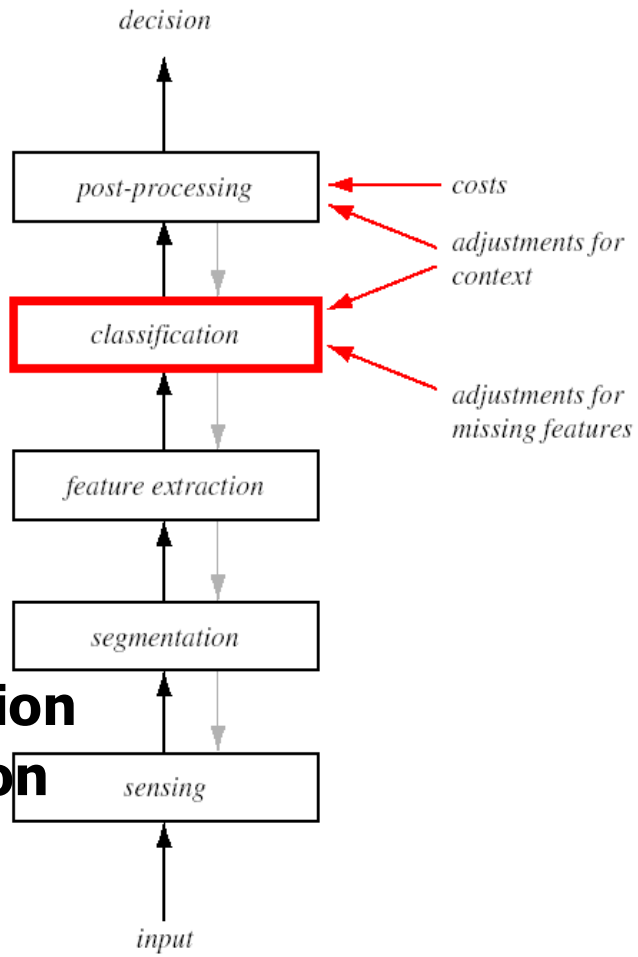
Challenges-II

- **Different fonts, popular for vanity plates**
- **Circumvention techniques**
- **Lack of coordination between countries or states. Two cars from different countries or states can have the same number but different design of the plate**

Examples of Michigan and Kentucky license plates

Pattern Recognition System

- **Domain-specific knowledge**
  - Acquisition, representation
- **Data acquisition**
  - camera, ultrasound, MRI,….
- **Preprocessing**
  - Image enhancement, segmentation
- **Representation**
  - Features: color, shape, texture,…
- **Classification or Decision making**
  - Statistical (geometric) pattern recognition
  - Syntactic (structural) pattern recognition
  - Artificial neural networks
- **Post-processing; use of context**

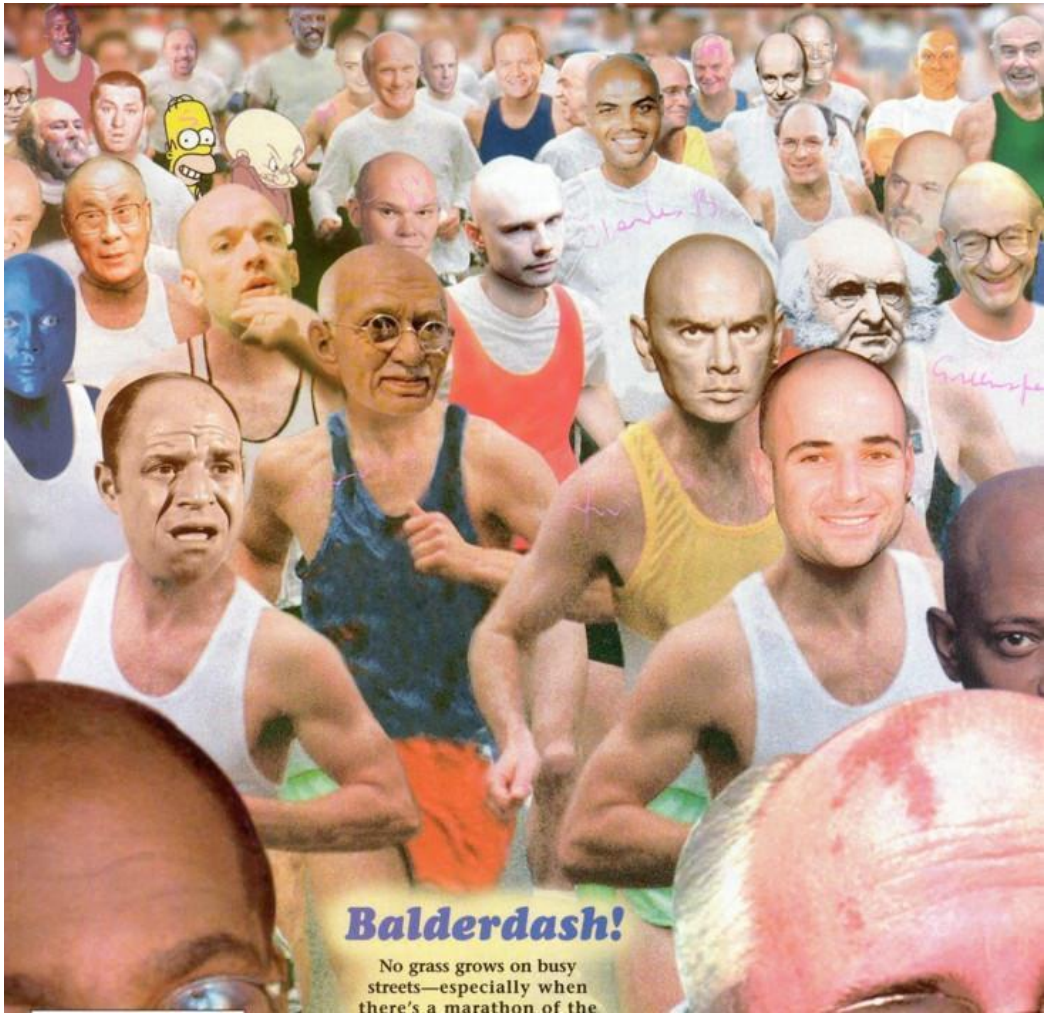


Pattern Recognition System

- **Challenges**
  - **Representation**
  - **Matching**
- **A pattern recognition system involves**
  - **Training or learning phase**
  - **Testing or evaluation phase**
    - **Error rate (Prob. of misclassification)**
    - **Speed (throughput)**
    - **Cost**
    - **Robustness**
    - **Reject option**

Segmentation: Face Detection

Segmentation: Face Detection



Games Magazine, September 2001

Difficulties of Representation

Are all these objects trees? Even a young child can answer correctly; a conventional computer, however, has enormous difficulty in doing so.


AMERICAN ELM


GINKGO


WEEPING WILLOW


SPRUCE

Although there is fair amount of regularity among the trees, there is also a major component of arboreal irregularity among them.


LARCH


CANYON LIVE OAK


TELEPHONE POLE


BIRCH

Any efficient program designed to recognize trees would essentially have to be a list of all types of trees, which cannot be done in a few lines of code.


MONTEREY CYPRESS


SCRUB PINE


DATE PALM

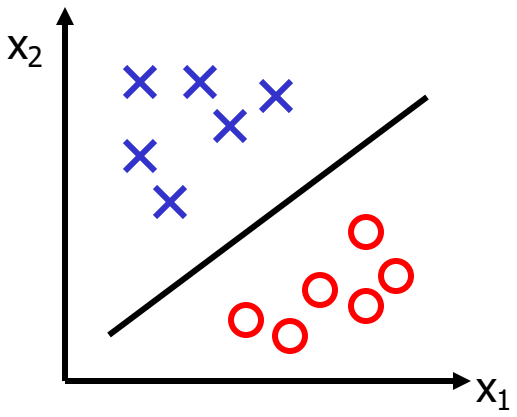

RED MANGROVE

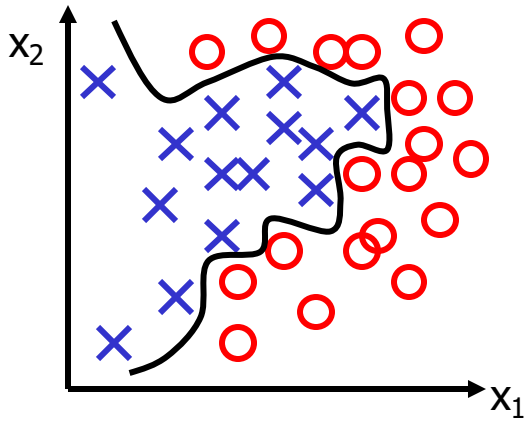
Good Representation

- A representation could consist of a vector of real-valued numbers, ordered list of attributes, parts and their relations….
- Should have some invariant properties (e.g., illumination, translation, rotation, scale…)
- Account for intra-class variations
- Ability to discriminate pattern classes of interest; low inter-class similarity
- Robustness to noise, occlusion,..
- Lead to simple matching or decision making strategies (e.g., linear decision boundary)
- Low measurement cost; real-time

Representation

- Each pattern is represented as a point in *d*-dimensional feature space
- Choice of features and their desired invariance properties are domain-specific





- Good representation implies (i) small intra-class variation, (ii) large interclass separation and (iii) simple decision boundary

Invariant Representation

- **Invariant to:**
  - **Translation**
  - **Rotation**
  - **Scale**
  - **Skew**
  - **Deformation**
  - **Color**

Not all invariant properties are needed for a given application

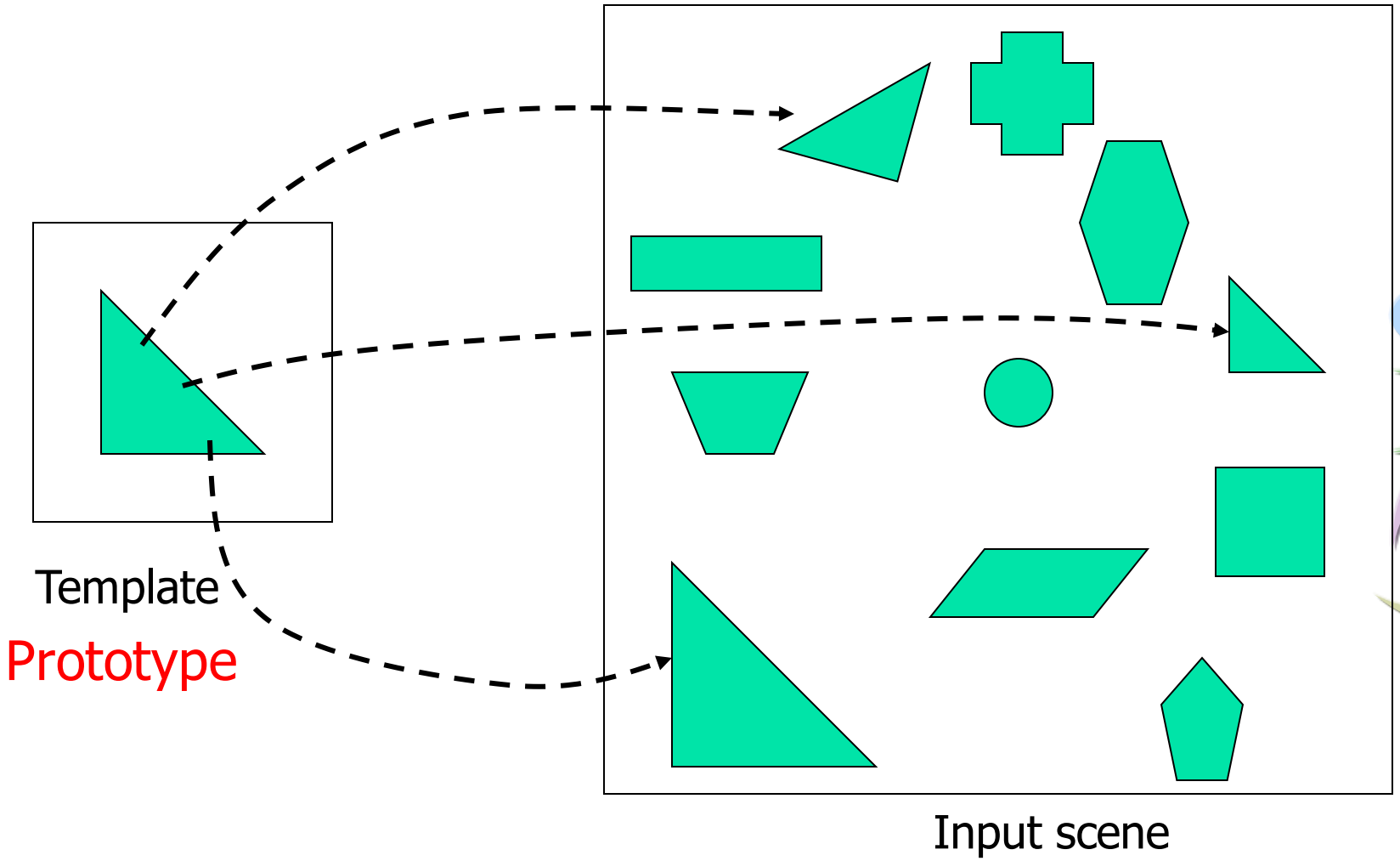
Feature Selection & Extraction

- **Selection vs. extraction**
- **How many and which subset of features to use in constructing the decision boundary?**
- **Some features may be redundant**
- **Curse of dimensionality—Error rate may in fact increase with too many features in the case of small number of training samples**

Models for Pattern Recognition

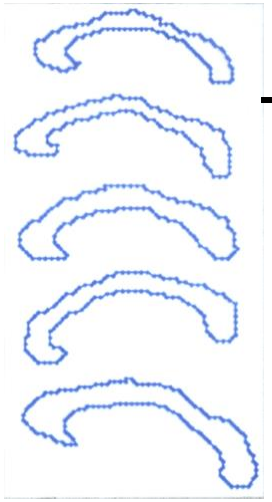
- Template matching
- Statistical (geometric)
- Syntactic (structural)
- Artificial neural networks
- Hybrid approach

Template Matching

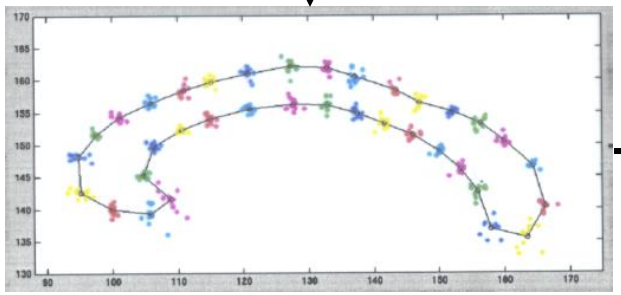


Deformable Template: Corpus Callosum Segmentation

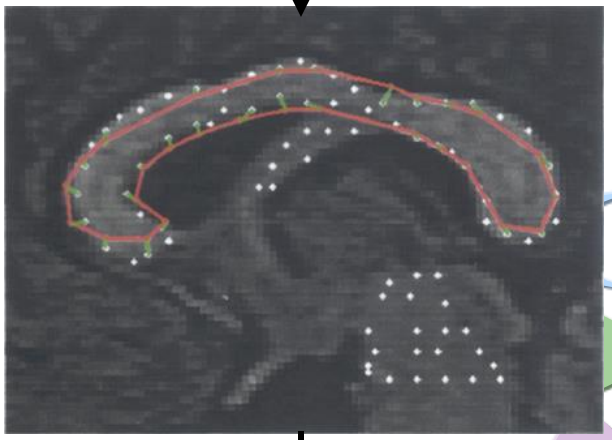
Shape training set



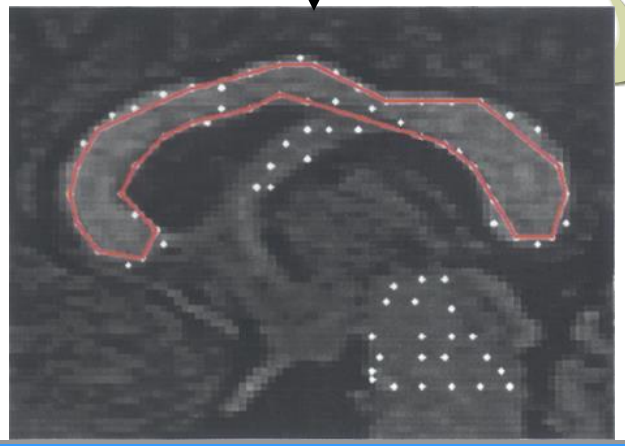
Prototype and variation learning

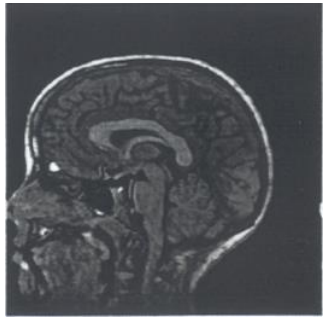


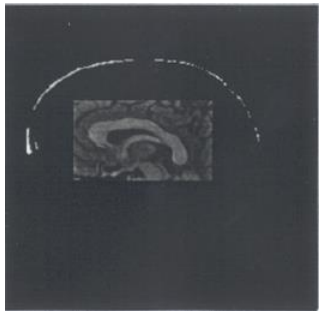
Prototype registration to the low-level segmented image



Prototype warping

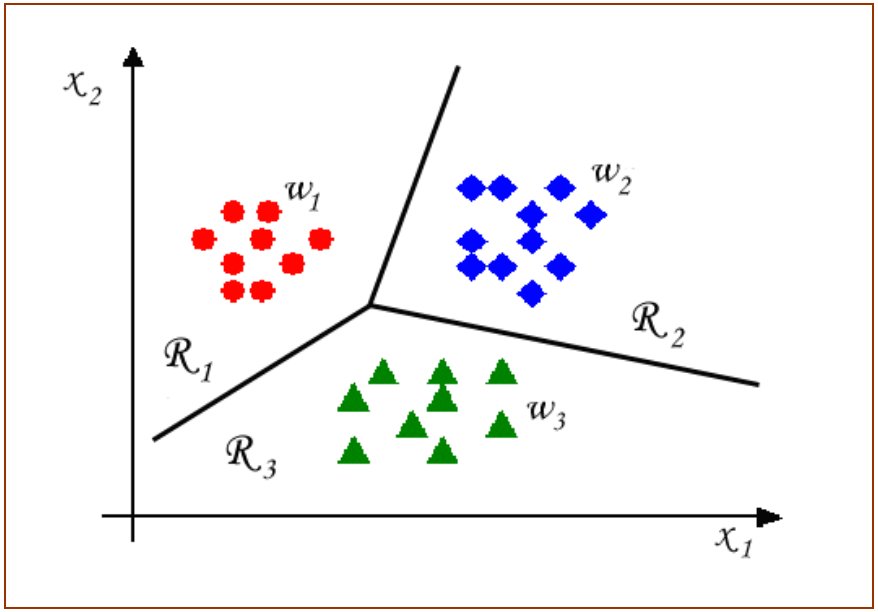






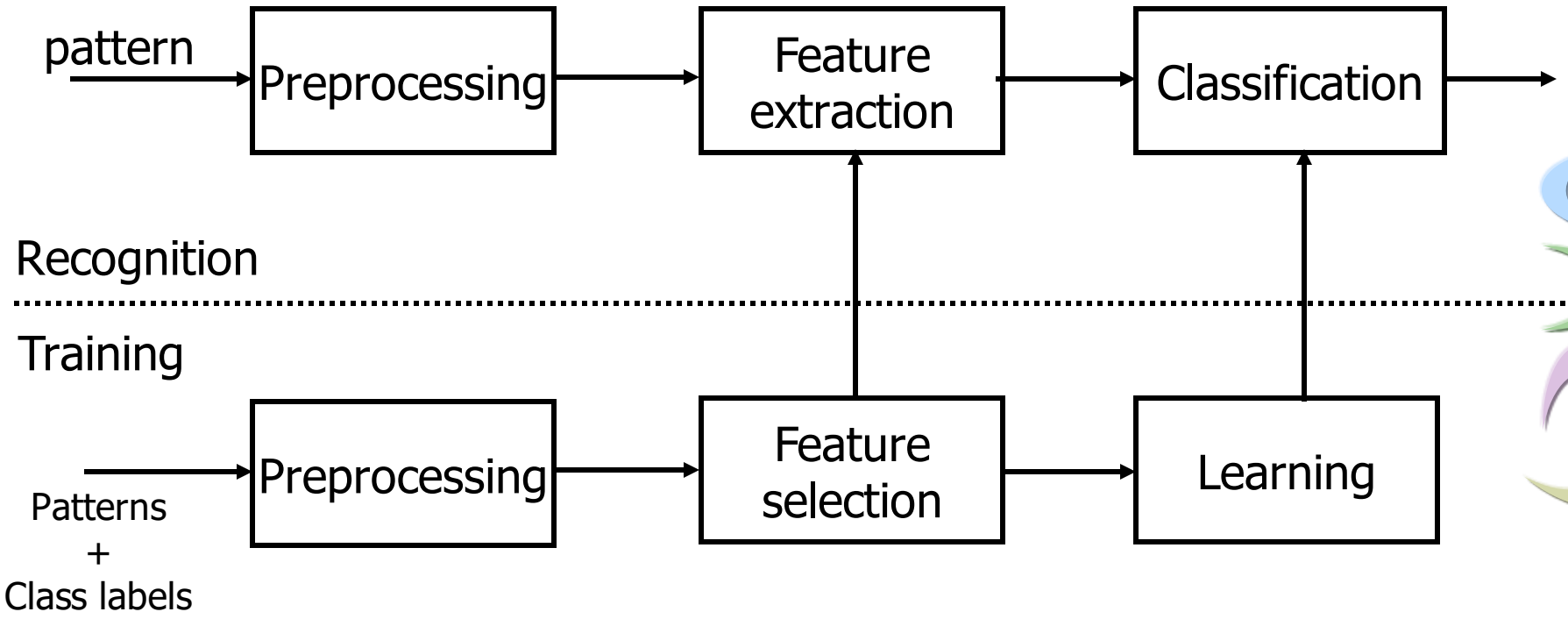
Statistical Pattern Recognition

- **Patterns represented in a feature space**
- **Statistical model for pattern generation**

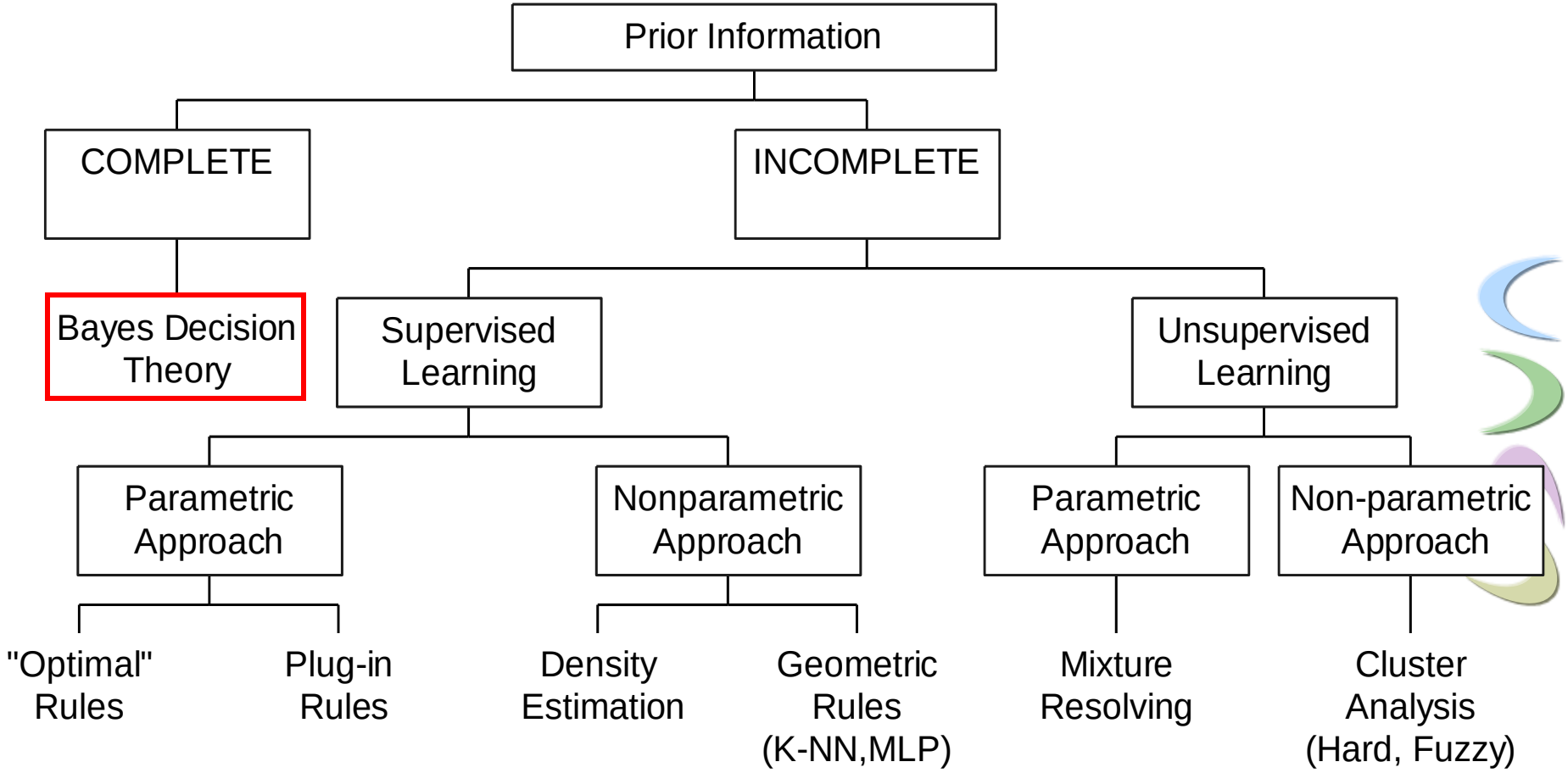


- **Given training patterns from each class, goal is to partition the feature space based on statistical parameters (mean, variance, *pdf*, …).**

Statistical Pattern Recognition

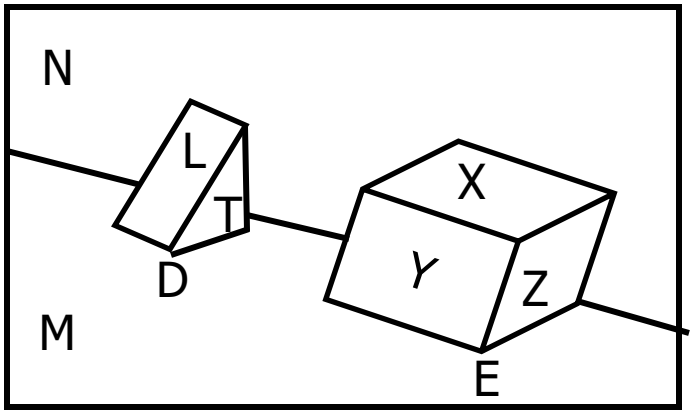


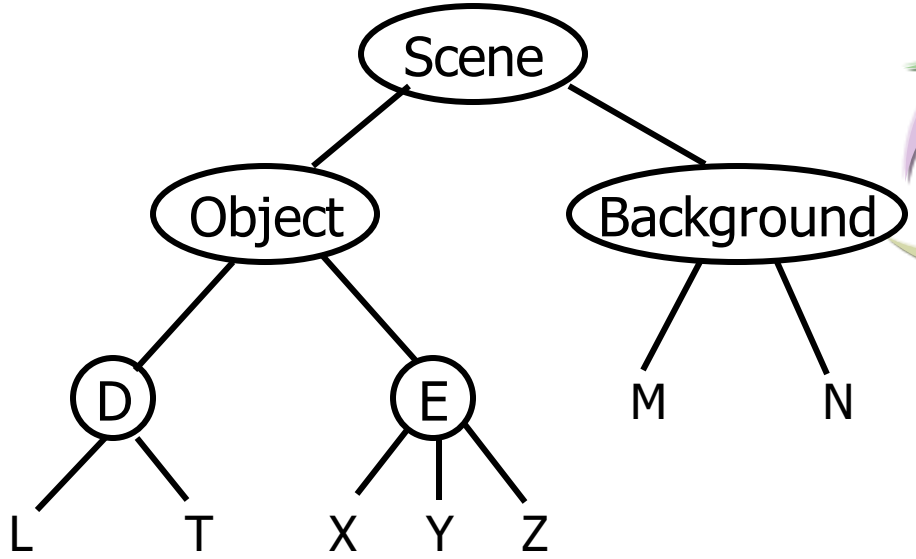
Statistical Pattern Recognition



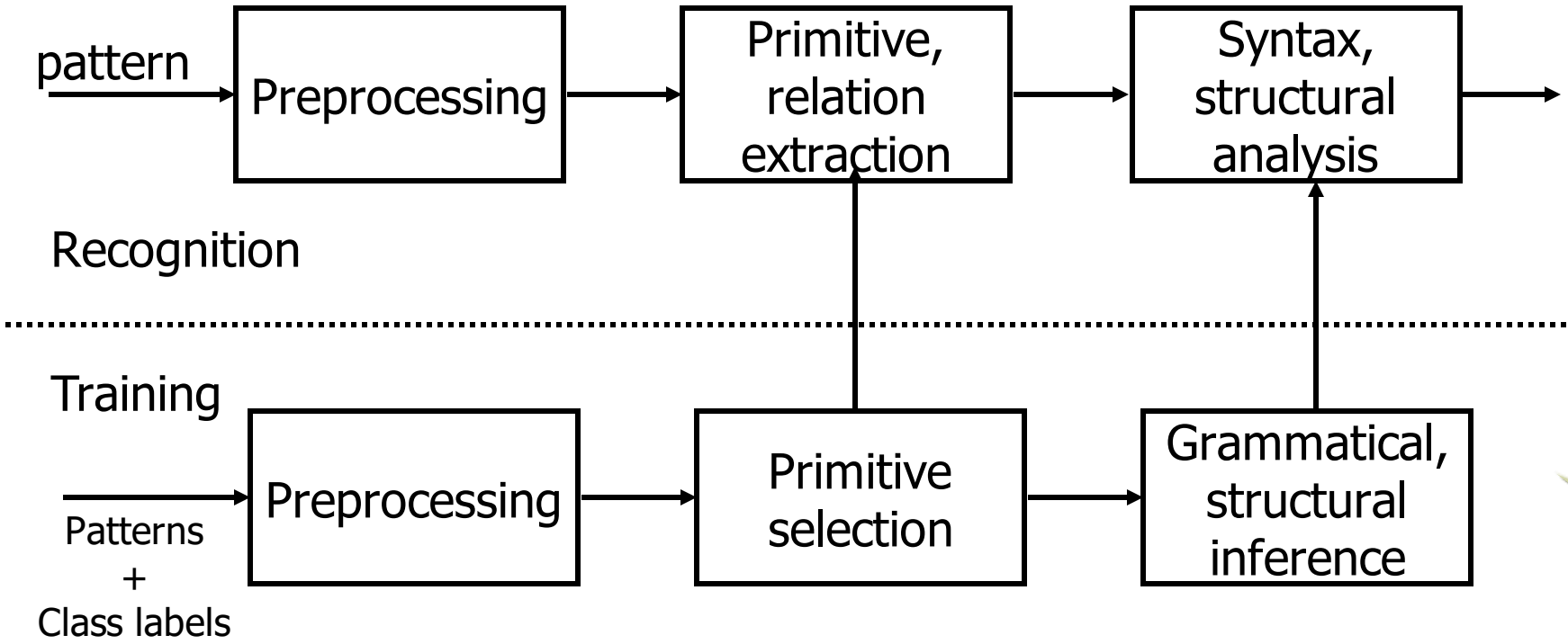
Structural Patten Recognition

- **Instead of describing an object in terms of a feature vector, describe it by its structure**
- **Describe complicated objects in terms of simple primitives and their relationship; parts based representation (represent face as eyes, mouth, nose,…)**





Structural Pattern Recognition



Example 1:
Fish Classification; Salmon v. Sea Bass

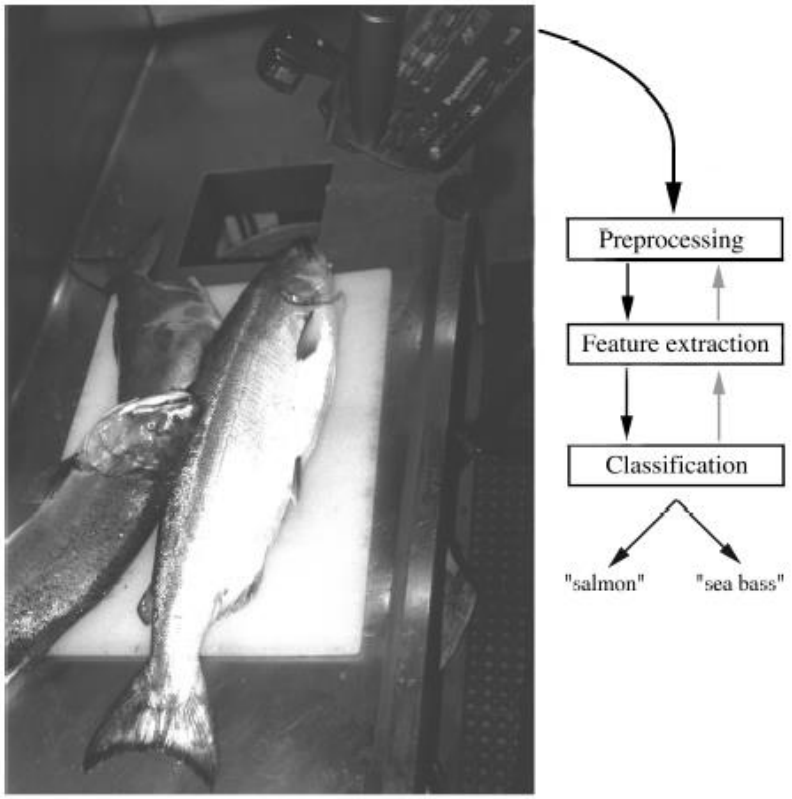
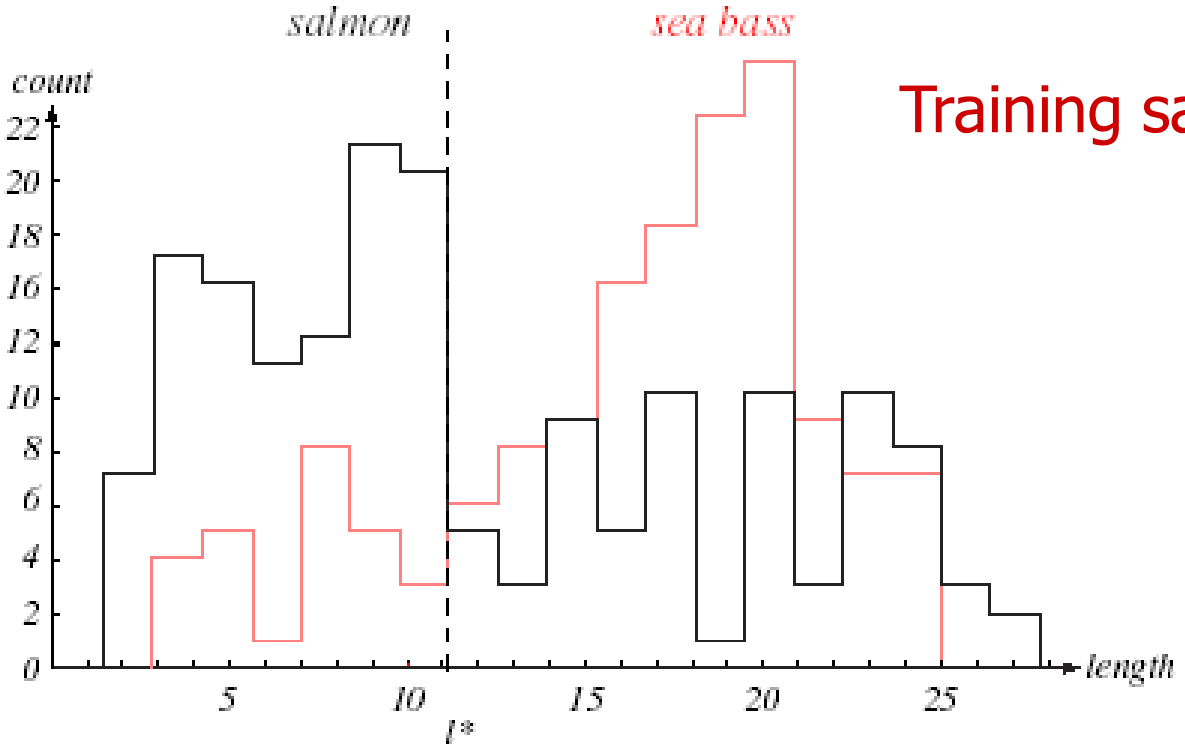


FIGURE 1.1. The objects to be classified are first sensed by a transducer (camera), whose signals are preprocessed. Next the features are extracted and finally the classification is emitted, here either "salmon" or "sea bass." Although the information flow is often chosen to be from the source to the classifier, some systems employ information flow in which earlier levels of processing can be altered based on the tentative or preliminary response in later levels (gray arrows). Yet others combine two or more stages into a unified step, such as simultaneous segmentation and feature extraction. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Preprocessing: image enhancement and segmentation; (i) separate touching or occluding fishes; (ii) extract fish contour

Representation: Fish Length as a Feature



Training samples

FIGURE 1.2. Histograms for the length feature for the two categories. No single threshold value of the length will serve to unambiguously discriminate between the two categories; using length alone, we will have some errors. The value marked *l** will lead to the smallest number of errors, on average. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Fish Lightness as a Feature

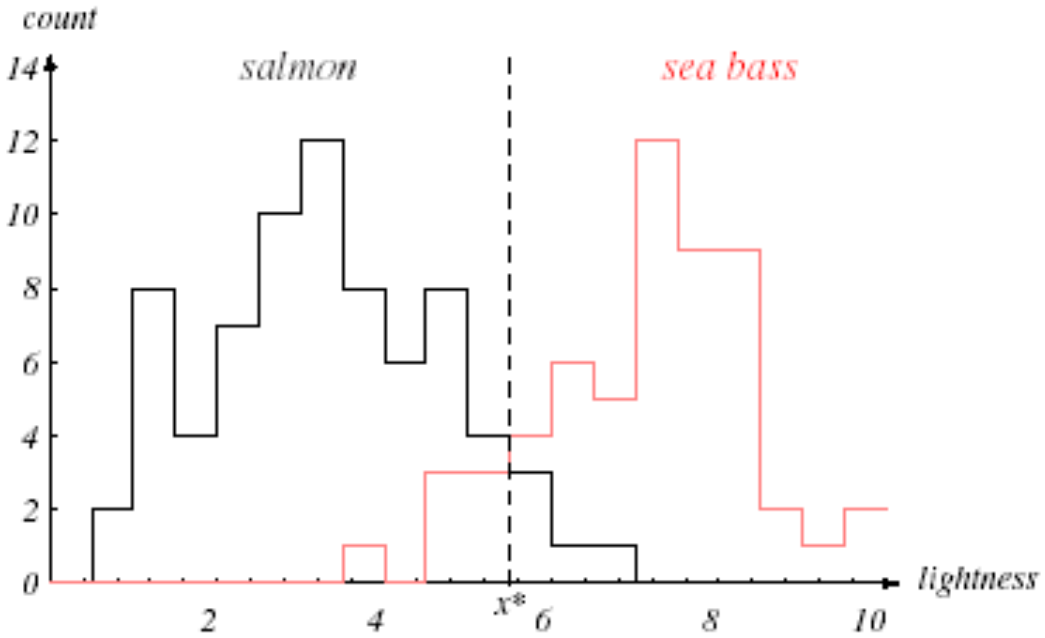
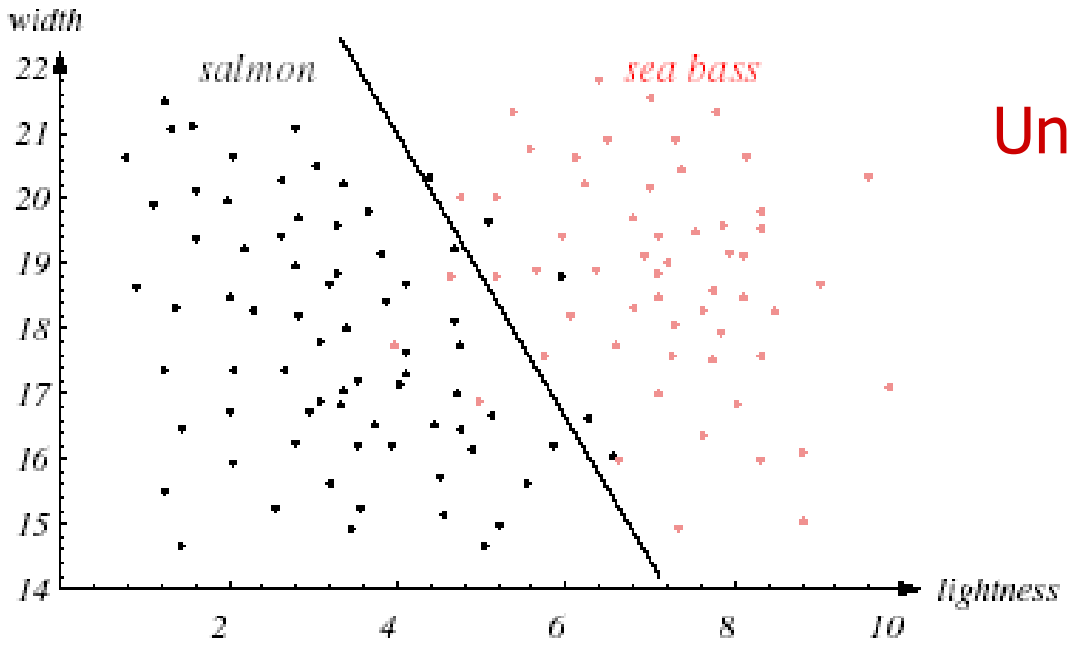


FIGURE 1.3. Histograms for the lightness feature for the two categories. No single threshold value x^* (decision boundary) will serve to unambiguously discriminate between the two categories; using lightness alone, we will have some errors. The value x^* marked will lead to the smallest number of errors, on average. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Overlap of these histograms is small compared to length feature

Two-dimensional Feature Space

Linear (simple) decision boundary; Cost of misclassification?

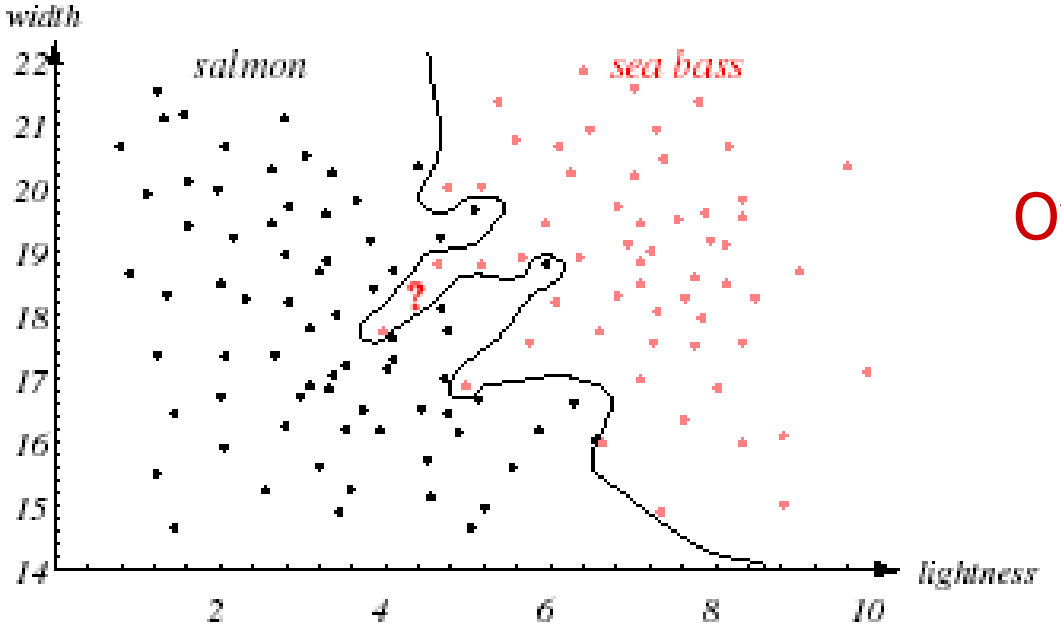


Underfitting

FIGURE 1.4. The two features of lightness and width for sea bass and salmon. The dark line could serve as a decision boundary of our classifier. Overall classification error on the data shown is lower than if we use only one feature as in Fig. 1.3, but there will still be some errors. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Joint distribution of two features leads to better separation

Complex Decision Boundary

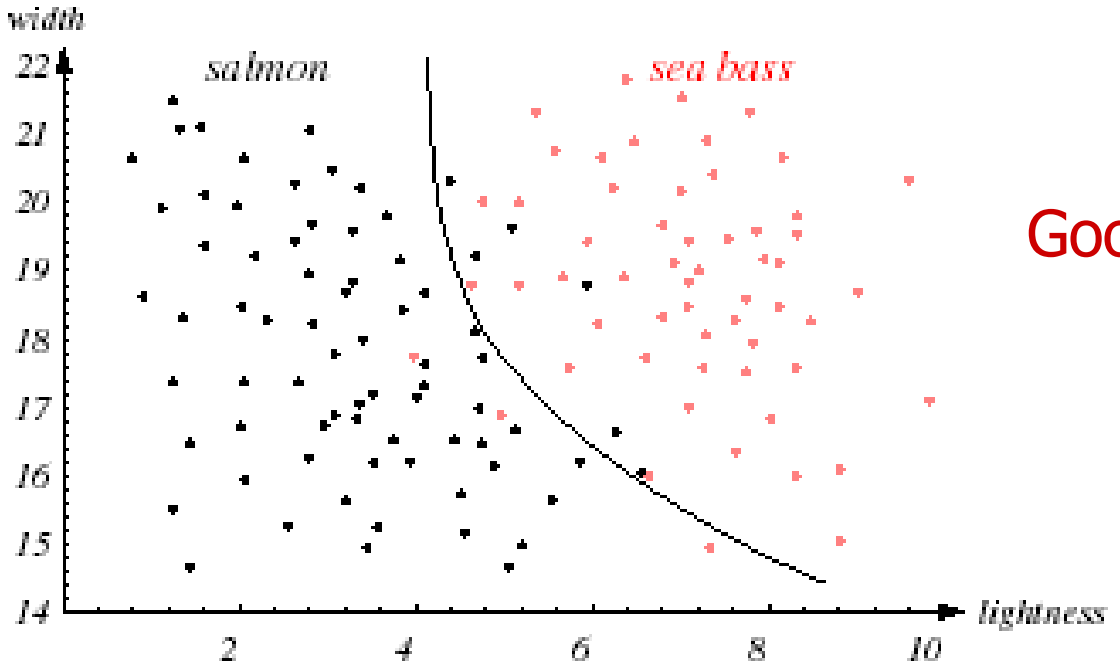


Overfitting

FIGURE 1.5. Overly complex models for the fish will lead to decision boundaries that are complicated. While such a decision may lead to perfect classification of our training samples, it would lead to poor performance on future patterns. The novel test point marked ? is evidently most likely a salmon, whereas the complex decision boundary shown leads it to be classified as a sea bass. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Generalization ability of the learned boundary

Boundary With Good Generalization



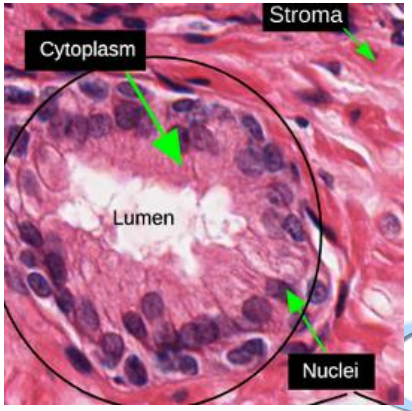
Good fit

FIGURE 1.6. The decision boundary shown might represent the optimal tradeoff between performance on the training set and simplicity of classifier, thereby giving the highest accuracy on new patterns. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

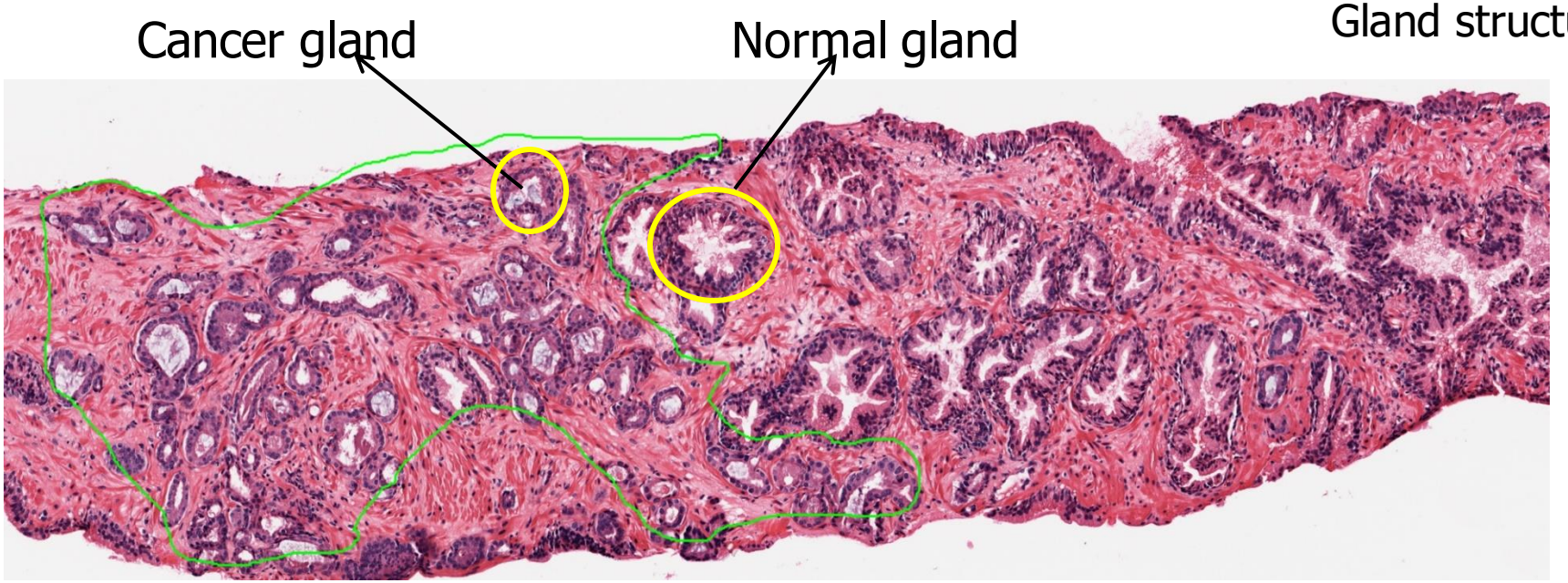
Simple decision boundaries are preferred

Example 2:
Automated Prostate Cancer Detection

Goal: Given a digitized microscopy image of a prostate tissue slide, we need to find cancer glands in the

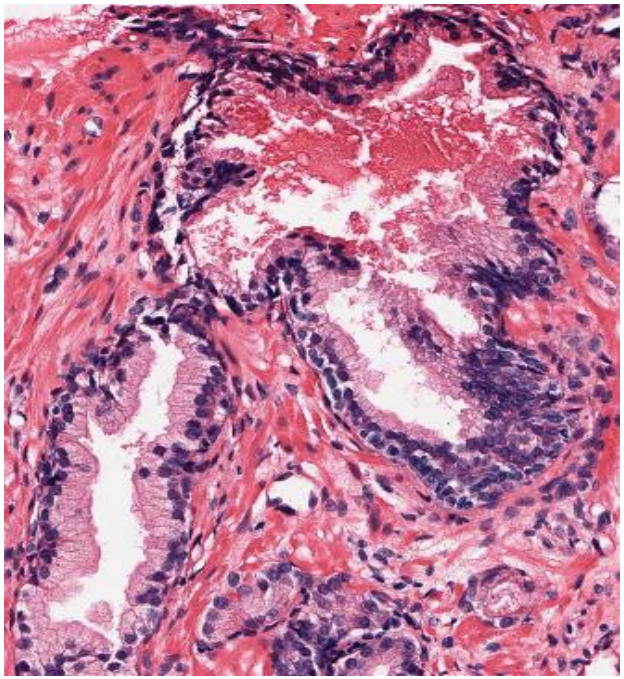


Gland structure

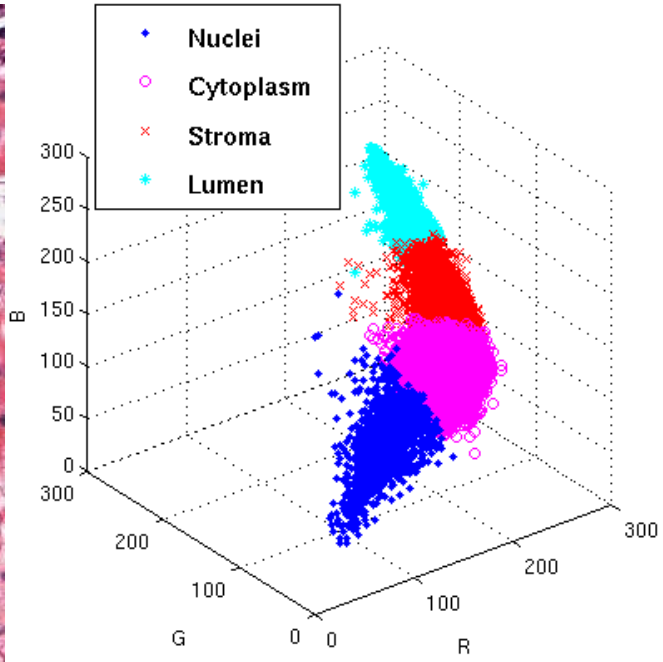


Identify tissue components

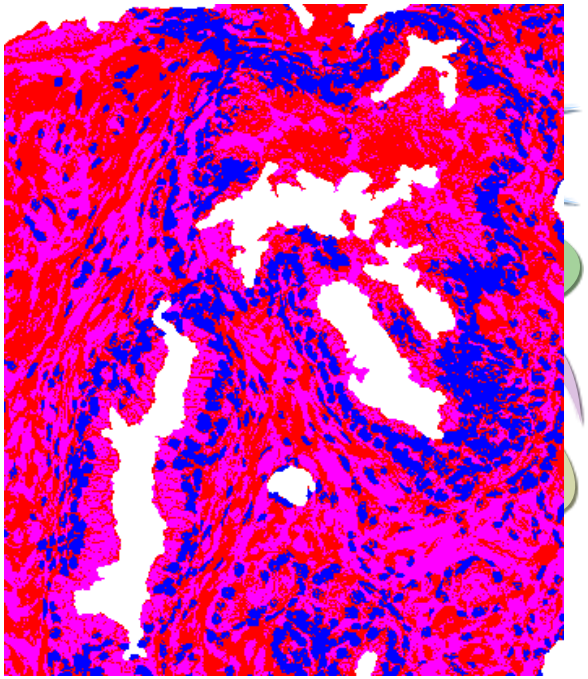
- **Apply K-means to cluster the image into 4 clusters corresponding to 4 tissue components: nuclei, stroma, cytoplasm, lumen**
- **The four components are identified by analyzing cluster centers**



Input image

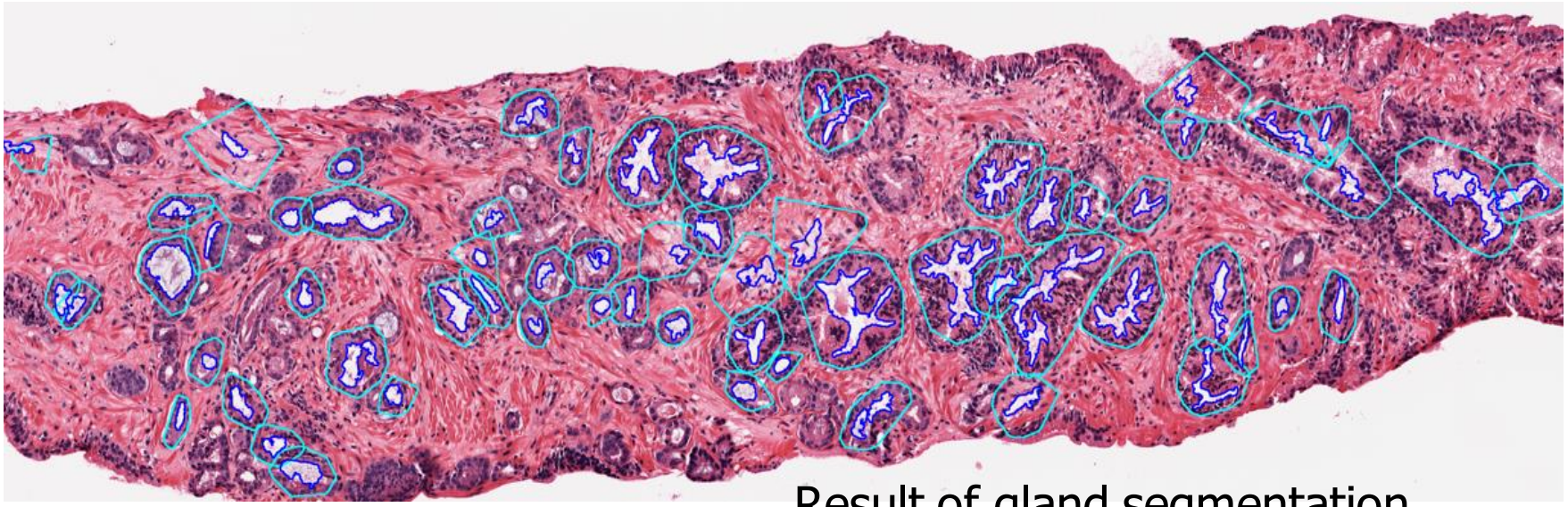


Clustering pixels in RGB color space



Clustering result. Blue: nuclei, white: lumen, red: stroma, pink: cytoplasm

Gland Segmentation



Result of gland segmentation

Segmentation process:

1. **Use lumen as the initial gland component**

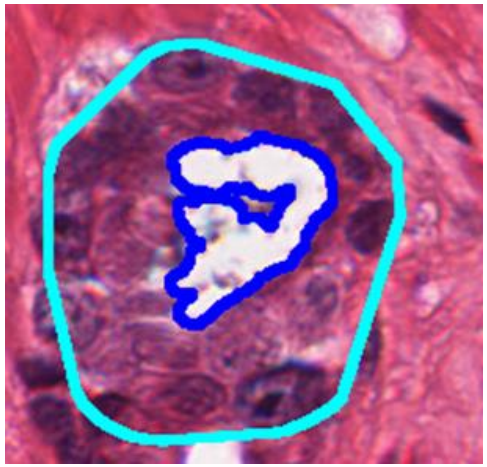
2. **Find nuclei associated with each lumen by searching on the normal direction to the lumen boundary contour**

Gland Classification

Razi University

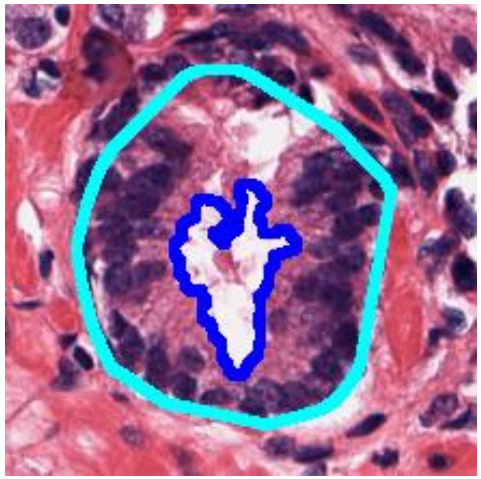
Artifact

- Stroma (pink) surrounding lumen
- Few nuclei surrounding lumen



Cancer gland

- Cytoplasm (purple) surrounding lumen
- More nuclei surrounding lumen

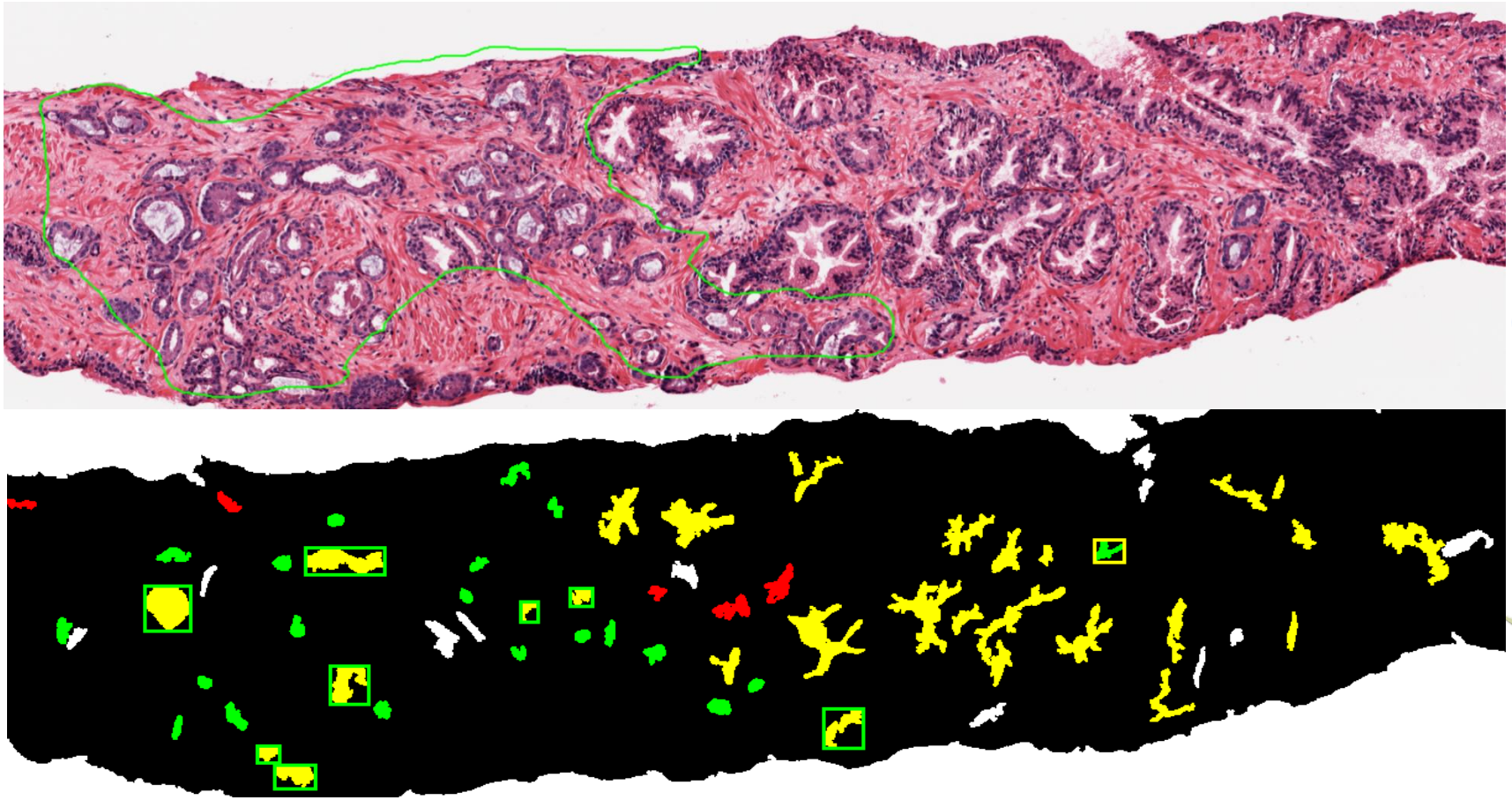


Normal gland

- Cytoplasm (purple) surrounding lumen
- Denser, darker nuclei surrounding lumen

Feature set	Features
Cytoplasm features	Mean and variance of L, a, b color on the blue contour neighborhood
Nuclei features	Mean and variance of L, a, b color on the cyan contour neighborhood
Gland features	Percentage of gland region that contain nuclei

Classification Result



Classification result. Yellow denotes normal glands, green denote cancer glands, red denotes artifacts, white denotes unused gland. Misclassified glands are highlighted by boxes (box color denotes true gland label)

Example 3:
Image Classification

- **Classify images into semantic object categories (ocean, beach, mountains, forest,..)**
- **Image annotation**
- **Application:**
  - **Content-based image indexing**
  - **Scene understanding**
  - **Online advertisement**
  - **Action recognition (monitoring human activities, home-based rehabilitation)…**

How should this image be Annotated?

Verification

- **Does it contain a ferry?**

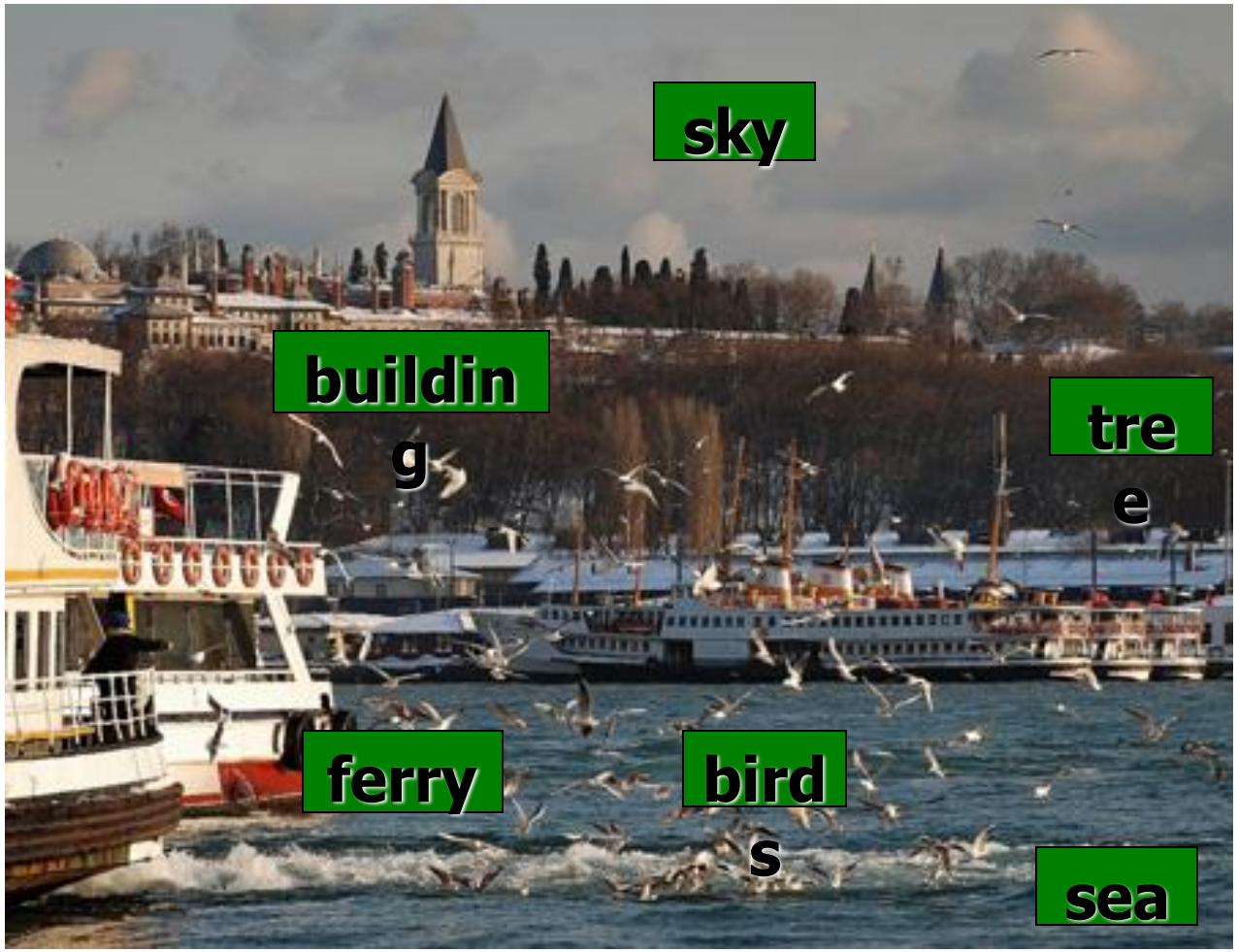
Detection

- **Are there birds in this image?**

Identification

- **IS This Topkapi Palace?**

Category recognition

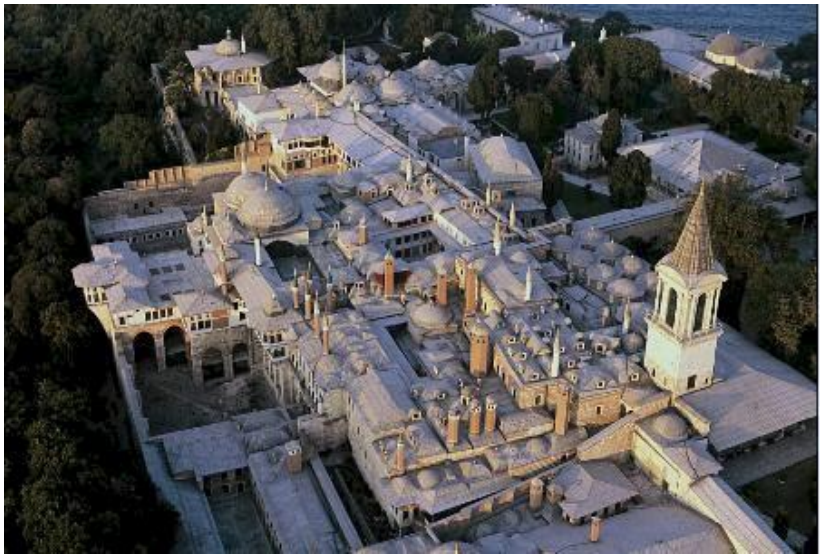


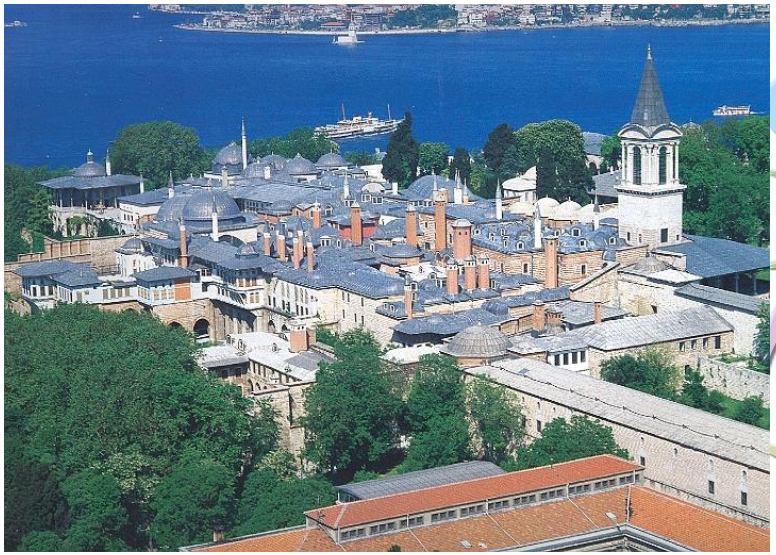
Challenges

- Variation in view point and lighting
- Variation in shape, pose and appearance
- Clutter and occlusion
- Function versus morphology
- Storage and speed requirements to process millions of images and thousands of categories

Challenges:
Variation in illumination

Challenges:
Viewpoint variation



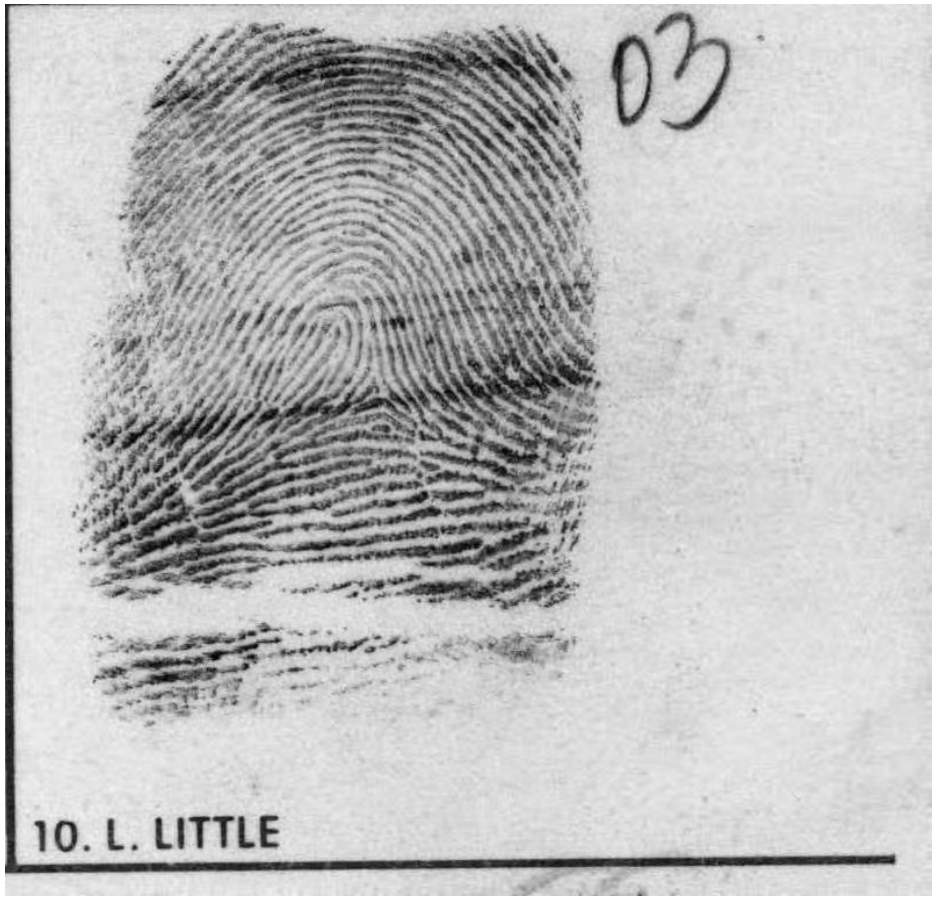


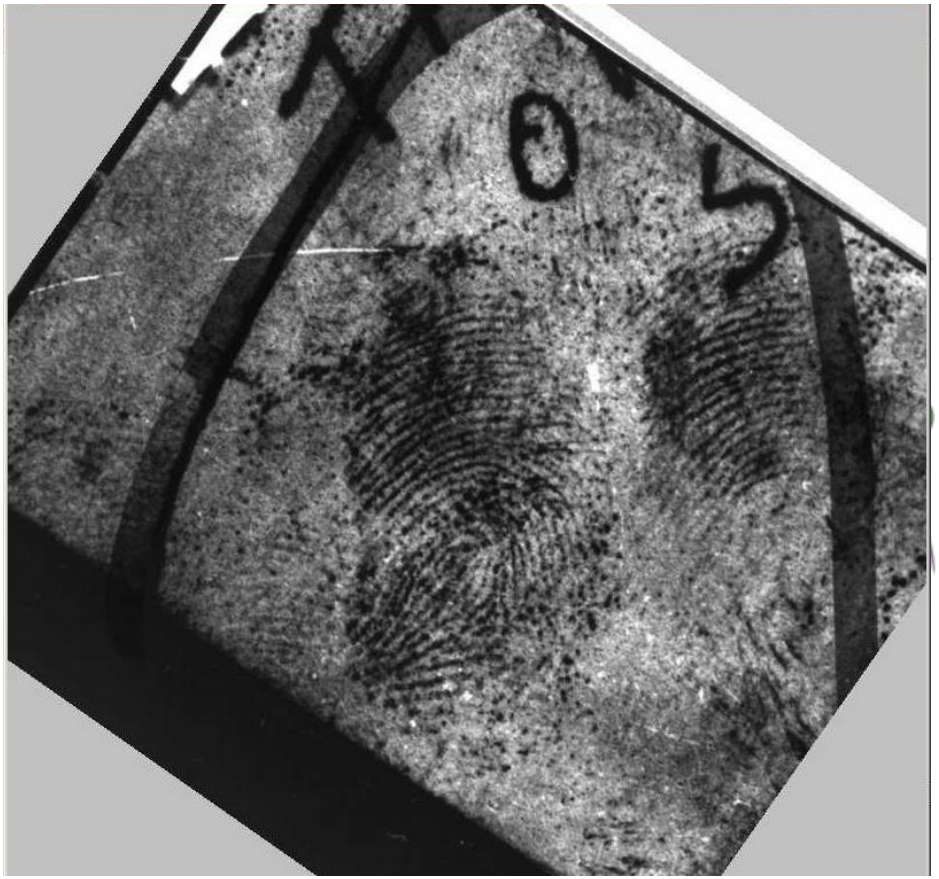
Challenges:
Viewpoint occlusion

Razi University

Challenges:
Deformation & background clutter

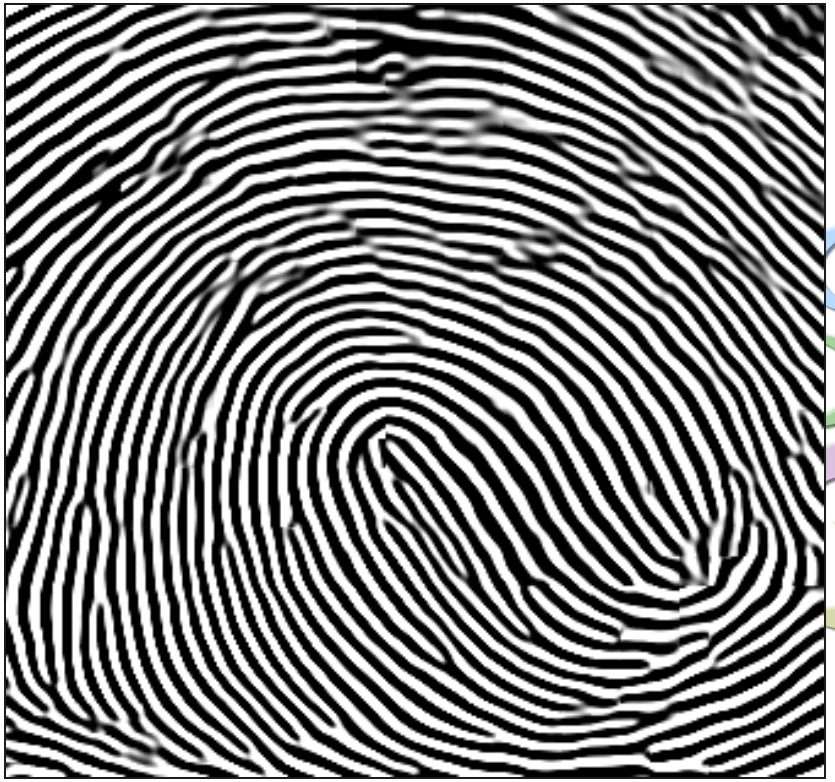
Challenges:
Foreground vs. Background





Challenges: Enhancement

Noisy image



Enhanced image

Comparing Pattern Recognition Models

- **Template Matching**
  - Assumes small intra-class variability
  - Learning is difficult for deformable templates
- **Syntactic or structural**
  - Primitive extraction is sensitive to noise
  - Describing a pattern in terms of primitives is difficult
- **Statistical or geometric**
  - Assumption of density model for each class
- **Neural Network**
  - Parameter tuning and local minima in learning

In practice, statistical/geometric approaches work well

Summary

- Pattern recognition is needed for
  - Automatic decision making
  - Assisting human decision makers
- General-purpose pattern recognition is a very difficult problem
- Successful systems available in well-constrained domains (e.g., handprinted character recognition)
- No single recognition approach has been found to be optimal for all pattern recognition problems
- Use of object models, constraints and context is necessary for identifying complex patterns
- Careful sensor design and feature extraction often lead to simple classifiers

Review of Basic Mathematics

Linear Algebra

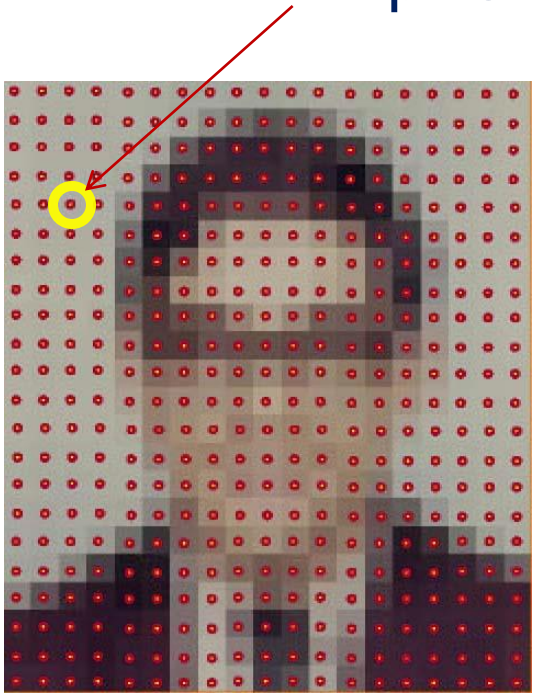
Linear Algebra has become as basic and as applicable as calculus, and fortunately it is easier.
--Gilbert Strang, MIT

Scalar

- **A quantity (*variable*), described by a single real number. e.g. 24, -98.5, 0.7,....**

e.g. Intensity of each digital

A pixel



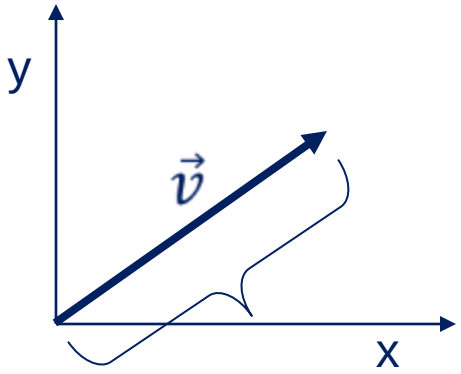
What is a Vector ?

- Think of a vector as a ***directed line segment in N-dimensions***! (has "length" and "direction")

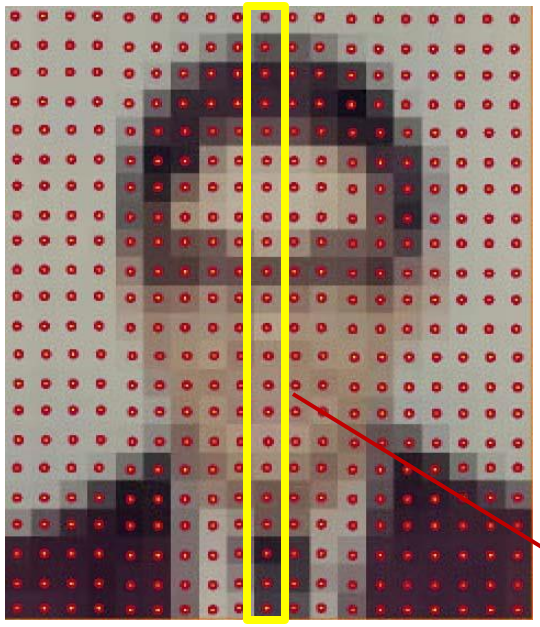
$$\vec{v} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

- Basic idea: convert geometry in higher dimensions into algebra!
  - Vector becomes a N x 1 matrix!
  - $\vec{v}$ = [a b c]^T
  - Geometry starts to become linear algebra on vectors like $\vec{v}$!

Vector means a column vector



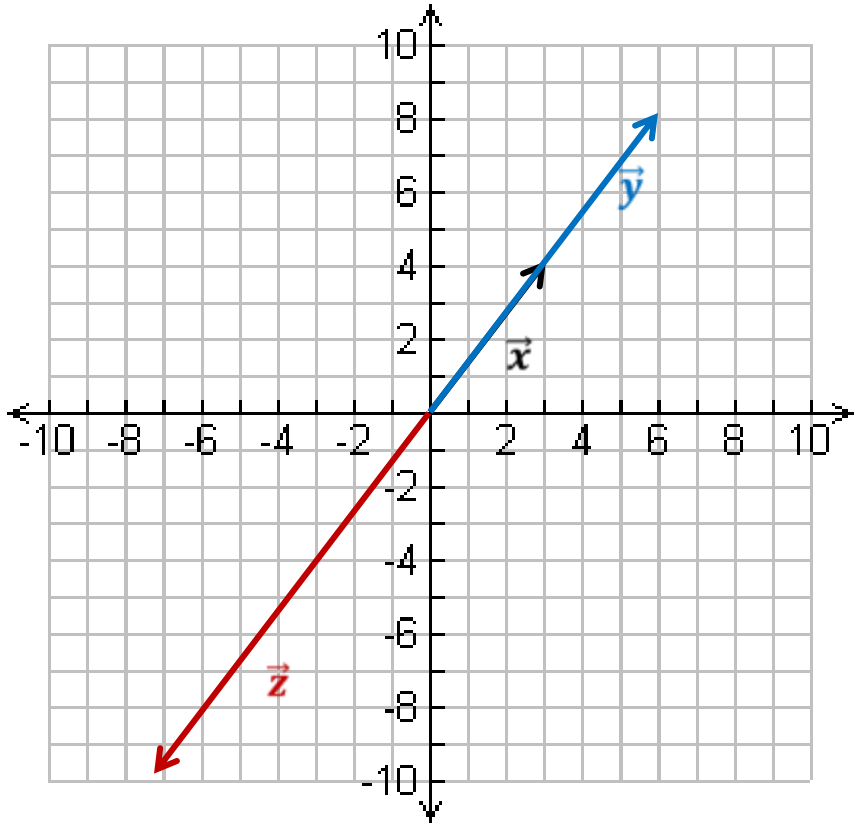
Vector

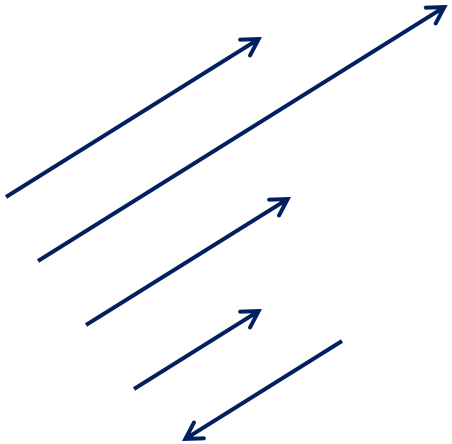


$$\text{VECTOR} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Parallel Vectors

- **Direction is same but length may vary**

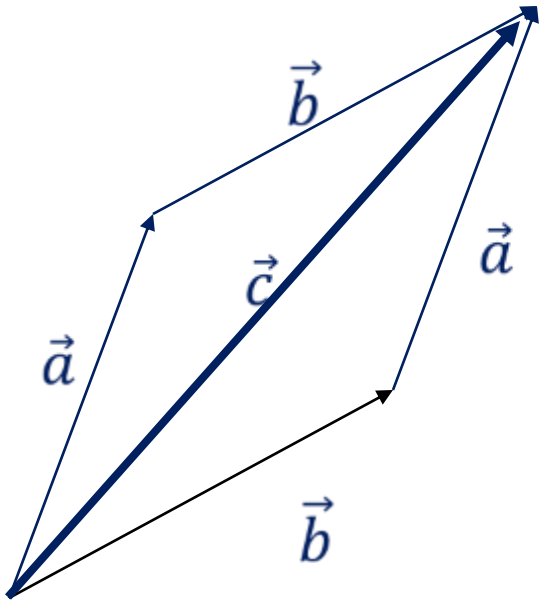




$$\vec{x} = a.\vec{y}$$
$$\vec{y} = b.\vec{z}$$
$$\vec{z} = c.\vec{x}$$

Vector Addition

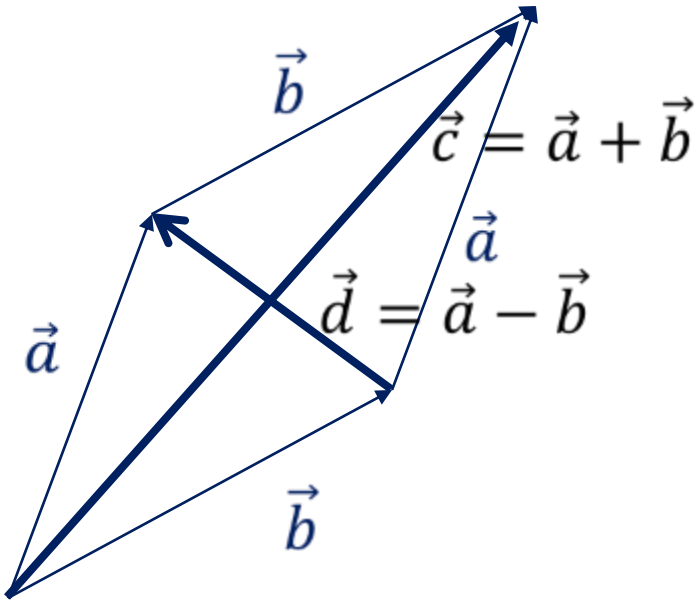
$$\vec{a} + \vec{b} = (x_1, x_2) + (y_1, y_2) = (x_1 + y_1, x_2 + y_2)$$



$\vec{a} + \vec{b} = \vec{c}$
(use the head-to-tail method to combine

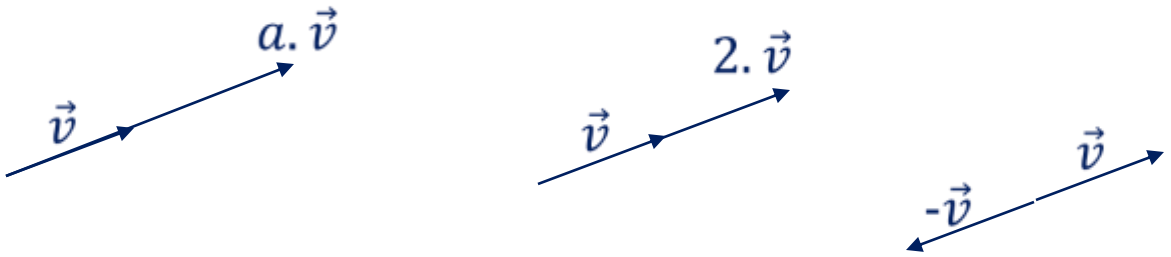
Vector Subtraction

$$\vec{a} - \vec{b} = (x_1, x_2) - (y_1, y_2) = (x_1 - y_1, x_2 - y_2)$$



Scalar Product

$$a.\vec{v} = a.(x_1, x_2) = (a.x_1, a.x_2); \quad a \text{ is scalr}$$



Change only the length ("scaling"), but keep *direction fixed*.

Sneak peek: matrix operation ($\mathbf{A}\vec{v}$) can change *length,*

Transpose of a Vector

$$\begin{bmatrix}1\\5\\7\end{bmatrix}^T = [1 \quad 5 \quad 7]$$

$$[11 \quad -12 \quad 9]^T = \begin{bmatrix}11\\-12\\9\end{bmatrix}$$

Vectors: Dot/Inner Product

- $\vec{v}.\,\vec{u} = \vec{v}^T\vec{u} = [v_1 \quad v_2 \quad v_3]\begin{bmatrix}u_1\\u_2\\u_3\end{bmatrix} = v_1u_1+v_2u_2+v_3u_3$

The inner product is a **SCALAR!**

- $\|\vec{v}\|^2 = \vec{v}^T\vec{v} = v_1v_1+v_2v_2+v_3v_3$

The length is the dot product of a vector with itself

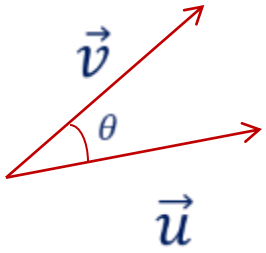
- A vector $\vec{v}$ is called unit vector if $\|\vec{v}\| = 1$.
- To make any vector $\vec{u}$, a unit vector divide with it's length, i.e. $\frac{\vec{u}}{\|\vec{u}\|}$

Vectors: Dot/Inner Product

The dot product is also related to the angle between the two vectors

$$\vec{v}.\ \vec{u} = \|\vec{v}\|\|\vec{u}\|\ cos\theta$$

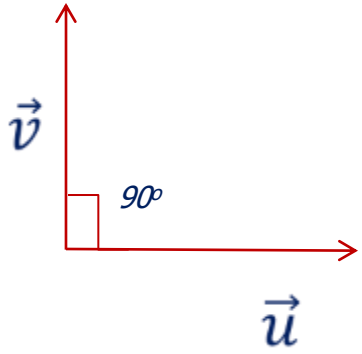
$$\Rightarrow\ cos\theta = \frac{\vec{v}.\ \vec{u}}{\|\vec{v}\|\|\vec{u}\|}$$



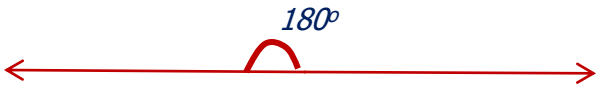
- If $\vec{v}$ and $\vec{u}$ are perpendicular to each other then $\vec{v}.\ \vec{u}=0$

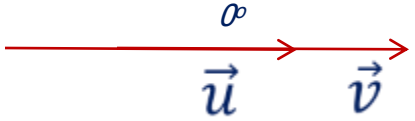
$$cos90^o = \frac{\vec{v}.\ \vec{u}}{\|\vec{v}\|\|\vec{u}\|}$$

$$\Rightarrow \vec{v}.\ \vec{u} = 0$$



- If $\vec{v}$ and $\vec{u}$ are parallel to each other then $\vec{v}.\ \vec{u} = \|\vec{v}\|\|\vec{u}\|$





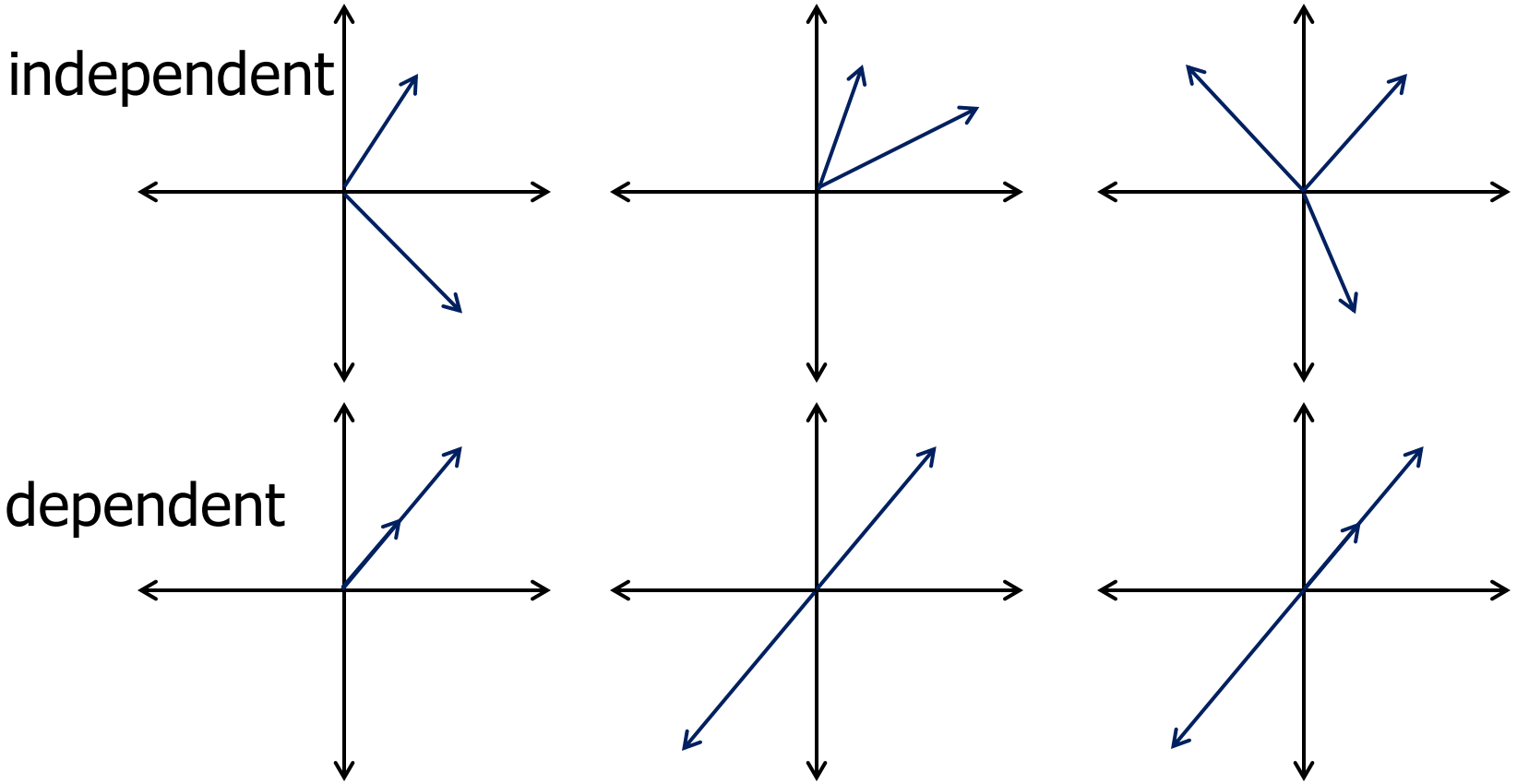
Outer Product

- **Outer product of two vectors is a matrix.**
- **Simple matrix multiplication of vectors**

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix} = \begin{bmatrix} a_1 b_1 & a_1 b_2 & a_1 b_3 \\ a_2 b_1 & a_2 b_2 & a_2 b_3 \\ a_3 b_1 & a_3 b_2 & a_3 b_3 \end{bmatrix}$$

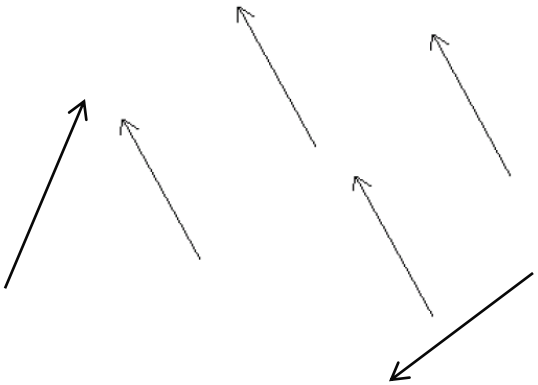
Linear Independence

- **Different direction vectors are independent.**
- **Parallel vectors are dependent**

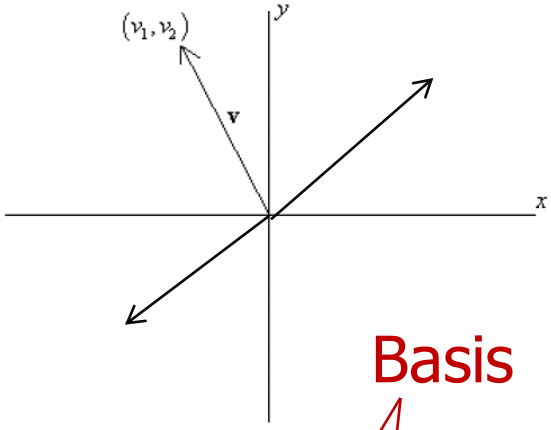


Basis & Orthonormal Bases

- **Basis (or axes): frame of reference**



vs



Basis

$$\begin{bmatrix}2\\3\end{bmatrix}=2.\begin{bmatrix}1\\0\end{bmatrix}+3.\begin{bmatrix}0\\1\end{bmatrix};\ \begin{bmatrix}-9\\6\end{bmatrix}=-9.\begin{bmatrix}1\\0\end{bmatrix}+6.\begin{bmatrix}0\\1\end{bmatrix}\longrightarrow\left\{\begin{bmatrix}1\\0\end{bmatrix},\begin{bmatrix}0\\1\end{bmatrix}\right\}$$ is a set of representative for $\mathbb{R}^2$

$$\begin{bmatrix}2\\3\end{bmatrix}=\frac{9}{7}.\begin{bmatrix}2\\1\end{bmatrix}+\frac{4}{7}.\begin{bmatrix}-1\\3\end{bmatrix};\begin{bmatrix}-9\\6\end{bmatrix}=-3.\begin{bmatrix}2\\1\end{bmatrix}+3.\begin{bmatrix}-1\\3\end{bmatrix}\longrightarrow\left\{\begin{bmatrix}2\\1\end{bmatrix},\begin{bmatrix}-1\\3\end{bmatrix}\right\}$$ is another Set of representative

Basis & Orthonormal Bases

Basis: a space is totally defined by a set of vectors – any point is a *linear combination* of the basis

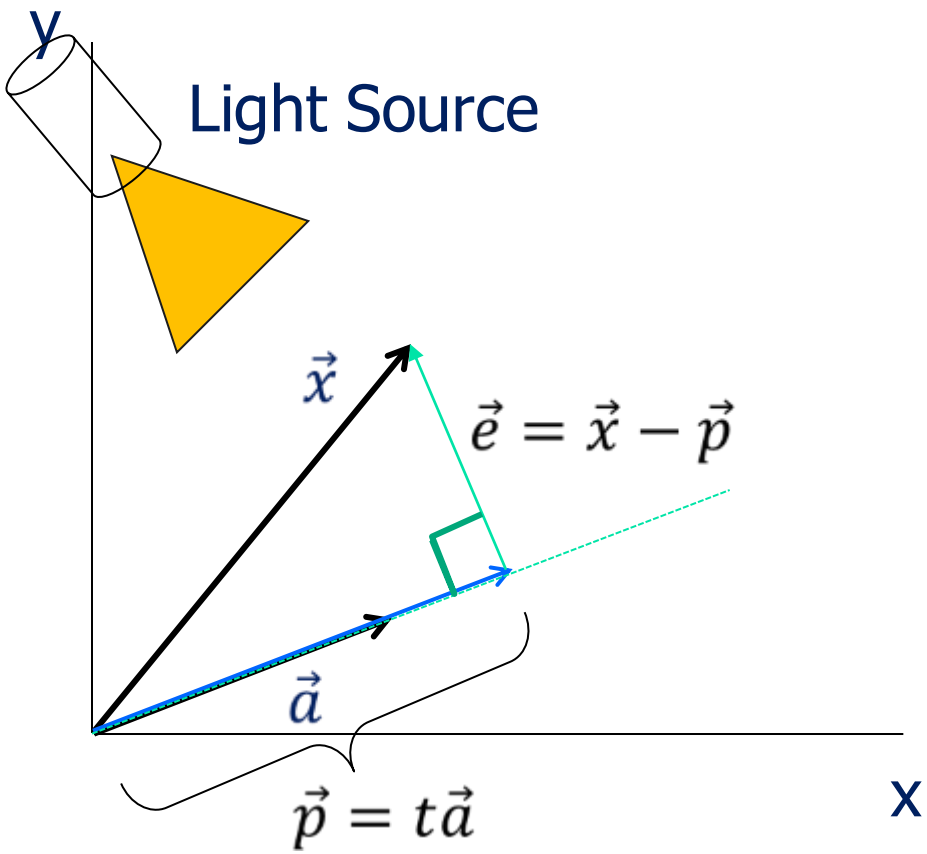
Ortho-Normal: orthogonal + normal

[**Orthogonal**: dot product is zero
Normal: magnitude is one]

$$\vec{x} = [1 \quad 0 \quad 0]^T \qquad \vec{x}.\vec{y} = 0$$
$$\vec{y} = [0 \quad 1 \quad 0]^T \qquad \vec{x}.\vec{z} = 0$$
$$\vec{z} = [0 \quad 0 \quad 1]^T \qquad \vec{y}.\vec{z} = 0$$

Projection: Using Inner Products

Projection of $\vec{x}$ along the direction $\vec{a}$



$$\cos 90^0 = \frac{\vec{a}.\vec{e}}{\|\vec{a}\|\|\vec{e}\|}$$

$$\Rightarrow \vec{a}.\vec{e} = 0$$

$$\Rightarrow \vec{a}.(\vec{x} - \vec{p}) = 0$$

$$\Rightarrow \vec{a}.(\vec{x} - t\vec{a}) = 0$$

$$\Rightarrow \vec{a}.\vec{x} - t\vec{a}.\vec{a} = 0$$

$$\Rightarrow t = \frac{\vec{a}.\vec{x}}{\vec{a}.\vec{a}}$$

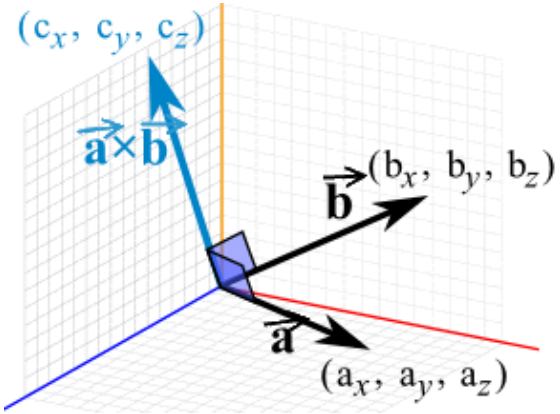
$$Projection\ vector\ \vec{p} = t\vec{a}$$

$$\vec{p} = \vec{a}\frac{\vec{a}^T\vec{x}}{\vec{a}^T\vec{a}} = \frac{\vec{a}\vec{a}^T}{\vec{a}^T\vec{a}}\vec{x}$$

Projection Matrix

Vectors: Cross Product

- The cross product of vectors $\vec{a}$ and $\vec{b}$ is a vector $\vec{c}$ which is perpendicular to $\vec{a}$ and $\vec{b}$
- The magnitude of $\vec{c}$ is proportional to the sin of the angle between $\vec{a}$ and $\vec{b}$
- The direction of $\vec{c}$ follows the right hand rule if we are working in a right-handed coordinate system



MAGNITUDE OF THE CROSS PRODUCT

$$\|A \times B\| = \|A\| \, \|B\| \, \sin\theta$$

Example:

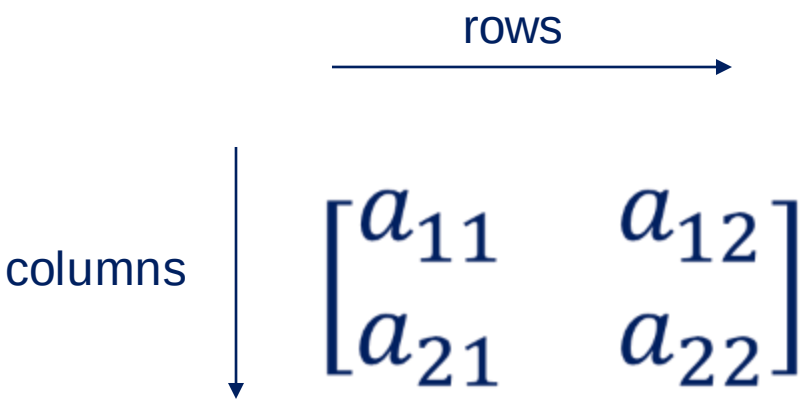
$$\vec{a} = \begin{bmatrix} 2 \\ 1 \\ 5 \end{bmatrix}; \qquad \vec{b} = \begin{bmatrix} -3 \\ 2 \\ 7 \end{bmatrix}$$

$$\vec{a} \times \vec{b} = \begin{vmatrix} i & j & k \\ 2 & 1 & 5 \\ -3 & 2 & 7 \end{vmatrix} = i(7-10) - j(14+15) + k(4+3)$$

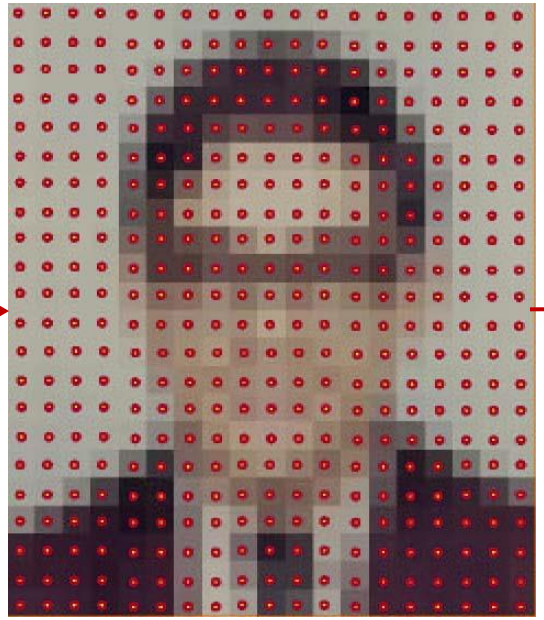
$$= -3i - 29j + 7k$$

What is a Matrix?

- **A matrix is a set of elements, organized into rows and columns**



Matrix



$$\begin{bmatrix} x11 & x12 & x13 \\ x21 & x22 & x23 \\ x31 & x32 & x33 \end{bmatrix}$$

Basic Matrix Operations

- **Addition, Subtraction, Multiplication: creating new matrices (or functions)**

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} - \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a-e & b-f \\ c-g & d-h \end{bmatrix}$$

Just add elements

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a+e & b+f \\ c+g & d+h \end{bmatrix}$$

Just subtract elements

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{bmatrix}$$

Multiply each row by each column

Note: Matrix multiplication is possible only when number of columns of first matrix is equal to number of rows of second one.

Multiplication

- **Is AB = BA? Maybe, but maybe not!**

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} e & f \\ g & h \end{bmatrix}=\begin{bmatrix} ae+bg & \dots \\ \dots & \dots \end{bmatrix} \qquad \begin{bmatrix} e & f \\ g & h \end{bmatrix}\begin{bmatrix} a & b \\ c & d \end{bmatrix}=\begin{bmatrix} ea+fc & \dots \\ \dots & \dots \end{bmatrix}$$

- **Heads up: multiplication is NOT commutative!**

Transpose & Symmetric Matrix

- **Transpose of a matrix A is formed by turning all the rows of a given matrix into columns and vice-versa.**

$$\begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}^T = \begin{bmatrix} 2 & 4 \\ 3 & 5 \end{bmatrix}$$

Matrix A is called a symmetric matrix if $A = A^T$

$$\begin{bmatrix} 2 & 3 & 6 \\ 3 & 10 & 22 \\ 6 & 22 & 5 \end{bmatrix}$$

Matrix operating on vectors

- **Matrix is like a function that transforms the vectors on a plane**
- **Matrix operating on a general point => transforms x- and y-components**
- ***System of linear equations* : matrix is just the bunch of coefficients !**

$$x' = a.x + b.y$$
$$y' = c.x + d.y$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

Linear Transformation

$$y = Ax \Longrightarrow \begin{aligned} y_1 &= a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ y_2 &= a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ &\vdots \\ y_d &= a_{d1}x_1 + a_{d2}x_2 + \cdots + a_{dn}x_n \end{aligned}$$

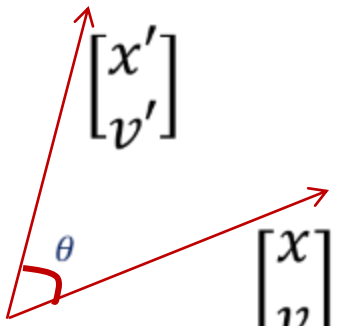
$$\Leftrightarrow \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_d \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{d1} & a_{d2} & \ldots & a_{dn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

Rotation

$$\text{Rotation Matrix} = R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

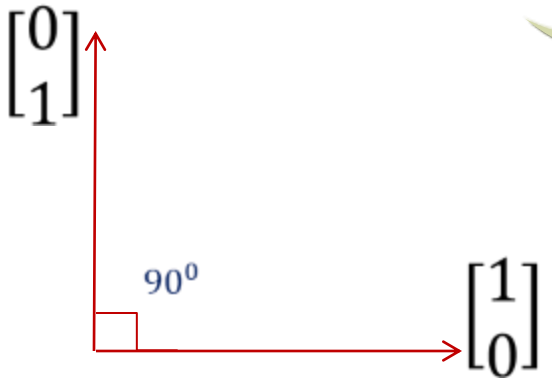
- We can rotate any 2-dimensional vector using $\boldsymbol{R(\theta)}$ as follows

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

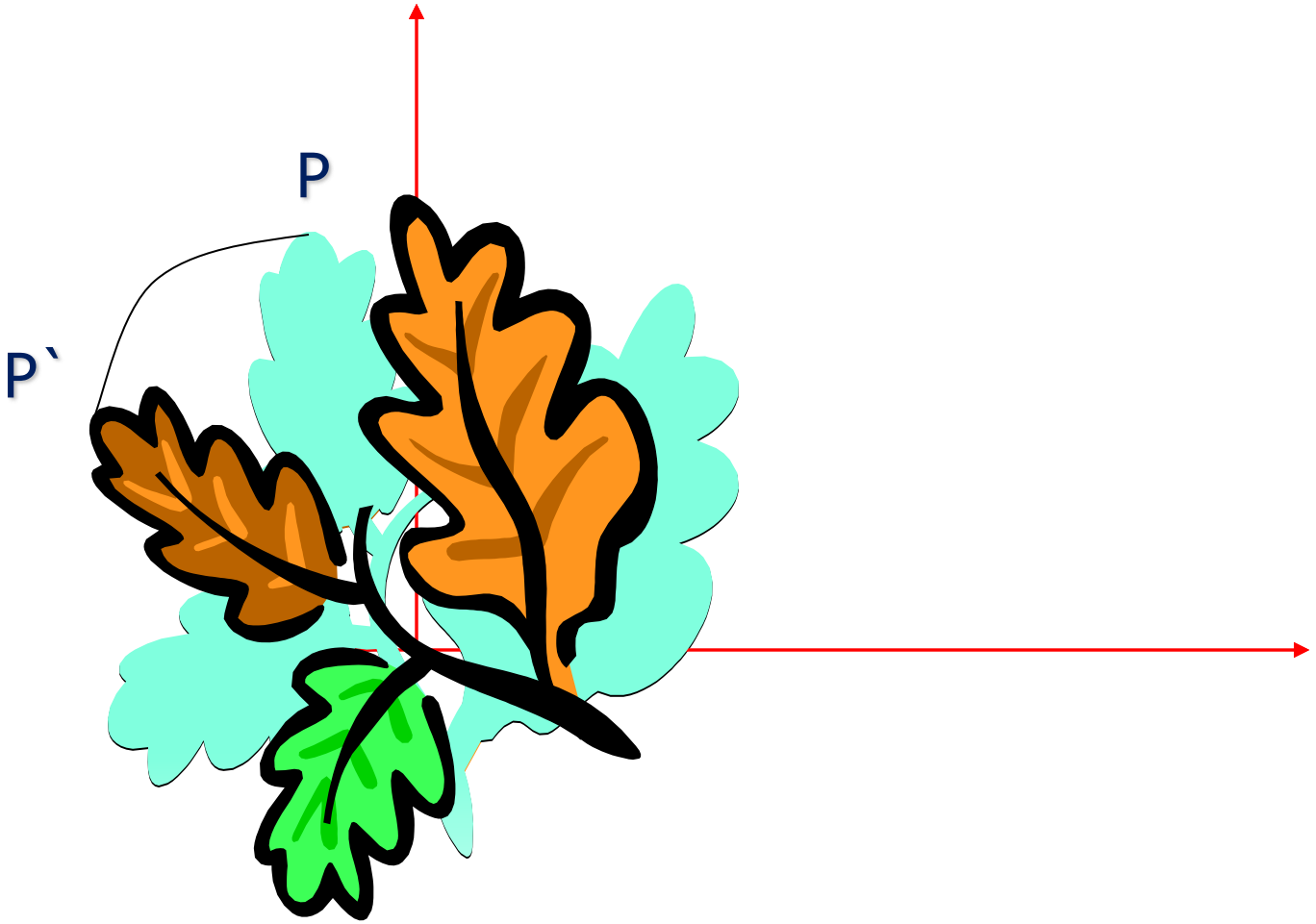


Example :

$$\begin{bmatrix} \cos 90 & -\sin 90 \\ \sin 90 & \cos 90 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$



Rotation



Scaling

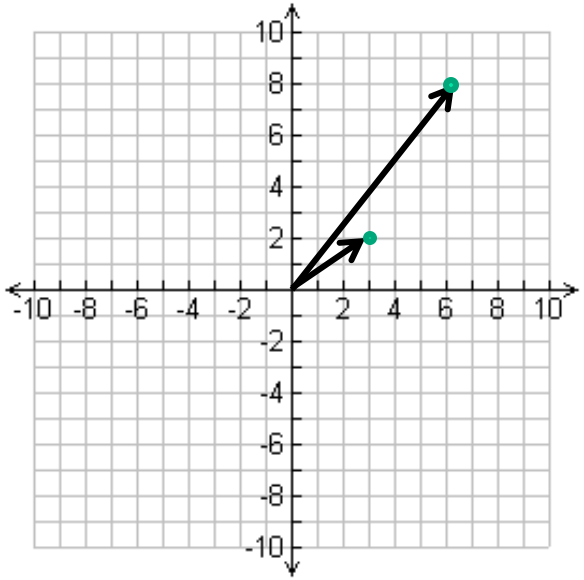
$$\text{Scaling Matrix} = \begin{bmatrix} r_1 & 0 \\ 0 & r_2 \end{bmatrix}$$

- **We can scale any 2-dimensional vector as follows**

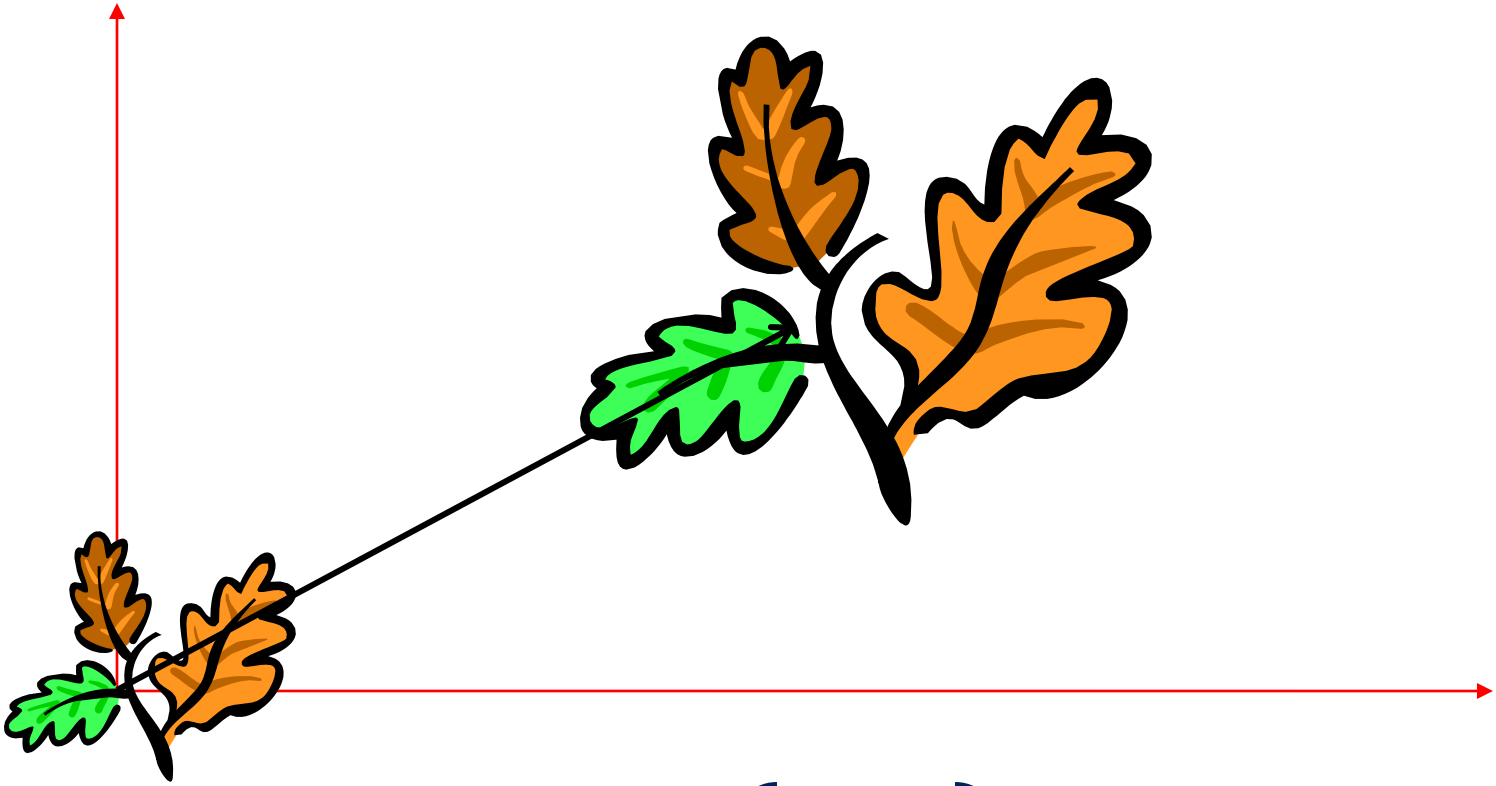
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} r_1 & 0 \\ 0 & r_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Example :

$$\begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 6 \\ 8 \end{bmatrix}$$



Scaling

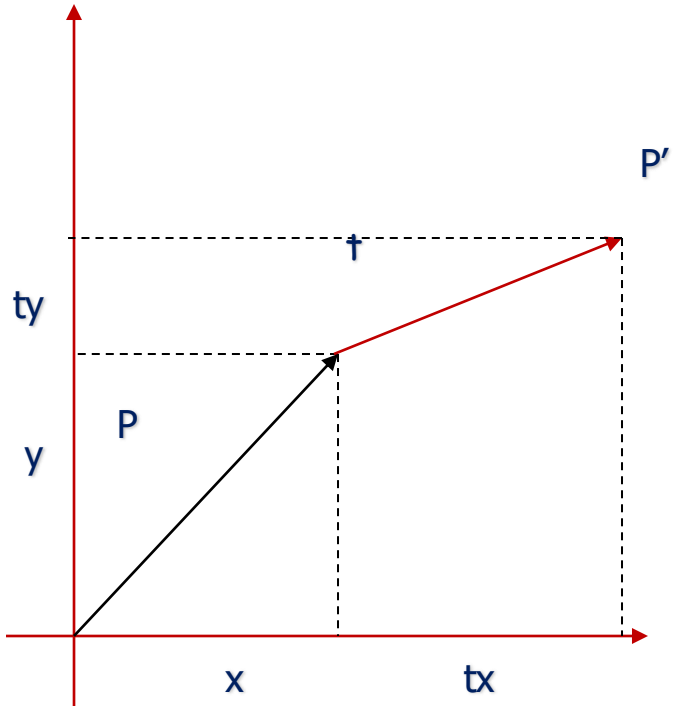


$$\begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix}$$

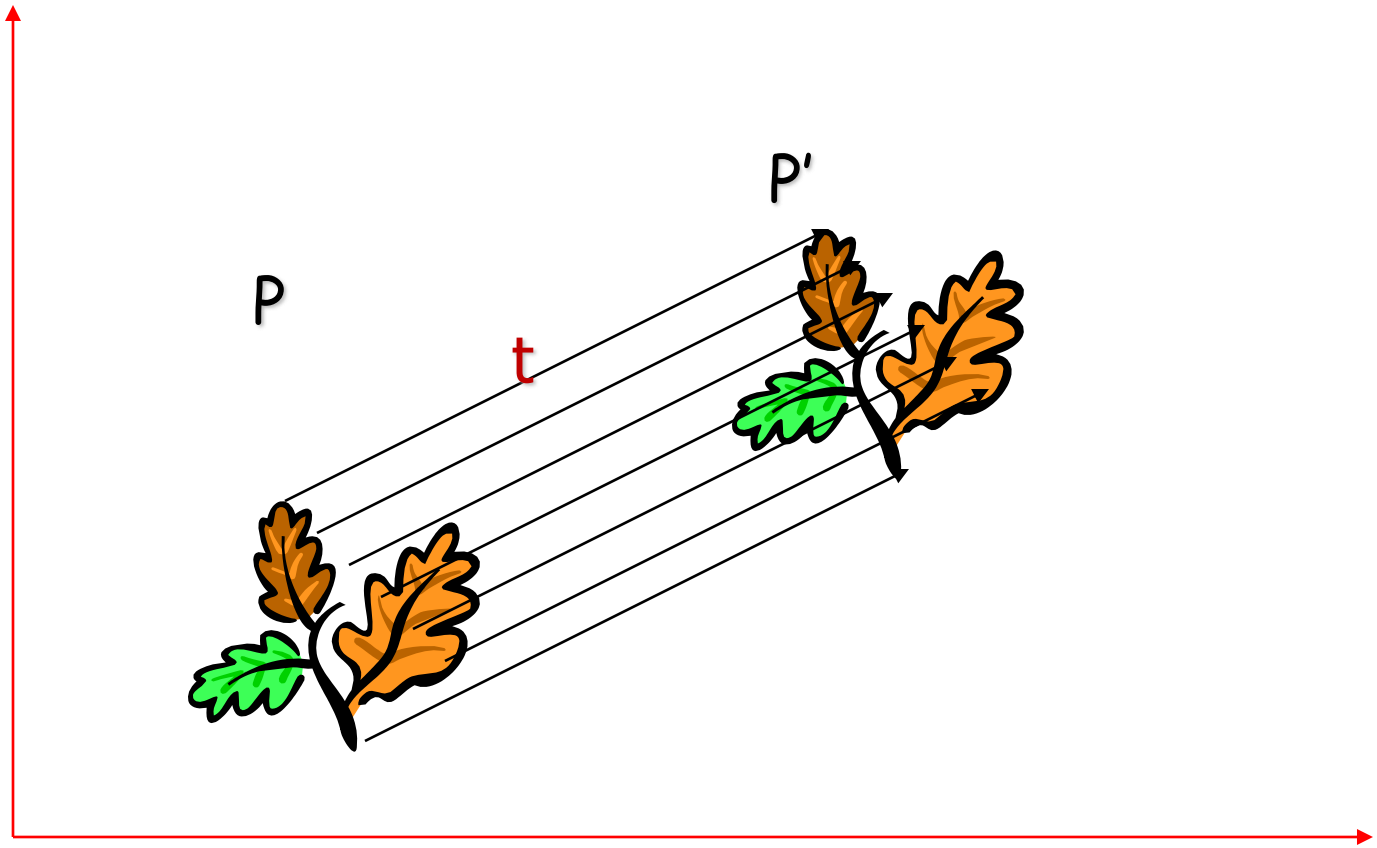
a.k.a: dilation (r >1), contraction (r <1)

Translation

$$P' = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = P + t$$

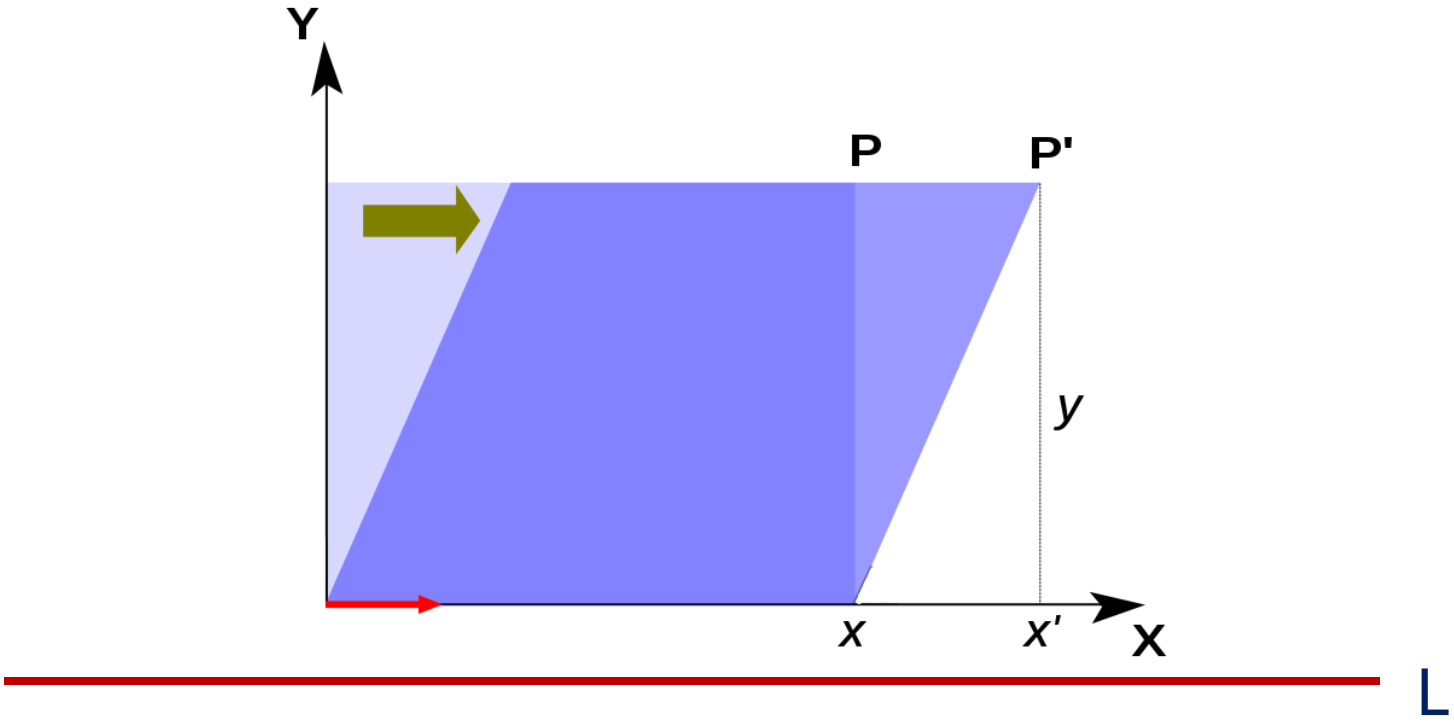


Translation



Shear

- **A transformation in which all points along a given line L remain fixed while other points are shifted parallel to L by a distance proportional to their perpendicular distance from L.**



Shear

Horizontal Shear:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & m \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x + my \\ y \end{bmatrix}$$

Vertical Shear:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ m & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \\ mx + y \end{bmatrix}$$

Identity Matrix

Identity ("do nothing") matrix = unit scaling, no rotation!

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Inverse of a Matrix

- **AI = IA = A**
- **Inverse exists only for square matrices that are non-singular**
- **Some matrices have an inverse, such that: $AA^{-1} = I$**
- **Inversion is tricky: $(ABC)^{-1} = C^{-1}B^{-1}A^{-1}$**

  Derived from non-commutativity property

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

Determinant & Trace of a Matrix

- Determinant used for inversion
- If det(A)=0, then A has no inversion

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad \det(A) = ad - bc$$

- Trace = sum of diagonal elements (a+d)

  = sum of eigenvalues

Eigenvalues & Eigenvectors

- **Eigenvectors (for a square *m*×*m* matrix S)**

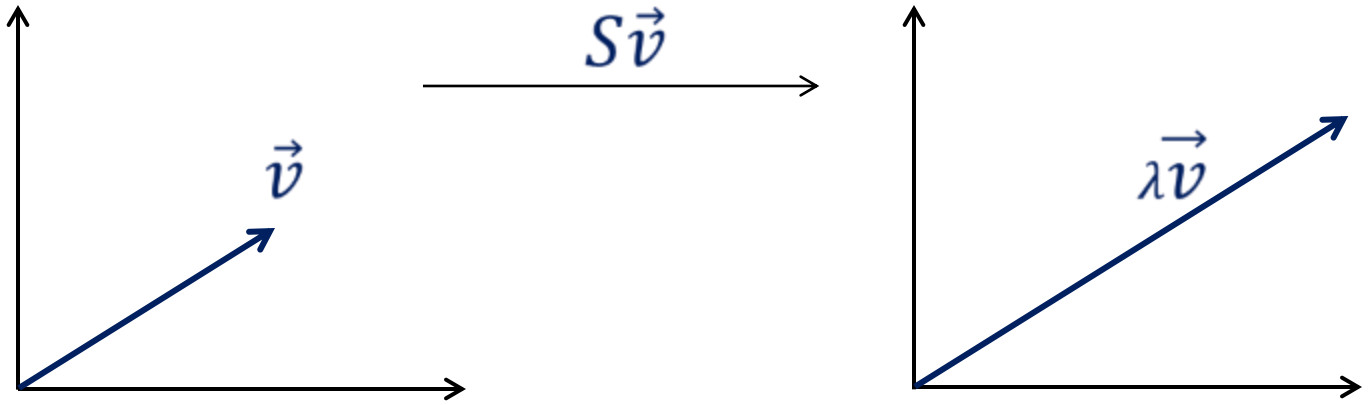
$$S\vec{v} = \lambda\vec{v}$$

(right) eigenvector $\vec{v} \epsilon \mathbb{R}^m \neq \vec{0}$

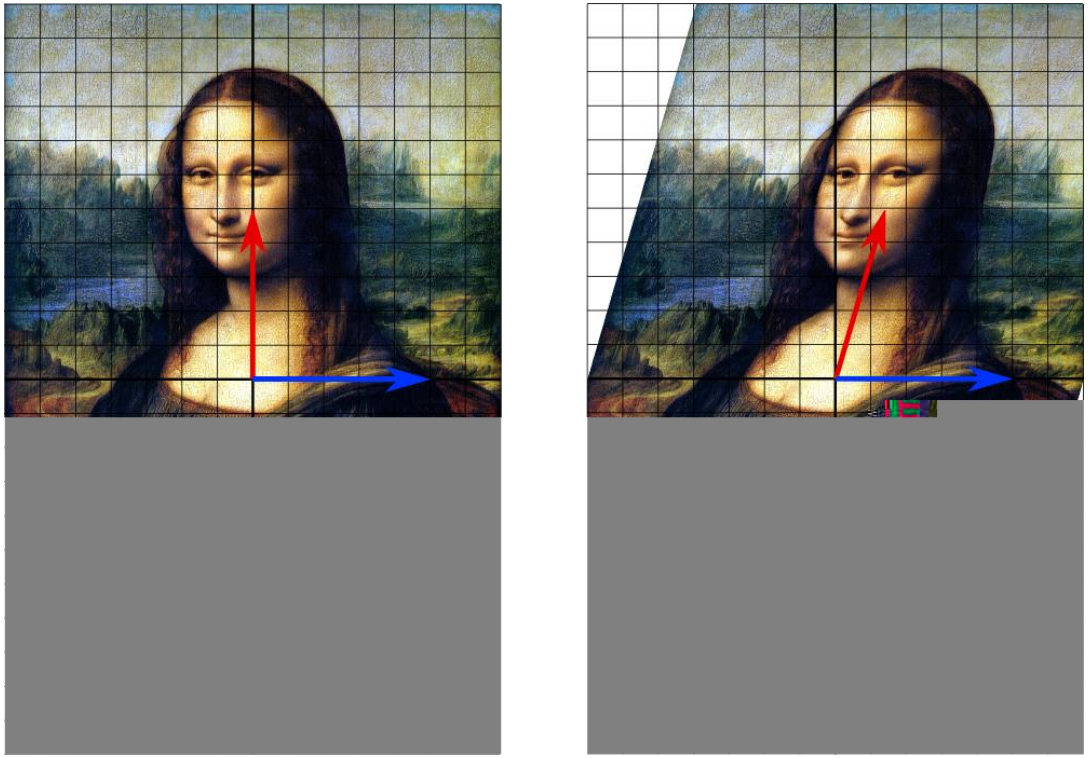
eigenvalue $\lambda \epsilon \mathbb{R}$

Example

$$\begin{pmatrix} 6 & -2 \\ 4 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \end{pmatrix} = 2 \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$



Eigenvalues & Eigenvectors



In the shear mapping the red arrow changes direction but the blue arrow does not. The blue arrow is an eigenvector of this shear mapping because it doesn't change direction, and since in this example its length is unchanged its eigenvalue is 1.

Computation of Eigenvalues

$$S\vec{v} = \lambda\vec{v} \implies (S - \lambda I)\,\vec{v} = 0$$

First find the eigenvalues λ and substitute λ in the above equation to ol eigenvectors

$$For\ eigenvalues\ solve, det(S - \lambda I) = 0$$

Example:

$$A = \begin{bmatrix} 0 & 1 \\ -6 & 5 \end{bmatrix} \qquad A - \lambda I = \begin{bmatrix} -\lambda & 1 \\ -6 & 5 - \lambda \end{bmatrix}$$

$$det\left(\begin{bmatrix} -\lambda & 1 \\ -6 & 5 - \lambda \end{bmatrix}\right) = -\lambda(5 - \lambda) + 6 = (\lambda - 2)(\lambda - 3)$$

$$Eigenvalues\ are\ 2, 3$$

Computation of Eigenvectors

$$(S - \lambda I)\,\vec{v} = 0$$

$\lambda = 2$:

$$\begin{bmatrix} -\lambda & 1 \\ -6 & 5-\lambda \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow \begin{matrix} -\lambda v_1 + v_2 = 0 \\ -6v_1 + (5-\lambda)v_2 = 0 \end{matrix}$$

$$\Rightarrow \left.\begin{matrix} -2v_1 + v_2 = 0 \\ -6v_1 + 3v_2 = 0 \end{matrix}\right\}$$ Both are same

$$\Rightarrow 2v_1 = v_2 \Rightarrow v_1 = t,\ v_2 = 2t; t > 0$$

$$\therefore \vec{v} = \begin{bmatrix} t \\ 2t \end{bmatrix} \text{ is a eigenvector}$$

Computation of Eigenvectors

$$\lambda = 3: \quad \begin{bmatrix} -\lambda & 1 \\ -6 & 5-\lambda \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow \begin{matrix} -\lambda v_1 + v_2 = 0 \\ -6v_1 + (5-\lambda)v_2 = 0 \end{matrix}$$

$$\Rightarrow \begin{matrix} -3v_1 + v_2 = 0 \\ -6v_1 + 2v_2 = 0 \end{matrix}$$

$$\Rightarrow 3v_1 = v_2 \Rightarrow v_1 = t, \; v_2 = 3t; t > 0$$

$$\therefore \vec{v} = \begin{bmatrix} t \\ 3t \end{bmatrix} \text{ is another eigenvector}$$

Singular Value Decomposition(SVD)

- Handy mathematical technique that has application to many problems
- Given any *m*×*n* matrix **A**, algorithm to find matrices **U**, **V** and **W** such that
- $$A_{m\times n} = U_{m\times n} W_{n\times n} V^T_{n\times n}$$

U and **V** are orthonormal

W is *n*×*n* and diagonal

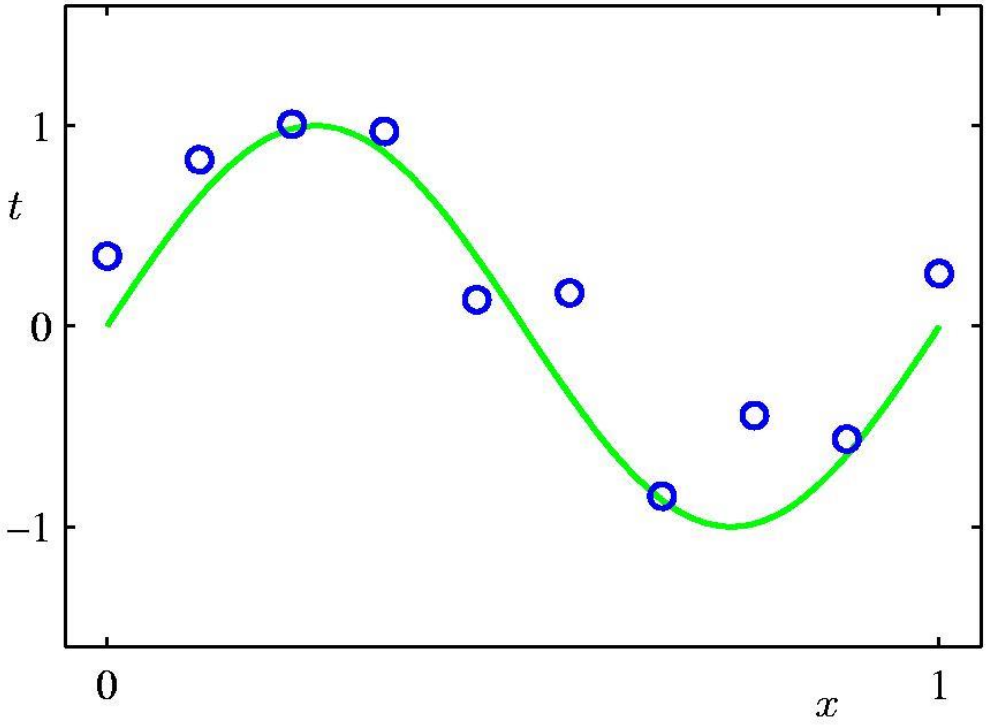
$$\begin{pmatrix} \\ \mathbf{A} \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \mathbf{U} \\ \\ \end{pmatrix} \begin{pmatrix} w_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & w_n \end{pmatrix} \begin{pmatrix} \\ \mathbf{V} \\ \\ \end{pmatrix}^{\mathbf{T}}$$

Singular Value Decomposition (SVD)

- The columns of **U** are orthonormal eigenvectors of AA^T
- The columns of **V** are orthonormal eigenvectors of A^TA
- **W** is a diagonal matrix containing the **square roots** of eigenvalues from **U** or **V** in descending order.
- The w_i are called the **singular/eigen values** of **A**
- If **A** is singular, some of the w_i will be 0
- In general ***rank***(**A**) = number of nonzero w_i

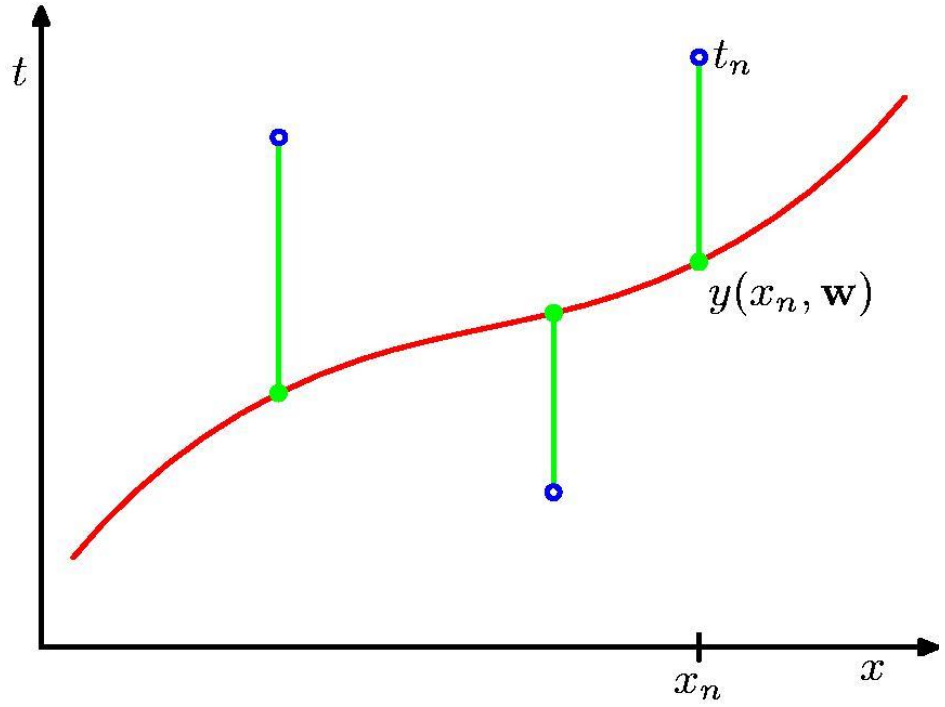
- **SVD is mostly unique.**

Polynomial Curve Fitting



$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

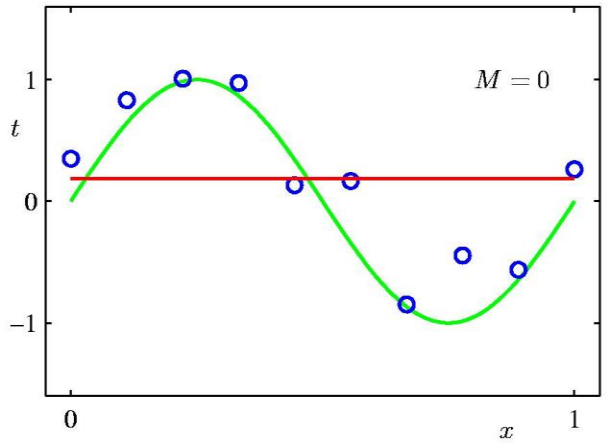
Sum-of-Squares Error Function



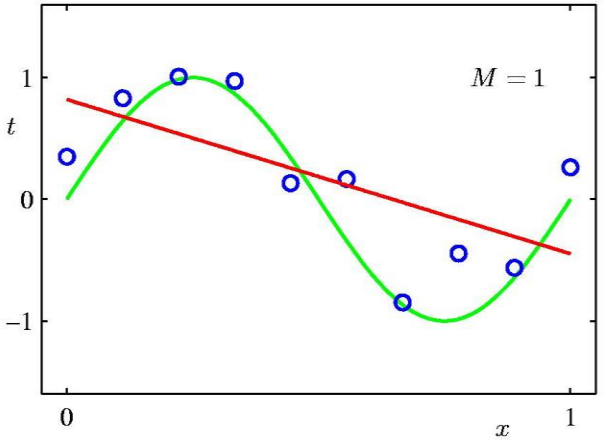
$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$

Effect of Polynomial Order

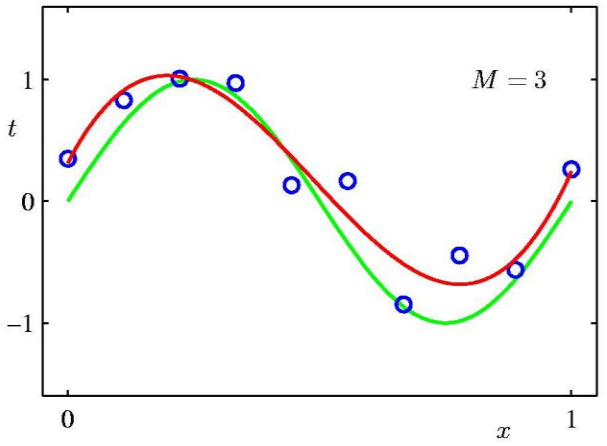
0th Order Polynomial

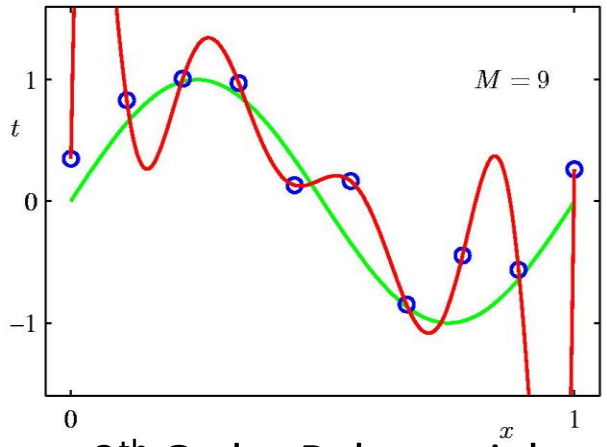


1th Order Polynomial

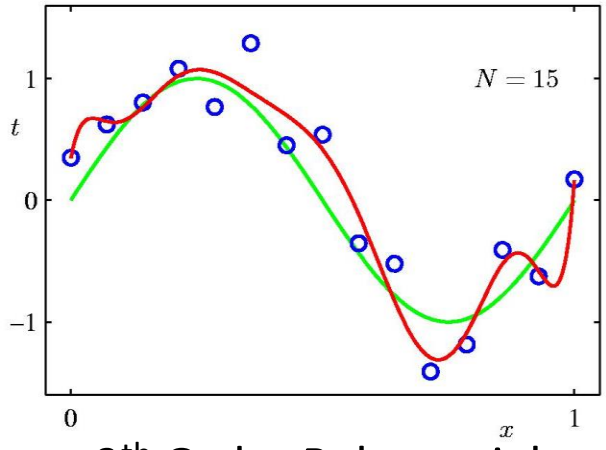


3th Order Polynomial

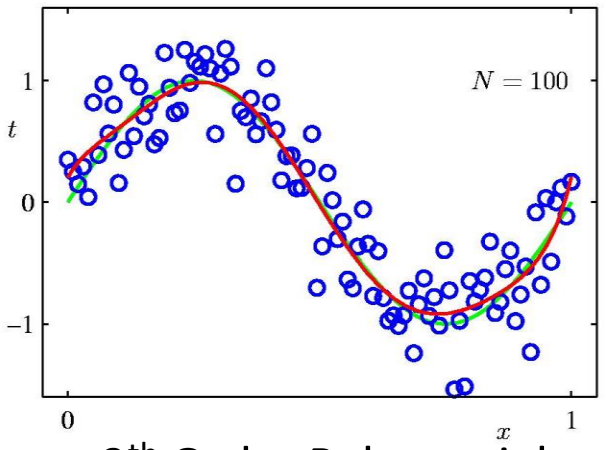




9th Order Polynomial



9th Order Polynomial



9th Order Polynomial

STATISTICAL MEASURES

Statistical Measures

- Center of the data
  - Mean
  - Median
- Variation
  - Range
  - Quartiles
  - Variance
  - Standard Deviation
  - Covariance
  - Correlation

Mean or Average or Expectation

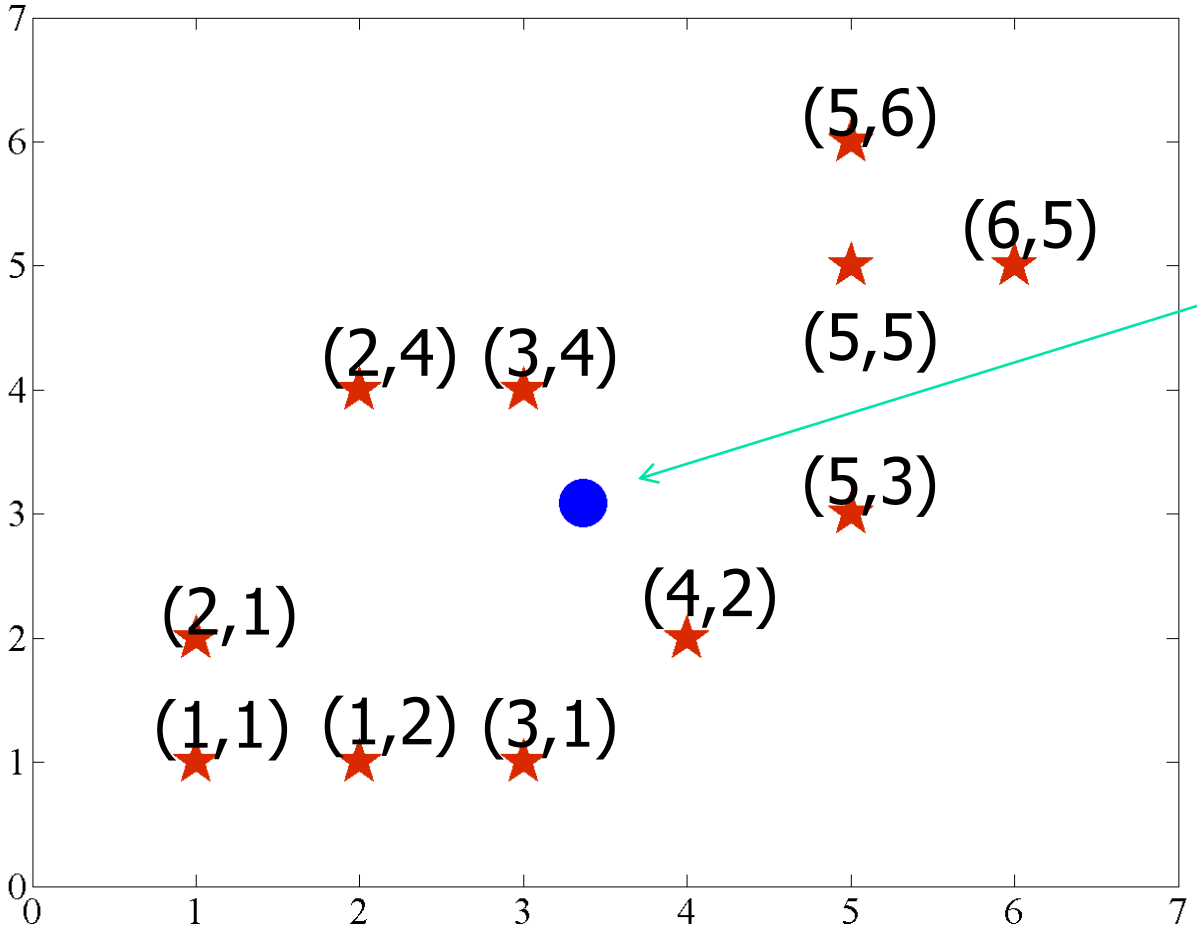
- **Traditional measure of center**
- **Sum the values and divide by the number of values**

$$E(\vec{x}) = \frac{1}{n}(\vec{x}_1 + \vec{x}_2 + \cdots + \vec{x}_n) = \frac{1}{n}\sum_{i=1}^{n} \vec{x}_i$$

- **In general**

$$E(\vec{x}) = (p_1\vec{x}_1 + p_2\vec{x}_2 + \cdots + p_n\vec{x}_n) = \sum_{i=1}^{n} p_i\vec{x}_i$$

Mean or Average



$$Mean = \frac{1}{11}[(1,1) + (1,2) + \cdots + (6,5)] = (3.3636,\ 3.0909)$$

Median (*M*)

- A *resistant measure* of the data's center
  - At least half of the *ordered* values are less than or equal to the median value
  - At least half of the *ordered* values are greater than or equal to the median value
- If *n* is odd, the median is the middle ordered value
- If *n* is even, the median is the average of the two middle ordered values
  - Location of the median: $L(M) = (n+1)/2$, where *n* = sample size.
  - **Example**: If 25 data values are recorded, the Median would be the $(25+1)/2 = 13^{th}$ ordered value.

Median

- **Example 1 data: 2 4 6**

  Median (M) = 4

- **Example 2 data: 2 4 6 8**

  Median = 5 (average of 4 and 6)

- **Example 3 data: 6 2 4**

  Median $\neq$ 2

  (order the values: 2 4 6 , so Median = 4)

Comparing the Mean & Median

- Computation of mean is easier.
- Finding median in higher dimension is much complex.
- Mean is prone to noise.
- The mean and median of data from a symmetric distribution should be close together. The actual (true) mean and median of a symmetric distribution are exactly the same.

Spread or Variability

- If all values are the same, then they all equal to the mean. There is no variability.
  - Eg: 2, 2, 2, 2, 2, 2; mean = 2
- Variability exists when some values are different from (above or below) the mean.
  - Eg: 10, 15,-20,-22,30, 22
- We will discuss the following measures of spread:
  - range,
  - quartiles,
  - variance,
  - and standard deviation

Range

- **One way to measure spread is to give the smallest (*minimum*) and the largest (*maximum*) values in the dataset;**

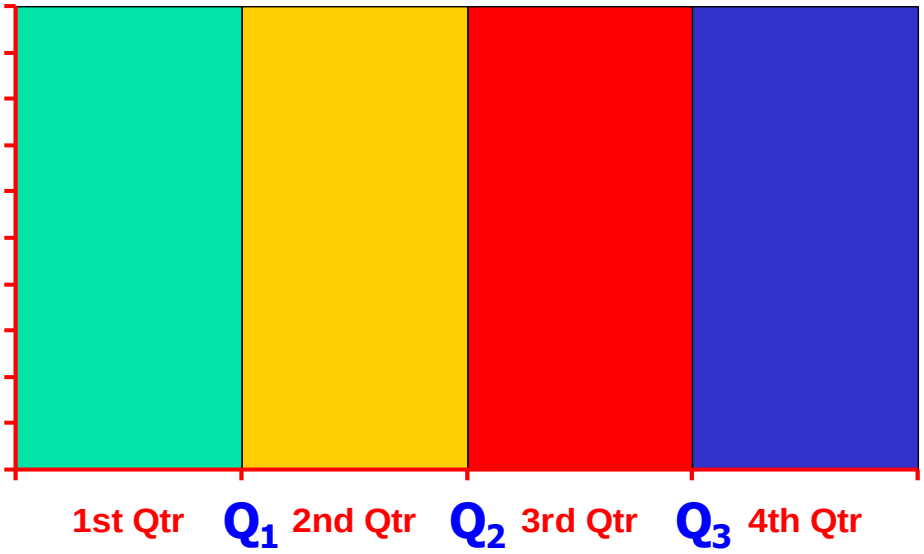
$$\text{Range} = \max - \min$$

  - **Eg: 10,-2,-7,22,0,11; Range = 22-(-7)=28**

- **The range is strongly affected by outliers**

Quartiles

- **Three numbers which divide the ordered data into four equal sized groups:**
  - **Q_1 has 25% of the data below it.**
  - **Q_2 has 50% of the data below it. (Median)**
  - **Q_3 has 75% of the data below it.**



Quartiles of Uniform Distribution Dataset

Variance and Standard Deviation

- **Recall that variability exists when some values are different from (above or below) the mean.**
- **Each data value has an associated *deviation from the mean*:**

$$x_i - \bar{x}$$

- **what is a *typical* deviation from the mean? *(standard deviation)***
  - **small values of this typical deviation indicate small variability in the data**
  - **large values of this typical deviation indicate large variability in the data**

Variance and Standard Deviation

Variance is the average squared deviation from the **mean** of a set of data. It is used to find the **standard deviation**.

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

standard deviation = square root of the variance

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Variance and Standard Deviation

Metabolic rates of 7 men (cal./24hr.) :

1792 1666 1362 1614 1460 1867 1439

$$\bar{X} = \frac{1792+1666+1362+1614+1460+1867+1439}{7}$$

$$= \frac{11,200}{7}$$

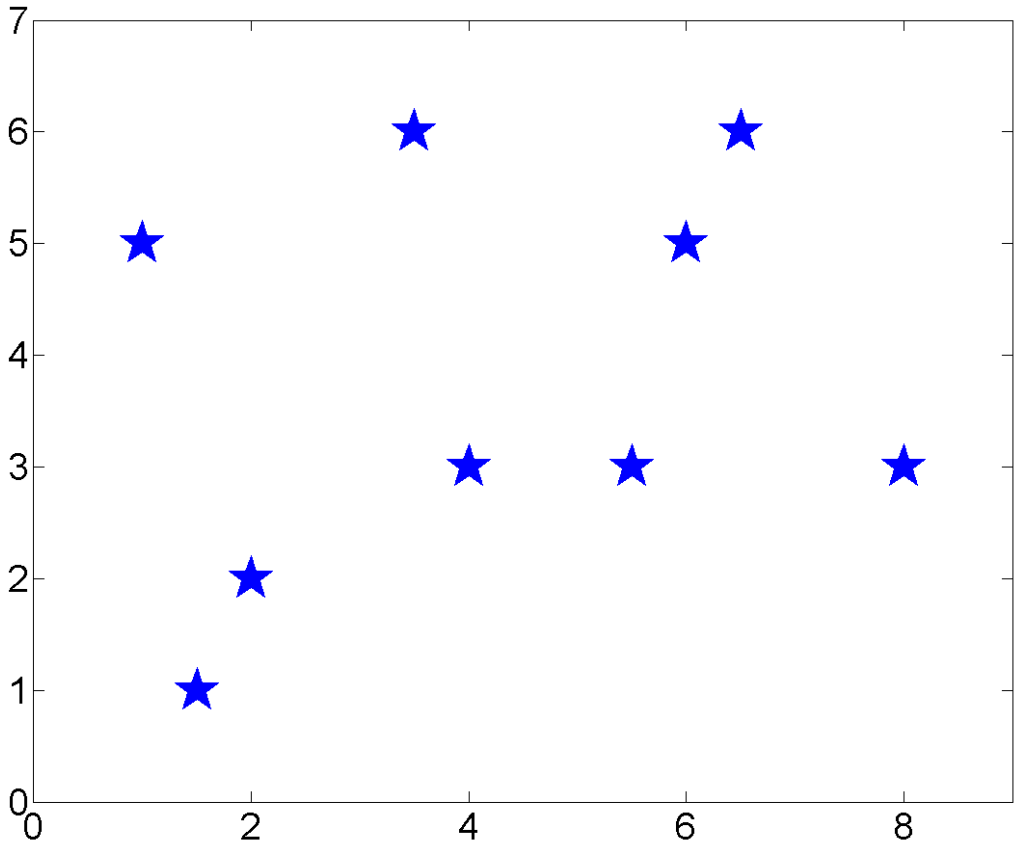
$$= 1600$$

Variance and Standard Deviation

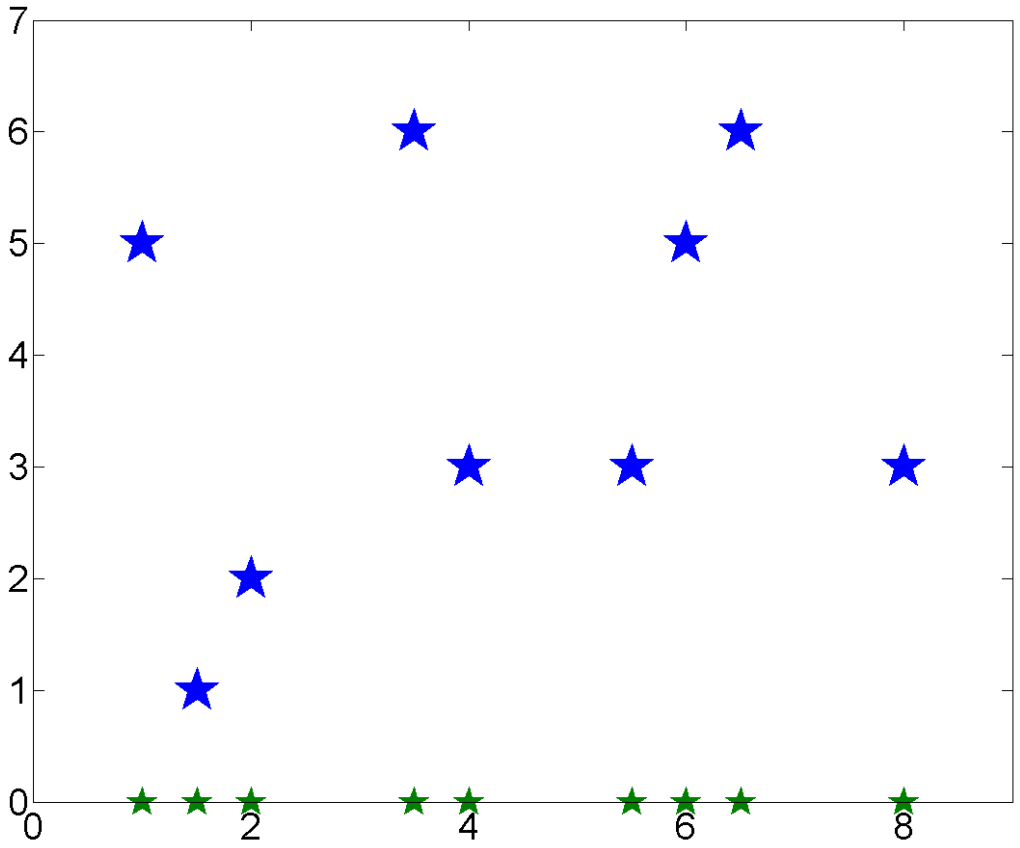
Observations x_i	Deviations $x_i - \bar{x}$	Squared deviations $(x_i - \bar{x})^2$
1792	1792−1600 = 192	$(192)^2$ = 36,864
1666	1666 −1600 = 66	$(66)^2$ = 4,356
1362	1362 −1600 = -238	$(-238)^2$ = 56,644
1614	1614 −1600 = 14	$(14)^2$ = 196
1460	1460 −1600 = -140	$(-140)^2$ = 19,600
1867	1867 −1600 = 267	$(267)^2$ = 71,289
1439	1439 −1600 = -161	$(-161)^2$ = 25,921
	sum = 0	sum = 214,870

$$\sigma^2 = \frac{214{,}870}{7} = 30695.71 \qquad \sigma = \sqrt{3069571} = 175.20 \text{ calories}$$

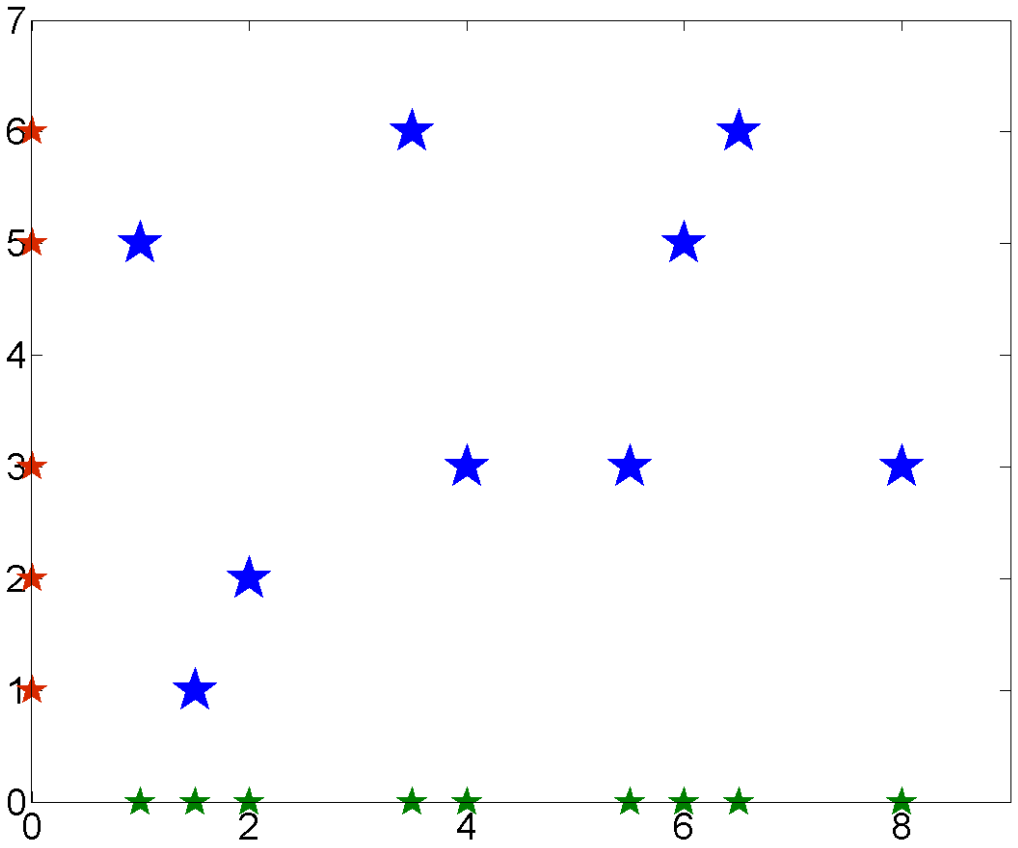
Variance (2D)



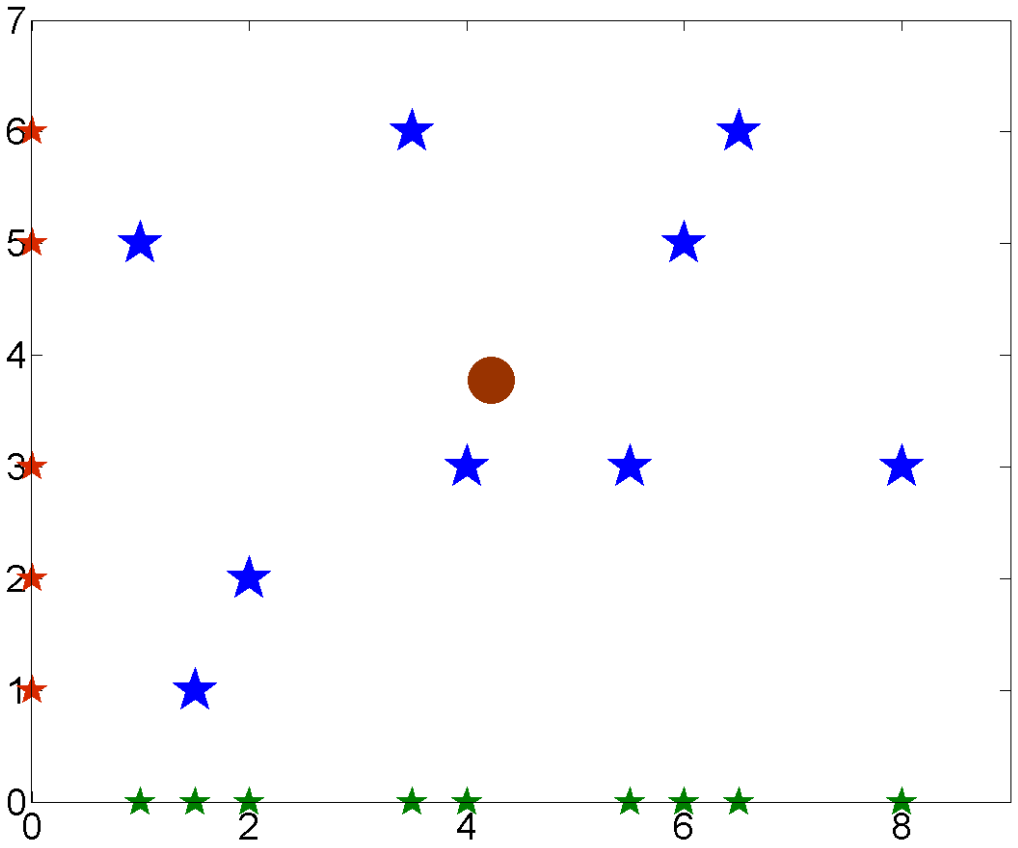
Variance (2D)



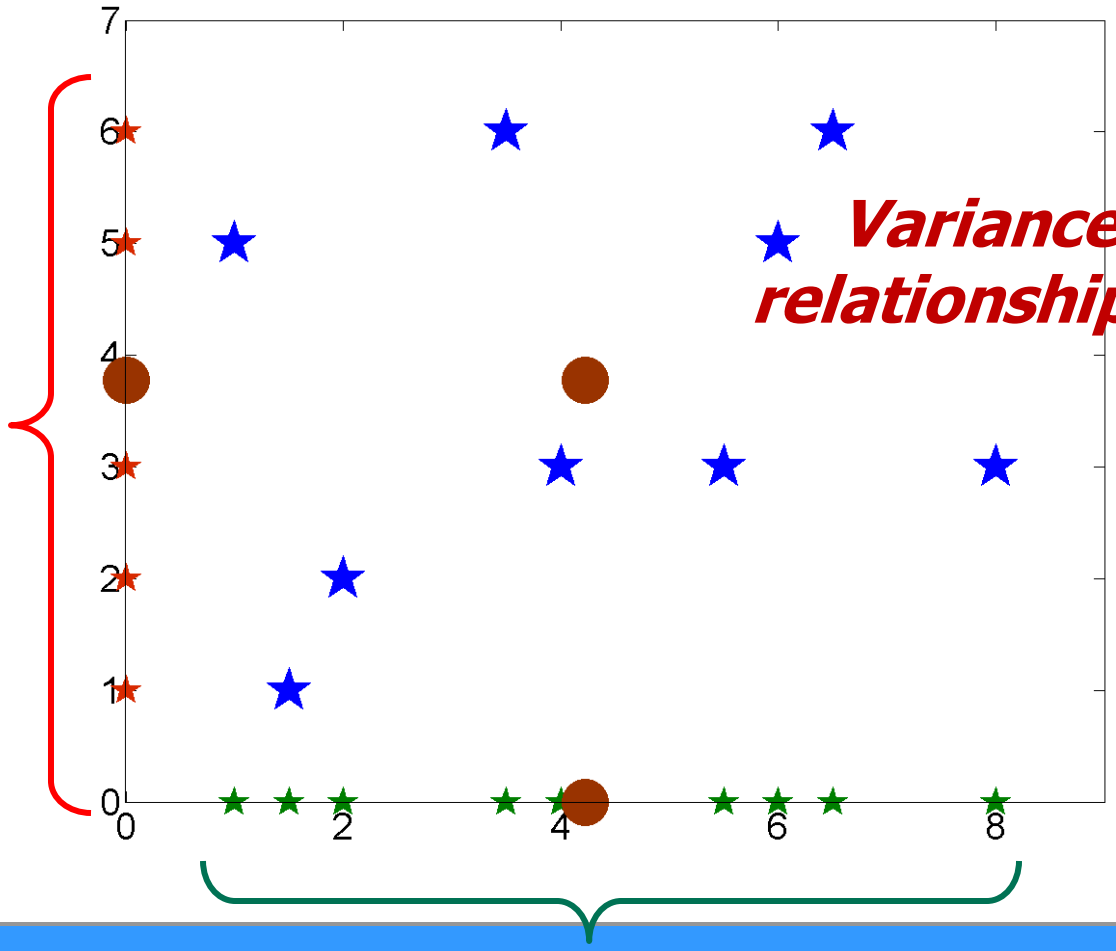
Variance (2D)



Variance (2D)



Variance (2D)



Variance doesn't explor
relationship between varia

Covariance

$$\text{Variance(x)} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

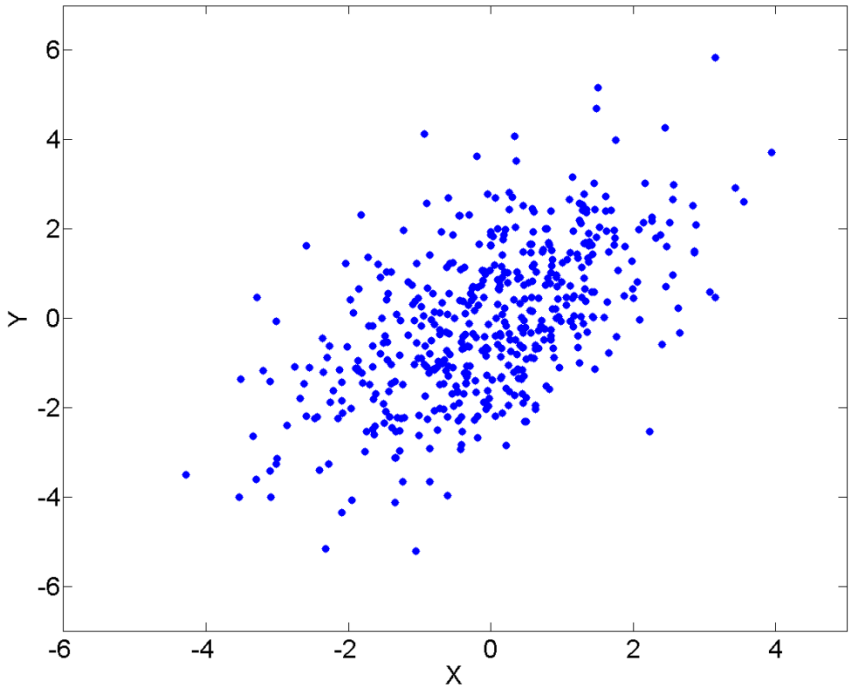
$$= \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})$$

$$\text{Covariance(x, y)} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

- $\text{Covariance(x, x)} = \text{var(x)}$
- $\text{Covariance(x, } y\text{)} = \text{Covariance(y, x)}$

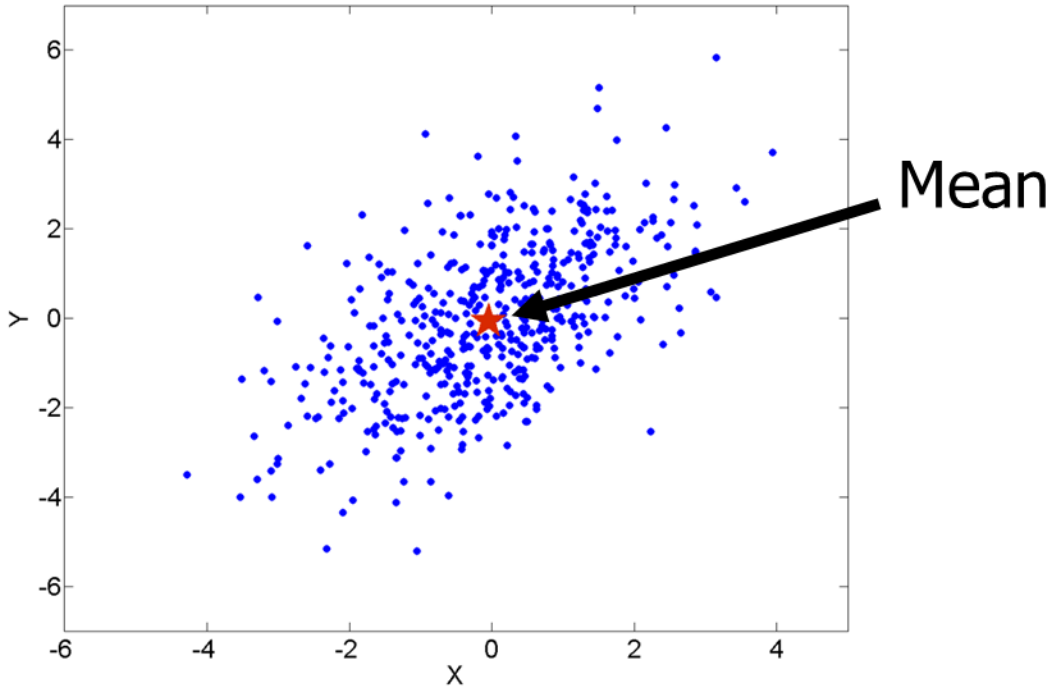
Covariance

$$\text{Covariance}(x, y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$



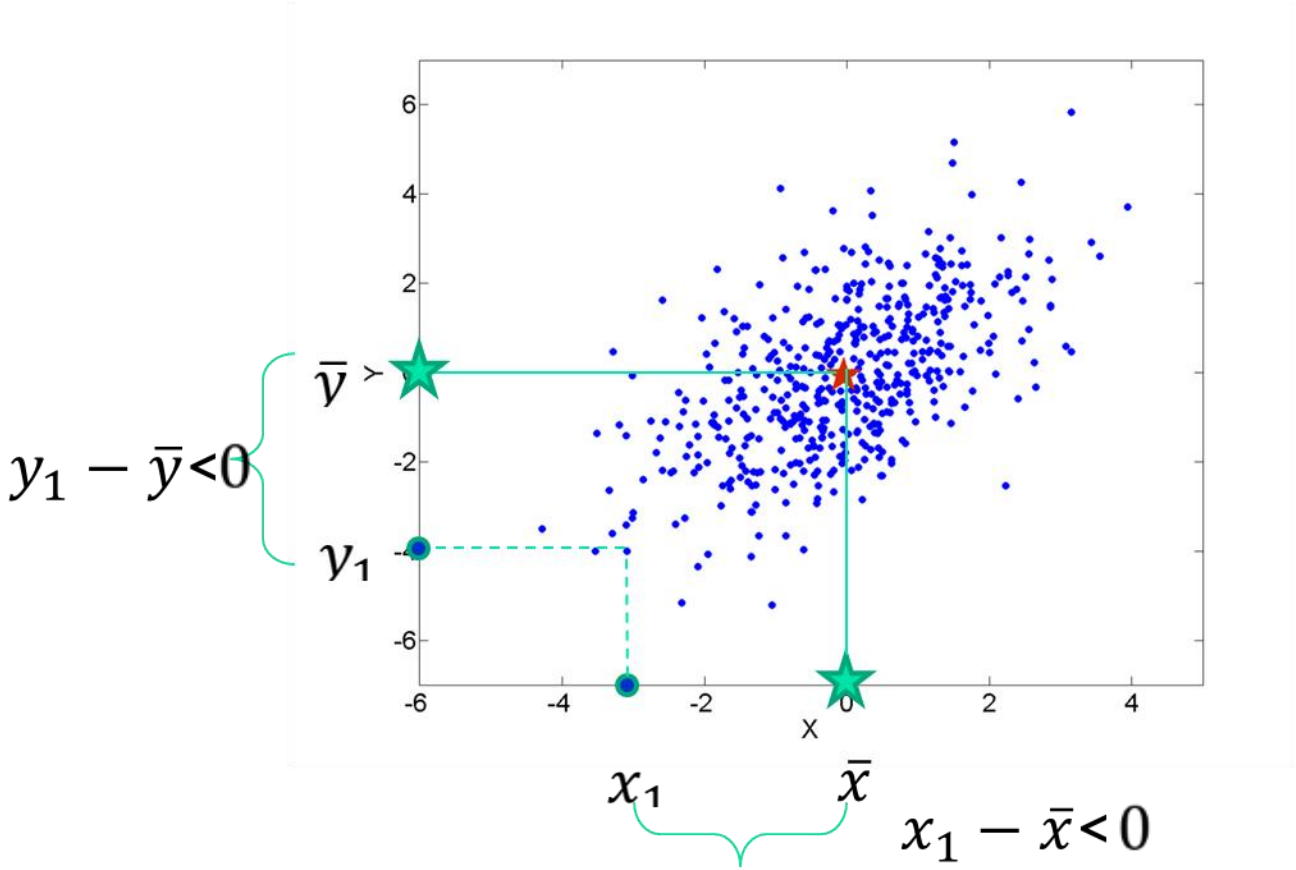
Covariance

$$\text{Covariance}(x, y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$



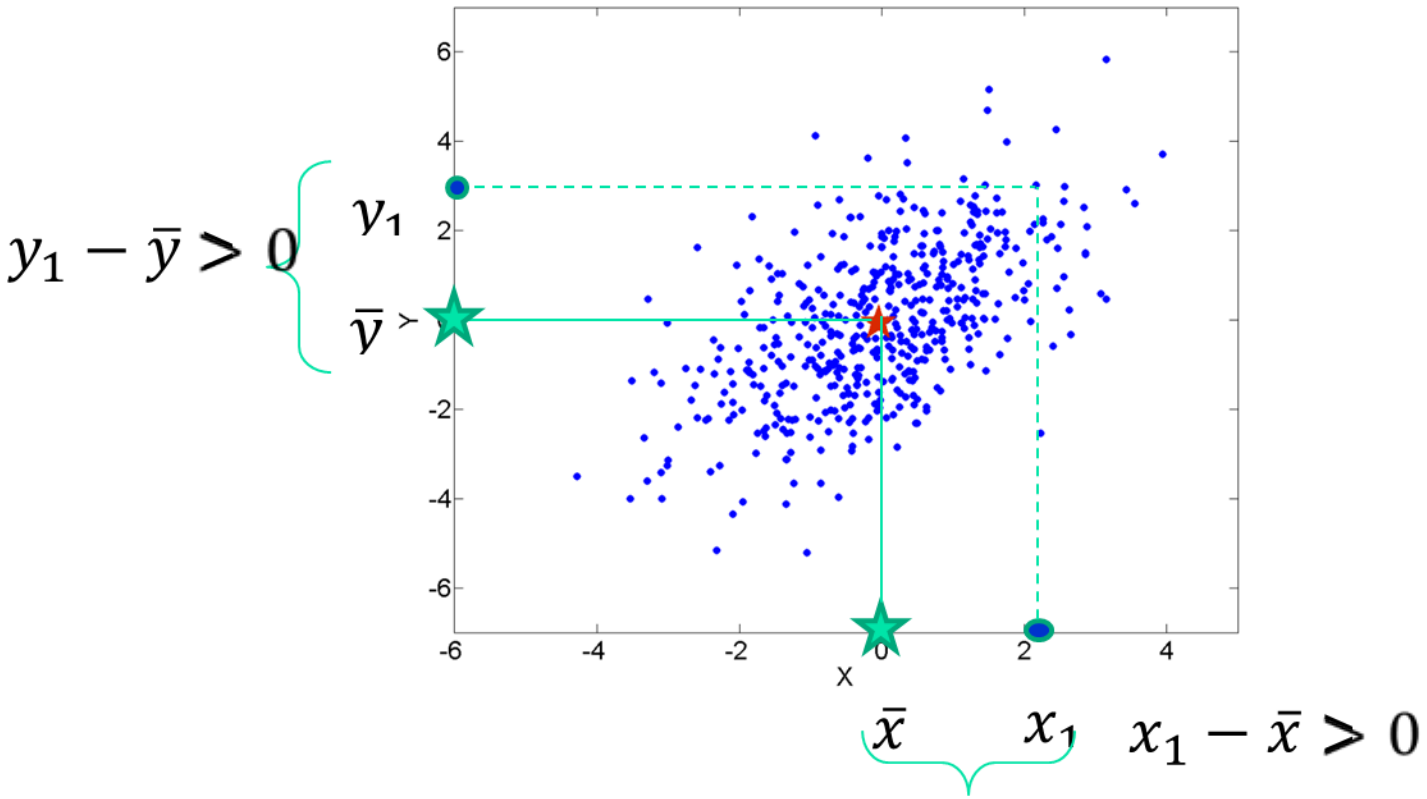
Covariance

$$\text{Covariance}(x, y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$



Covariance

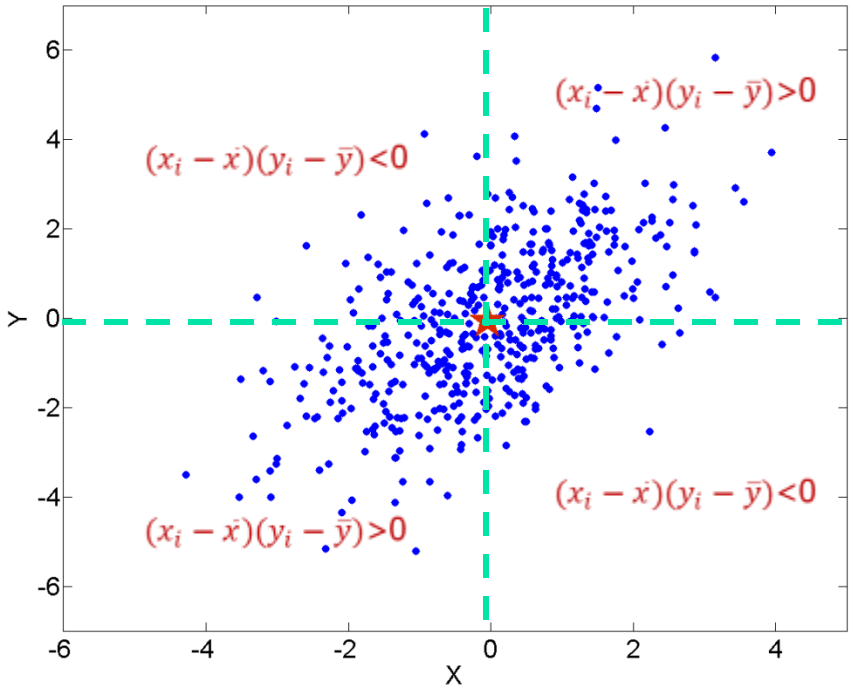
$$\text{Covariance}(x, y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$



Covariance

$$\text{Covariance}(x, y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

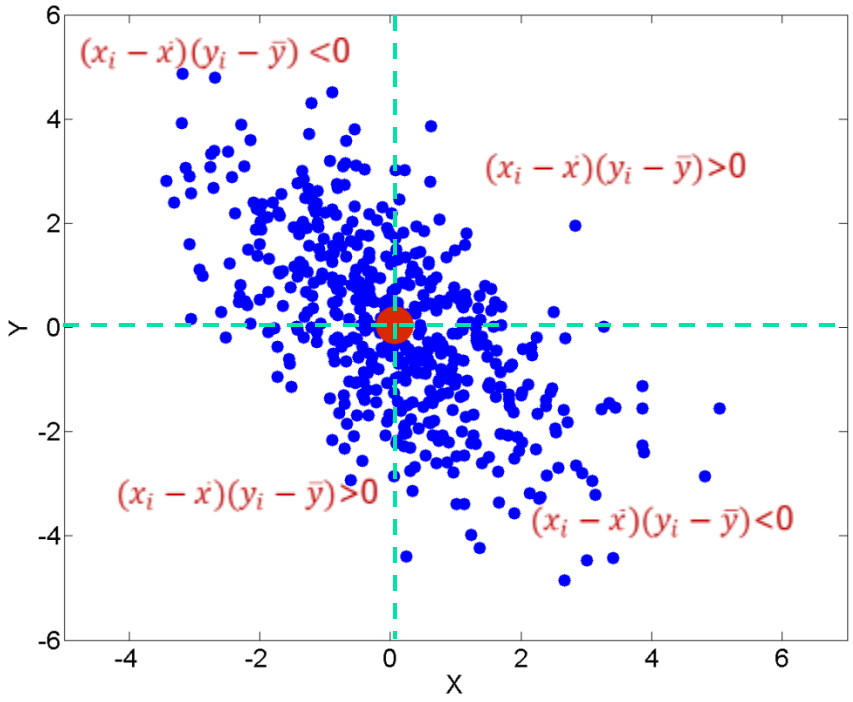
Positive Relation



Covariance

$$\text{Covariance}(x, y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

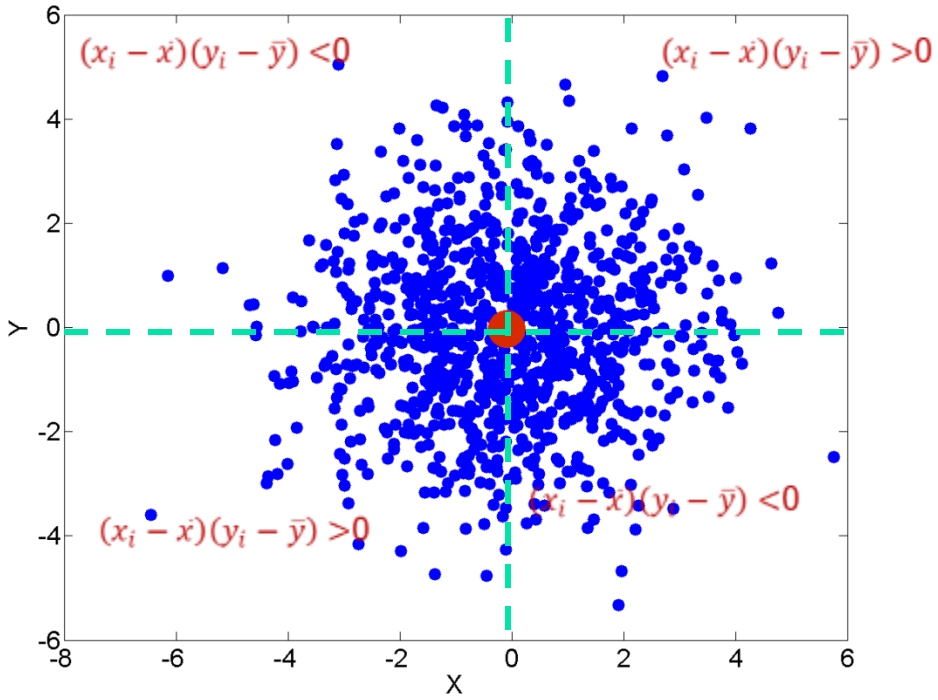
Negative Relation



Covariance

$$\text{Covariance}(x, y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

No Relation



Covariance Matrix

$$Cov\ (\sum) = \begin{bmatrix} cov(x_1, x_1) & cov(x_1, x_2) & \cdots & cov(x_1, x_n) \\ cov(x_2, x_1) & cov(x_2, x_2) & \cdots & cov(x_2, x_n) \\ \vdots & \vdots & \vdots & \vdots \\ cov(x_n, x_1) & cov(x_n, x_2) & \cdots & cov(x_n, x_n) \end{bmatrix}$$

- Diagonal elements are variances, i.e. $\mathrm{Cov}(x, x) = var(x)$
- Covariance Matrix is symmetric.
- It is a positive semi-definite matrix.

Diagonal covariance matrix

- If all the variables are independent from each other,
- The covariance matrix will be an diagonal one.
- Reverse is also true:
- If the covariance matrix is a diagonal one they are independent

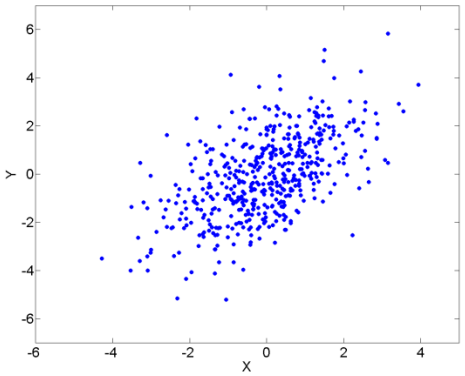
$$\begin{bmatrix} \sigma^2_1 & 0 \\ 0 & \sigma^2_2 \end{bmatrix}$$

Diagonal matrix: m matrix where off-diagonal terms are zero

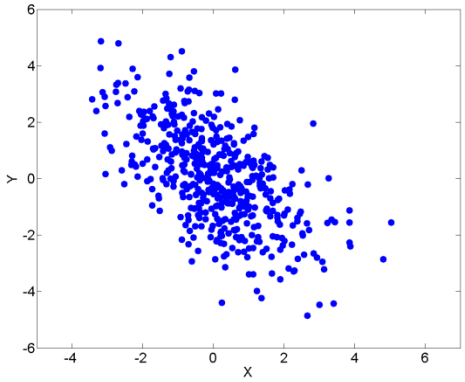
$$\sigma_{ij}^2 = \mathrm{E}[(x_i - \mu_i)(x_j - \mu_j)] = 0$$

$$i \neq j$$

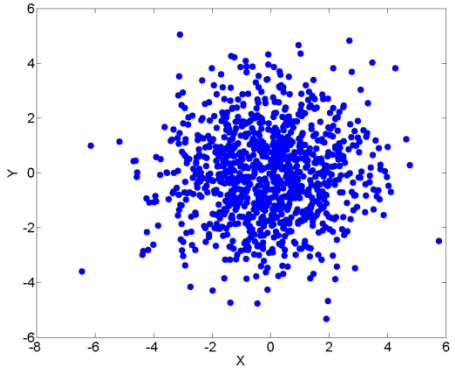
Correlation



Positive relation



Negative relation



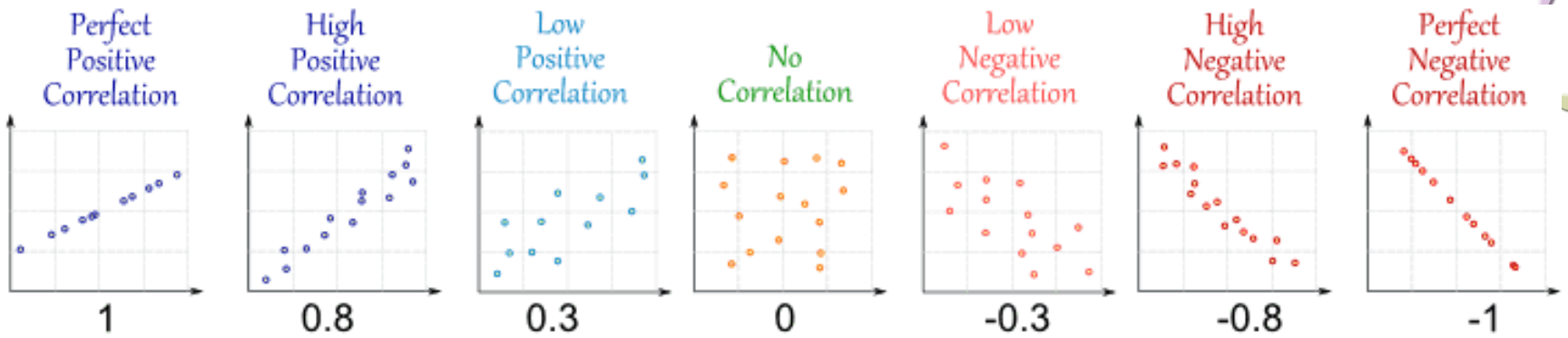
No relation

- **Covariance determines whether relation is positive or negative, but it was impossible to measure the degree to which the variables are related.**
- **Correlation is another way to determine how two variables are related.**
- **In addition to whether variables are positively or negatively related, correlation also tells the degree to which the variables are related each other.**

Correlation

$$\rho_{xy} = Correlation\ (x, y) = \frac{cov(x, y)}{\sqrt{var(x)}\sqrt{var(y)}.}$$

$$-1 \leq Correlation(x, y) \leq +1$$



Probability

Idea of Probability

- **Probability is the science of chance behavior.**

- **Chance behavior is unpredictable in the short run but has a regular and predictable pattern in the long run.**

Randomness & Random Experiment

- **Random: individual outcomes are uncertain.**
  - **But there is a regular distribution of outcomes in a large number of repetitions.**
  - **If an experiment has n possible outcomes [all equally likely to occur].**
  - **A *random experiment* is an action or process that leads to one of several possible outcomes. For example:**

Experiment	Outcomes
Flip a coin	Heads, Tails
Selecting a color ball	Green, red, blue
Rolling a die	1,2,3,4,5,6
Picking a card from a deck	52 cards

Relative-Frequency Probabilities

- ***Relative frequency*** **(proportion of occurrences) of an outcome settles down to one value over the long run. That *one value* is then defined to be the *probability* of that outcome.**
- **Can be determined (or checked) by observing a long series of *independent* trials (empirical data)**
  - **experience with many samples**
  - **simulation**

Probability Models

- The **sample space S** of a random phenomenon is the set of all possible outcomes.
- An **event** is an outcome or a set of outcomes (subset of the sample space).
- A **probability model** is a mathematical description of long-run regularity consisting of a **sample space S** and a **way of assigning probabilities** to events.

Example of Sample Space and Events

Event 1:
Rolling an even number={2,4,6}

Event 2:
Rolling an odd number={1,3,5}

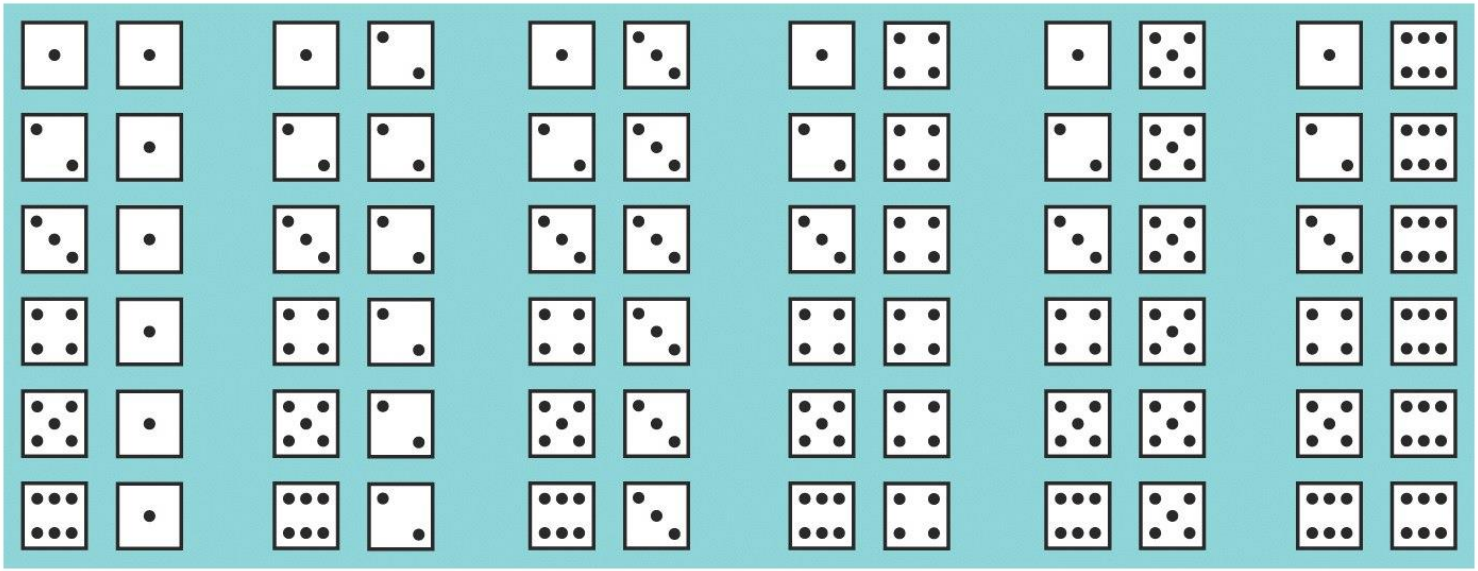
Sample Space ={1,2,3,4,5,6}

Event 3:
Rolling a prime number={2,3,5}

Probability Model for Two Dice

Random phenomenon: roll pair of fair dice.

Sample space:



Event: rolling even numbers on both dice

What is a Probability?

- **Probability is the chance that some event will happen.**

- **It is the ratio of the number of ways a certain event can occur to the number of possible outcomes:**

$$P(\text{event}) = \frac{\text{number of favorable outcomes}}{\text{number of possible outcomes}}$$

Probability of Simple Events

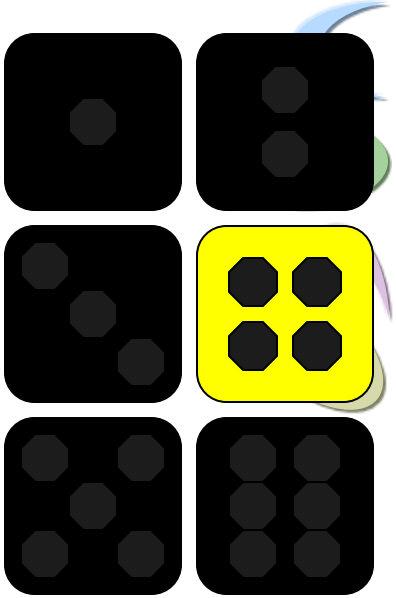
Example 1: Roll a dice.

What is the probability of rolling a 4?

$$P(event) = \frac{number\ of\ favorable\ outcomes}{number\ of\ possible\ outcomes}$$

$$P(rolling\ a\ 4) = \frac{1}{6}$$

The probability of rolling a 4 is 1 out of 6

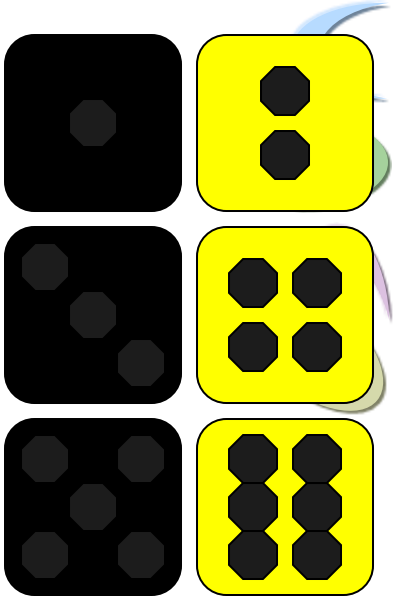


Probability of Simple Events

Example 2: Roll a dice.

What is the probability of rolling an even number?

$$P(even\ \#) = \frac{\#\ favorable\ outcomes}{\#possible\ outcomes} = \frac{3}{6} = \frac{1}{2}$$



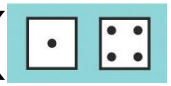
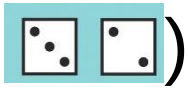
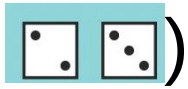
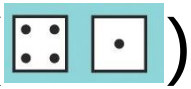
The probability of rolling an even number is 3 out of 6.

Probability of Simple Events

Example 3: Roll a dice.

Random phenomenon: roll pair of fair dice and count the number of pips on the up-faces.

Find the probability of rolling a 5.

$P(\text{roll a 5}) = P($  $)+P($  $)+P($  $)+P($  $)$

$= 1/36 + 1/36 + 1/36 + 1/36$

$= 4/36$

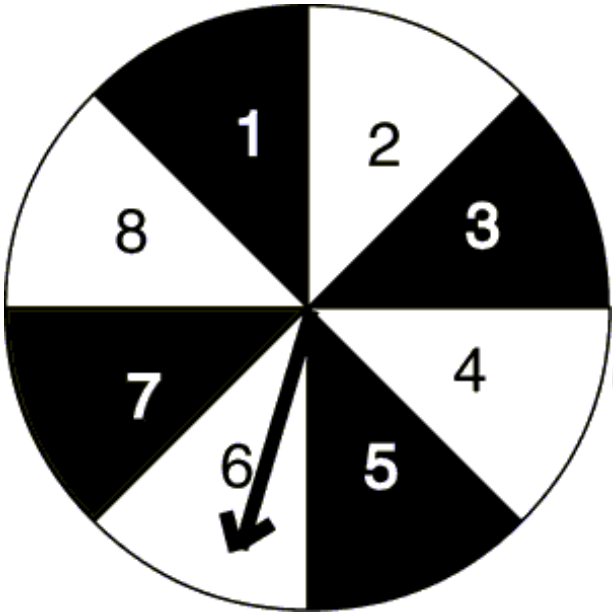
$= 0.111$

Probability of Simple Events

Example 4: Guided Practice

Calculate the probability of each independent event.

1) P(black) = 1/2
2) P(1) = 1/8
3) P(odd) = 1/2
4) P(prime) = 1/2
5) P(even) = 1/2
6) P(not black) = 1/2



Probability of Simple Events

Real World Example:

A computer company manufactures 2,500 computers each day. An average of 100 of these computers are returned with defects. What is the probability that the computer you purchased is *not* defective?

$$P(\text{not defective}) = \frac{\#\ favorable\ outcomes}{\#possible\ outcomes} = \frac{2400}{2500} = \frac{24}{25}$$

Complementary Events

- The complement of an event E is the set of all outcomes in a sample space that are not included in event E.

- The complement of an event E is denoted by $E' or\ \bar{E}$

Properties of Probability:

$$0 \le P(E) \le 1$$

$$P(E) + P(\bar{E}) = 1$$

$$P(E) = 1 - P(\bar{E})$$

$$P(\bar{E}) = 1 - P(E)$$

Complementary Events

- Example I: A sequence of 5 bits is randomly generated. What is the probability that at least one of these bits is zero?
- Solution: There are $2^5 = 32$ possible outcomes of generating such a sequence.

Define event E as at least one of the bits is zeros

Then event $\bar{E}$, "none of the bits is zero", includes only one

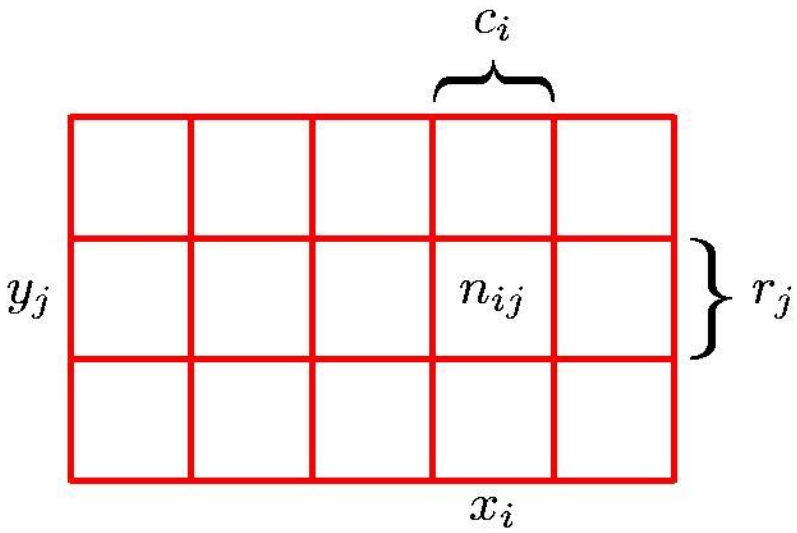
of these outcomes, namely the sequence 11111.

Therefore, $p(\bar{E}) = 1/32$.

Now p(E) can easily be computed as

$p(E) = 1 - p(\bar{E}) = 1 - 1/32 = 31/32$.

Probability Theory



- **Marginal Probability**

$$p(X = x_i) = \frac{c_i}{N}.$$

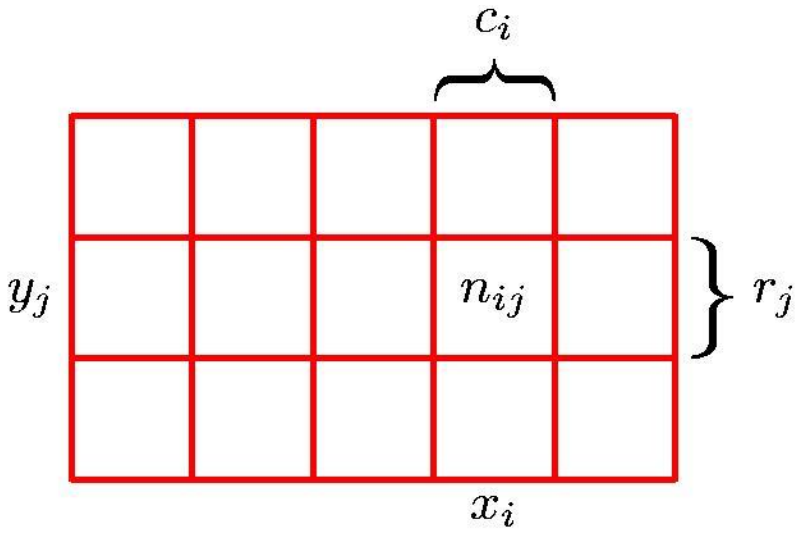
- **Joint Probability**

$$p(X = x_i, Y = y_j) = \frac{n_{ij}}{N}$$

- **Conditional Probability**

$$p(Y = y_j | X = x_i) = \frac{n_{ij}}{c_i}$$

Probability Theory



- **Sum Rule**

$$p(X = x_i) = \frac{c_i}{N} = \frac{1}{N}\sum_{j=1}^{L} n_{ij}$$

$$= \sum_{j=1}^{L} p(X = x_i, Y = y_j)$$

- **Product Rule**

$$p(X = x_i, Y = y_j) = \frac{n_{ij}}{N} = \frac{n_{ij}}{c_i} \cdot \frac{c_i}{N}$$

$$= p(Y = y_j | X = x_i)p(X = x_i)$$

- **Sum Rule** $\quad p(X) = \sum_{Y} p(X, Y)$
- **Product Rule** $\quad p(X, Y) = p(Y|X)p(X)$

The Multiplication Rule

- If events A and B are **independent**, then the probability of two events, A and B occurring in a sequence (or simultaneously) is:

$$P(A \cap B) = P(A) \times P(B)$$

- This rule can extend to any number of independent events.

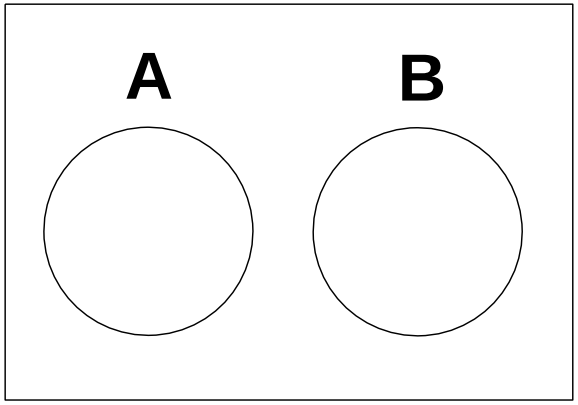
- Two events are independent if the occurrence of the first event does not affect the probability of the occurrence of the second event.

Mutually Exclusive

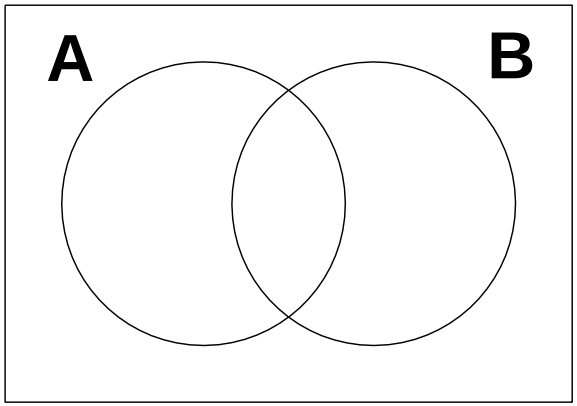
Two events A and B are mutually exclusive if and only if:

$$P(A \cap B) = 0$$

In a Venn diagram this means that event A is disjoint from event B.



A and B are M.E.



A and B are not M.E.

The Addition Rule

- **The probability that at least one of the events A or B will occur, P(A or B), is given by:**

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

- **If events A and B are mutually exclusive, then the addition rule is simplified to:**

$$P(A \cup B) = P(A) + P(B)$$

- **This simplified rule can be extended to any number of mutually exclusive events.**

Conditional Probability

The probability that both event A and event B occur is given by

$$P(\text{A and B}) = P(\text{A}|\text{B}) \times P(\text{B})$$

We often use this in the form

$$P(\text{A}|\text{B}) = \frac{P(\text{A and B})}{P(\text{B})}$$

In words, this is "the probability of event A given that B has occurred, equals the probability of both A and B occurring divided by the probability of B".

Reminder :

***P*(A and B) can also be written as *P*(A ∩ B)**

Bayes' Theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$P(A|B) = \frac{P(B|A)P(A)}{\sum_n P(B|A_n)P(A_n)}$$

- **The important consequence of Bayes' Theorem is that it relates inverse probabilities: P(A|B) *and* P(B|A)**

- **posterior ∝ likelihood × prior**

Random Variables

- A **random variable** is a variable whose value is a numerical outcome of a random experiment
  - often denoted with capital alphabetic symbols (*X*, *Y*, etc.)
  - a normal random variable may be denoted as $X \sim N(\mu, \sigma)$
- The **probability distribution** of a random variable *X* tells us what values *X* can take and how to assign probabilities to those values

Discrete Random Variables

- **Random variables that have a finite (countable) list of possible outcomes, with probabilities assigned to *each* of these outcomes, are called discrete**
- **Discrete random variables**
  - **numerical day of the month (1, 2, …, 31)**
  - **the total number of tails you get if you flip 100 coins**
- **Important discrete distributions in epidemiology**
  - **Binomial**
    - **Yes/no outcomes (dead/alive, treated/untreated, smoker/non-smoker, sick/well, etc.)**
  - **Poisson**
    - **Counts (e.g., how many cases of disease in a given area)**

Continuous Random Variables

- **Random variables that can take on any value in an interval, with probabilities given as areas under a density curve, are called continuous.**

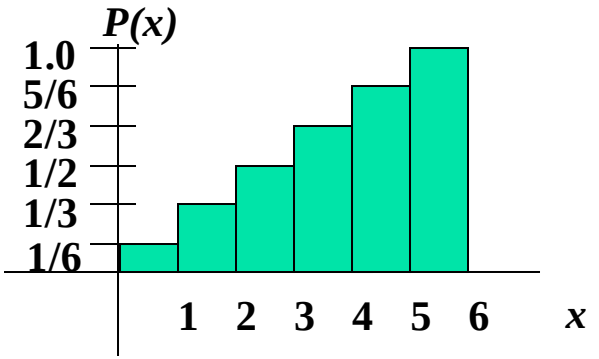
- **Continuous random variables**
  - **weight**
  - **temperature**

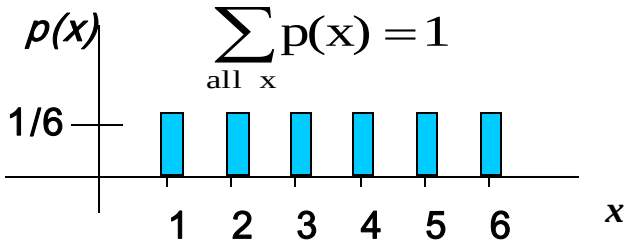
Probability Density Function (PDF)

- The probability function that accompanies a continuous random variable is a continuous mathematical function that integrates to 1.
- The probabilities associated with continuous functions are just areas under the curve (integrals!).
- Probabilities are given for a range of values, rather than a particular.

Probability Distribution Function (PDF)
Cumulative Distribution Function (CDF)

variable	*pdf*	*CDF*
X	*p(x)*	*P(x≤A)*
1	*p(x=1)* = 1/6	*P(x≤1)* = 1/6
2	*p(x=2)* = 1/6	*P(x≤2)* = 2/6
3	*p(x=3)* = 1/6	*P(x≤3)* = 3/6
4	*p(x=4)* = 1/6	*P(x≤4)* = 4/6
5	*p(x=5)* = 1/6	*P(x≤5)* = 5/6
6	*p(x=6)* = 1/6	*P(x≤6)* = 6/6





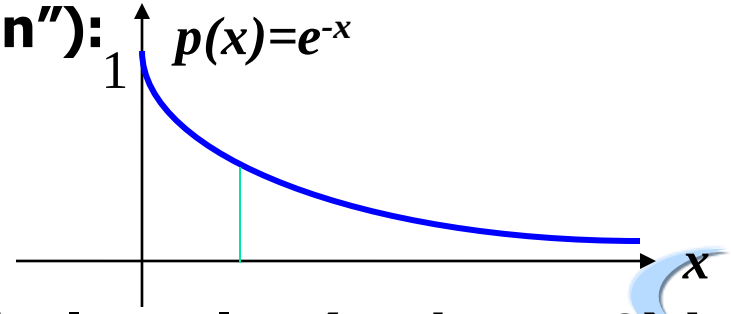
1. **What's the probability that you roll a 3 or less?**
   ***P(x≤3)*=1/2**

2. **What's the probability that you roll a 5 or higher?**
   ***P(x≥5)* = 1 – *P(x≤4)* = 1-2/3 = 1/3**

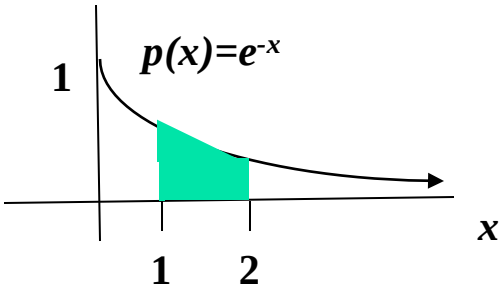
Probability Density Function (PDF) & Cumulative Density Function (CDF)

- **For example, the negative exponential function (in probability, this is called an "exponential distribution"):**

$$p(x) = e^{-x}$$



- **The probability that *x* is any exact particular value (such as 1.9) is 0; we can only assign probabilities to possible ranges of x. For example, the probability of *x* falling within 1 to 2:**



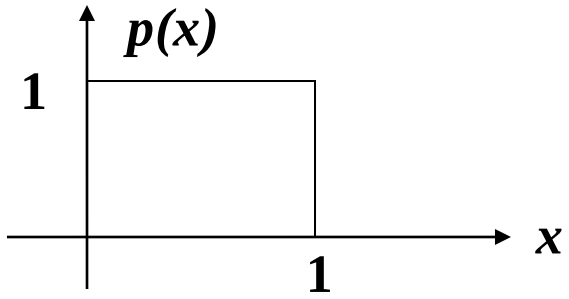
$$P(1 \le x \le 2) = \int_1^2 e^{-x} = -e^{-x} \Big|_1^2 = -e^{-2} - -e^{-1} = -.135 + .368 = .23$$

- **The CDF here :** $P(x \le A) = \int_0^A e^{-x} = -e^{-x} \Big|_0^A = -e^{-A} - -e^{-0} = 1 - e^{-A}$

Uniform Density

- The uniform distribution: all values are equally likely
- The uniform distribution:

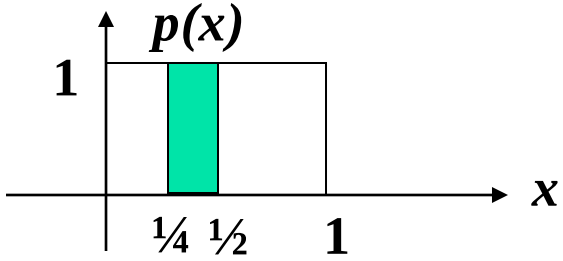
  $f(x) = 1$, for $1 \geq x \geq 0$



- We can see it's a probability distribution because it integrates to 1 (the area under the curve is 1):

$$\int_0^1 1 = x\Big|_0^1 = 1 - 0 = 1$$

- What's the probability that x is between ¼ and ½?



P(1/4 ≤ x≤ 1/2)= 1/4

Normal (Gaussian) Distribution

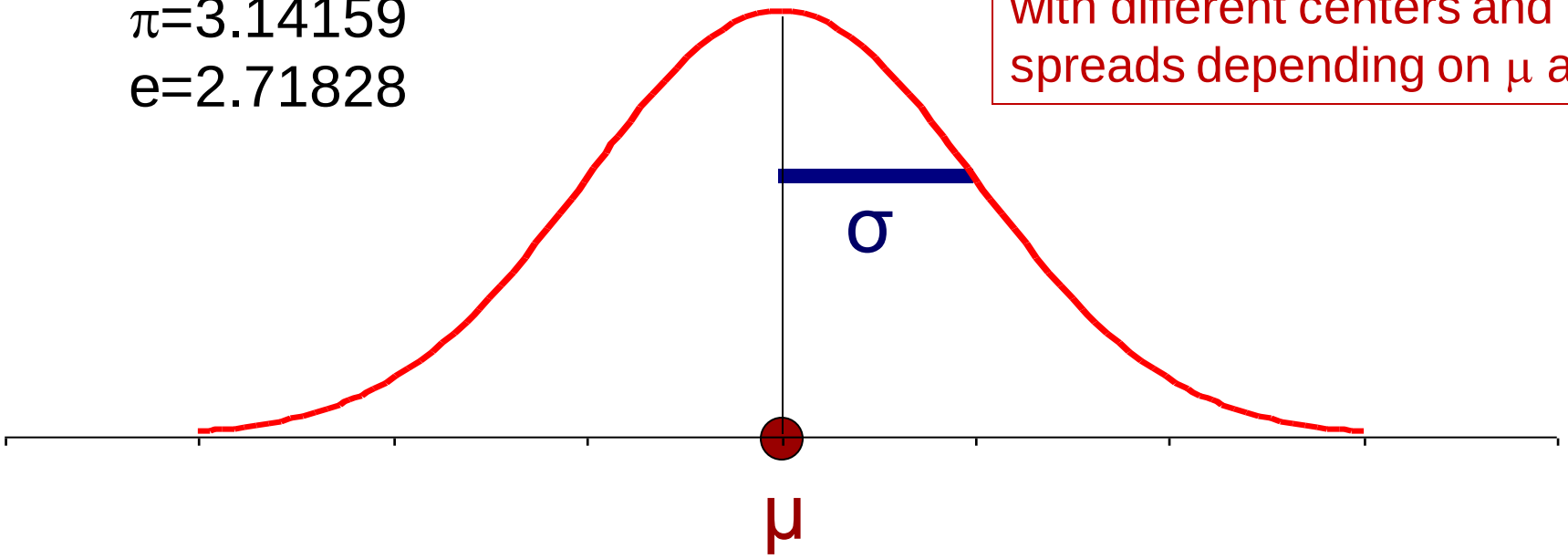
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

This is a bell shaped curve with different centers and spreads depending on μ and σ

Note constants:

π=3.14159

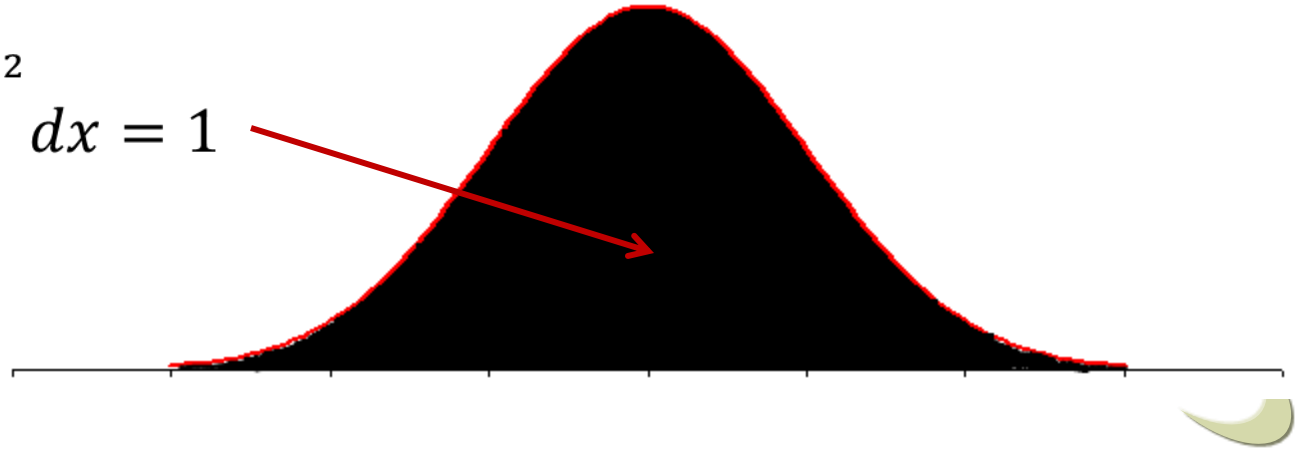
e=2.71828



Normal (Gaussian) Distribution

- **It's a probability function, so no matter what the values of μ and σ, must integrate to 1!**

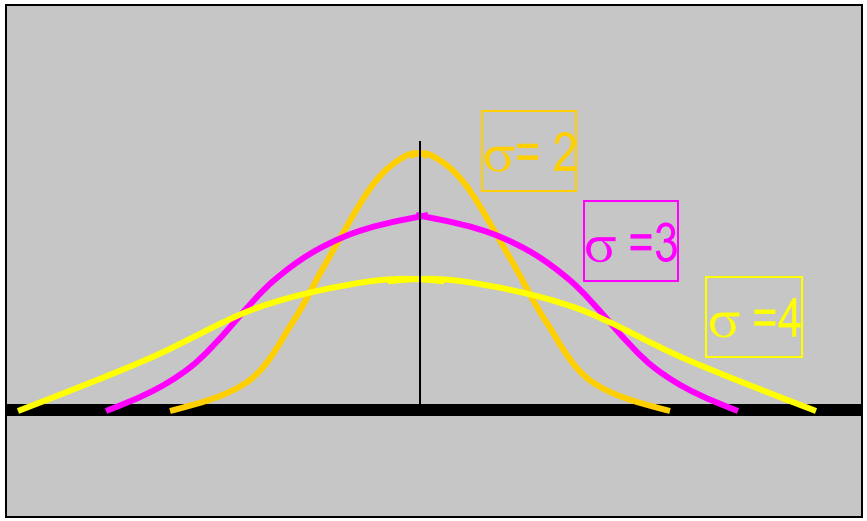
$$\int_{-\infty}^{+\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx = 1$$

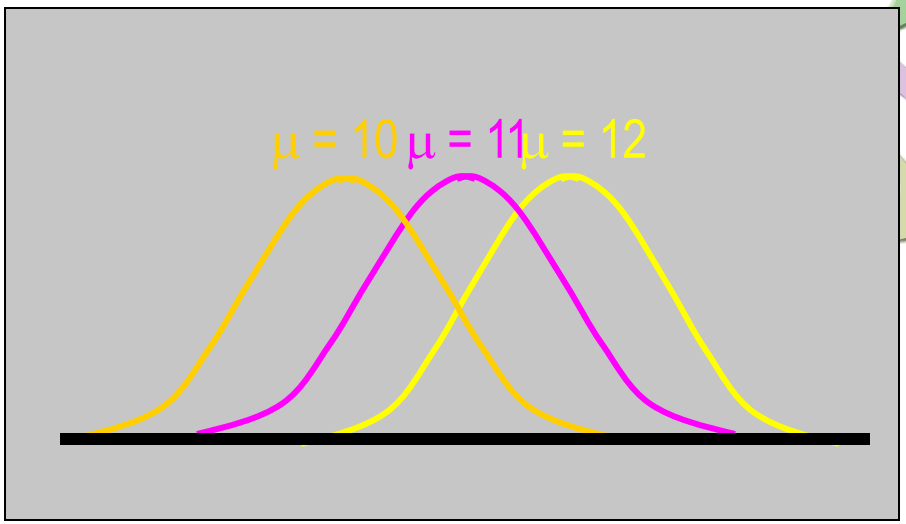


- **Normal distribution is bell shaped, and symmetrical around mean.**

Normal (Gaussian) Distribution

- The expected value (also called the mean) E(X) (or μ) can be any number
- The standard deviation σ can be any nonnegative number
- The total area under every normal curve is 1
- There are infinitely many normal distributions





Normal (Gaussian) Distribution

- **Univariate N(μ, σ²):** $p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$

- **Multivariate N(μ, Σ):**

$$p(x) = \frac{1}{\sqrt{2\pi\,|\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

with the mean μ and the covariance matrix

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1l} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2l} \\ \vdots & \vdots & \vdots & \vdots \\ \sigma_{l1} & \sigma_{l2} & \cdots & \sigma_l^2 \end{bmatrix}$$

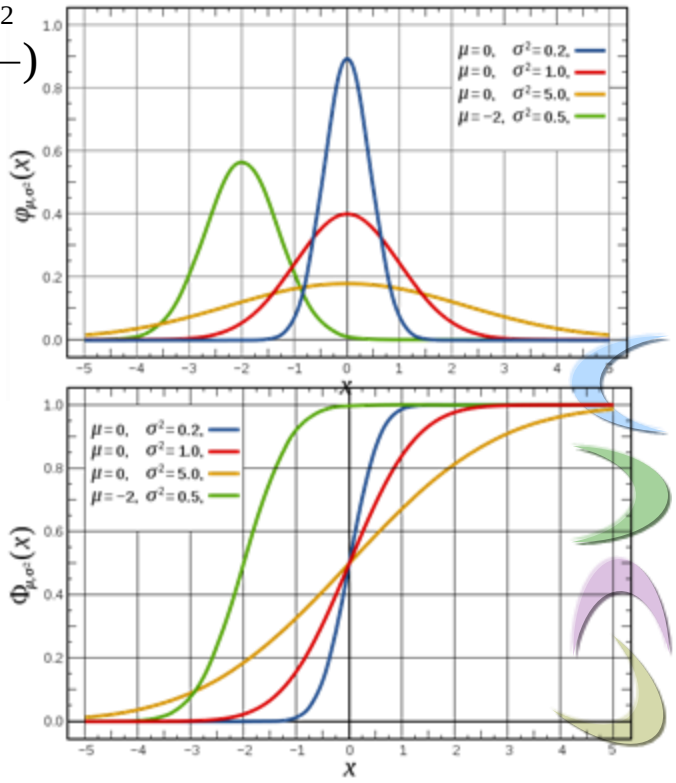
where $\sigma_i^2 = E[(x_i - \mu_i)^2]$ and

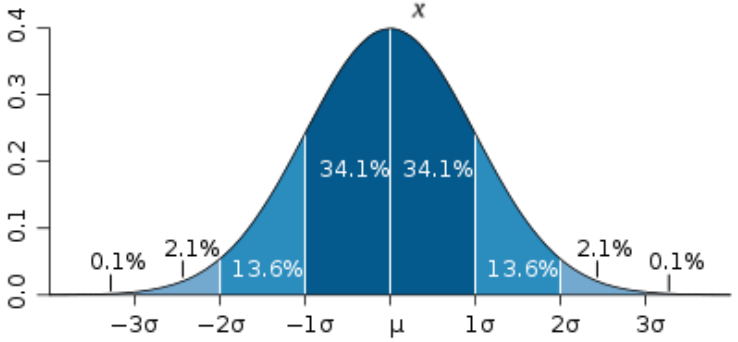
$\sigma_{ij} = \sigma_{ji} = E[(x_i - \mu_i)(x_j - \mu_j)]$

- **Central limit theorem:**

Let $z = \sum_{i=1}^{n} x_i$, then $\frac{z-\mu}{\sigma} \sim N(0,1)$ when $n \to \infty$

irrespective of the pdf's of x_i's.





An example: Two variate case

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$$

The pdf of the multivariate will be:

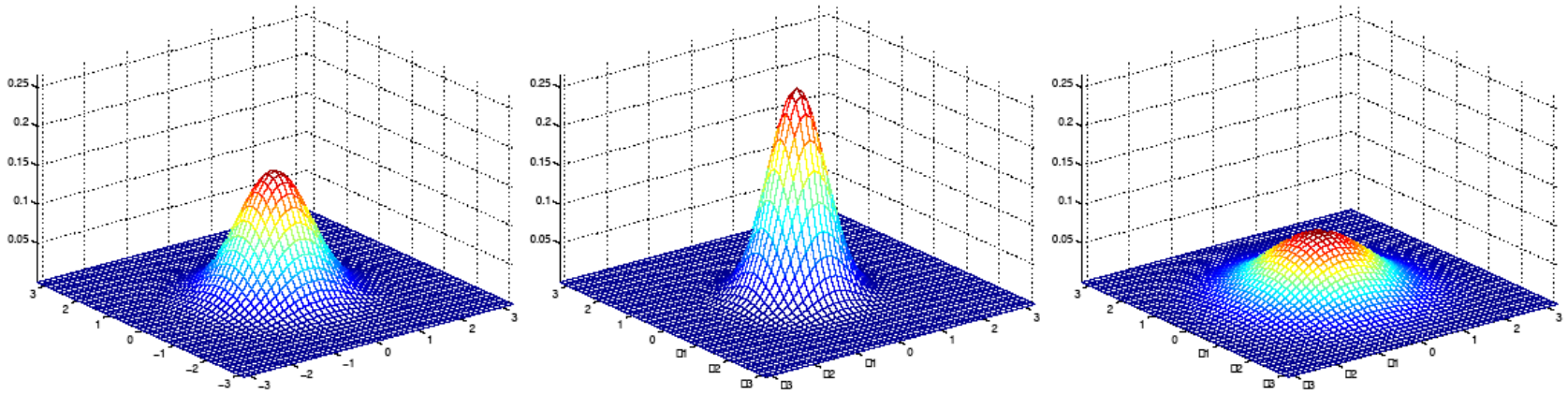
Covariance matrix

$$p(x;\mu,\Sigma) = \frac{1}{2\pi \begin{vmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{vmatrix}^{1/2}} \exp\left(-\frac{1}{2}\begin{bmatrix} x_1-\mu_1 \\ x_2-\mu_2 \end{bmatrix}^T \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}^{-1} \begin{bmatrix} x_1-\mu_1 \\ x_2-\mu_2 \end{bmatrix}\right)$$

Determinant

$$= \frac{1}{2\pi(\sigma_1^2\cdot\sigma_2^2 - 0\cdot 0)^{1/2}} \exp\left(-\frac{1}{2}\begin{bmatrix} x_1-\mu_1 \\ x_2-\mu_2 \end{bmatrix}^T \begin{bmatrix} \frac{1}{\sigma_1^2} & 0 \\ 0 & \frac{1}{\sigma_2^2} \end{bmatrix} \begin{bmatrix} x_1-\mu_1 \\ x_2-\mu_2 \end{bmatrix}\right)$$

Gaussian Intuitions: Size of Σ



$\mu = [0\ 0]$ $\quad$ $\mu = [0\ 0]$ $\quad$ $\mu = [0\ 0]$

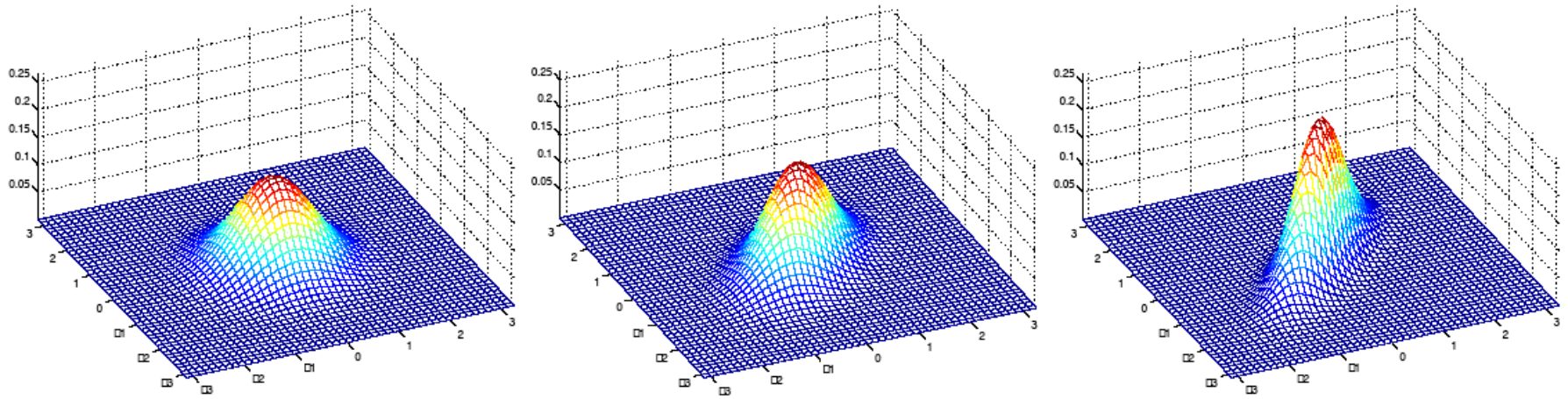
$\Sigma = I$ $\quad$ $\Sigma = 0.6I$ $\quad$ $\Sigma = 2I$

Identity matrix

As Σ becomes larger,

Gaussian becomes more spread out

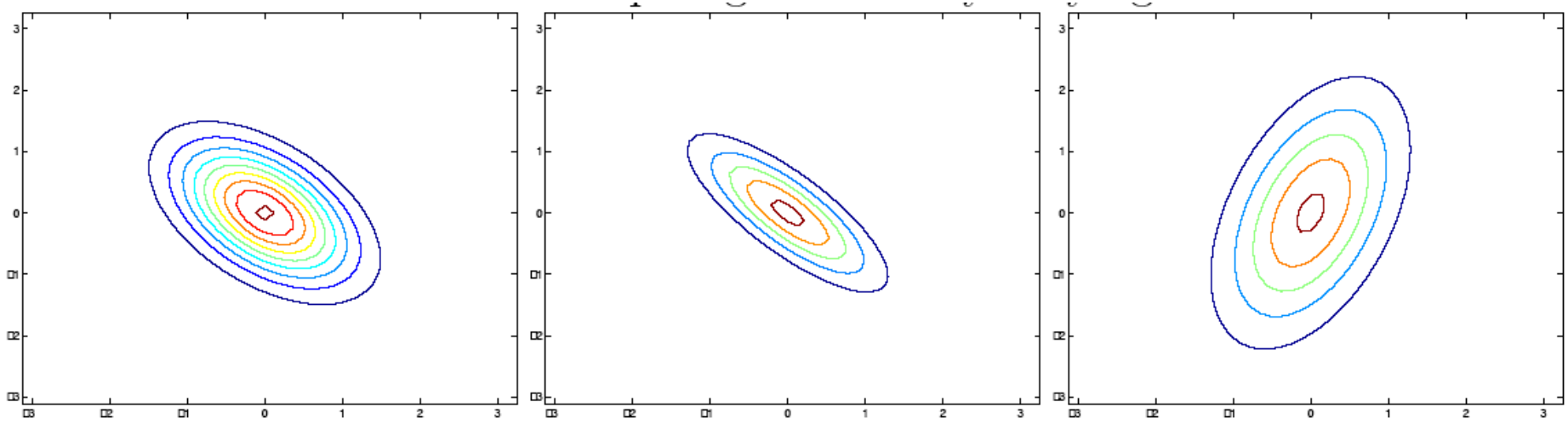
Gaussian Intuitions: Off-diagonal



$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}; \quad .\Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

As the off-diagonal entries increase, more correlation between value of x and value of y

Gaussian Intuitions: off-diagonal and diagonal



$$\Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}; \quad .\Sigma = \begin{bmatrix} 3 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

- **Decreasing non-diagonal entries (#1-2)**
- **Increasing variance of one dimension in diagonal (#3)**

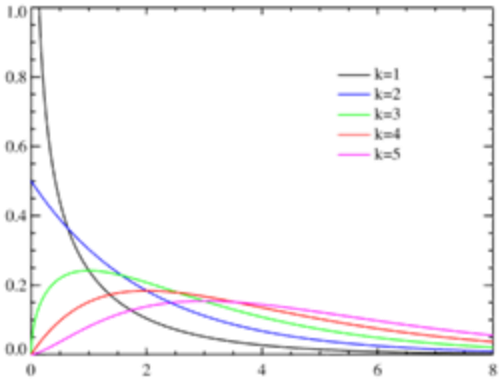
Other Continuous Distributions

- **Chi-square (X^2)distribution of k degrees of freedom:**

  distribution of a sum of squares of *k* independent standard normal random variables, that is, $\chi^2 = x_1^2 + x_2^2 + \cdots + x_k^2$ where $x_i \approx N(0,1)$

$$p(y) = \frac{1}{2^{k/2}\Gamma(k/2)} y^{k/2-1} e^{-y/2} \text{step}(y),$$

$$\text{where } \Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt$$



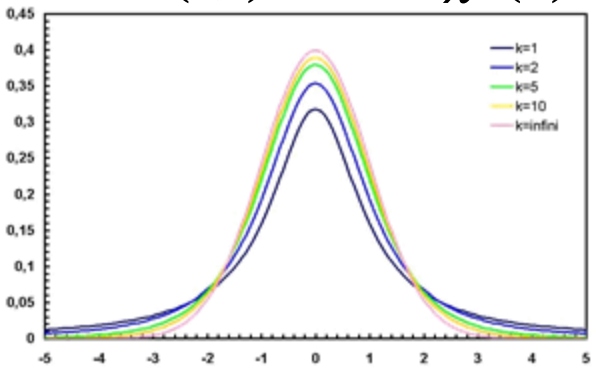
- **Mean: k, Variance: 2k**
- Assume $x \sim \chi^2(k)$
  - Then $(x-k)/\sqrt{2k} \sim N(0,1)$ as $k \to \infty$ by central limit theorem.
  - Also $\sqrt{2x}$ is approximately normally distributed with mean $\sqrt{2k-1}$ and unit variance.

Other Continuous Distributions

- **t-distribution:** estimating mean of a normal distribution when sample size is small.

  A t-distributed variable $q = x/\sqrt{z/k}$ where $x \approx N(0,1)$ and $z \approx \chi^2(k)$

$$p(q) = \frac{\Gamma((k+1)/2)}{\sqrt{\pi k}\,\Gamma(k/2)}\left(1+\frac{q^2}{k}\right)^{-(k+1)/2}$$



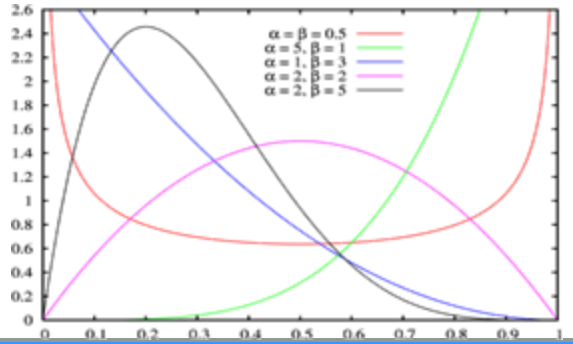
  Mean: 0 for k > 1,

  variance: k/(k-2) for k > 2

- **β-distribution:** Beta(α,β): the posterior distribution of p of a binomial distribution after α−1 events with p and β−1 with 1-p.

$$p(x) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}x^{\alpha-1}(1-x)^{\beta-1}$$

$$= \frac{1}{\mathrm{B}(\alpha,\beta)}x^{\alpha-1}(1-x)^{\beta-1}$$



Discrete Distributions

- **Binomial distribution B(n,p):**

  Repeatedly grab n balls, each with a probability p of getting a black ball. The probability of getting k black balls:

$$P(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

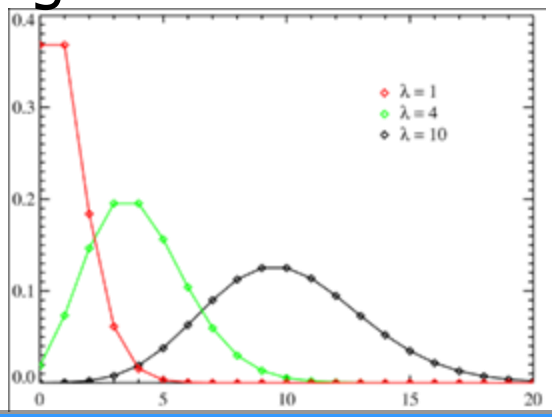
- **Poisson distribution**

  probability of # of events occurring in a fixed period of time if these events occur with a known average.

$$P(k;\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$

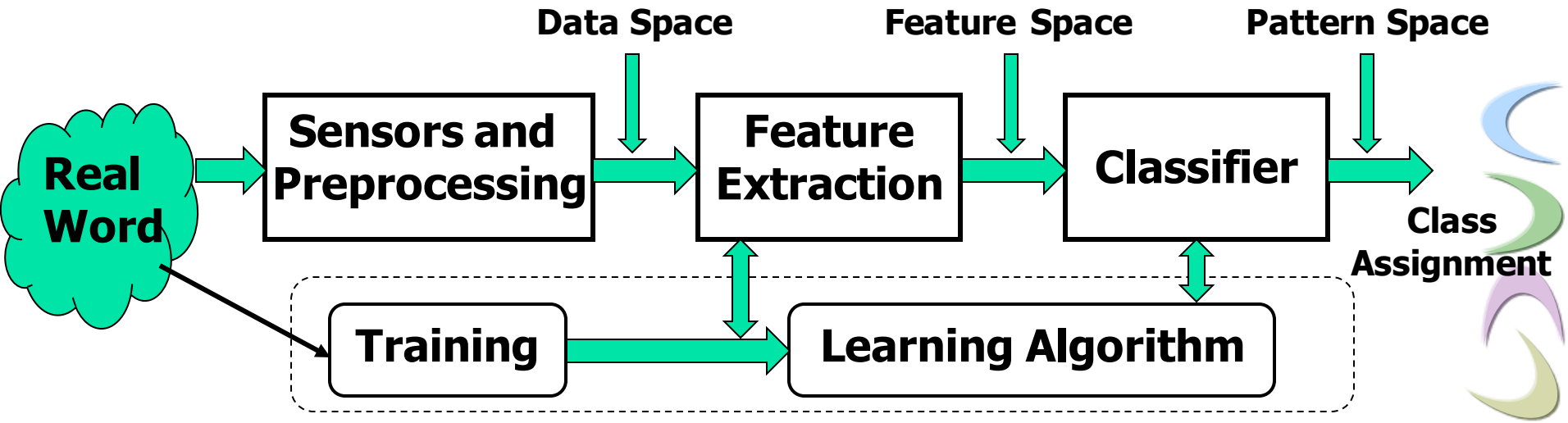
When $n \to \infty$ and np remains constant,

$$B(n, p) \to Poisson(np)$$



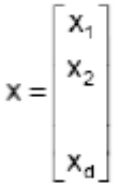
Feature Extraction and Dimensionality Reduction

PATTERN RECOGNITION SYSTEM

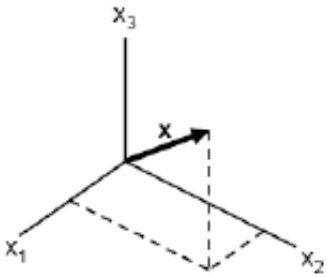


What is feature?

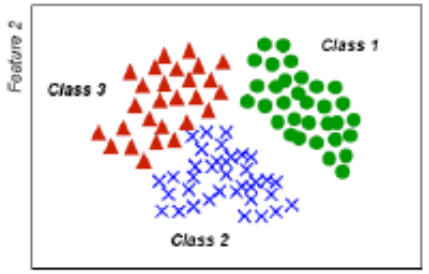
- **Feature**
  - **Feature is any distinctive aspect, quality or characteristic of an object (population)**
    - Features may be symbolic (e.g. color) or numeric (e.g. height).
- **Definitions**
  - **Feature vector**
    - The combination of d features is presented as a d-dimensional column vector called a feature vector.
  - **Feature space**
    - The d-dimensional space defined by the feature vector is called the feature space.
  - **Scatter plot**
    - Objects are represented as points in feature space. This representation is called a scatter plot.



Feature vector



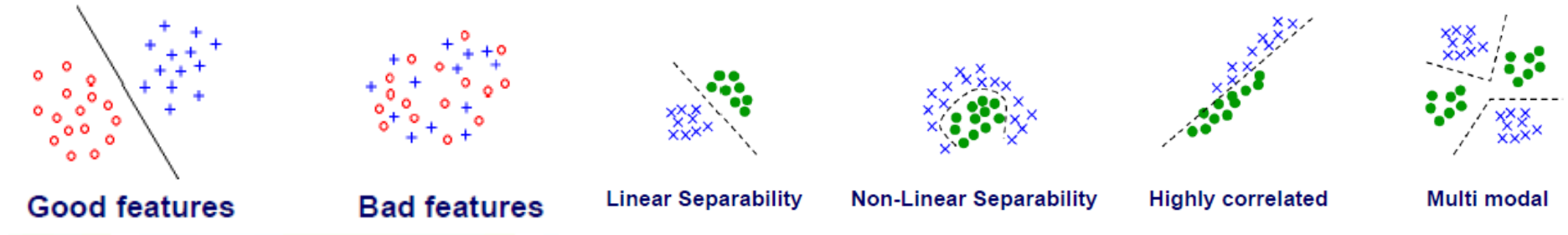
Feature Space (3D)



Scatter plot (2D)

Feature and Patterns

- **Pattern is a composite of traits or features corresponding to characteristics of an object or population**
  - In classification; a pattern is a pair of feature vector and label
- **What makes a good feature vector**
  - **The quality of a feature vector is related to its ability to discriminate samples from different classes**
    - Samples from the same class should have similar feature values
    - Samples from different classes should have different feature values



Feature Extraction

What is Feature Extraction?

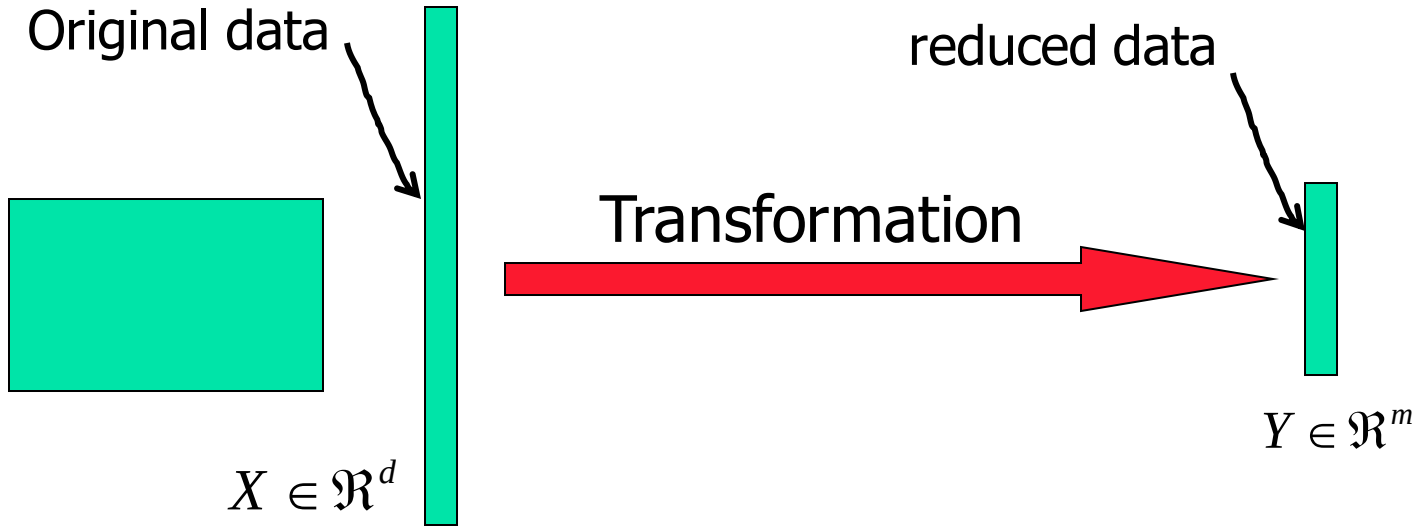
- **Feature extraction refers to the mapping of the original high-dimensional input data onto a lower-dimensional space.**
  - **Criterion for feature extraction can be different based on different problem settings.**
    - **Supervised or Unsupervised transformation to maximize the class discrimination or minimize the information loss.**
    - **Linear or Non-linear transformation to reduce the computational cost or increase class discrimination**
    - **Local or Global transformation to employ the limited number of input data (local view) or all of them (global view)**
    - **Frequency or Spatial (Time) domain transformation.**
    - **…**

Feature Extraction

- **Given a set of data points of *d* features** $\{x_1, x_2, \cdots, x_d\}$

Compute the desired transformation (projection)

$$f \in \Re^{m \times d} : x \in \Re^d \rightarrow y = f(x) \in \Re^m \ (m << d)$$



Some Feature Extraction Methods

- Statistical based features
  - Mean, variance, energy, entropy and etc. of whole input data or a limited widow (block) of input data.
  - Histogram of input data or transformed data (as *pdf* of the data).
- Local features
  - LBP(Local Binary Patterns)
  - LTP(Local Ternary Patterns)
  - LPQ(Local Phase Quantization)
  - HOG (Histogram of Oreinted Gradient)
  - MFCC (Mel Frequency Cepstral Coefficient)
  - ...
- Linear feature extraction methods
  - PCA (Principal Component Analysis)
  - LDA (Linear Discriminant Analysis)
- Learning based features
  - HDLF(High Discriminative Local Feature)
  - ANN (Artificial Neural Network)
  - Deep Learning

Local Binary Pattern (LBP)

For each PIXEL of an image, a BINARY CODE is produced
→ to make a new matrix with the new value (binary to decimal value).

$$LBP_{p,r}(N_c) = \sum_{p=0}^{P-1} g\left(N_p - N_c\right)2^p$$

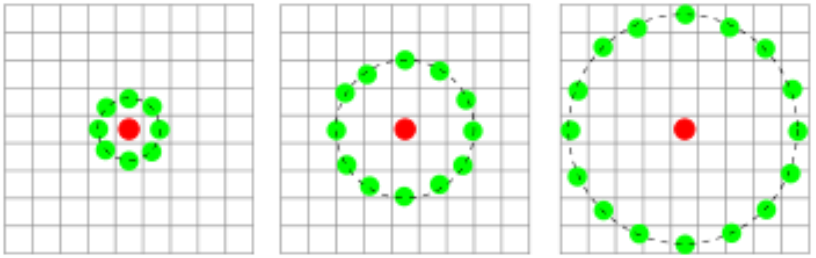
where, neighborhood pixels (N_p) in each block is thresholded by its center pixel value (N_c)

p→ sampling points (e.g., p = 0, 1, …, 7 for a 3x3 cell, where P = 8)

r → radius (for 3x3 cell, it is 1).

Binary threshold function $g(x)$ is,

$$g(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$$



Computation of LBP

Binary code for > N_c

0	1	0
1	N_c	0
0	0	1

	3	7	2
	8	**4**	1
	2	3	5

Component-wise multiplication

1	2	4
128		8
64	32	16

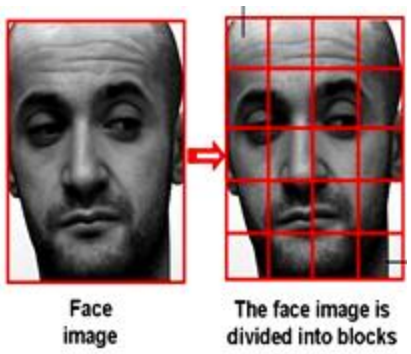
0	2	0
128	**?**	0
0	0	16

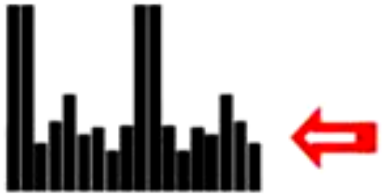
Representation

$\sum$ **Sum**

LBP

146



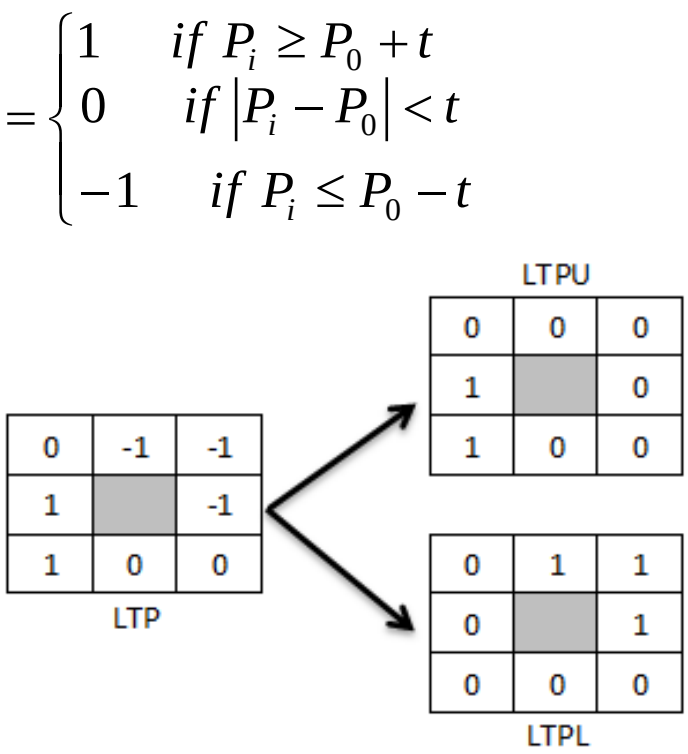


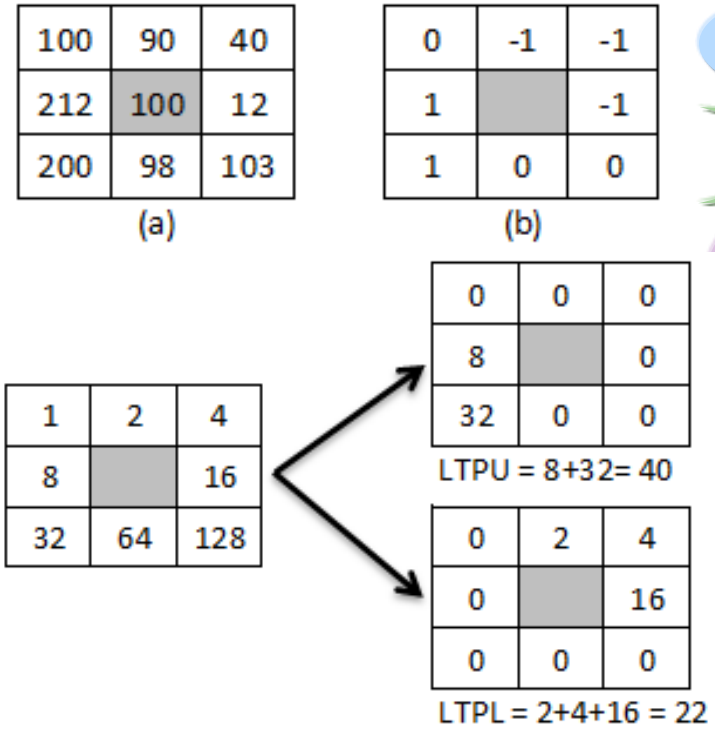
- **Optionally normalize the histogram.**
- **Concatenate normalized histograms of all cells. This gives the feature vector for the input image.**

Computation of LTP

- **LTP is similar to LBP, but :**
  - LBP is sensitive to noise in near uniform regions.
  - LTP solves this later problem.
  - LTP extends LBP to 3-valued (1, 0, -1) codes, in which local pixels are compared to a user defined thresholds –t and +t.

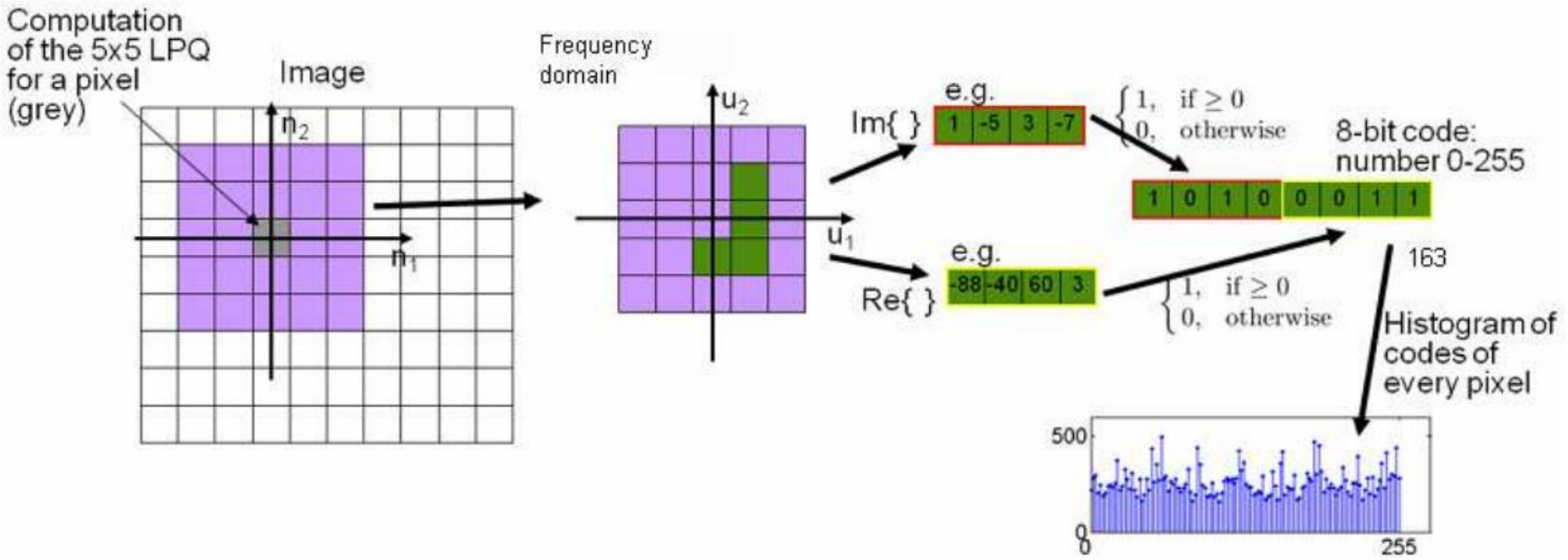
$$P_i' = \begin{cases} 1 & if\ P_i \geq P_0 + t \\ 0 & if\ |P_i - P_0| < t \\ -1 & if\ P_i \leq P_0 - t \end{cases}$$





Computation of LPQ

- **LPQ computation steps are:**
  - Compute Fourier Transform
  - For each pixel Real and Imaginary parts of four neighbors are binarized and perform LPQ code as below:



Computation of HOG

1. Partition the input image to cells
2. Compute gradients in the region to be described
2. Put them in 9 bins according to orientation
3. Group the obtained histograms of cells (by concatenation) into large blocks (typically a cell is of 8x8 pixels and a block is of 2x2 cells)
4. Normalize the histogram of each block
5. Concatenate the histogram of all blocks and construct HOG feature vector.

Computation of HOG

$G_x = I \otimes [-1\ 0\ 1]$

$G_y = I \otimes [-1\ 0\ 1]^T$

Apply the 1-D centered, point discrete derivative mask

Group the cells into connected blocks to normalize the gradient strength

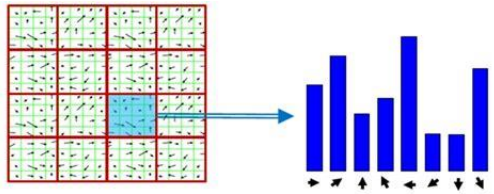
Gradient Computation → Orientation Binning → Descriptor Blocks

Magnitude

$$G = \sqrt{G_x^2 + G_y^2}$$

Angle

$$\alpha = ac\tan\left(\frac{G_y}{G_x}\right)$$

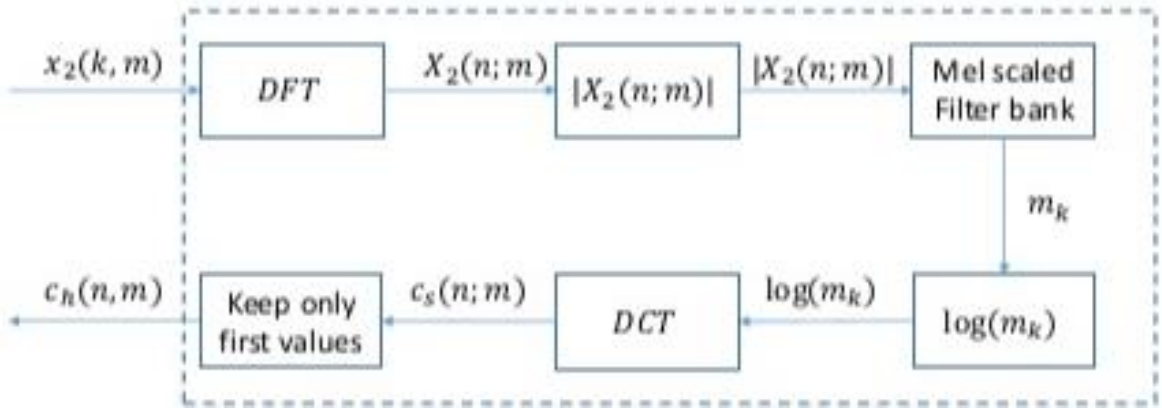


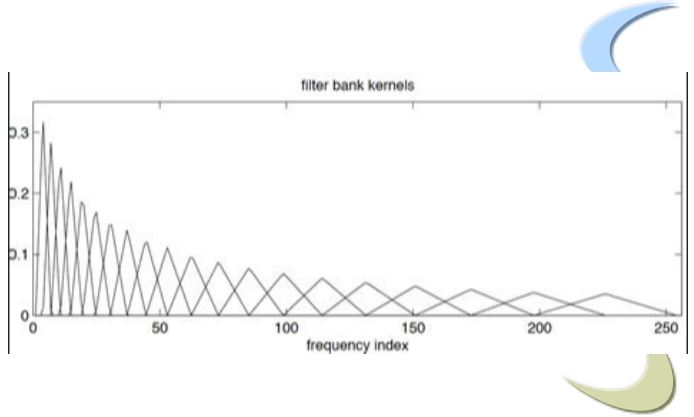
From each pixel within the cell is created an orientation-based histogram

The HOG Descriptor is the vector generated from the normalized cell histograms from all block regions

Computation of MFCC

- **This feature is specialized for speech data and extract powerful features based on below scenario:**





- **Energy of Ch, ΔCh and ΔΔCh are usually used as feature.**

Global Features

- In most of global feature descriptors at first some kernels or filters are applied on raw data (the whole data or blocks of data) to transform them into different spaces like DFT, DCT, DWT, Gabor, … .
- Then some statistical functions such as energy and entropy and also some moment based operators are applied to the transformed data and construct a vector as global feature.
  - Global Gabor Zernike (GGZ) descriptor is one example that combined the Gabor transform and Zernike moments to obtain the global feature of input image and its blocks.

THE KARHUNEN–LOÈVE TRANSFORM
OR
PRINCIPAL COMPONENT ANALYSIS

Principal Component Analysis (PCA)

- The Karhunen–Loève transform (KLT) or principal component analysis (PCA), is known as one of the most popular methods for feature generation and dimensionality reduction in pattern recognition.
- Principal component analysis (PCA)
  - Reduce the dimensionality of a data set by finding a new set of variables, smaller than the original set of variables.
  - Retains most of the input data information.
  - Useful for the compression and noise removal in the data.
- By information we mean the variation present in the sample, given by the correlations between the original variables.
  - The new variables, called principal components (PCs), are uncorrelated, and are ordered by the fraction of the total information each retains.

Algebraic definition of PCs

Given a sample of '*n*' observations on a vector of '*d*' variables

$$\{x_1, x_2, \cdots, x_n\} \in \Re^d$$

Define the first principal component of the sample by the linear transformation

$$z_1 = a_1^T x_j = \sum_{i=1}^{d} a_{i1} x_{ij}, \quad j = 1,2,\cdots,n.$$

Here a_1 is chosen such that var[z_1] is maximum.

where the vector $a_1 = (a_{11}, a_{21}, \cdots, a_{d1})$

$$x_j = (x_{1j}, x_{2j}, \cdots, x_{dj})$$

Algebraic definition of PCs

To find a_1 first note that

$$\text{var}[z_1] = E((z_1 - \overline{z_1})^2) = \frac{1}{n}\sum_{i=1}^{n}\left(a_1^T x_i - a_1^T \overline{x}\right)^2$$

$$= \frac{1}{n}\sum_{i=1}^{n} a_1^T \left(x_i - \overline{x}\right)\left(x_i - \overline{x}\right)^T a_1 = a_1^T S a_1$$

where

$$S = \frac{1}{n}\sum_{i=1}^{n}\left(x_i - \overline{x}\right)\left(x_i - \overline{x}\right)^T \text{ is the covariance matrix.}$$

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \text{ is the mean.}$$

Algebraic definition of PCs

To find a_1 that maximizes $\text{var}[z_1]$ subject to $a_1^T a_1 = 1$

Let λ be a Lagrange multiplier

$$L = a_1^T S a_1 - \lambda(a_1^T a_1 - 1)$$

$$\frac{\partial}{\partial a_1} L = S a_1 - \lambda a_1 = 0$$

$$\Rightarrow (S - \lambda I) a_1 = 0$$

therefore a_1 is an eigenvector of S

corresponding to the largest eigenvalue $\lambda = \lambda_1$.

Algebraic definition of PCs

We find that a_2 is also an eigenvector of S

whose eigenvalue $\lambda = \lambda_2$ is the second largest.

In general

$$\text{var}[z_k] = a_k^T S a_k = \lambda_k$$

- The k^{th} largest eigenvalue of S is the variance of the k^{th} PC.
- The k^{th} PC z_k retains the k^{th} greatest fraction of the variation in the sample.

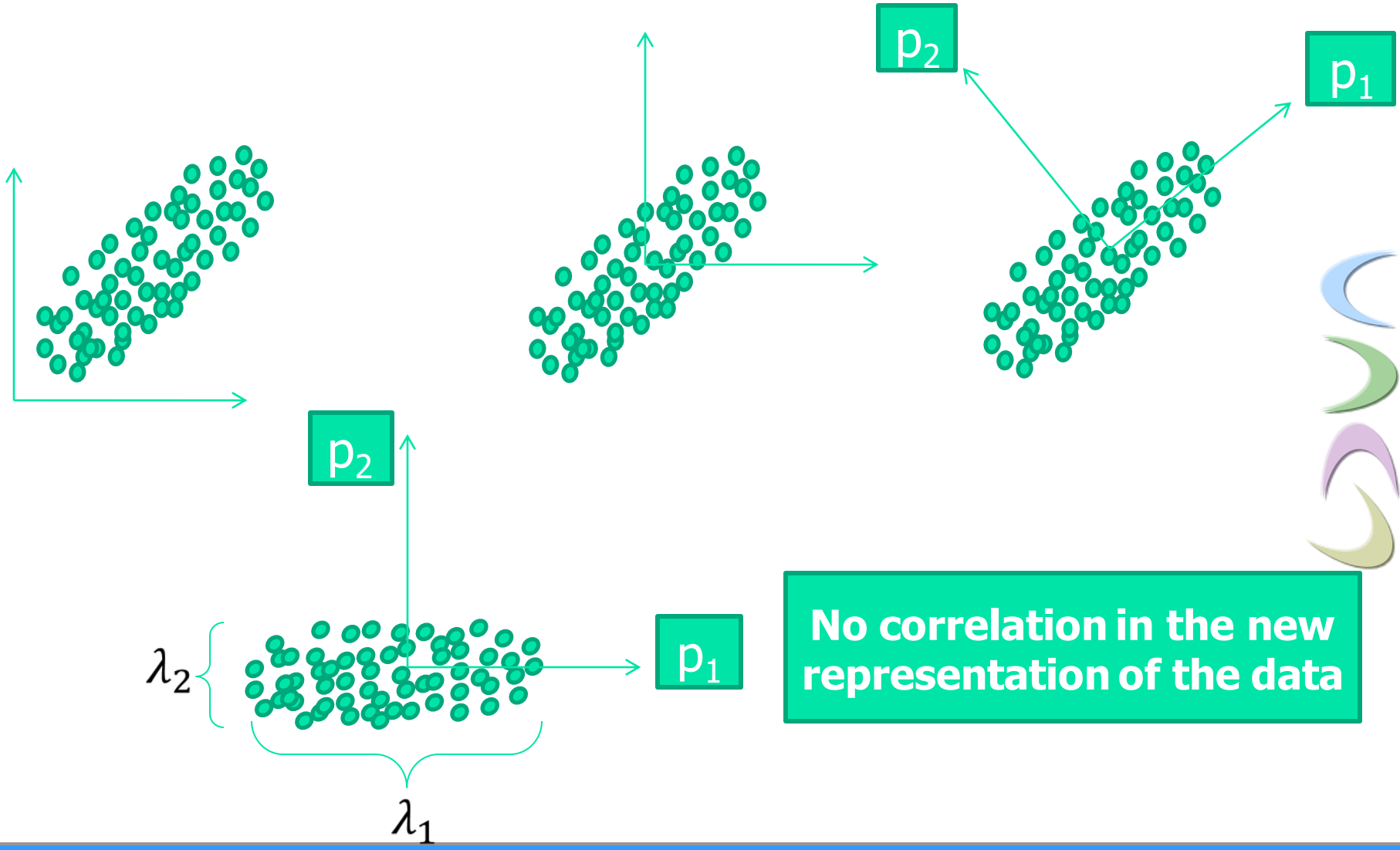
Algebraic definition of PCs

- **Main steps for computing PCs for n training samples**

  - **Form the covariance matrix S.**
  - **Compute its eigenvectors:** $\{a_i\}_{i=1}^{d}$
  - **The first p eigenvectors** $\{a_i\}_{i=1}^{p}$ **corresponds to 'p' largest eigenvalues form the '*p*' PCs.**
  - **The transformation G consists of the p PCs:** $G \leftarrow [a_1, a_2, \cdots, a_p]$
  - **Applying G on each new raw data (X) to obtained PCA based features P**

$$P = X * G$$

Principal Component Analysis

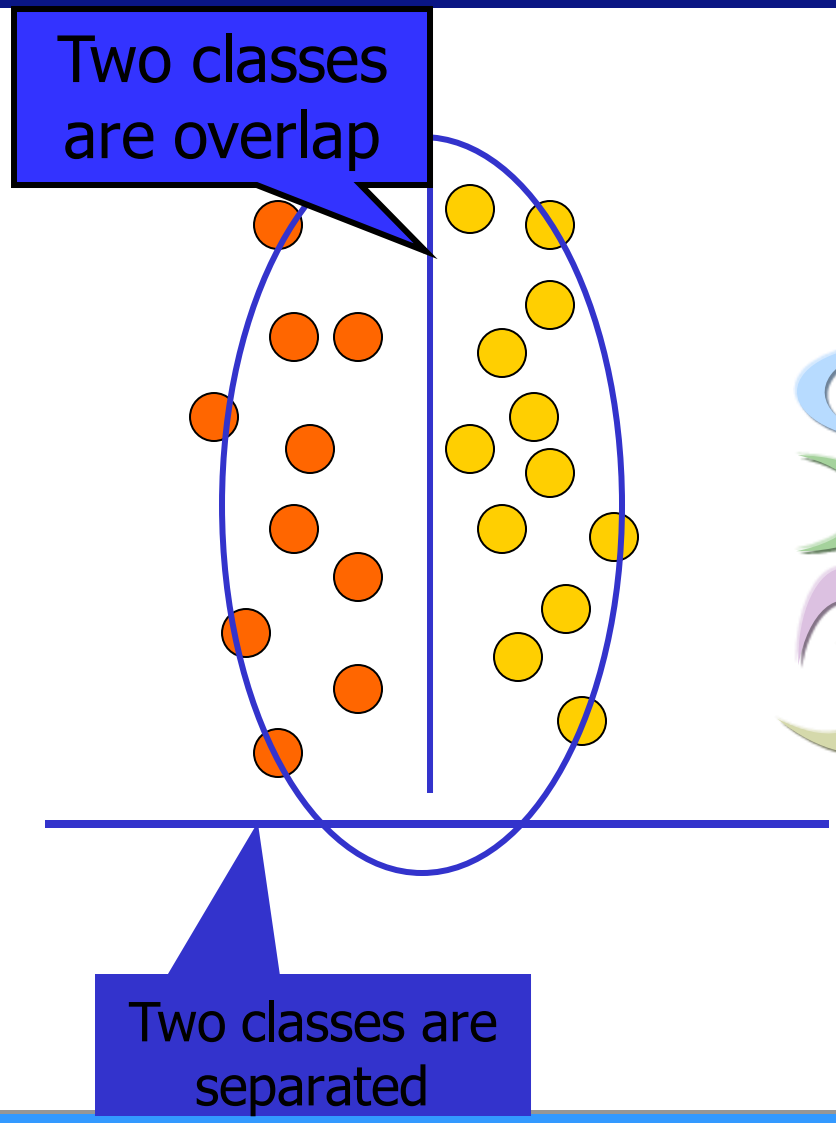


No correlation in the new representation of the data

Linear Discriminant Analysis

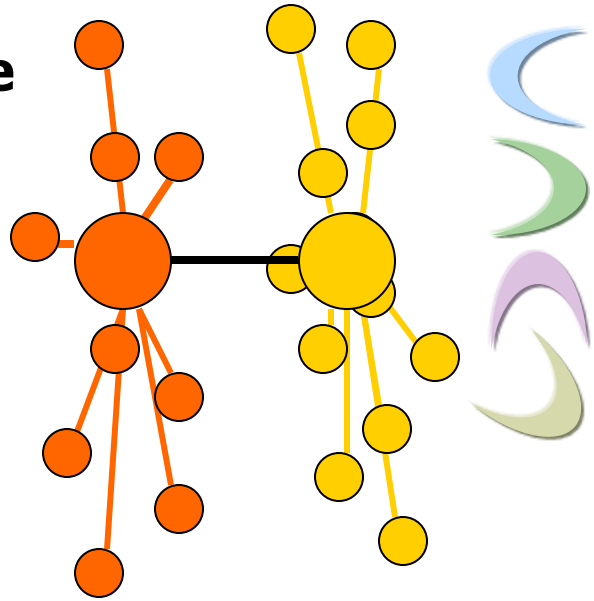
Is PCA a good criterion for classification?

- **Data variation determines the projection direction**
- **What's missing?**
  - **Class information**
- **Similarly, what is a good criterion?**
  - **Separating different classes**



What class information may be useful?

- **Between-class distance**
  - Distance between the centroids of different classes
- **Within-class distance**
  - Accumulated distance of an instance to the centroid of its class
- **Linear discriminant analysis (LDA)**
  - finds most discriminant projection by
    - maximizing between-class distance
    - and minimizing within-class distance



Within-class distance

Between-class distance

Linear Discriminant Analysis (LDA)

Training data from different classes 1, 2, …, k

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_k \end{bmatrix}_{n \times d}$$

Data matrix

Linear Discriminant Analysis (LDA)

- **Between-class scatter**

$$S_b = H_b^T H_b$$

C_i is the centroid of i^{th} Class
C is the centroid of all classes

$$H_b = \begin{bmatrix} c_1 - c \\ c_2 - c \\ \vdots \\ c_k - c \end{bmatrix}_{n \times d}$$

- **Within-class scatter**

$$S_w = H_w^T H_w$$

$$H_w = \begin{bmatrix} A_1 - c_1 \\ A_2 - c_2 \\ \vdots \\ A_k - c_k \end{bmatrix}_{m \times n}$$

- **Properties:**
  - **Between-class distance = trace of between-class scatter (I.e., the summation of diagonal elements of the scatter)**
  - **Within-class distance = trace of within-class scatter**

Linear Discriminant Analysis (LDA)

- **Discriminant criterion in mathematical formulation**

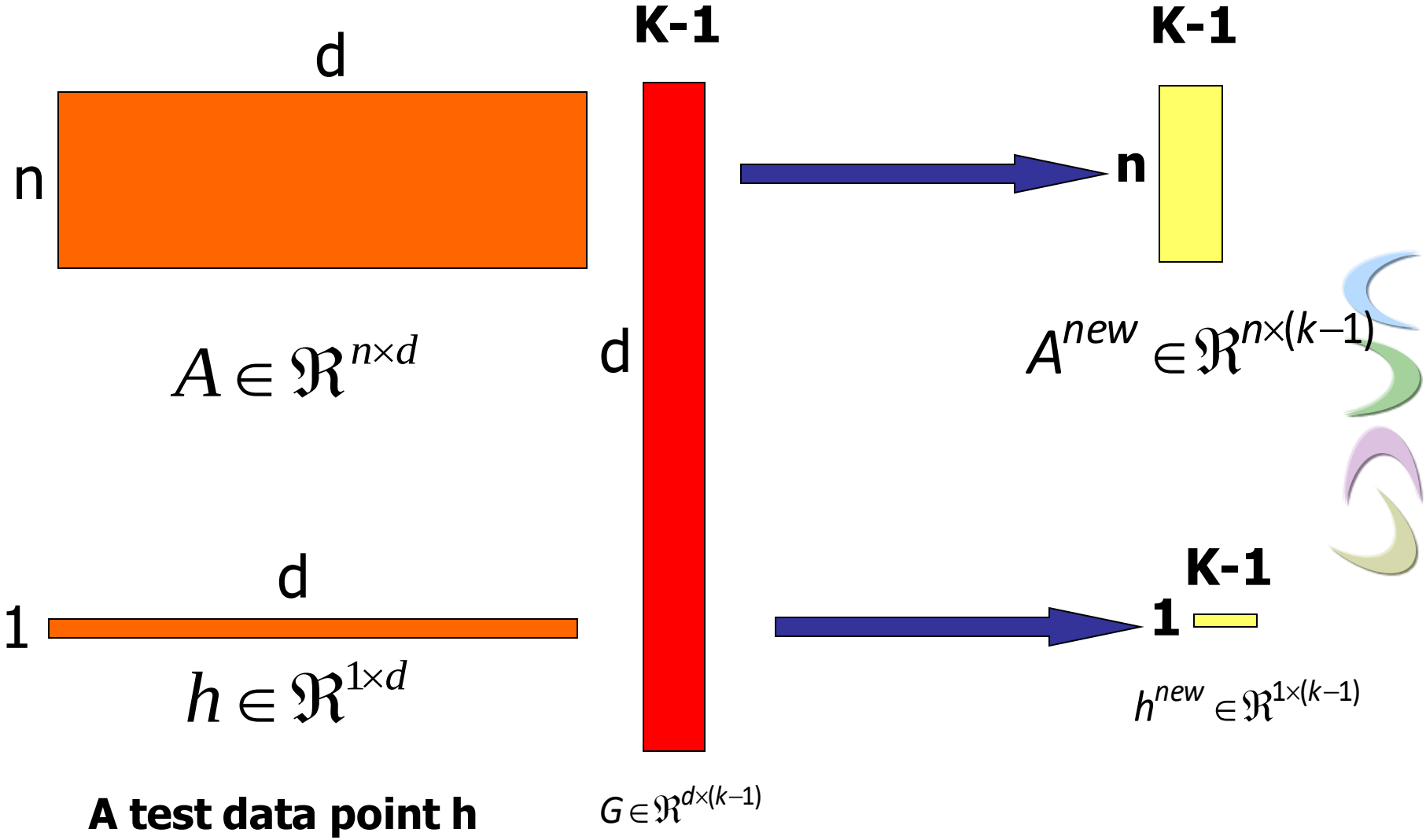
$$\arg\max_{G} \frac{\text{trace}(G^T S_b G)}{\text{trace}(G^T S_w G)}$$

  - **Between-class scatter matrix** S_b
  - **Within-class scatter matrix** S_w

G_{dxd} is the projection matrix

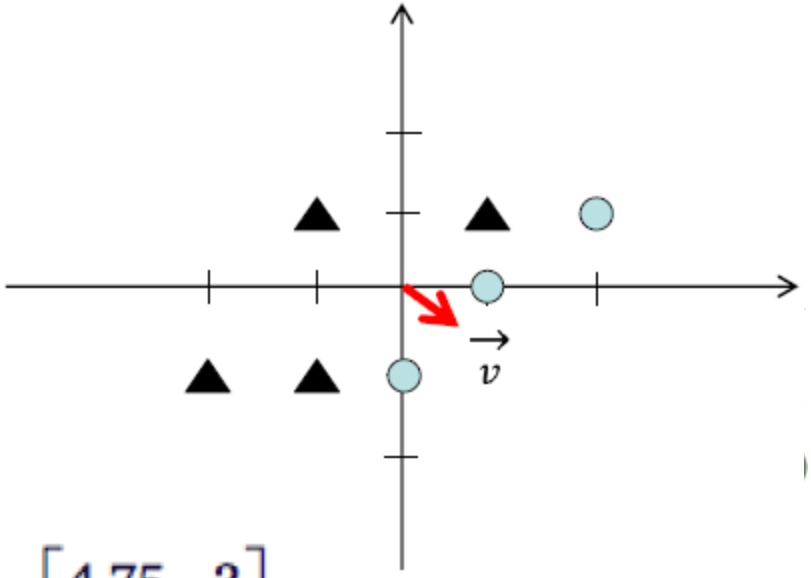
- **The optimal transformation (G) is given by solving a generalized eigenvalue problem of resulting** $S_w^{-1} S_b$ **matrix.**
  - **Eigenvectors of it will be used as optimal transform matrix**

Graphical view of projection



An Example of LDA

- Example: Consider 2 classes below:
  - $C1=\{(0,-1),(1,0),(2,1)\}$
  - $C2=\{(1,1),(-1,1),(-1,-1),(-2,-1)\}$



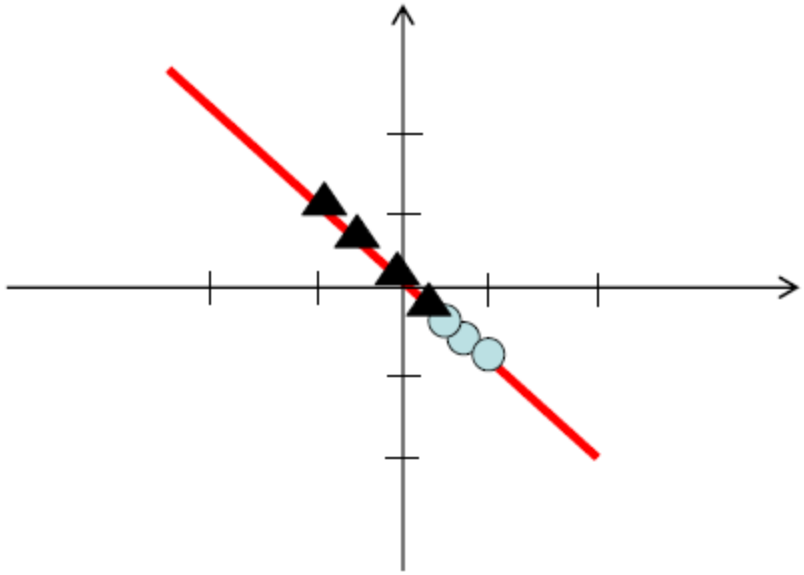
$$\mu_1=(1,0)^T,\mu_2=(-.75,0)^T \qquad S_1=\begin{bmatrix}2 & 2\\ 2 & 2\end{bmatrix},S_2=\begin{bmatrix}4.75 & 3\\ 3 & 4\end{bmatrix}$$

$$S_w=S_1+S_2=\begin{bmatrix}6.75 & 5\\ 5 & 6\end{bmatrix} \qquad \Rightarrow S_w^{-1}=\frac{1}{15.5}\begin{bmatrix}6 & -5\\ -5 & 6.75\end{bmatrix}\approx\begin{bmatrix}.4 & -.33\\ -.33 & .43\end{bmatrix}$$

$$\Rightarrow v=S_w^{-1}(\mu_1-\mu_2)=\underbrace{\begin{bmatrix}.4 & -.33\\ -.33 & .43\end{bmatrix}}_{S_b}\begin{bmatrix}1.75\\ 0\end{bmatrix}=\begin{bmatrix}.7\\ -.6\end{bmatrix} \qquad \Rightarrow \vec{v}=\frac{v}{\|v\|}=\begin{bmatrix}.75\\ -.65\end{bmatrix}$$

An Example of LDA

- Example: Consider 2 classes below:
  - $C1=\{(0,-1),(1,0),(2,1)\}$
  - $C2=\{(1,1),(-1,1),(-1,-1),(-2,-1)\}$



Projected C1 data points = $v^T C_1 = \{.63, .77, .9\}$
Projected C2 data points = $v^T C_2 = \{.13, -1.4, -.13, -.9\}$

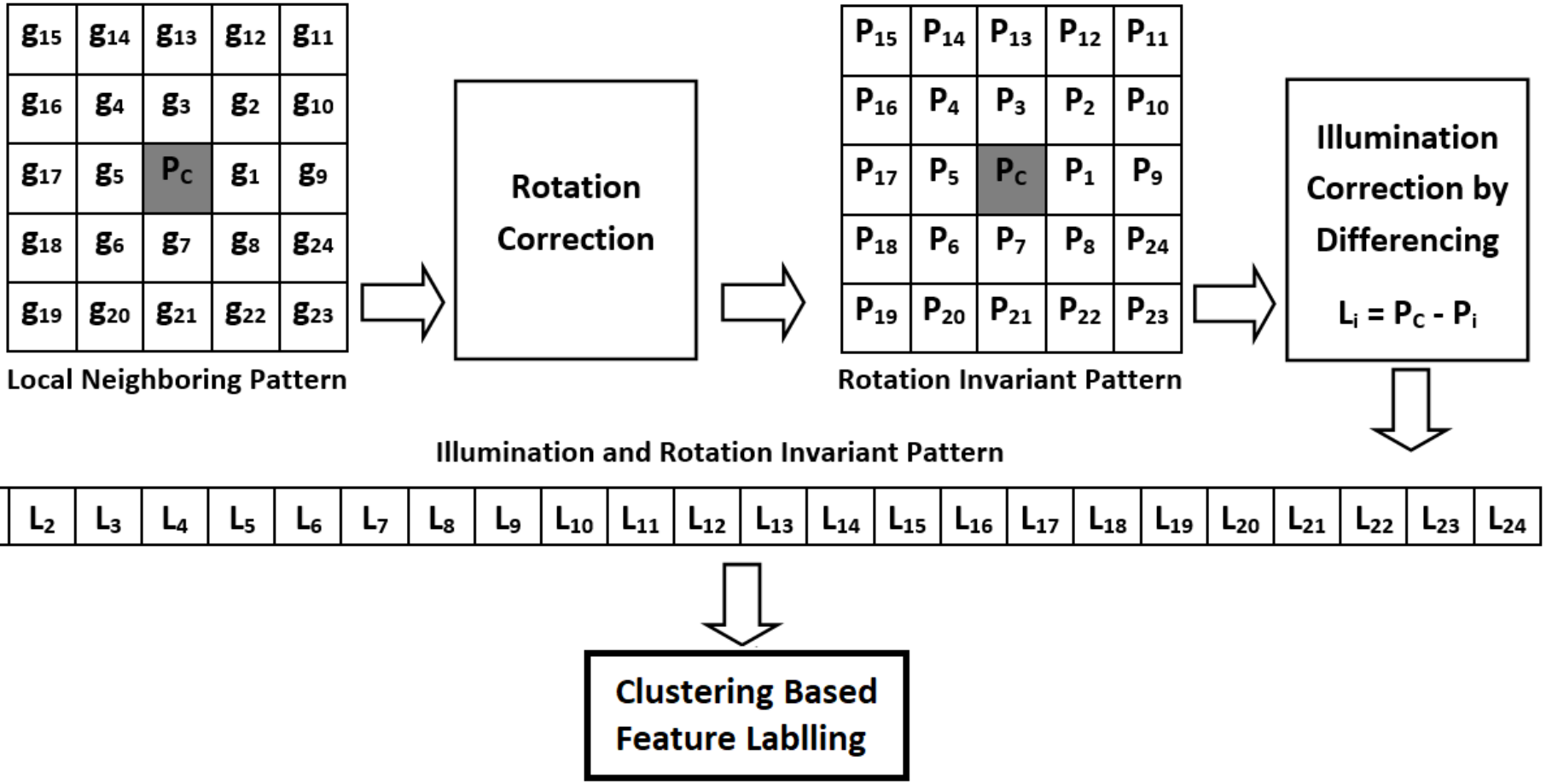
PCA vs. LDA

	PCA	LDA
Sample mean	$\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x_i}$	$\boldsymbol{\mu_i} = \frac{1}{N_i}\sum_{\mathbf{x}\in\omega_i} \mathbf{x}$ Mean for each class
Scatter matrix	$\mathbf{S} = \sum_{i=1}^{n}(x_i - \mu)(x_i - \mu)^T$	$\mathbf{S_w} = \sum_{i=1}^{c}\sum_{x_j \in C_i}(x_j - \mu_i)(x_j - \mu_i)^T$ Within-class scatter
		$\mathbf{S_b} = \sum_{i=1}^{c} N_i(\mu_i - \mu)(\mu_i - \mu)^T$ Between-class scatter
Eigen decomposition	$\mathbf{Sw} = \lambda\mathbf{w}$	$\mathbf{S_W^{-1}S_B w} = \lambda\mathbf{w}$ Eigen decomposition

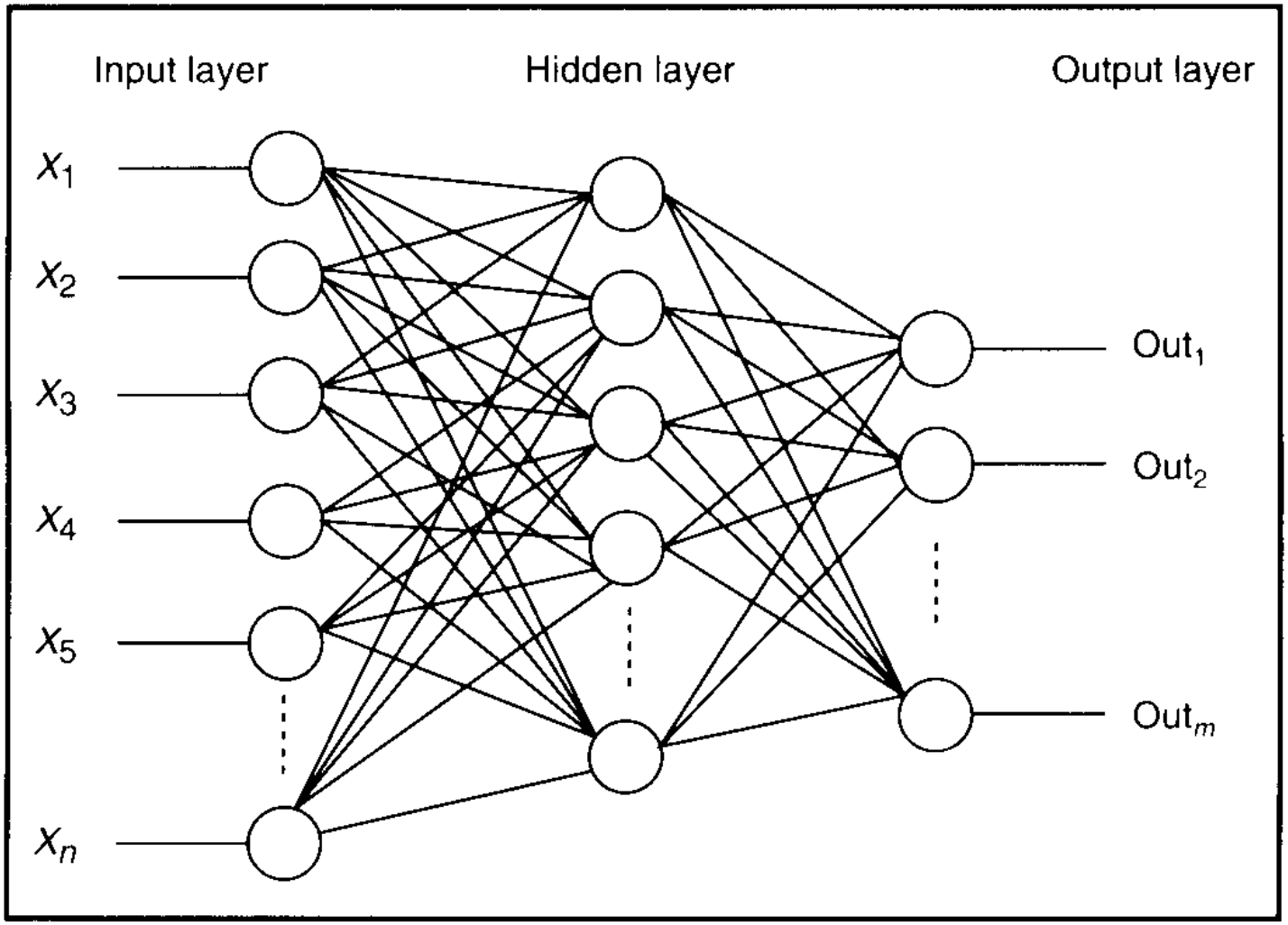
$$\mathbf{Y} = \mathbf{w}^T\mathbf{X}$$

X is transformed to Y using w

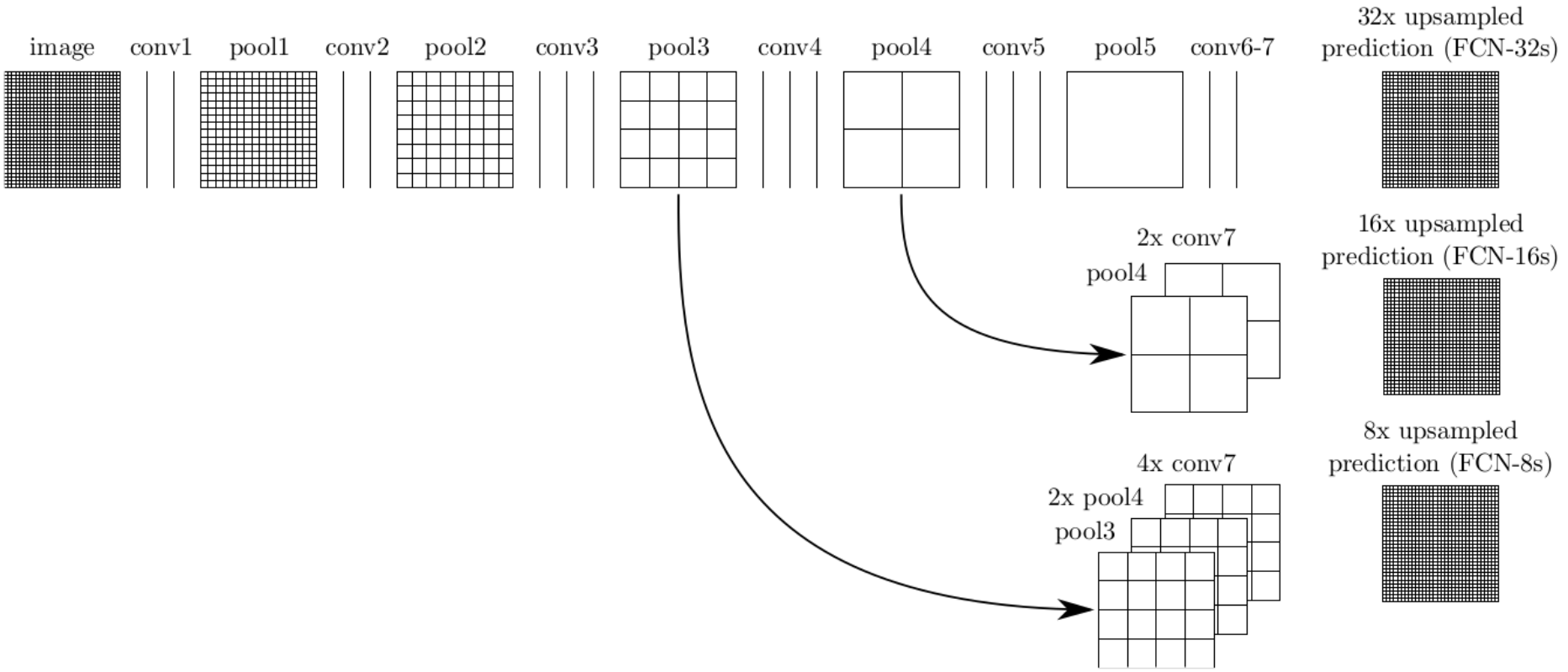
Computation of HDLF



ANN-based Feature Extraction



CNN-based Feature Extraction

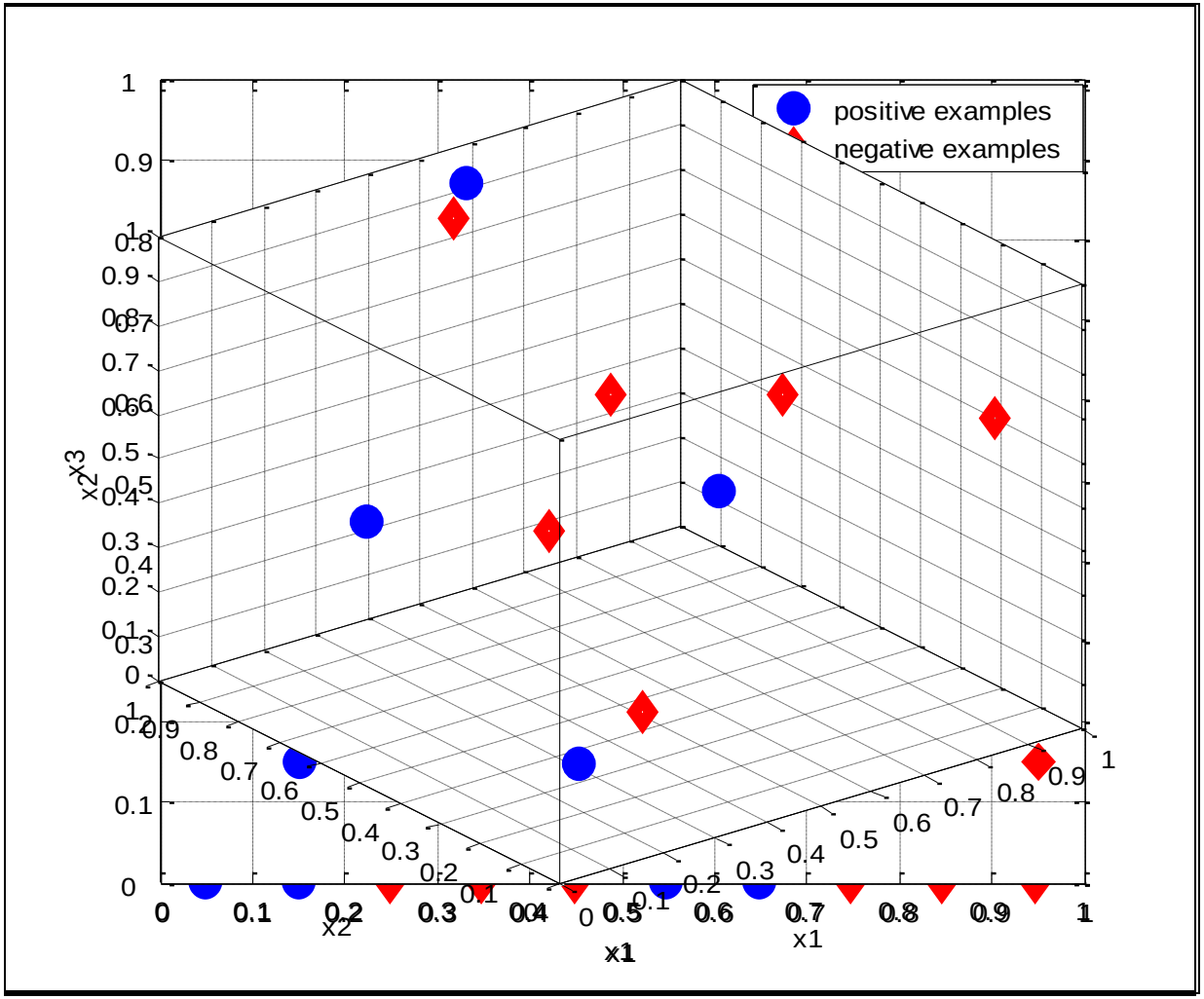


Dimensionality Reduction
(Feature Selection)

Why Dimensionality Reduction?

- **Most machine learning and data mining techniques may not be effective for high-dimensional data**
  - **In all the methods, so far, we saw that the highest the number of points, N, the better the resulting estimate.**
  - **If in the one-dimensional space an interval, filled with N points, is adequate (for good estimation), in the two-dimensional space the corresponding square will require N^2 and in the ℓ-dimensional space the ℓ-dimensional cube will require N^ℓ points.**
  - **The exponential increase in the number of necessary points in known as the curse of dimensionality. This is a major problem one is confronted with in high dimensional spaces.**
  - **The intrinsic dimension may be small.**

Curse of dimensionality

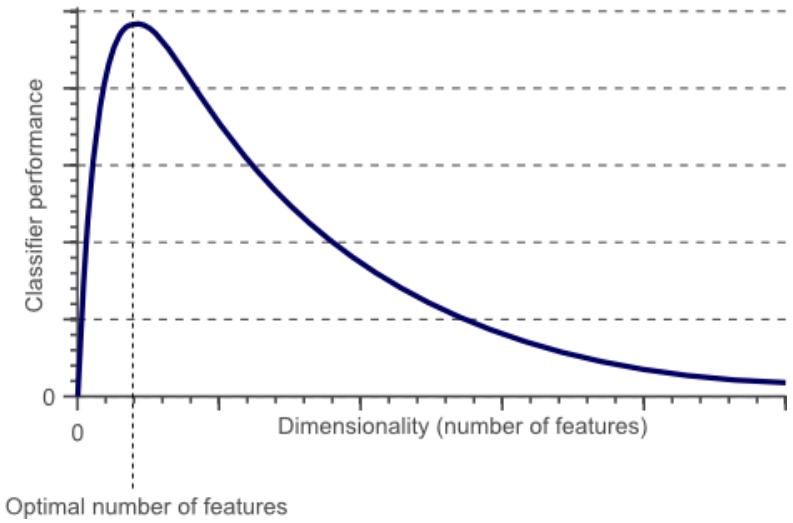


Curse of dimensionality

- **The required number of samples (to achieve the same accuracy) grows exponentionally with the number of variables!**

- **In practice: number of training examples is fixed!**

  => the classifier's performance usually will degrade for a large number of features!



In fact, after a certain point, increasing the dimensionality of the problem by adding new features would actually degrade the performance of classifier.

Curse of dimensionality

- **Many explored domains have hundreds to tens of thousands of variables/features with many irrelevant and redundant ones!**
- **In domains with many features the underlying probability distribution can be very complex and very hard to estimate (e.g. dependencies between variables) !**
- **Irrelevant and redundant features can "confuse" learners!**
- **Limited training data!**
- **Limited computational resources!**

Feature Extraction vs. Selection

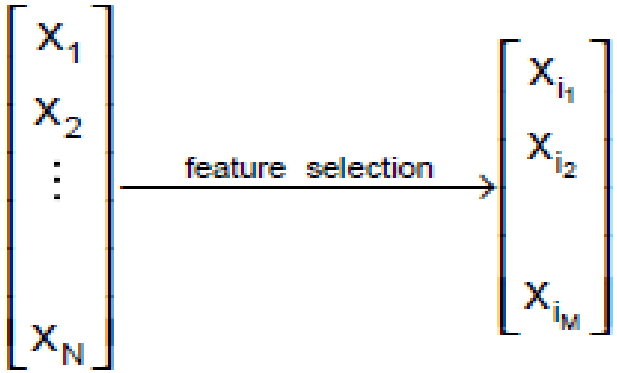
- **Feature extraction**
  - All original data are used and they are transferred
  - The transformed features are linear/nonlinear combinations of the original data
- **Feature selection**
  - Only a subset of the original features are selected

Feature selection

- **Feature selection:**

  Problem of selecting some subset from a set of input features upon which it should focus attention, while ignoring the rest

- **Humans/animals do that constantly!**
- **Given a set of N features, the role of feature selection is to select a subset of size M (M < N) that leads to the smallest classification/clustering error.**

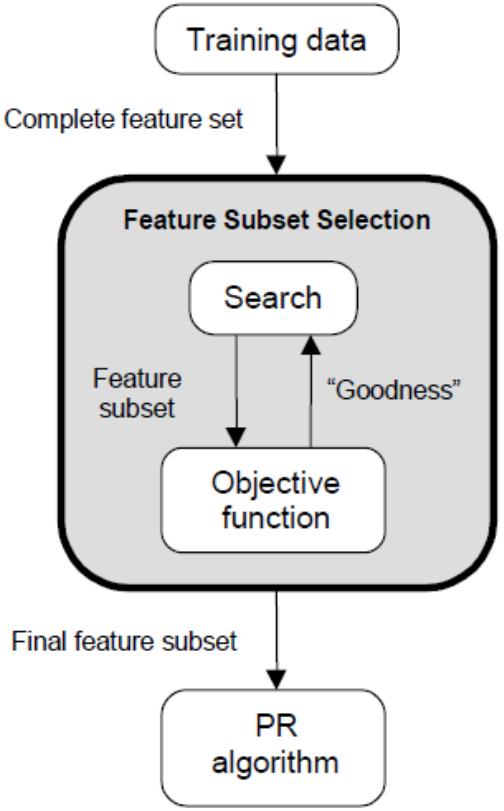


Feature Selection Methods

- **Feature selection is an optimization problem.**
  - Search the space of possible feature subsets.
  - Pick the subset that is optimal or near-optimal with respect to a certain criterion.

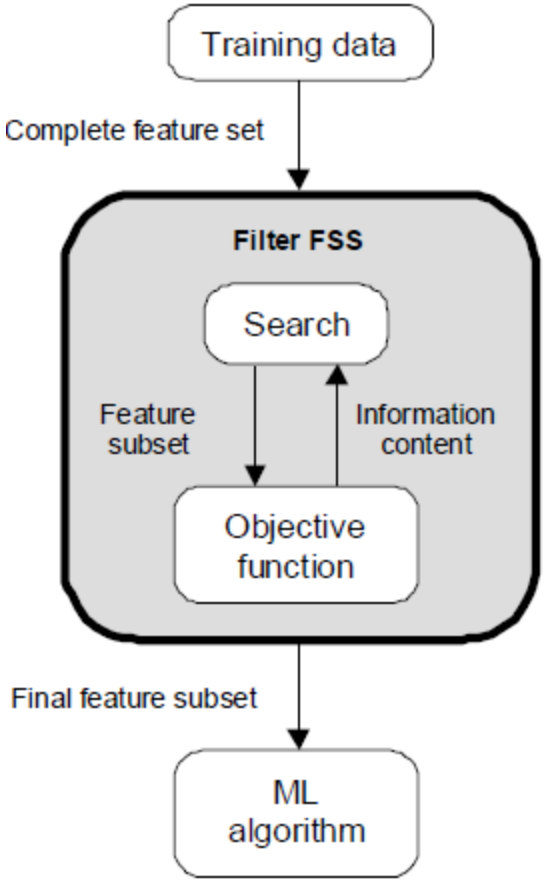
Search strategies
- Optimum
- Heuristic
- Randomized

Evaluation strategies
- Filter methods
- Wrapper methods



Evaluation Strategies

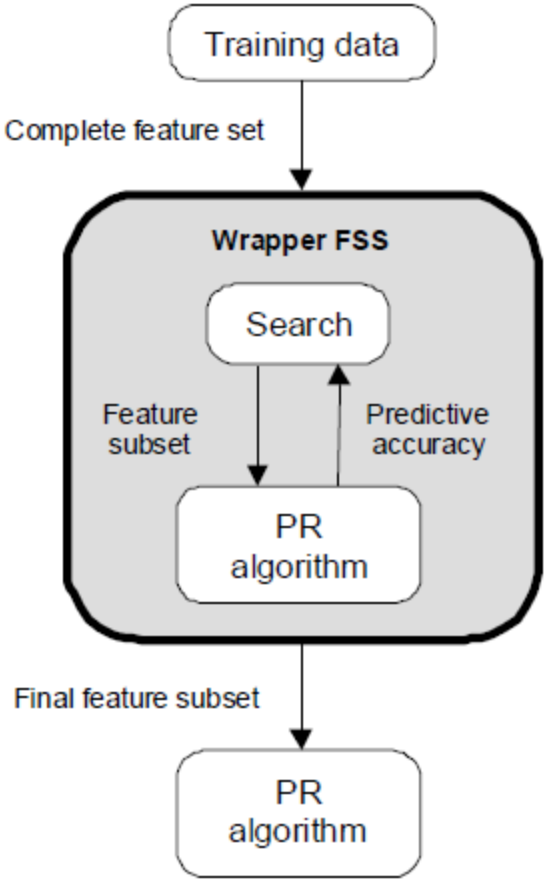
- **Filter Methods**
  - Evaluation is independent of the classification algorithm.
  - The objective function evaluates feature subsets by their information content, typically interclass distance, statistical dependence or information-theoretic measures.



- **Advantages**
  - **Fast execution:** Filters generally involve a non-iterative computation on the dataset, which can execute much faster than a classifier training session
  - **Generality:** Since filters evaluate the intrinsic properties of the data, rather than their interactions with a particular classifier, their results exhibit more generality; the solution will be "good" for a large family of classifiers
- **Disadvantages**
  - **Tendency to select large subsets:** Filter objective functions are generally monotonic

Evaluation Strategies

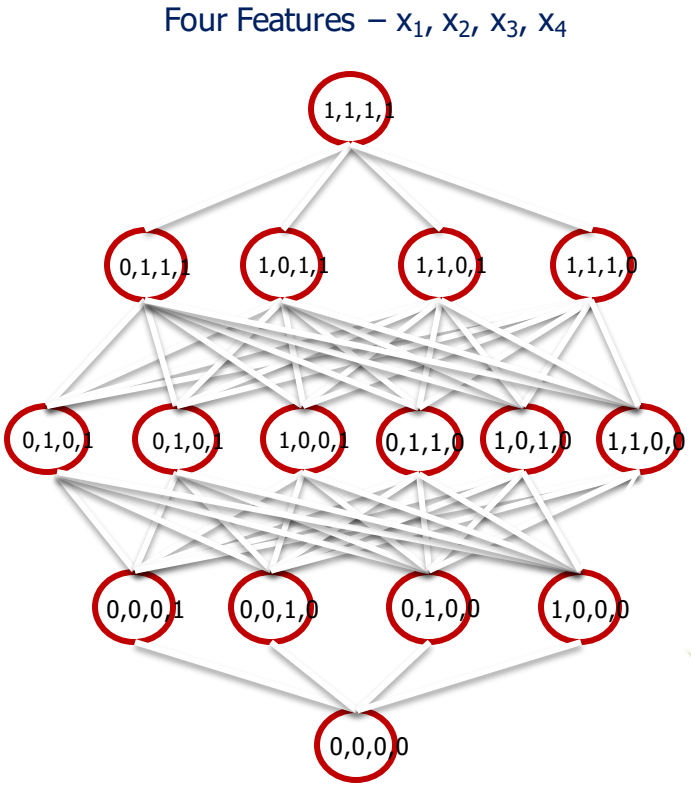
- **Wrapper Methods**
  - Evaluation uses criteria related to the classification algorithm.
  - The objective function is a pattern classifier, which evaluates feature subsets by their predictive accuracy (recognition rate on test data) by statistical resampling or cross-validation.



- **Advantages**
  - **Accuracy:** wrappers generally have better recognition rates than filters since they tuned to the specific interactions between the classifier and the features.
  - **Ability to generalize:** wrappers have a mechanism to avoid over fitting, since they typically use cross-validation measures of predictive accuracy.
- **Disadvantages**
  - **Slow execution**

Search Strategies

- Assuming *N* features, an exhaustive search would require:
  - Examining all $\binom{N}{M}$ possible subsets of size M.
  - Selecting the subset that performs the best according to the criterion function.
- The number of subsets grows combinatorially, making exhaustive search impractical.
- Iterative procedures are often used based on heuristics but they cannot guarantee the selection of the optimal subset.



$1 \rightarrow x_i$ is selected;
$0 \rightarrow x_i$ is not selected

Statistical test and Naïve Search

- **Naïve Search**
  - Sort the given N features in order of their probability of correct recognition.
  - Select the top M features from this sorted list.
  - Disadvantage
    - Feature correlation is not considered.
    - Best pair of features may not even contain the best individual feature.
- **Statistical test**
  - To select the best feature, employ two pair t-test and compare the significance of each feature against others. The most significant feature is then determined.
  - Repeat this for the next significant features.

Sequential forward selection (SFS) (heuristic search)

- First, the best single feature is selected (i.e., using some criterion function).
- Then, pairs of features are formed using one of the remaining features and this best feature, and the best pair is selected.
- Next, triplets of features are formed using one of the remaining features and these two best features, and the best triplet is selected.
- This procedure continues until a predefined number of features are selected.

SFS performs best when the optimal subset is small.

Sequential forward selection (SFS) (heuristic search)

Feature Set $\{x_1, x_2, x_3, x_4\}$

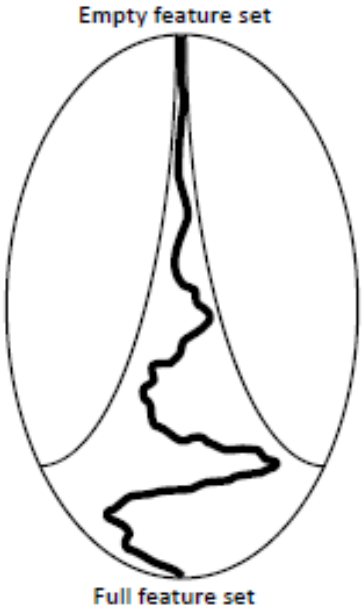
$\{x_1\}$ $\{x_2\}$ $\{x_3\}$ $\{x_4\}$ $J(x_2)>=J(x_i);\ i=1,3,4$

$\{x_2, x_1\}$ $\{x_2, x_3\}$ $\{x_2, x_4\}$ $J(x_2, x_3)>=J(x_2, x_i);\ i=1,4$

$\{x_2, x_3, x_1\}$ $\{x_2, x_3, x_4\}$ $J(x_2, x_3, x_1)>=J(x_2, x_3, x_4)$

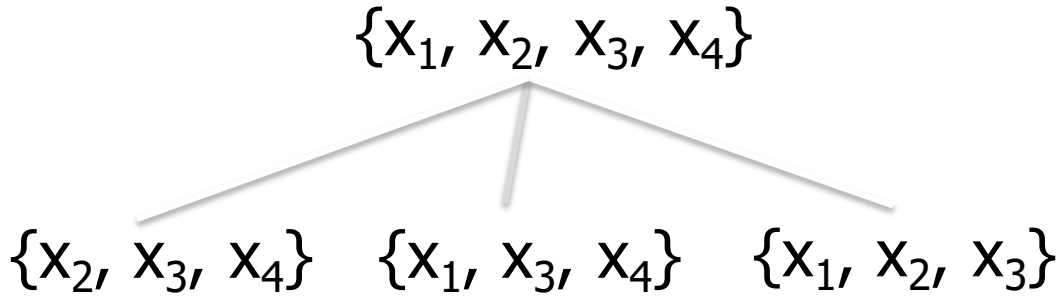
Sequential backward selection (SBS) (heuristic search)

- First, the criterion function is computed for combination of all N features.
- Then, each feature is deleted one at a time, the criterion function is computed for all remaining subsets with n-1 features, and the worst feature is discarded.
- Next, each feature among the remaining n-1 is deleted one at a time, and the worst feature is discarded to form a subset with n-2 features.
- This procedure continues until a predefined number of features are left.



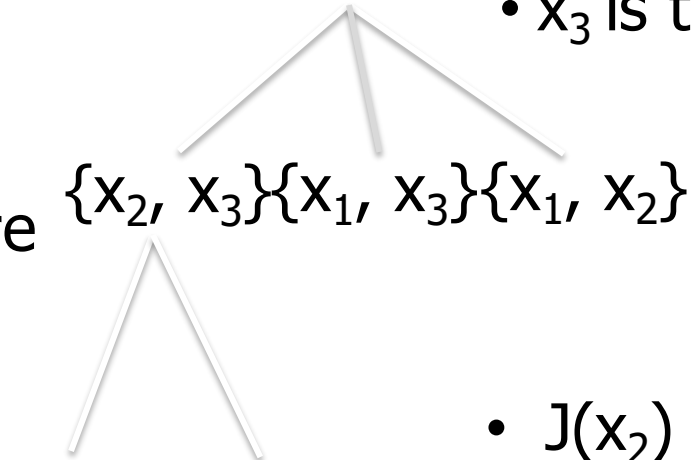
SBS performs best when the optimal subset is large.

Sequential backward selection (SBS) (heuristic search)



- $J(x_1, x_2, x_3)$ is maximum
- x_3 is the worst feature

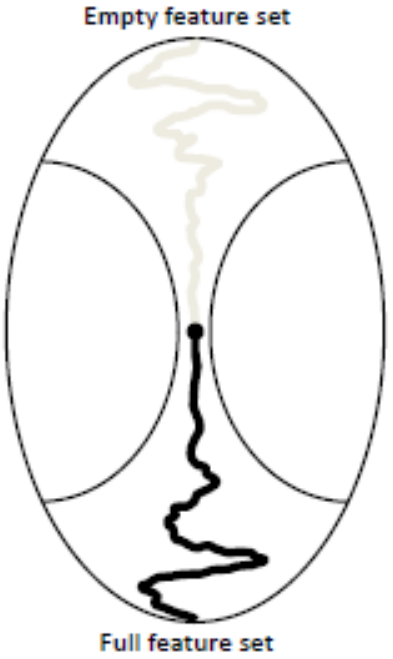
- $J(x_2, x_3)$ is maximum
- x_1 is the worst feature



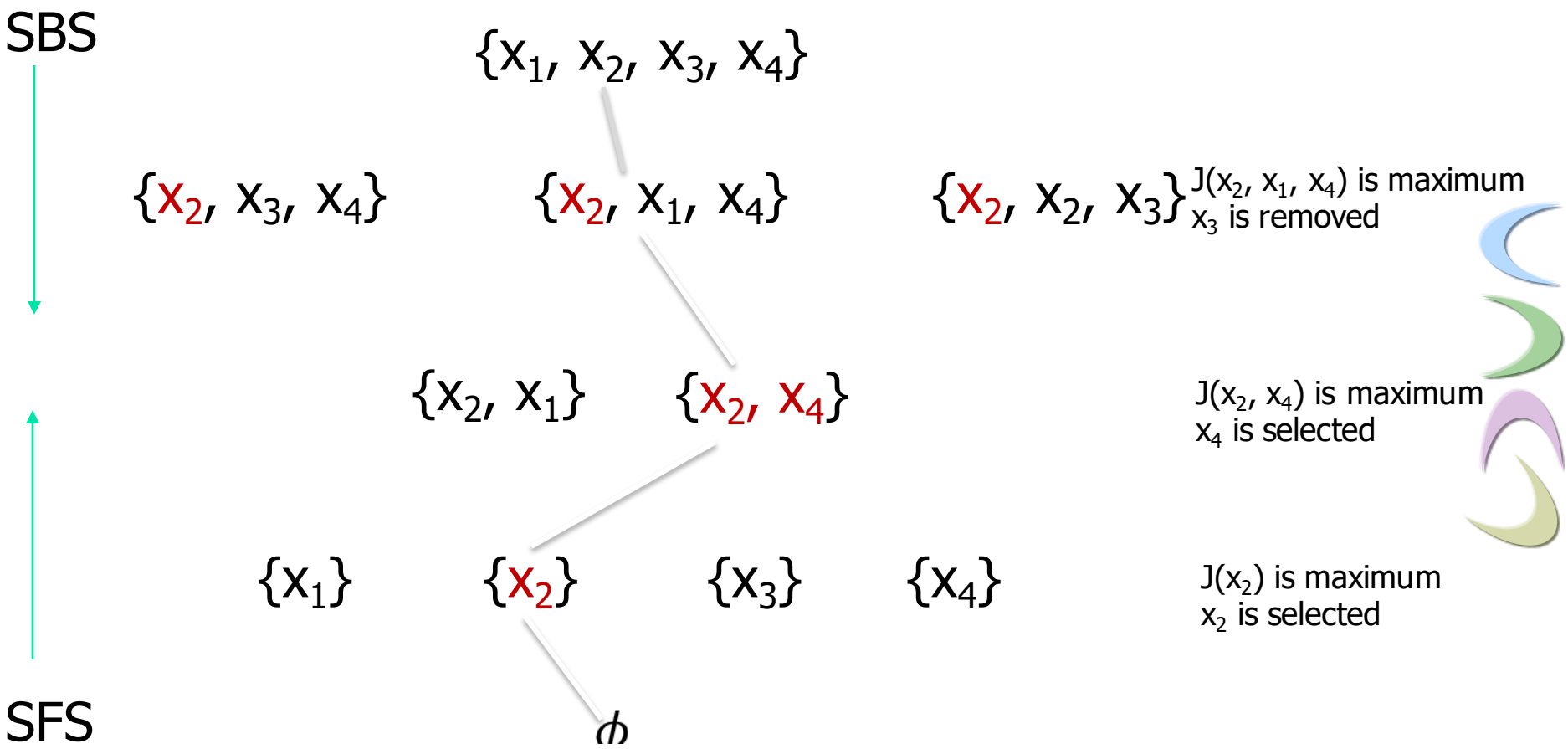
- $J(x_2)$ is maximum
- x_3 is the worst feature

Bidirectional Search (BDS) (heuristic search)

- BDS applies SFS and SBS simultaneously:
  - SFS is performed from the empty set
  - SBS is performed from the full set
- To guarantee that SFS and SBS converge to the same solution
  - Features already selected by SFS are not removed by SBS
  - Features already removed by SBS are not selected by SFS



Bidirectional Search (BDS)



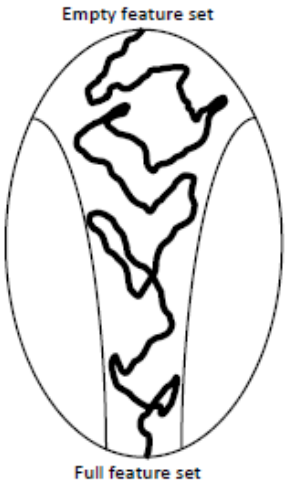
"Plus-L, minus-R" selection (LRS) (heuristic search)

- A generalization of SFS and SBS
  - If L>R, LRS starts from the empty set and:
    - Repeatedly add L features
    - Repeatedly remove R features
  - If L<R, LRS starts from the full set and:
    - Repeatedly removes R features
    - Repeatedly add L features

- LRS attempts to compensate for the weaknesses of SFS and SBS with some backtracking capabilities.
- There is no way of predicting the best values of L and R

Sequential floating selection (SFFS and SFBS) (heuristic search)

- **An extension to LRS with flexible backtracking capabilities**
  - **Rather than fixing the values of L and R, floating methods determine these values from the data.**
  - **The dimensionality of the subset during the search can be thought to be "floating" up and down**



- **There are two floating methods:**
  - **Sequential Floating Forward Selection (SFFS)**
  - **Sequential Floating Backward Selection (SFBS)**

Sequential floating selection (SFFS and SFBS)

- **SFFS**
  - **Sequential floating forward selection (SFFS) starts from the empty set.**
  - **After each forward step, SFFS performs backward steps as long as the objective function increases.**
- **SFBS**
  - **Sequential floating backward selection (SFBS) starts from the full set.**
  - **After each backward step, SFBS performs forward steps as long as the objective function increases.**

Dimensionality Reduction

- Another way of coping with the problem of high dimensionality is to reduce the dimensionality by combining features.
- Issues in feature reduction:
  - Linear vs. non-linear transformations.
  - Use of class labels or not (depends on the availability of training data).
- Training objective:
  - minimizing classification error (discriminative training),
  - minimizing reconstruction error (PCA),
  - maximizing class separability (LDA),
  - making features as independent as possible (ICA),
  - ...

Linear Dimensionality Reduction

- **Linear combinations are particularly attractive because they are simple to compute and are analytically tractable to project the high-dimensional data onto a lower dimensional space.**
  - reduced complexity in estimation and classification,
- **Two classical approaches for finding optimal linear transformations are:**
  - **Principal Components Analysis (PCA):** Seeks a projection that best represents the data in a least-squares sense.
  - **Linear Discriminant Analysis (LDA):** Seeks a projection that best separates the data in a least-squares sense.
- **Given $x \in \mathbb{R}^d$, the goal is to find a linear transformation A that gives $y = A^T x \in \mathbb{R}^{d'}$ where $d' < d$.**

Classifiers and Decision Functions

Introduction

- **Classification**
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model), based on the training set and the class labels, and uses it in classifying new data

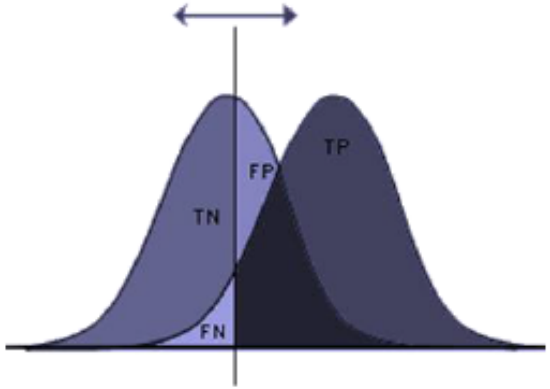
- **Model construction**
  - Each sample is assumed to belong to a predefined class, as determined by the class label
  - The set of samples used for model construction is called "training set"
  - The model is represented as classification rules, decision trees, probabilistic model, mathematical formulae and etc.

Evaluating Classification Methods

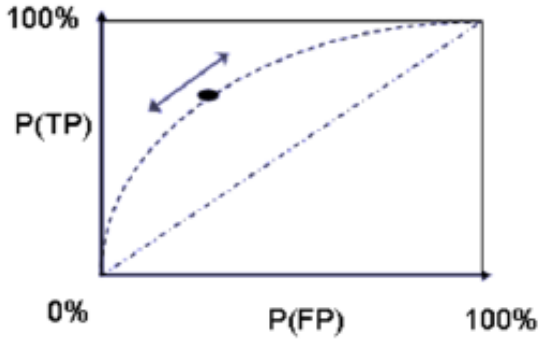
- **Performance**
  - classifier performance: predicting class label of new data
    - Accuracy, {true positive, true negative}, {false positive, false negative}, Sensitivity, Specificity, Precision, …
- **Time Complexity**
  - time to construct the model (training time)
    - the model will be constructed once
    - can be large
  - time to use the model (classification time)
    - must be tolerable
    - need for good data structures
- **Robustness**
  - handling noise and missing values
  - handling incorrect training data

Performance Measures

- **Accuracy is not a good measure for classifier performance always (Why?)**
  - **Suppose a "cancer detection" problem**
- **Presentation of Classifier Performance**
  - **Use a confusion matrix or a receiver-operating characteristic (ROC) curve**



Predicted \ Real	P	N
P	TP	FP
N	FN	TN



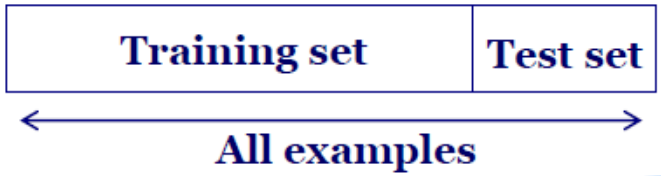
- **We can extract some performance measures from the above matrix (or curve)**

Performance Measures

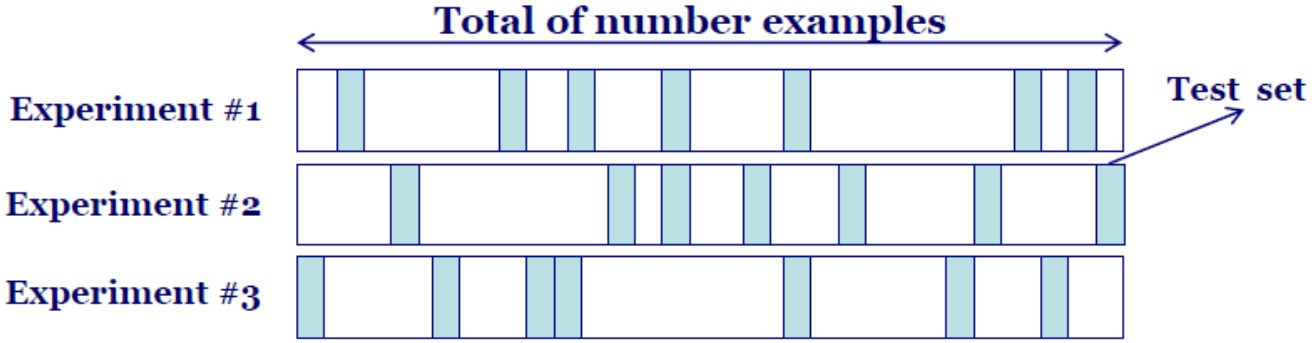
- **Performance Measures**
  - **Accuracy: (TP+TN) / (TP+TN+FN+FP)**
  - **Specificity: TN / (FP+TN)**
  - **Sensitivity: TP / (FN+TP)**
  - **Index of Merit: (Specificity + Sensitivity) / 2 = (TP%+TN%) / 2**
    - **Also known as "percentage correct classifications"**
- **Performance measured using test set results**
  - **Test set should be distinct and different from the train (learning) set.**
  - **Several methods are available to partition the data into separated training and testing sets, resulting in different estimates of the "true" index of merit**
    - **Goal: validating the classifier and its parameters**
      - **Choose the best parameter set**
    - **Idea: use a part of training data as the validation set**
      - **Validation set must be a good representative for the whole data**

Data Partitioning (Holdout)

- **Holdout methods: Random Sampling**
  - data is randomly partitioned into two independent sets
  - Always size of train set is twice of test set
  - Assumption: data is uniformly distributed

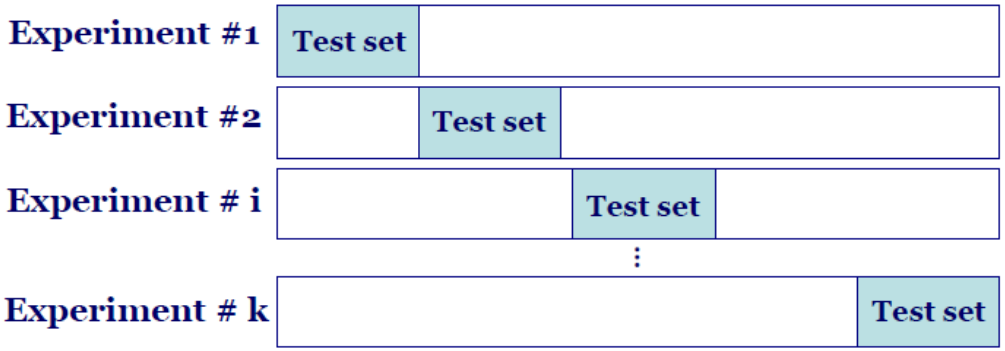


- **Holdout methods: Bootstrap**
  - Resample with replacement N samples of original data as training set.
  - Some numbers in the original sample may be included several times in the bootstrap sample (63.2% of samples are distinct)
- **Holdout methods: Multiple train-and-test experiment Bootstrap**



Data Partitioning(Cross-Validation)

- **Cross-validation (k-fold, where k = 10 is most popular)**
  - Randomly partition the data into k mutually exclusive subsets, each approximately equal size
  - At i^{th} iteration, use D_i as test set and others as training set
  - The mean of measures obtained in all iterations used as output performance measure



- **Leave-one-out**
  - k folds where *k* = #the number of samples, for small sized data
- **Stratified cross-validation**
  - folds are stratified so that class distributions in each fold is approximate the same as that in the initial data

Data Partitioning(Cross-Validation)

- How many folds are needed?
  - With a large number of folds
    - + The bias of the true error rate estimator will be small (the estimator will be very accurate)
    - -The variance of the true error rate estimator will be large
    - -The computational time will be very large as well (many experiments)
  - With small number of folds
    - + The number of experiments and, computation time are reduced
    - + The variance of the estimator will be small
    - -The bias of the estimator will be large(conservative or higher than the true error rate)
  - In practice, the choice of the number of folds depends on the size of the dataset
    - For large datasets, even 3-Fold Cross Validation will be quite accurate
    - For very sparse datasets, we may have to use leave-one-out in order to train on as many examples as possible

Data Partitioning (Three-way)

- If **model selection** and **true error estimates** are to be computed simultaneously, the data needs to be divided into three disjoint sets
  - **Training set:** a set of examples used for learning: to fit the parameters of the classifier
  - **Validation set:** a set of examples used to tune the parameters of a classifier
  - **Test set:** a set of examples used only to assess the performance of a fully-trained classifier
- Why separate test and validation sets?
  - The error rate estimate of the final model on validation data will be biased(smaller than the true error rate) since the validation set is used to select the final model
  - After assessing the final model with the test set, **YOU MUST NOT** tune the model any further

Parametric Models

Parametric Classifiers

- Bayesian Decision Theory
  - Prior Probabilities
  - Class-Conditional Probabilities
  - Posterior Probabilities
  - Probability of Error
  - Conditional Risk
  - MinMax Criteria
  - Naive Bayes
- Discriminant Functions
- Probability Density Function Estimation

Classifiers based on Bayes Decision Theory

- **Bayesian Decision Theory is a fundamental statistical approach that quantifies the tradeoffs between various decisions using probabilities and costs that accompany such decisions.**
  - First, we will assume that all probabilities are known.
  - Then, we will study the cases where the probabilistic structure is not completely known.
- **Feature vectors are treated as random vectors.**
- **For input feature vector** $\underline{x} = [x_1, x_2, \ldots, x_l]^T$
- **Assign the pattern represented by feature vector** $\underline{x}$ **to the most probable of the available classes** $\omega_1, \omega_2, \ldots, \omega_M$

That is $\underline{x} \rightarrow \omega_i : P(\omega_i | \underline{x})$ **maximum**

Bayesian Classifiers: Prior Probability

- **Prior probabilities reflect our knowledge of how likely each class type will appear before we actually see it.**
- **How can we choose $P(w_1)$ and $P(w_2)$?**
  - **Set $P(w_1) = P(w_2)$ if they are equiprobable (uniform priors).**
  - **May use different values depending on the problem.**
- **Assume there are no other types** $P(\omega_1) + P(\omega_2) = 1$
  - **(exclusivity and exhaustivity).**
- **How can we make a decision with only the prior information?**
  - **Decide** $\begin{cases} \omega_1 & P(\omega_1) > P(\omega_2) \\ \omega_2 & Otherwise \end{cases}$
- **What is the probability of error for this decision?**
  - **$P(error) = \min\{P(w_1), P(w_2)\}$**

Bayesian Classifiers: Class-Conditional Probability

- **Let's try to improve the decision using the lightness measurement x.**
  - Let x be a continuous random variable.
  - Define $P(x|w_j)$ as the class-conditional probability density (probability of x given that the state of nature is w_j for j = 1, 2).
  - $P(x|w_1)$ and $P(x|w_2)$ describe the hypothetical class-conditional probability density functions for two Classes.
- **How can we make a decision with only the class-conditional probabilities?**
  - Decide $\begin{cases} \omega_1 & P(x|\omega_1) > P(x|\omega_2) \\ \omega_2 & Otherwise \end{cases}$
  - **Class-conditional is known as "Maximum Likelihood".**
  - **Looks good, but prior information are not used. It may degrade decision performance**

Bayesian Classifiers: Posterior Probability

- **Computation of a-posteriori probabilities**
  - **Assume known**
    - **a-priori probabilities** $P(\omega_1), P(\omega_2)\ldots, P(\omega_M)$

      and
    - **likelihood probabilities** $p(\underline{x}|\omega_i), i = 1,2,\ldots, M$

      (This is also known as the likelihood of $\underline{x}$ $w.r.$ to ω_i. **)**
  - **The Bayes rule (For M=2)**

$$p(\underline{x})P(\omega_i|\underline{x}) = p(\underline{x}|\omega_i)P(\omega_i) \Rightarrow$$

$$P(\omega_i|\underline{x}) = \frac{p(\underline{x}|\omega_i)P(\omega_i)}{p(\underline{x})} \approx \frac{\text{Likelihood}\times\text{Priori}}{\text{Evidence}}$$

where

$$p(\underline{x}) = \sum_{i=1}^{2} p(\underline{x}|\omega_i)P(\omega_i)$$

Bayesian Classifiers: Posterior Probability

- **The Bayes classification rule (for two classes $M=2$)**
  - **Given $\underline{x}$ classify it according to the rule**

$$If\ P(\omega_1|\underline{x}) > P(\omega_2|\underline{x})\ \ \underline{x} \rightarrow \omega_1$$

$$If\ P(\omega_2|\underline{x}) > P(\omega_1|\underline{x})\ \ \underline{x} \rightarrow \omega_2$$

  - **Equivalently: classify $\underline{x}$ according to the rule**

$$p(\underline{x}|\omega_1)P(\omega_1)(><)p(\underline{x}|\omega_2)P(\omega_2)$$

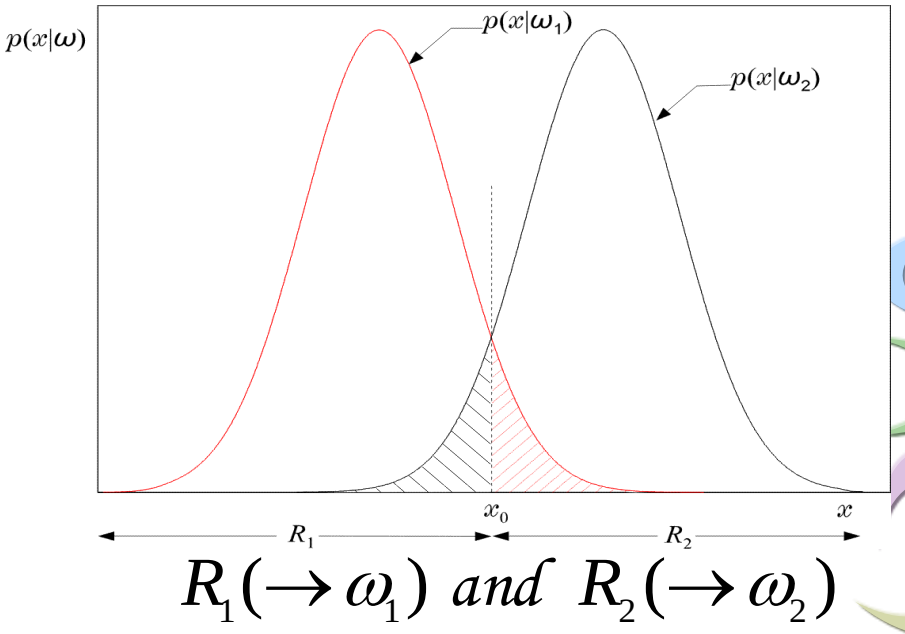
  - **For equiprobable classes the test becomes**

$$p(\underline{x}|\omega_1)(><)p(\underline{x}|\omega_2)$$

Bayesian Classifiers: Posterior Probability

- **Equivalently in words: Divide space in two regions**

$$\text{If } \underline{x} \in R_1 \Rightarrow \underline{x} \text{ in } \omega_1$$
$$\text{If } \underline{x} \in R_2 \Rightarrow \underline{x} \text{ in } \omega_2$$



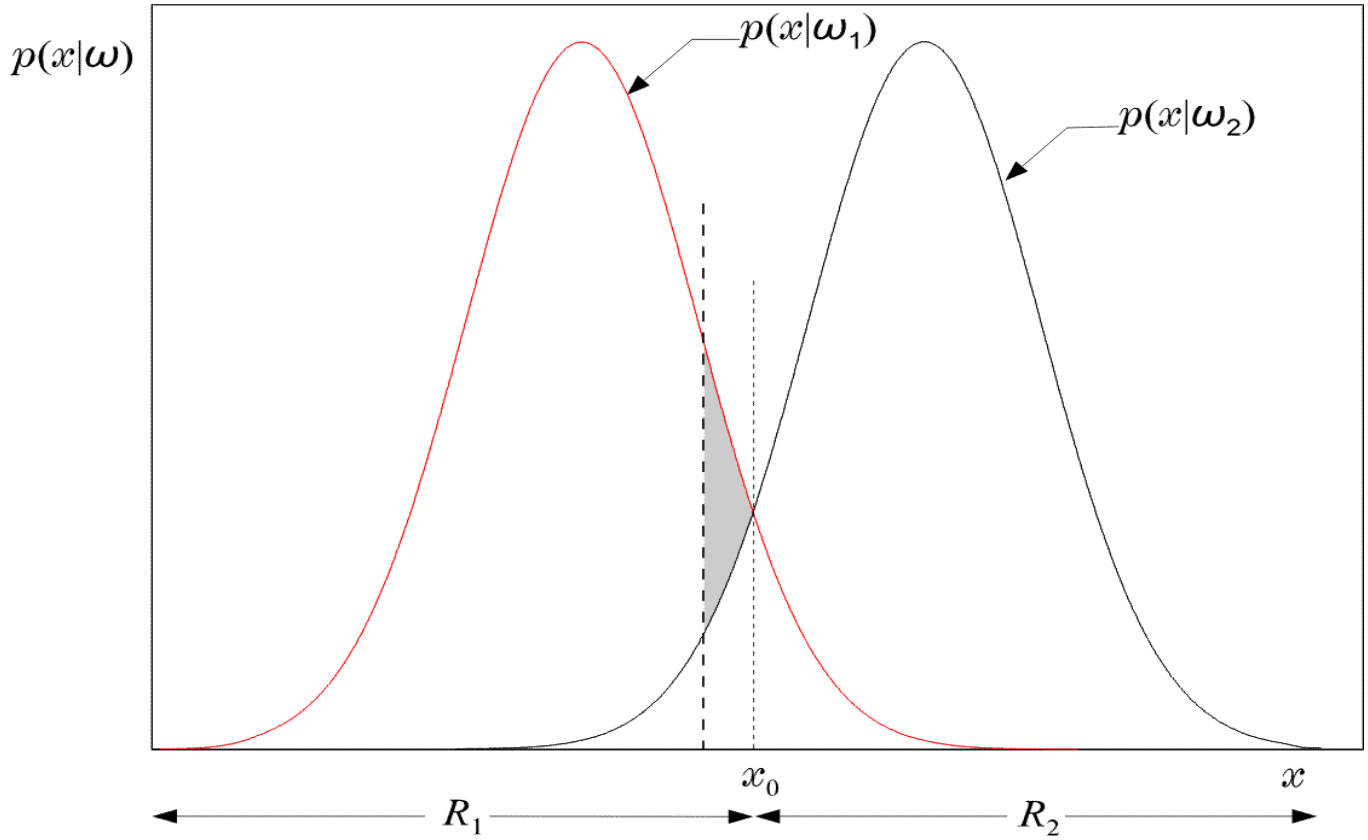
$$R_1(\rightarrow \omega_1) \text{ and } R_2(\rightarrow \omega_2)$$

- **Probability of error**
  - **Total shaded area**

$$P_e = \frac{1}{2}\int_{-\infty}^{x_0} p(x|\omega_2)dx + \frac{1}{2}\int_{x_0}^{+\infty} p(x|\omega_1)dx$$

- **Bayesian classifier is OPTIMAL with respect to minimizing the classification error probability!!!!**

Bayesian Classifiers: Posterior Probability



- **Indeed: Moving the threshold the total shaded area INCREASES by the extra "gray" area.**

Bayesian Classifiers: Probability Error

- **What is the probability of error for each decision?**

$$p(error \mid x) = \begin{cases} p(\omega_1 \mid x) & if\ we\ decide\ \omega_2 \\ p(\omega_2 \mid x) & if\ we\ decide\ \omega_1 \end{cases}$$

- **What is the average probability of error?**

$$p(error) = \int_{-\infty}^{+\infty} p(error, x)dx = \int_{-\infty}^{+\infty} p(error \mid x)p(x)dx$$

- **Bayes decision rule minimizes this error because**

$$p(error \mid x) = \min\{p(\omega_1 \mid x), p(\omega_2 \mid x)\}$$

Bayesian Classifiers: Conditional Risk

- **The Bayes classification rule for many classes(M>2):**
  - **Given $\underline{x}$ classify it to ω_i if:**

$$P(\omega_i|\underline{x}) > P(\omega_j|\underline{x}) \quad \forall j \neq i$$

  - **Such a choice also minimizes the classification error probability**

- **Minimizing the average risk**
  - **For each wrong decision, a penalty term is assigned since some decisions are more sensitive than others**

Bayesian Classifiers: Naive Bayes

- **NAIVE – BAYES CLASSIFIER**
  - **Let $\underline{x} \in \Re^{\ell}$ and the goal is to estimate $p(\underline{x} \mid \omega_i)$ for $i = 1, 2, \ldots, M$. For a "good" estimate of the pdf one would need, say, N^{ℓ} points.**
  - **Assume $x_1, x_2, \ldots, x_{\ell}$ mutually independent. Then:**

$$p(\underline{x} \mid \omega_i) = \prod_{j=1}^{\ell} p(x_j \mid \omega_i)$$

  - **In this case, one would require, roughly, N points for each pdf. Thus, a number of points of the order $N \cdot \ell$ would suffice.**
  - **It turns out that the Naïve – Bayes classifier works reasonably well even in cases that violate the independence assumption.**

DISCRIMINANT FUNCTIONS DECISION SURFACES

- **discriminant function divides the feature space by a decision surface**
- **If R_i, R_j are contiguous:** $g(\underline{x}) \equiv P(\omega_i|\underline{x}) - P(\omega_j|\underline{x}) = 0$

$$R_i : P(\omega_i|\underline{x}) > P(\omega_j|\underline{x})$$

$$+ \quad \text{---------- } g(\underline{x}) = 0$$

$$-$$

$$R_j : P(\omega_j|\underline{x}) > P(\omega_i|\underline{x})$$

g(x) is the surface separating the regions. On the one side is positive (+), on the other is negative (-).

It is known as Decision Surface.

DISCRIMINANT FUNCTIONS DECISION SURFACES

- **If $f(.)$ monotonically increasing, the rule remains the same if we use:**

$$\underline{x} \to \omega_i \text{ if : } f(P(\omega_i|\underline{x})) > f(P(\omega_j|\underline{x})) \ \forall i \neq j$$

- $g_i(\underline{x}) \equiv f(P(\omega_i|\underline{x}))$ **is a discriminant function.**

- **In general, discriminant functions can be defined independent of the Bayesian rule.**
  - **They lead to suboptimal solutions, yet, if chosen appropriately, they can be computationally more tractable.**
  - **Moreover, in practice, they may also lead to better solutions. This, for example, may be case if the nature of the underlying pdf's are unknown.**

Linear Discriminant Function (LDF)

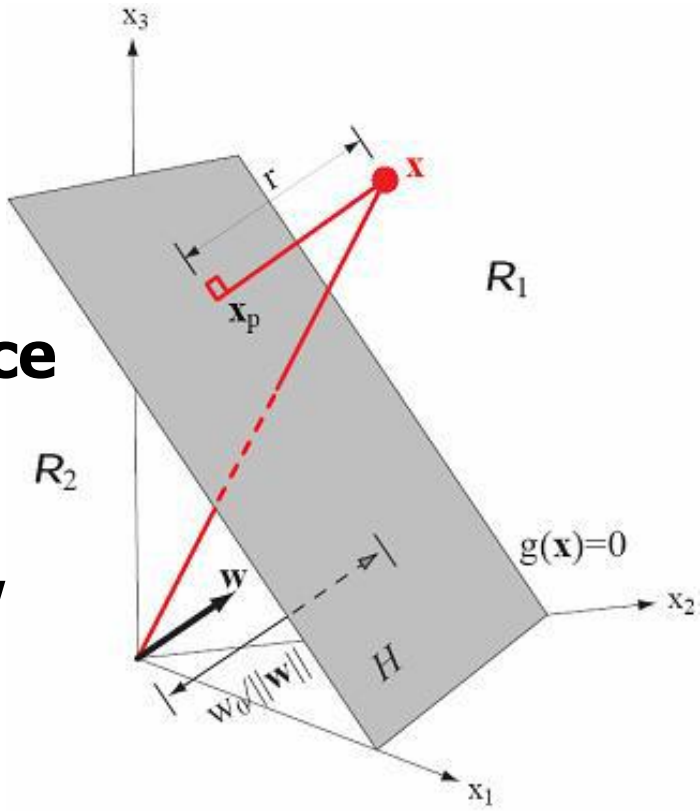
- **Definition:**
  - **LDF is a function that is a linear combination of the components of x**

$$g(x) = w^t x + w_0$$

    - where w is the weight vector and w_0 the bias, or threshold weight.
- **A two-category classifier with a discriminant function of the above form uses the following rule:**
  - **Decide ω_1 if g(x) > 0 and ω_2 if g(x) < 0**
    - Decide ω_1 if $w^t x > -w_0$ and ω_2 otherwise
    - The value g(x) of is called functional margin
  - **If g(x) = 0 then x can be assigned to either class**
    - The equation g(x) = 0 defines the decision surface that separates points assigned to the category ω_1 from points assigned to the category ω_2
    - When g(x) is linear, the decision surface is a hyperplane.

Linear Discriminant Function (LDF)

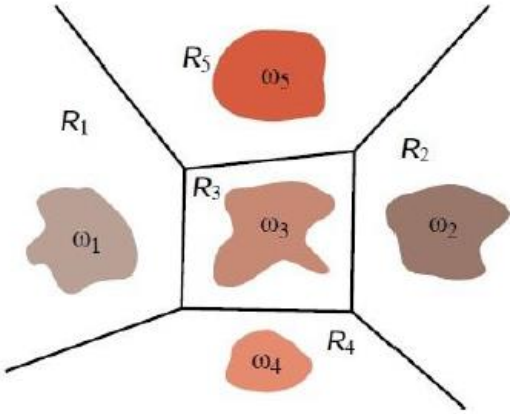
- A linear discriminant function divides the feature space by a hyperplane decision surface
- Decision boundary *g(x)=0* corresponds to (d-1)-dimensional hyperplane in d-dimensional x-space
- The orientation of the surface is determined by the normal vector w and the location of the surface is determined by the bias w_0
  - We can view Fisher method (LDA) as a linear discriminant function, too.

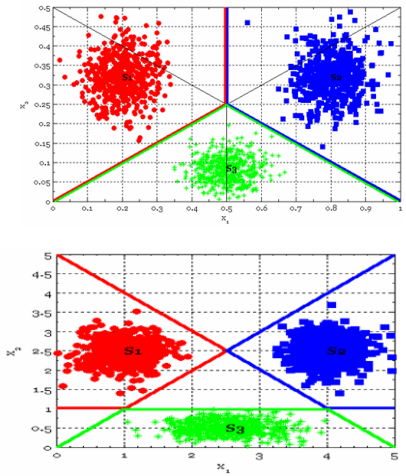


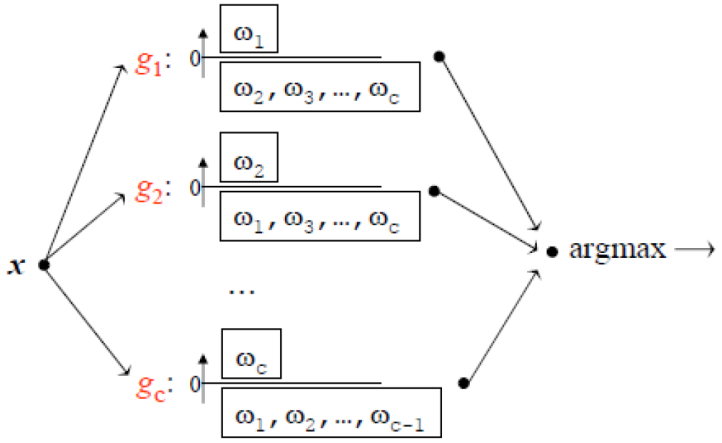
Linear Discriminant Function (LDF) Linear Machine

- **Suppose we have an n-classes classification problem, and we want to separate them with linear discriminant functions.**
  - **If we use the following rule for classification, this is linear machine rule:**

$$x \in \omega_i \Leftrightarrow g_i(x) > g_j(x);\ \forall j \neq i$$







  - **If we use the following rule for classification, this is completely linearly separation rule:**

$$if\ g_i(x) > 0 \ \Rightarrow\ x \in \omega_i\ and\ \ if\ g_i(x) < 0 \ \Rightarrow\ x \notin \omega_i$$

- **The decision regions for linear machine are convex and this restriction limits the flexibility of the classifier.**

Linear Discriminant Function Design

- **Definition:**
  - **LDF is a function that is a linear combination of the components of x**

$$g(x) = w^t x + w_0$$

    - where w is the weight vector and w_0 the bias, or threshold weight.
- **Main problem**
  - **How to create the discriminant functions for each class(how obtain w)?**
- **Many methods exist for this purpose, such as:**
  - **Probabilistic Methods**
  - **Error Minimization Methods**
    - **Least Mean Squared Error Method**
    - **Sum of Squared Error Method**
    - **Stochastic Minimization Methods**
    - **Ho-Kashyap Method**
  - **Perceptorn Method**
  - **etc.**

Linear Discriminant Function Design (Probabilistic Methods)

- **Maximum likelihood**
  - $g_i(x) = P(x|\omega_i)$
- **Bayesian Classifier**
  - $g_i(x) = P(\omega_i|x)$
    - $g_i(x) = P(x|\omega_i)P(\omega_i)$
    - $g_i(x) = Ln(P(x|\omega_i)) + Ln(P(\omega_i))$
- **Expected Loss (Conditional Risk)**
  - $g_i(x) = -R(a_i|x)$

Linear Discriminant Function Design (Augmented Space)

- **Consider a linear discriminant function $g(x)$ in feature space x**

$$g(x) = w^t x + w_0$$

- **We can use augmented space to get ride of w_0 in new augmented feature space x^* which map each data point x in to new augmented space x^* according:**

$$x^* = \begin{bmatrix} x \\ 1 \end{bmatrix} \quad and \quad w^* = \begin{bmatrix} w \\ w_0 \end{bmatrix}$$

- **So the new discriminant function will be:** $g(x^*) = w^{*t} x^*$
- **Then:**

$$g(x^*) > 0 \; for \; positive \; calss$$
$$g(x^*) < 0 \; for \; negative \; calss$$

- **If we invert the training data of negative class then for all data point we must have**

$$x^* = \begin{bmatrix} x_{11} & \cdots & x_{1d} & 1 \\ x_{21} & \cdots & x_{2d} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{k1} & \cdots & x_{kd} & 1 \\ x_{(k+1)1} & \cdots & x_{(k+1)d} & -1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & \vdots & x_{nd} & -1 \end{bmatrix} \Rightarrow for \; all \; data \quad g(x^*) > 0$$

Linear Discriminant Function Design (Least Mean Squared Error)

- **The steps to Compute a solution, i.e., a hyperplane w^***
  - **Define a cost function ($J(w^*)$) to be minimized**
  - **Choose an algorithm to minimize the cost function**
  - **The minimum corresponds to a solution**

- **We want to choose the w^* that minimizes the mean-squared-error criterion function:**

$$J(w^*) = E\left[|y - g(x^*)|^2\right] = E\left[(y - w^{*t}x^*)^2\right]$$

$$\widehat{w^*} = arg \min_{w^*} J(w^*)$$

  - **y is the corresponding desired responses**

Linear Discriminant Function Design (Least Mean Squared Error)

- **Minimizing**

$J(\underline{w}^*)$ w.r. to $\underline{w}^*$ results in :

$$\frac{\partial J(\underline{w}^*)}{\partial \underline{w}^*} = \frac{\partial}{\partial \underline{w}^*} E[(y - \underline{w}^{*T} x^*)^2] = 2E[x^*(y - x^{*T}\underline{w}^*)] = 0$$

$$\Rightarrow E[x^* x^{*T}]\underline{w}^* = E[x^* y] \Rightarrow \quad \boxed{\underline{\hat{w}} = R_{x^*}^{-1} E[x^* y]}$$

- **where R_x is the autocorrelation matrix**

$$R_{x^*} \equiv E[\underline{x}^* \underline{x}^{*T}] = \begin{bmatrix} E[x_1^* x_1^*] & E[x_1^* x_2^*] \ldots & E[x_1^* x_n^*] \\ \ldots\ldots\ldots & \ldots\ldots\ldots\ldots & \ldots\ldots\ldots \\ E[x_n^* x_1^*] & E[x_n^* x_2^*] \ldots & E[x_n^* x_n^*] \end{bmatrix}$$

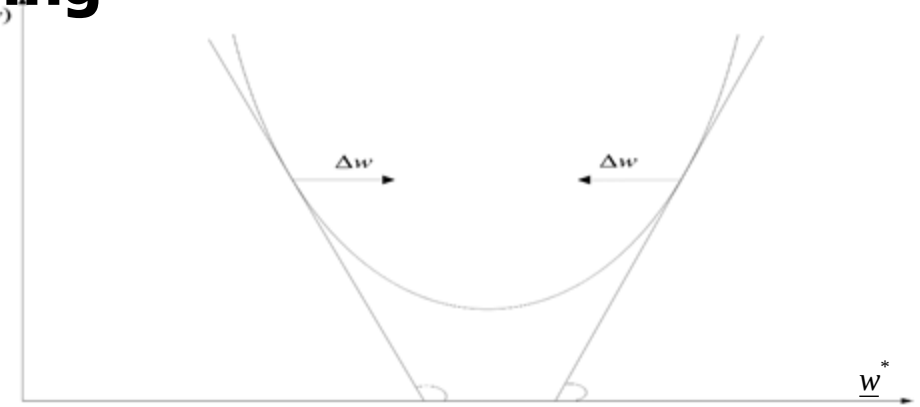
and $E[x^* y] = \begin{bmatrix} E[x_1^* y] \\ \ldots \\ E[x_n^* y] \end{bmatrix}$ **the crosscorrelation vector**

Linear Discriminant Function Design (Least Mean Squared Error)

- **Remark:** The MSE criterion belongs to a more general class of cost function with the following important property:
  - The value of $g_i(x^*)$ is an estimate, in the MSE sense, of the a-posteriori probability $P(\omega_i|x^*)$, provided that the desired responses used during training are $y_i = 1\ if\ x^* \in \omega_i$ and 0 otherwise.
- We can also use the gradient descent rule for updating w^* instead of analytical solving

$$\underline{w}^*(\text{new}) = \underline{w}^*(\text{old}) + \Delta\underline{w}^*$$

$$\Delta\underline{w}^* = -\mu \frac{\partial J(\underline{w}^*)}{\partial \underline{w}^*}\Big|_{\underline{w}^* = \underline{w}^*(\text{old})}$$



Linear Discriminant Function Design (Sum of Squared Error)

- **In SSE we want to choose the w^* that minimizes the sum of squared error as objective function**
  - **Also known as Pseudo inverse matrix method**

$$J(\underline{w}^*) = \sum_{i=1}^{N} (y_i - \underline{w}^{*T} \underline{x}_i^*)^2$$

$(y_i, \underline{x}_i^*)$: training pairs that, the input is $\underline{x}_i$ and its corresponding class label is y_i (±1).

$$\frac{\partial J(\underline{w}^*)}{\partial \underline{w}^*} = \frac{\partial}{\partial \underline{w}^*} \sum_{i=1}^{N} (y_i - \underline{w}^{*T} \underline{x}_i^*)^2 = 0 \Rightarrow$$

$$(\sum_{i=1}^{N} \underline{x}_i^* \underline{x}_i^{*T}) \underline{w}^* = \sum_{i=1}^{N} \underline{x}_i^* y_i$$

Linear Discriminant Function Design (Sum of Squared Error)

- **Pseudo inverse Matrix**
  - **Define**

$$X = \begin{bmatrix} \underline{x}_1^T \\ \underline{x}_2^T \\ \dots \\ \underline{x}_N^T \end{bmatrix} \quad (N \times (d+1) \text{ matrix}) \qquad \underline{y} = \begin{bmatrix} y_1 \\ \dots \\ y_N \end{bmatrix} \text{ corresponding desired responses}$$

  - $X^T = [\underline{x}_1, \underline{x}_2, \dots, \underline{x}_N] \quad (\text{an } (d+1) \times N \text{ matrix})$
  - $X^T X = \sum_{i=1}^{N} \underline{x}_i \underline{x}_i^T$ **and** $X^T \underline{y} = \sum_{i=1}^{N} \underline{x}_i y_i$

- **Thus** $(\sum_{i=1}^{N} \underline{x}_i^T \underline{x}_i)\underline{\hat{w}} = (\sum_{i=1}^{N} \underline{x}_i y_i) \Rightarrow (X^T X)\underline{\hat{w}} = X^T \underline{y} \Rightarrow \underline{\hat{w}} = (X^T X)^{-1} X^T \underline{y} = X^{\neq} \underline{y}$

$$X^{\neq} \equiv (X^T X)^{-1} X^T$$ **Pseudo inverse of X**

Linear Discriminant Function Design (Sum of Squared Error)

- **Assume $N = d+1 \Rightarrow X$ square and invertible. Then**

$$(X^T X)^{-1} X^T = X^{-1} X^{-T} X^T = X^{-1} \Rightarrow \boxed{X^{\neq} = X^{-1}}$$

- **Assume $N > d+1$. Then, in general, there is no solution to satisfy all equations simultaneously:**

$$X\underline{w} = \underline{y}: \begin{cases} \underline{x}_1^T \underline{w} = y_1 \\ \underline{x}_2^T \underline{w} = y_2 \\ \cdots \\ \underline{x}_N^T \underline{w} = y_N \end{cases} \quad N \text{ equations} > d+1 \text{ unknowns}$$

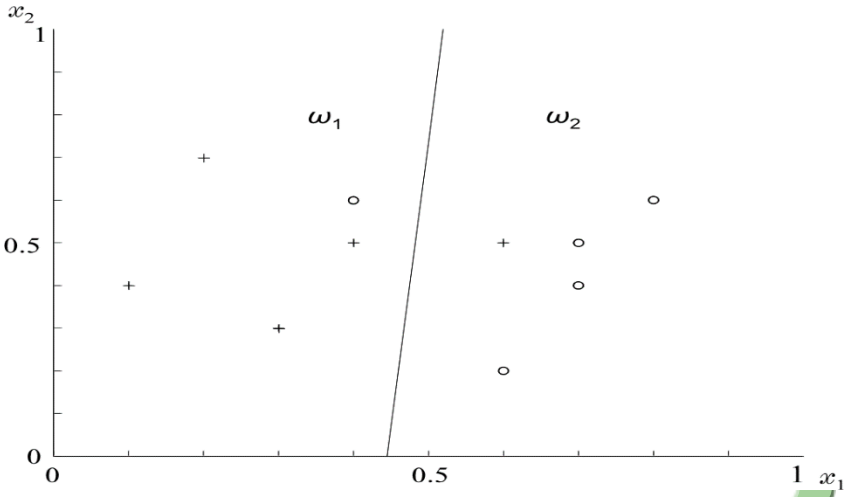
- **The "solution" $\underline{w} = X^{\neq} \underline{y}$ corresponds to the minimum sum of squares solution**

Linear Discriminant Function Design (Sum of Squared Error)

- **Example:**

$$\omega_1 : \begin{bmatrix}0.4\\0.5\end{bmatrix}, \begin{bmatrix}0.6\\0.5\end{bmatrix}, \begin{bmatrix}0.1\\0.4\end{bmatrix}, \begin{bmatrix}0.2\\0.7\end{bmatrix}, \begin{bmatrix}0.3\\0.3\end{bmatrix}$$

$$\omega_2 : \begin{bmatrix}0.4\\0.6\end{bmatrix}, \begin{bmatrix}0.6\\0.2\end{bmatrix}, \begin{bmatrix}0.7\\0.4\end{bmatrix}, \begin{bmatrix}0.8\\0.6\end{bmatrix}, \begin{bmatrix}0.7\\0.5\end{bmatrix}$$



$$X = \begin{bmatrix}0.4 & 0.5 & 1\\0.6 & 0.5 & 1\\0.1 & 0.4 & 1\\0.2 & 0.7 & 1\\0.3 & 0.3 & 1\\0.4 & 0.6 & 1\\0.6 & 0.2 & 1\\0.7 & 0.4 & 1\\0.8 & 0.6 & 1\\0.7 & 0.5 & 1\end{bmatrix} \text{ and } \underline{y} = \begin{bmatrix}1\\1\\1\\1\\1\\-1\\-1\\-1\\-1\\-1\end{bmatrix} \Rightarrow$$

$$X^T X = \begin{bmatrix}2.8 & 2.24 & 4.8\\2.24 & 2.41 & 4.7\\4.8 & 4.7 & 10\end{bmatrix}, \; X^T \underline{y} = \begin{bmatrix}-1.6\\0.1\\0.0\end{bmatrix}$$

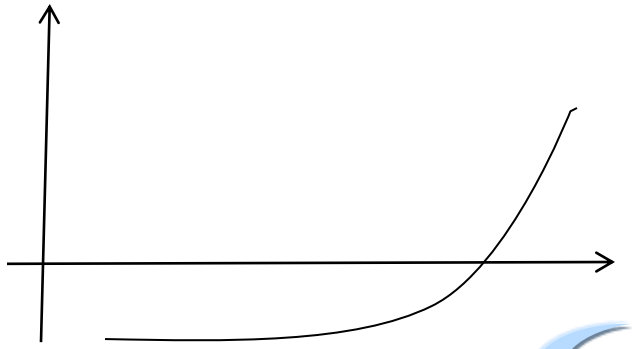
$$\underline{w} = (X^T X)^{-1} X^T \underline{y} = \begin{bmatrix}-3.13\\0.24\\1.34\end{bmatrix}$$

Discriminant Function Design (Stochastic Minimization)

- **Our goal:**

$$E\{F(x_k, w)\} = 0, \quad x_k : \text{Sequence of samples}$$

$$w_k = w_{k-1} + \varepsilon_k F(x_k, w_{k-1}), \quad \begin{cases} \sum \varepsilon_k \to \infty \\ \sum \varepsilon_k^2 \to 0 \end{cases}$$



- **Simple Model:**

$$E\{x_k - w\} = 0, \quad x_k : \text{Sequence of samples}$$

$$w_k = w_{k-1} + \frac{1}{k}(x_k - w_{k-1})$$

$$w_k = \frac{k-1}{k} w_{k-1} + \frac{1}{k} x_k \Rightarrow w_k = \frac{1}{k} \sum_{r=1}^{k} x_r$$

- **Widrow-Hoff Learning:**

$$w_k = w_{k-1} + \varepsilon_k x_k \left(y_k - x_k^T w_{k-1}\right)$$

LMS Method with fixed step size: $0 < \varepsilon < \frac{2}{trace(R_x)}$

Linear Discriminant Function Design (Ho-Kashyap Method)

- **The main limitation of the SSE is lack of guarantees that a separating hyperplane will be found in the linearly separable case**
  - The SSE rule tries to minimize $\|w^t x + y\|^2$
  - Finding a separating hyperplane depends on how suitably the outputs **y** are selected

- **If the two classes are linearly separable, there must exists vectors w and b such that $w^t x = y > 0$**
  - if **y** were known, to compute the separating hyperplane, the SSE solution will be $w = x - y$
  - Nevertheless, since **y** is unknown, one must solve the equation for both w and y

- **A possible algorithm is the Ho-Kashyap procedure:**
  - **1. Find the target values y with gradient descent**
  - **2. compute the weight vector w from the SSE solution**
  - **3. Repeat 1 and 2 until convergence**

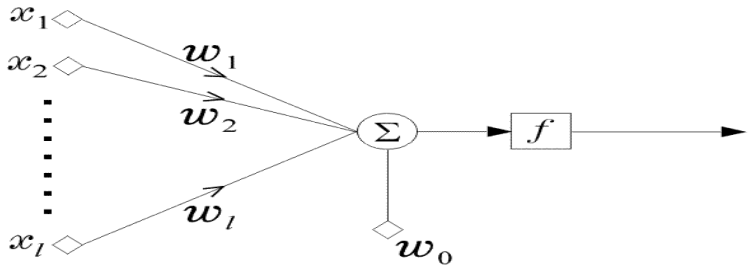
Linear Discriminant Function Design (Ho-Kashyap Method)

- $g(x) > 0$ **can be rewrite as** $g(x) = y\,; y > 0$
  - **How we can determine** y**?**
- **Objective function in this case is** $J(w, y) = \|w^t x - y\|^2$
- **Ho-Kashyap method offers an iterative method for obtaining w and** y**, using following steps:**
  - **Keep** y **constant and optimize** J **relative to** w **(using obtained** y **from last step)**
    - Using previous method we have: $w(t+1) = x^t y(t)$
  - **Keep** w **constant and optimize** J **relative to** y **(using obtained** w **from last step)**
    - The objective is to minimize $\frac{dJ}{dw} = -2w^t x - y$
    - Using Gradient descent method we have: $y(t+1) = y(t) + 2\eta\, w^t(t)x - y$
    - To hold the constraint $y > 0$, we set $(w^t x - y)$ in this rule to zero if it becomes negative, then the rule will be:

$$y(t+1) = y(t) + \eta[w^t(t)x - y + |w^t(t)x - y|]$$

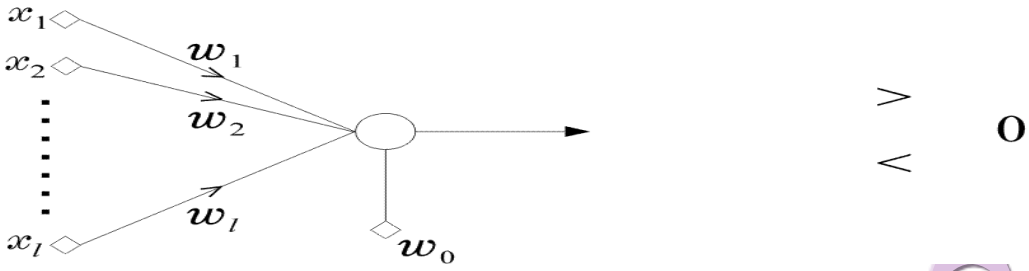
Linear Discriminant Function Design (Perceptorn Method)

- **The network is called perceptron or neuron**



$w_i's$ synapses or synaptic weights

w_0 threshold



- **It is a learning machine that learns from the training vectors via the perceptron algorithm**

$$\mathbf{w}(k+1)=\begin{cases}\mathbf{w}(k)+\mu\mathbf{x} & x\in\omega_1 \text{ and } \mathbf{w}^T(k)\mathbf{x}<0\\ \mathbf{w}(k)-\mu\mathbf{x} & x\in\omega_2 \text{ and } \mathbf{w}^T(k)\mathbf{x}>0\end{cases}$$

$$\mu > \frac{\left|\mathbf{w}^T(k)\mathbf{x}\right|}{\mathbf{x}^T\mathbf{x}}$$

Non-parametric Methods

Nonparametric Classification
K Nearest Neighbor (KNN)

- **The Nearest Neighbor Rule**
  - Choose k out of the N training vectors, identify the k nearest ones to $\underline{x}$
  - Out of these k identify k_i that belong to class ω_i
  - $$\text{Assign } \underline{x} \rightarrow \omega_i : \; k_i > k_j \quad \forall i \neq j$$
  - **The simplest version is : *k=1* !!!**
  - **For large N this is not bad. It can be shown that: if P_B is the optimal Bayesian error probability, then:**

$$P_B \leq P_{NN} \leq P_B\left(2 - \frac{M}{M-1} P_B\right) \leq 2P_B$$

    - **M is the number of classes, and** $k \rightarrow \infty, \; P_{kNN} \rightarrow P_B$

Nonparametric Classification
Decision Trees (DT)

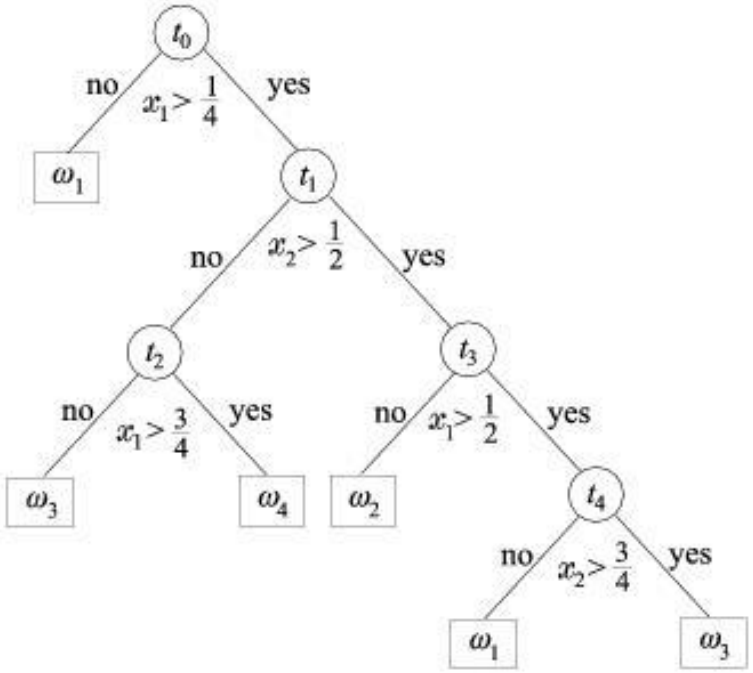
- This is a family of non-linear classifiers. They are **multistage** decision systems, in which classes are **sequentially** rejected, until a finally accepted class is reached. To this end:
  - The feature space is split into unique regions in a sequential manner.
  - Upon the arrival of a feature vector, sequential decisions, assigning features to specific regions, are performed along a path of **nodes** of an appropriately constructed **tree**.
  - The sequence of decisions is applied to **individual** features, and the queries performed in each node are of the **type**:

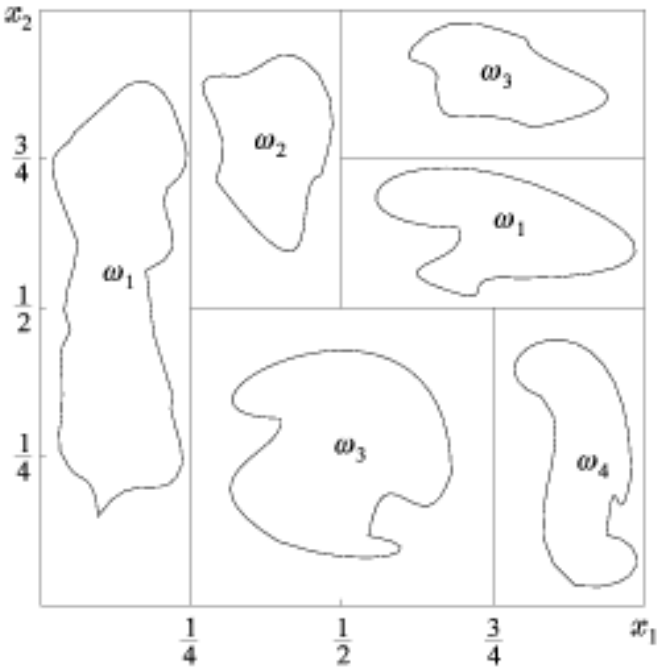
$$is\ Feature\ x_i \leq \alpha$$

  - where α is a pre-chosen (during training) threshold.

Nonparametric Classification
Decision Trees (DT)

- **The figures below are such examples. This type of trees is known as Ordinary Binary Classification Trees (OBCT).**
  - **The decision hyperplanes, splitting the space into regions, are parallel to the axis of the spaces. Other types of partition are also possible, yet less popular.**





Nonparametric Classification
Decision Trees (DT)

- **Design Elements that define a decision tree.**
  - **Each node, t, is associated with a subset $X_t \subseteq X$, where X is the training set. At each node, X_t is split into two (binary splits) disjoint descendant subsets $X_{t,Y}$ and $X_{t,N}$, where**

$$X_{t,Y} \cap X_{t,N} = \emptyset$$
$$X_{t,Y} \cup X_{t,N} = X_t$$

$X_{t,Y}$ is the subset of X_t for which the answer to the query at node t is YES. $X_{t,N}$ is the subset corresponding to NO. The split is decided according to an adopted question (query).

Nonparametric Classification
Decision Trees (DT)

Razi University

- A **splitting** criterion must be adopted for the **best** split of X_t into $X_{t,Y}$ and $X_{t,N}$.
- A **stop-splitting** criterion must be adopted that controls the growth of the tree and a node is declared as **terminal (leaf)**.
- A rule is required that assigns each (terminal) leaf to a class.
- **Set of Questions:** In OBCT trees the set of questions is of the type

$$is \quad x_i \leq \alpha \ ?$$

The choice of the specific x_i and the value of the threshold α, for each node t, are the results of searching, during training, among the features and a set of possible threshold values. The final combination is the one that results to the **best value** of a criterion.

Nonparametric Classification
Decision Trees (DT)

- **Splitting Criterion:** The main idea behind splitting at each node is the resulting descendant subsets $X_{t,Y}$ and $X_{t,N}$ to be more **class homogeneous** compared to X_t. Thus the criterion must be in harmony with such a goal. A commonly used criterion is the **node impurity**:

$$I(t) = -\sum_{i=1}^{M} P(\omega_i \mid t) \log_2 P(\omega_t \mid t)$$

and

$$P(\omega_i \mid t) \approx \frac{N_t^i}{N_t}$$

where N_t^i is the number of data points in X_t that belong to class ω_i. The **decrease in node impurity** is defined as:

$$\Delta I(t) = I(t) - \frac{N_{t,\mathrm{Y}}}{N_t} I(t_{\mathrm{Y}}) - \frac{N_{t,N}}{N_t} I(t_N)$$

Nonparametric Classification
Decision Trees (DT)

- The goal is to choose the parameters in each node (feature and threshold) that result in a split with the highest decrease in impurity.

- Why highest decrease? Observe that the highest value of $I(t)$ is achieved if all classes are equiprobable, i.e., X_t is the least homogenous.

- Stop - splitting rule. Adopt a threshold T and stop splitting a node (i.e., assign it as a leaf), if the impurity decrease is less than T. That is, node t is "pure enough".

- Class Assignment Rule: Assign a leaf to a class ω_j, where:

$$j = \arg\max_i P(\omega_i \mid t)$$

Nonparametric Classification
Decision Trees (DT)

- **Remarks:**
  - **A critical factor in the design is the size of the tree. Usually one grows a tree to a large size and then applies various pruning techniques.**
  - **Decision trees belong to the class of unstable classifiers.**
    - **This can be overcome by a number of "averaging" techniques. Bagging is a popular technique. Using bootstrap techniques in X, various trees are constructed, T_i, $i=1, 2, \ldots, B$. The decision is taken according to a majority voting rule.**

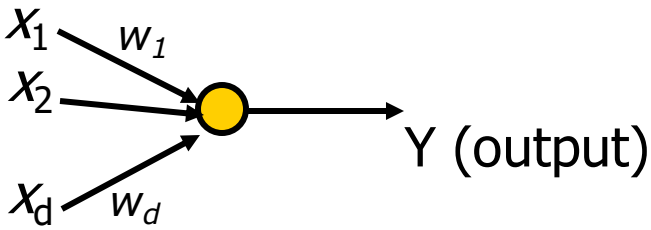
Nonparametric Classification
Artificial Neural Network (ANN)

- **Biological networks achieve excellent recognition performance via dense interconnection of simple computational elements (neurons)**
  - **Number of neurons ≈ $10^{10} - 10^{12}$**
  - **Number of interconnections/neuron ≈ $10^{3} - 10^{4}$**
  - **Total number of interconnections ≈ 10^{14}**

- **Massive parallelism is essential for complex recognition tasks (e.g., speech & image recognition)**
  - **Humans take only a few hundred milliseconds for most cognitive tasks; this suggests parallel computation in human brain**

Nonparametric Classification
Artificial Neural Network (ANN)

- **Nodes in neural networks are simple computational nonlinear elements, typically analog**



$$Y = f(\sum_{i=1}^{d} w_i x_i - \theta)$$

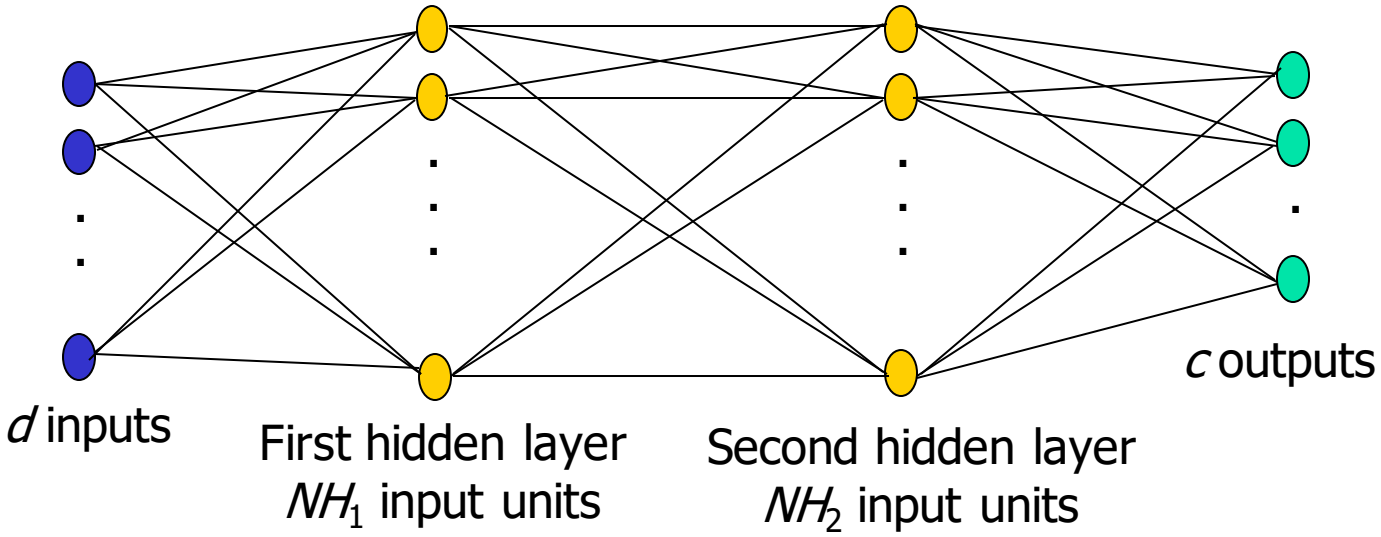
where θ is internal threshold or offset

- **Feed-forward networks with one or more layers (hidden) between input & output nodes**

- **Networks are trained by back-propagation training algorithm**

Nonparametric Classification
Artificial Neural Network (ANN)

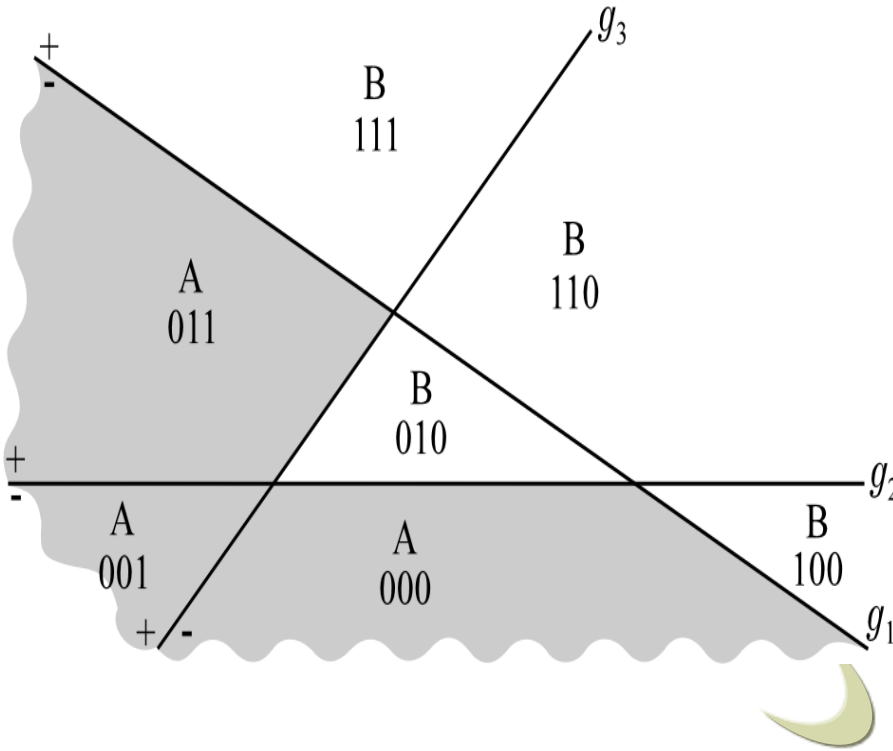
- **A three-layer network can generate arbitrary complex decision regions**



- **This is capable to classify vectors into classes consisting of ANY union of polyhedral regions.**
- **Intersections of these hyperplanes form regions in the l-dimensional space. Each region corresponds to a vertex of the H_p unit hypercube.**

Nonparametric Classification Artificial Neural Network (ANN)

- **The reasoning**
  - **For each vertex, corresponding to class, say A, construct a hyperplane which leaves THIS vertex on one side (+) and ALL the others to the other side (-).**
  - **The output neuron realizes an OR gate**



- **Overall:**

The first layer of the network forms the hyperplanes, the second layer forms the regions and the output neuron forms the classes.

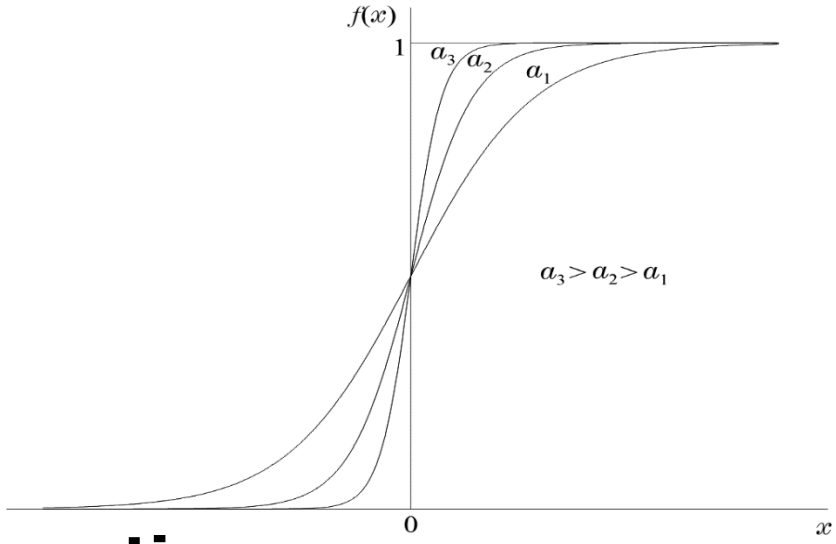
- **The Backpropagation Algorithm**
  - **This is an algorithmic procedure that computes the synaptic weights iteratively, so that an adopted cost function is minimized (optimized)**
  - **In a large number of optimizing procedures, computation of derivatives are involved. Hence, discontinuous activation functions pose a problem, i.e.,**

$$f(x)=\begin{cases}1 & x>0\\ 0 & x<0\end{cases}$$

  - **There is always an escape path!!!**
  - **The logistic/Sigmoid function** $f(x)=\dfrac{1}{1+\exp(-ax)}$

is an example. Other functions are also possible and in some cases more desirable.

Nonparametric Classification
Artificial Neural Network (ANN)



- **Common Alternatives:**
  - **Sigmoid** $f(x)=\frac{1}{1+e^{-x}} \Rightarrow f'(x)=f(x)\left(1-f(x)\right)!$
  - **Hyperbolic** $f(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \Rightarrow f'(x)=1-f(x)^{2}$
  - **Linear** $f(x)=x \Rightarrow f'(x)=1$

Nonparametric Classification
Artificial Neural Network (ANN)

- **The steps:**
  - **Adopt an optimizing cost function, e.g.,**
    - **Least Squares Error**
    - **Relative Entropy**

    between desired responses and actual responses of the network for the available training patterns.

    That is, from now on we have to live with errors. We only try to minimize them, using certain criteria.
  - **Adopt an algorithmic procedure for the optimization of the cost function with respect to the synaptic weights e.g.,**
    - **Gradient descent** $\underline{w}_1^r(\text{new}) = \underline{w}_1^r(\text{old}) + \Delta\underline{w}_1^r \qquad where \quad \Delta\underline{w}_1^r = -\mu\frac{\partial J}{\partial w_1^r}$
    - **Newton's algorithm**
    - **Conjugate gradient**

Nonparametric Classification
Artificial Neural Network (ANN)

- **The Procedure:**
  - Initialize unknown weights randomly with small values.
  - Compute the gradient terms backwards, starting with the weights of the last (3^{rd}) layer and then moving towards the first
  - Update the weights
  - Repeat the procedure until a termination procedure is met

- **Two major philosophies:**
  - **Batch mode**: The gradients of the last layer are computed once ALL training data have appeared to the algorithm, i.e., by summing up all error terms.
  - **Pattern mode**: The gradients are computed every time a new training data pair appears. Thus gradients are based on successive individual errors.

Some Important Points

- **Algorithm Termination:**
  - **Condition on Cost function ($J<\varepsilon$)**
  - **Condition on Gradient of Cost Function ($Grad(J)<\varepsilon$) –Between two successive Epochs**
  - **Small Changes in Weights!**
- **Problem with Local Minimum:**
  - **Problem comes from high complexity of mapping function**
  - **Solving methods:**
    - **Re-Initialization**
    - **Add white noise to data**
    - **Change the cost function**
    - **Data Re-Shuffling (some case!)**

Some Important Points

- **What do, when see no improvement in cost function:**
  - **Add complexity is NOT best and first choice!**
  - **A common case is local minima!**
  - **Add complexity if required!**

- **How to use from our data:**
  - **For large dataset (Holdout Method)**
    - **Training Set:**
      - **Estimation Set**
      - **Validation Set**
    - **Test Set**
  - **For small dataset (Leave-One-Out Method):**
    - **Use N-1 samples for training and 1 excluding for testing (N possible experiments) and choose the best one!**

Some Heuristic Interpretation

- **For large Dataset uses Pattern-Mode**
  - **Problem of Correlated data in Batch-Mode**
- **Use Valuable Training set (Rich-Contents):**
  - **Use Samples with high training Errors**
  - **Use Samples which differ from others**
  - **Random Re-shuffling in each Epoch**
  - **Train hard samples (High Training Error) more**
- **Use Anti-Symmetric (odd) function:**

$$f(x)=\alpha\tanh(\beta x)\xrightarrow[\beta=0.6667]{\alpha=1.7150}\begin{cases} f(1)=-f(-1)=1 \\ f'(0)=\alpha\beta=1.1424 \\ f''(1)=Max f''(x) \end{cases}$$

Some Heuristic Interpretation

- **Normalized output [0 1], [-1 +1]**
  - **Consider a margin [ε,1-ε], [-1+ε , 1-ε]**
  - **Former example: ε=0.7159→[-1 +1] as target!**

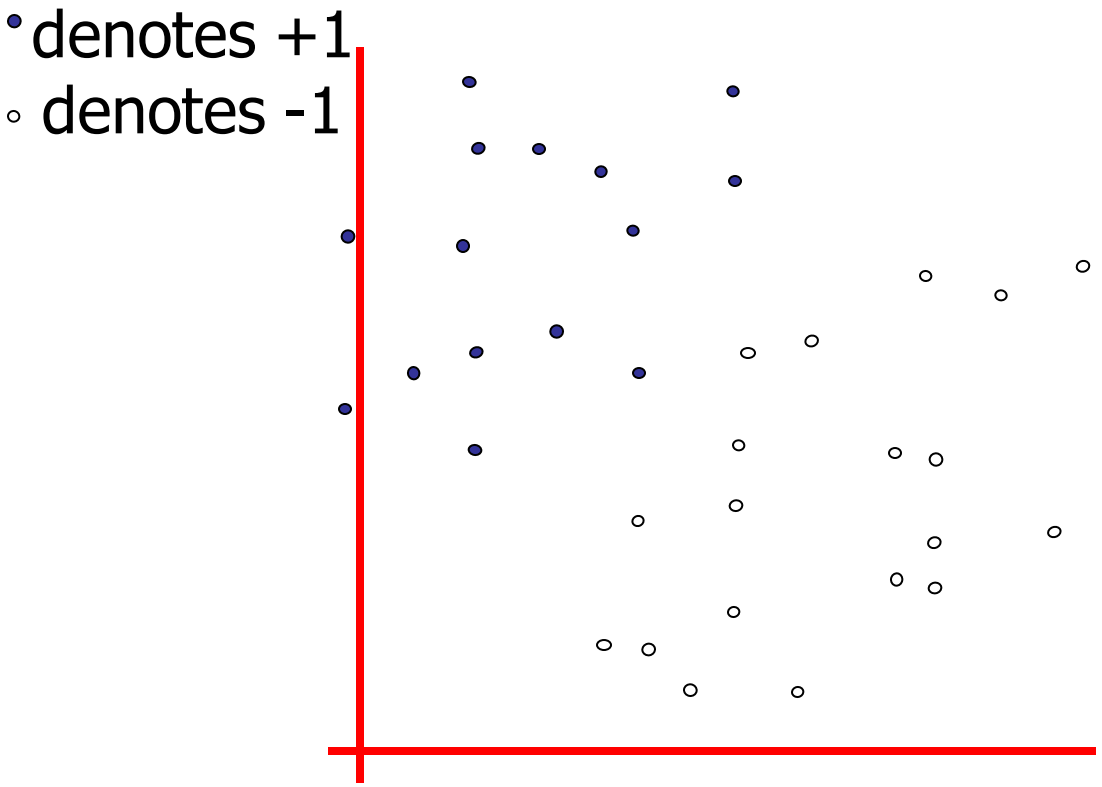
- **Input Normalization, zero mean**
  - **Input Whitening (Uncorrelated data), PCA/KL Transform**
  - **Normalized to unique Covariance value (Same Speed of convergence for each dimension)**
  - **Gram-Schmidt orthogonalization**

- **Initialization**
  - **Medium value for all weights (Problem with large and small values)**
  - **Uniform $(0,\sigma^2)$**

Nonparametric Classification
Support Vector Machine (SVM)

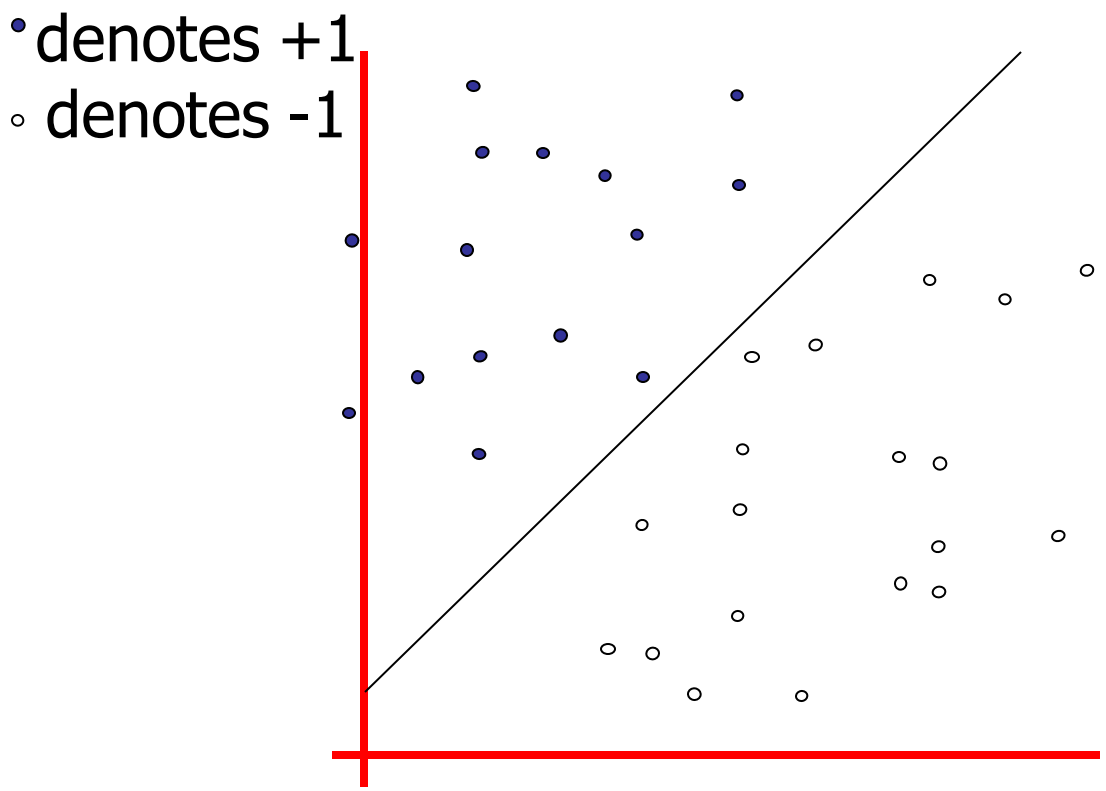
- **Separable Classes**
  - If we have two-class (ω_1, ω_2) linearly separable and $\boldsymbol{x} = \{x_i, i = 1,2, \dots, N\}$ as feature vectors of the training set, then we design linear classifiers as:
$$g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b$$
  - To correctly classifies all the training vectors. Such a hyperplane is not unique. The perceptron algorithm may converge to any one of the possible solutions.
  - Let us now quantify the term *margin* that a hyperplane leaves from both classes.
  - Every hyperplane is characterized by its direction (determined by $\boldsymbol{w}$) and its exact position in space (determined by b). Since we want to give no preference to either of the classes, then it is reasonable for each direction to select that hyperplane which has the same distance from the respective nearest points in ω_1 and ω_2.
  - ***Our goal is to search for the direction that gives the maximum possible margin.***

Nonparametric Classification
Support Vector Machine (SVM)



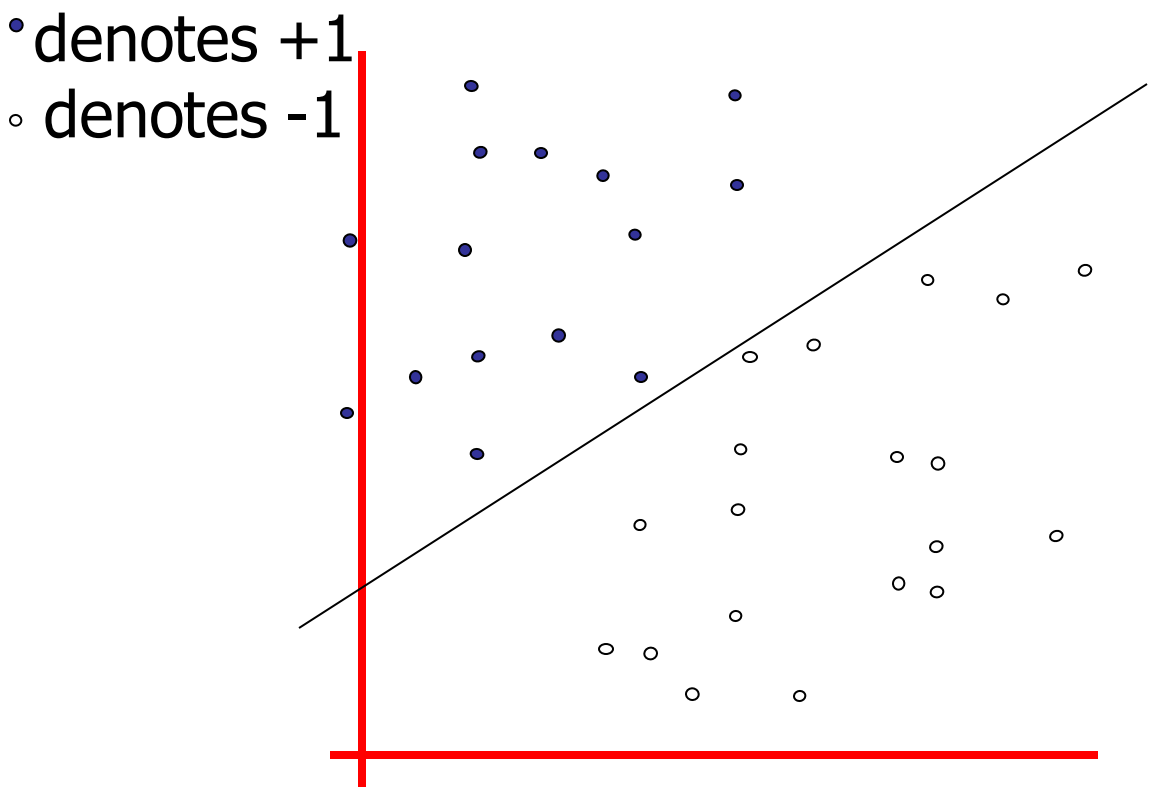
How would you classify this data?

Nonparametric Classification
Support Vector Machine (SVM)



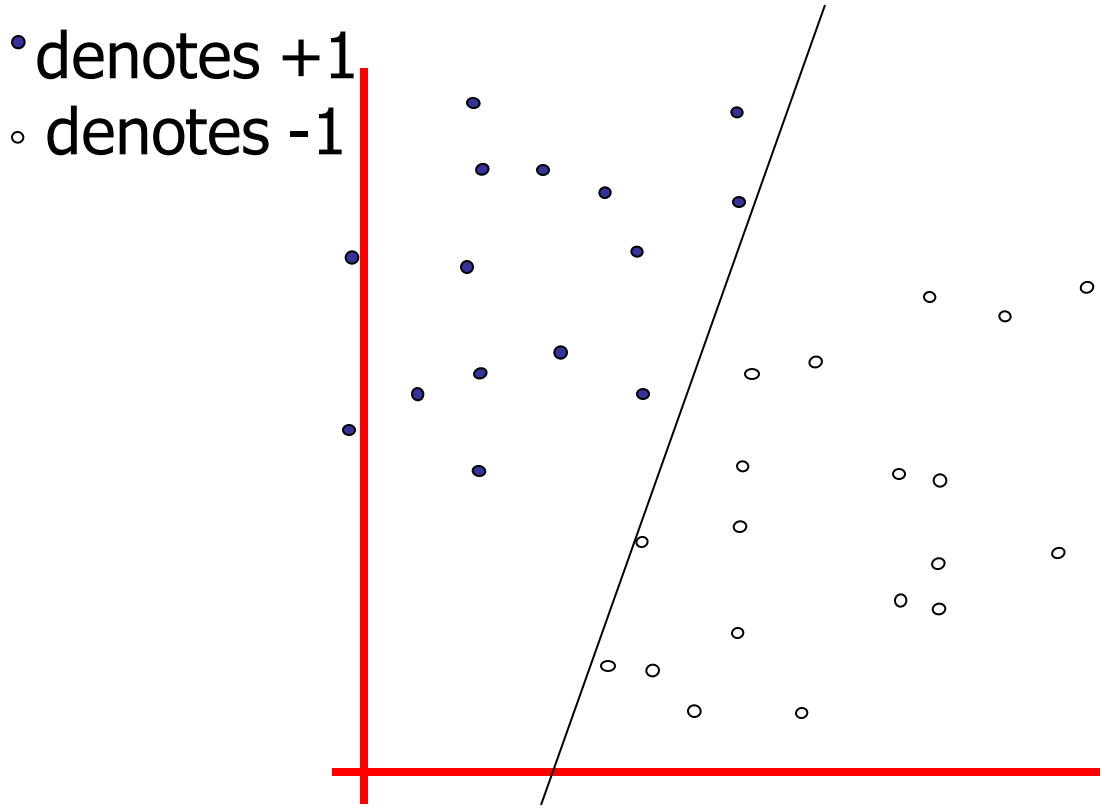
How would you classify this data?

Nonparametric Classification
Support Vector Machine (SVM)



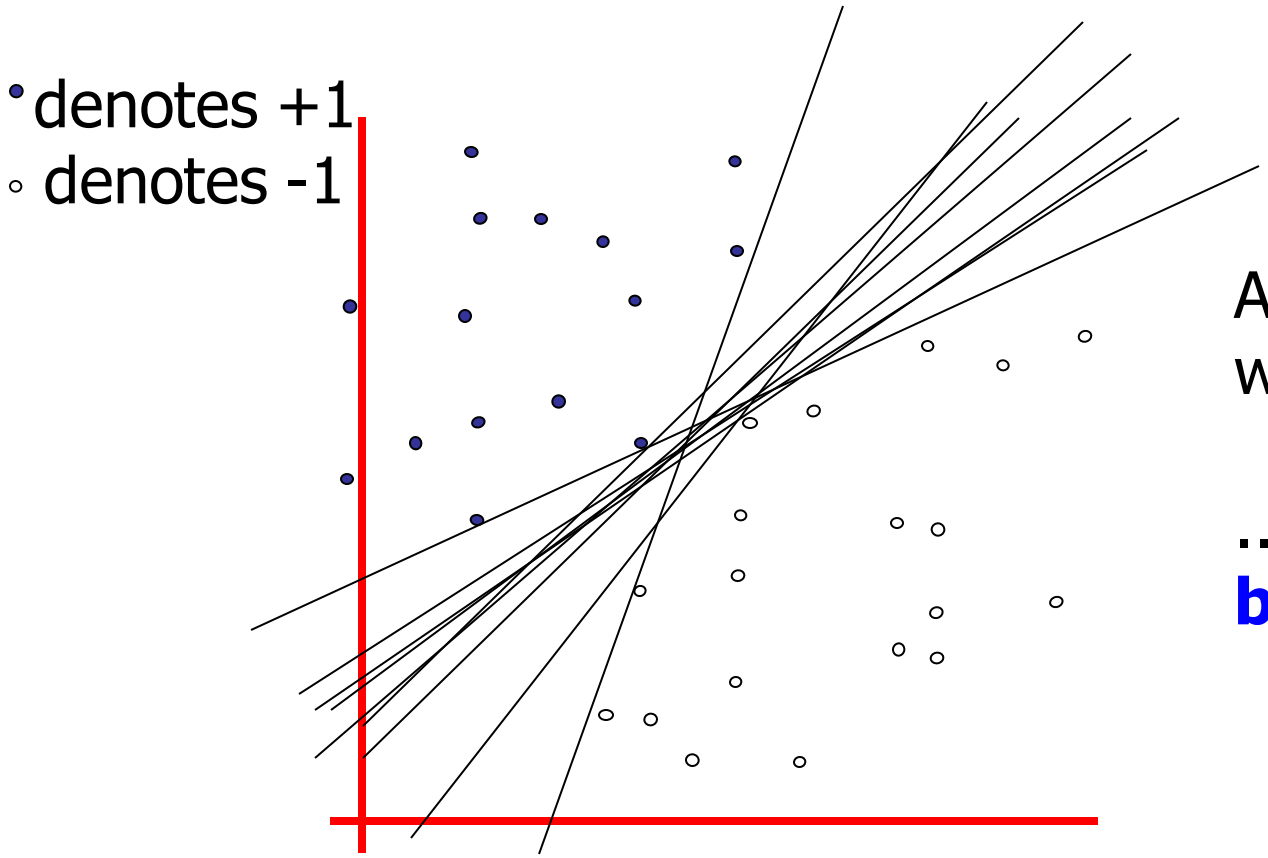
How would you classify this data?

Nonparametric Classification
Support Vector Machine (SVM)



How would you classify this data?

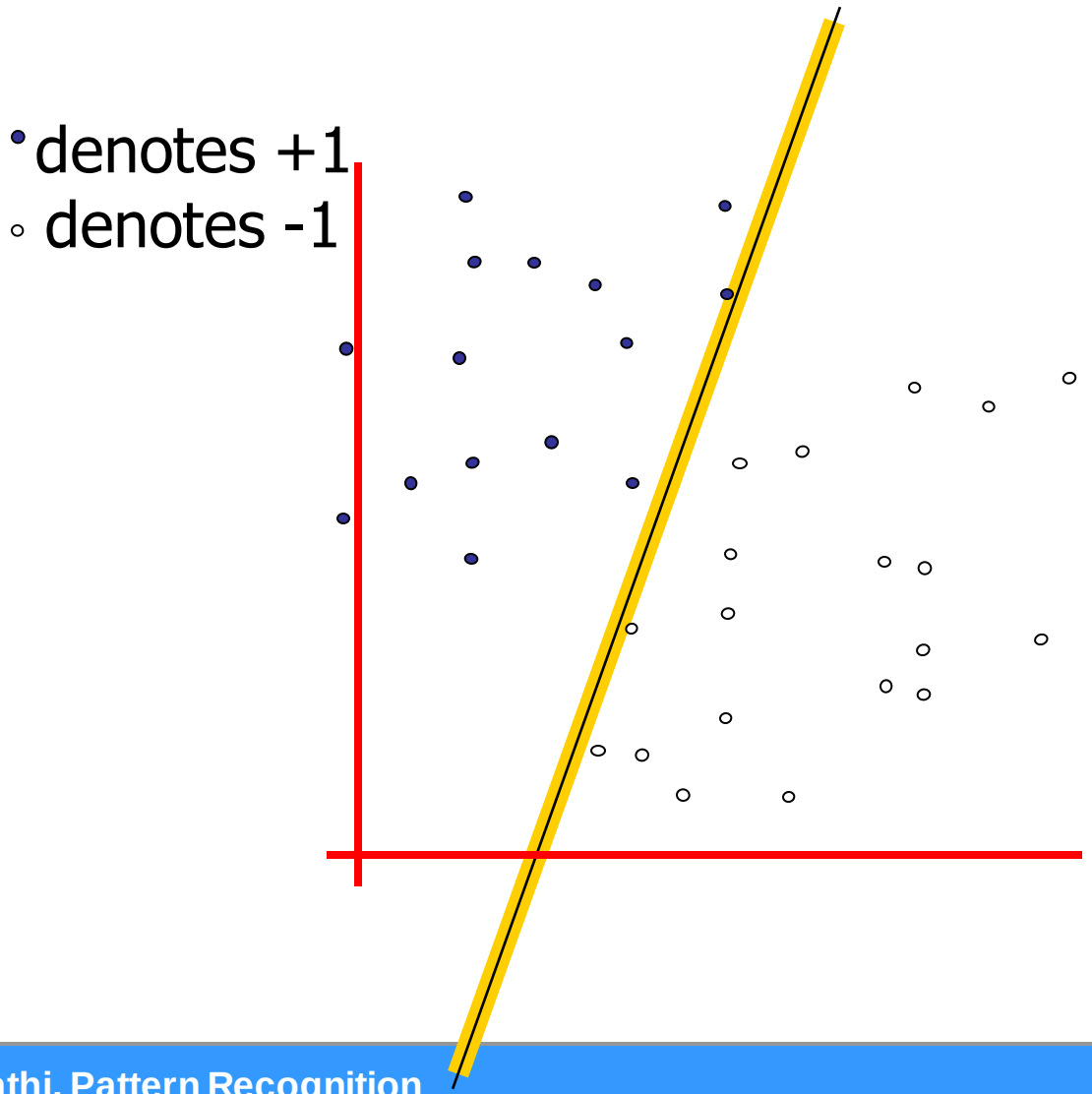
Nonparametric Classification
Support Vector Machine (SVM)



Any of these would be fine..

..but which is **best**?

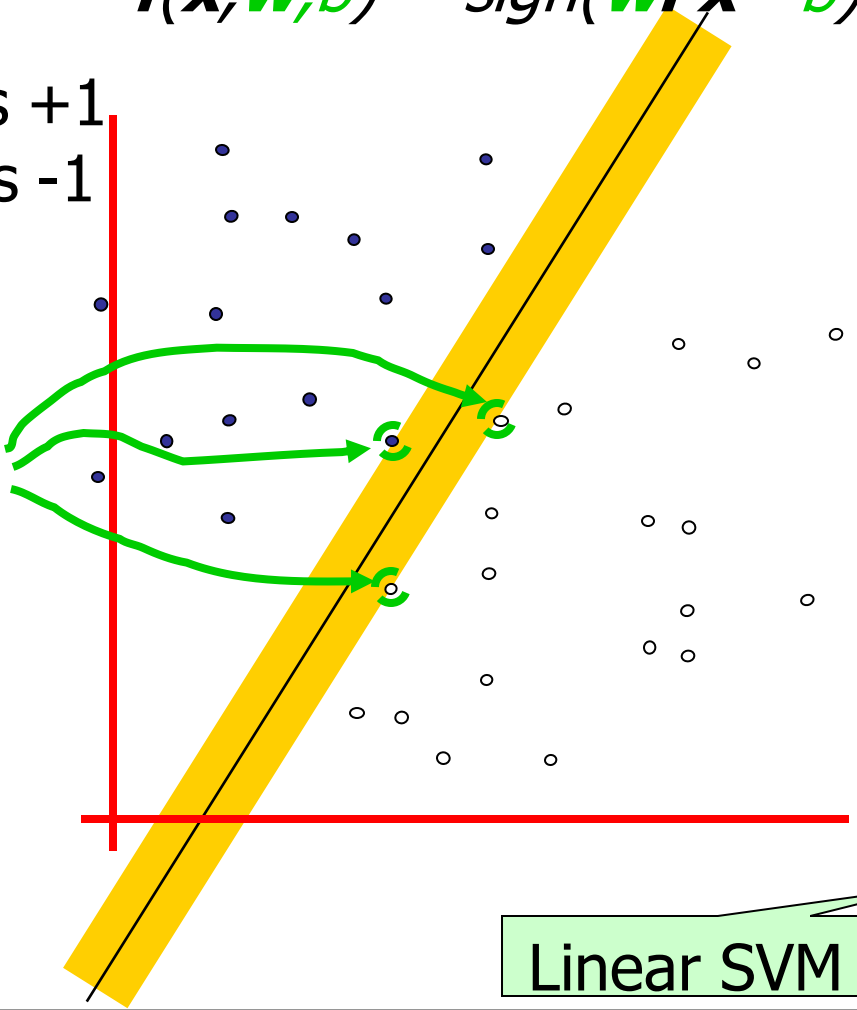
Support Vector Machine (SVM) Classifier Margin



Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Support Vector Machine (SVM) Maximum Margin

$$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}.\ \mathbf{x} - b)$$

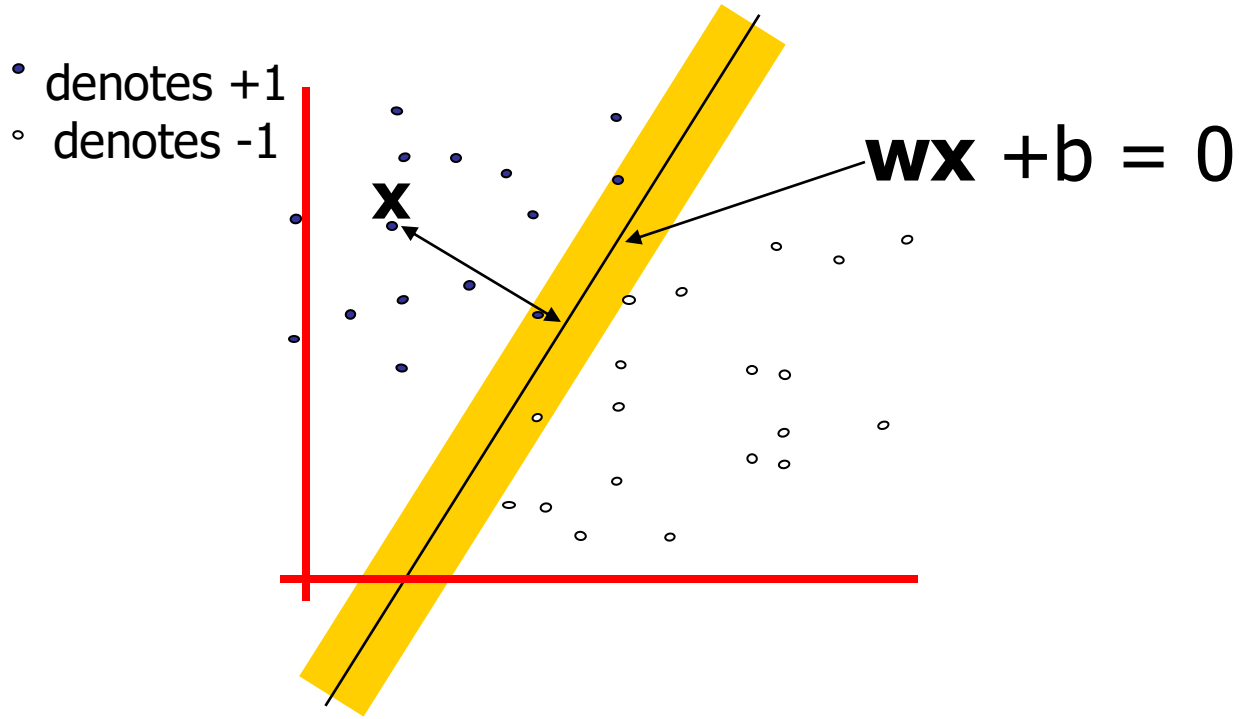


Support Vectors are those datapoints that the margin pushes up against

The maximum margin linear classifier is the linear classifier with the, um, maximum margin. This is the simplest kind of SVM (Called an LSVM)

Linear SVM

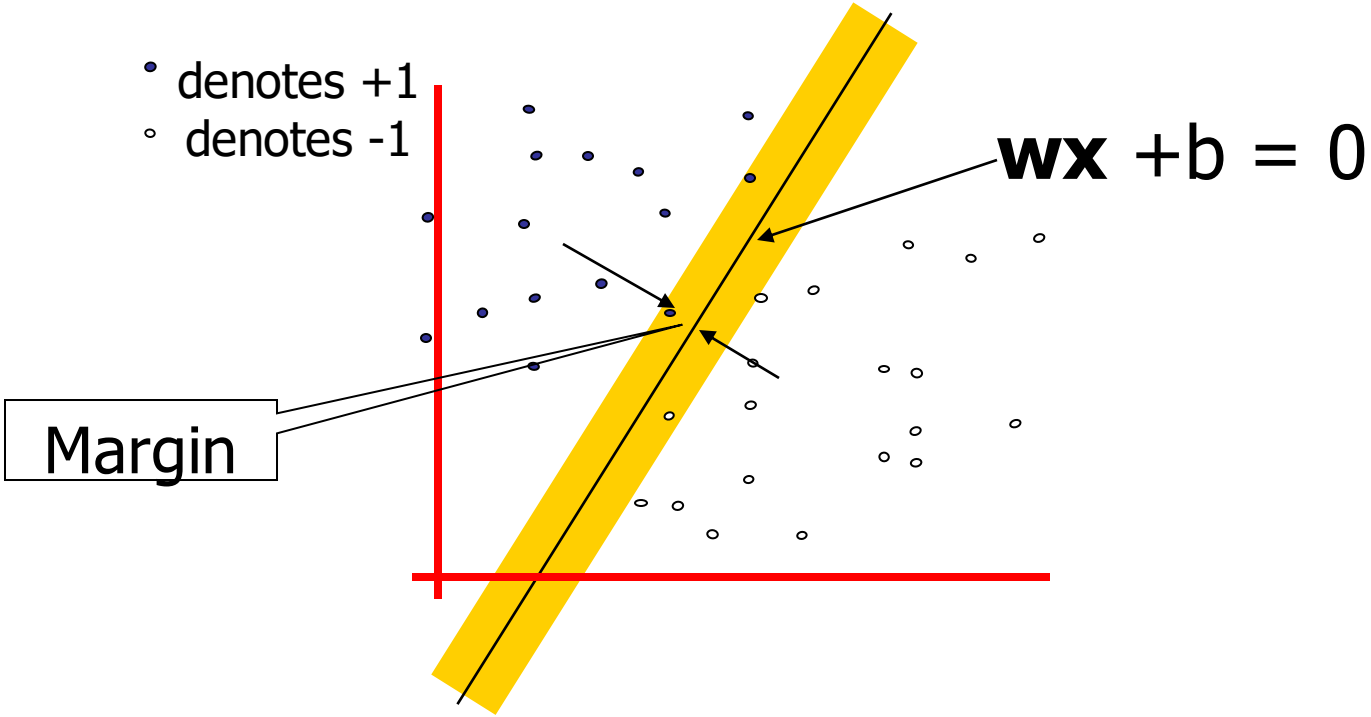
Support Vector Machine (SVM) Estimate the Margin



- **What is the distance expression for a point x to a line wx+b= 0?**

$$d(\mathbf{x}) = \frac{|\mathbf{x}\cdot\mathbf{w}+b|}{\sqrt{\|\mathbf{w}\|_2^2}} = \frac{|\mathbf{x}\cdot\mathbf{w}+b|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

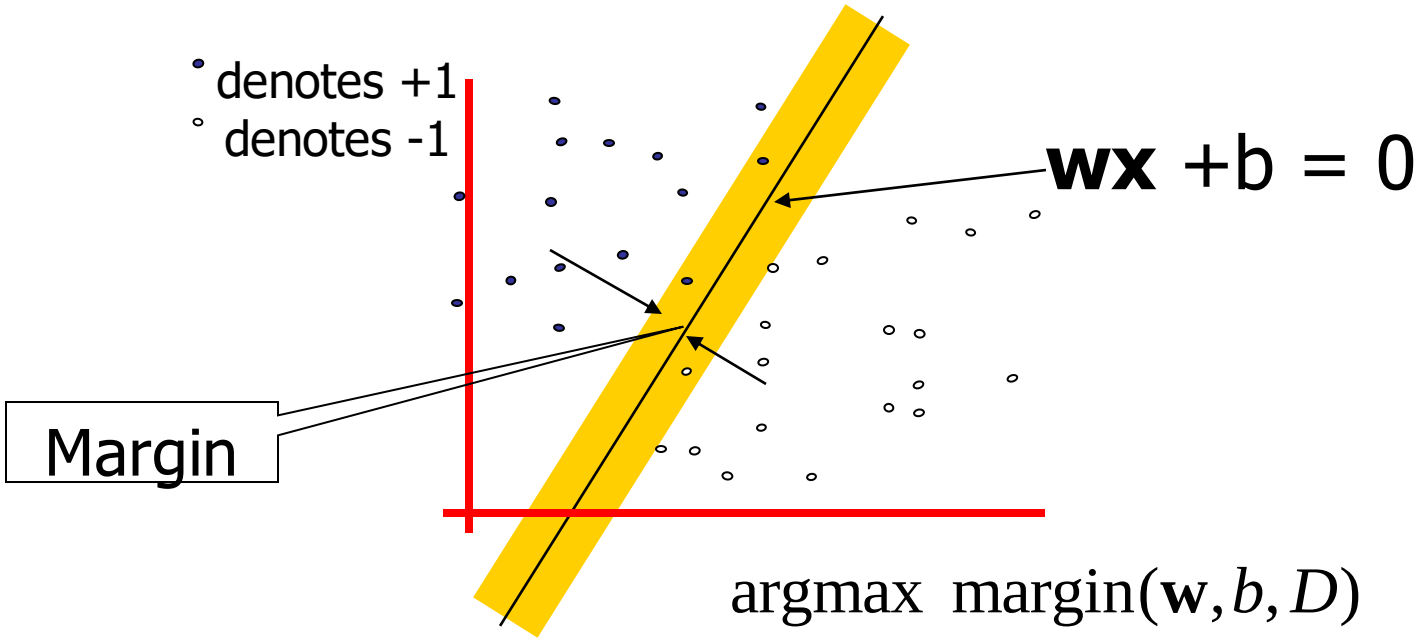
Support Vector Machine (SVM) Estimate the Margin



- **What is the expression for margin?**

$$\text{margin} \equiv \min_{\mathbf{x}\in D} d(\mathbf{x}) = \min_{\mathbf{x}\in D} \frac{\left|\mathbf{x}\cdot\mathbf{w}+b\right|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

Support Vector Machine (SVM) Maximize Margin

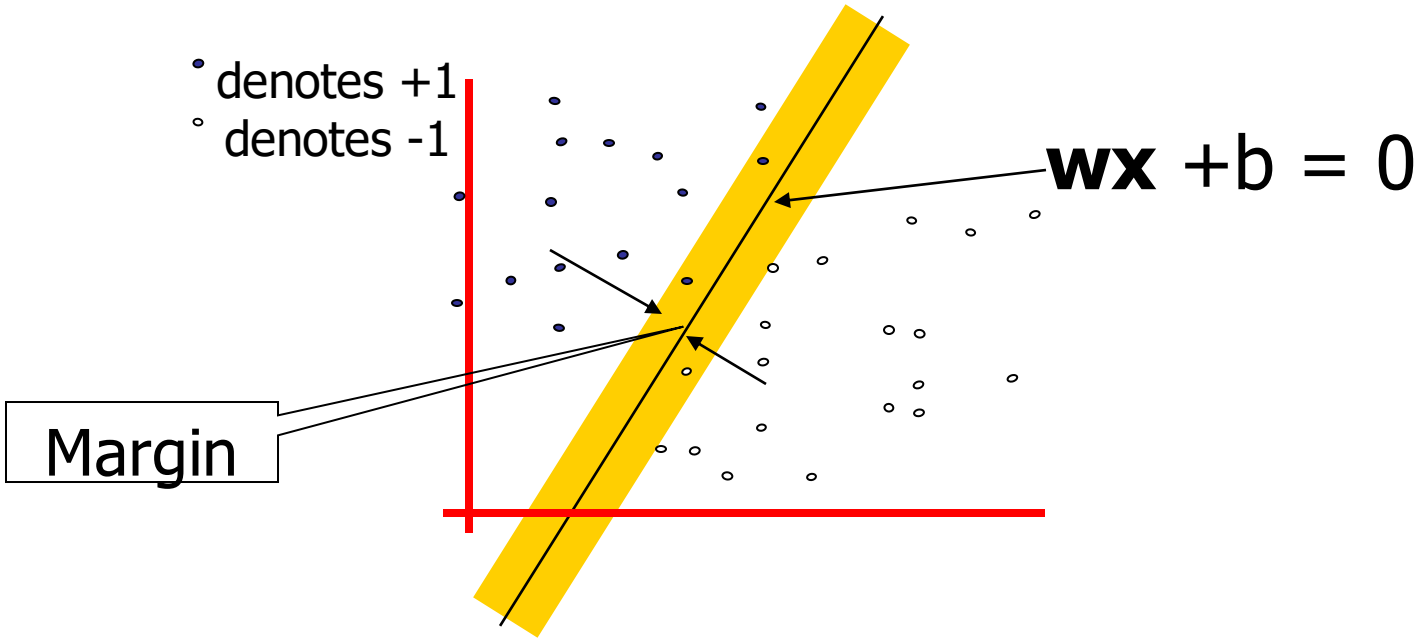


$$\underset{\mathbf{w},b}{\text{argmax}}\ \text{margin}(\mathbf{w},b,D)$$

$$= \underset{\mathbf{w},b}{\text{argmax}} \min_{\mathbf{x}_i \in D} d(\mathbf{x}_i)$$

$$= \underset{\mathbf{w},b}{\text{argmax}} \min_{\mathbf{x}_i \in D} \frac{\left| b + \mathbf{x}_i \cdot \mathbf{w} \right|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

Support Vector Machine (SVM) Maximize Margin

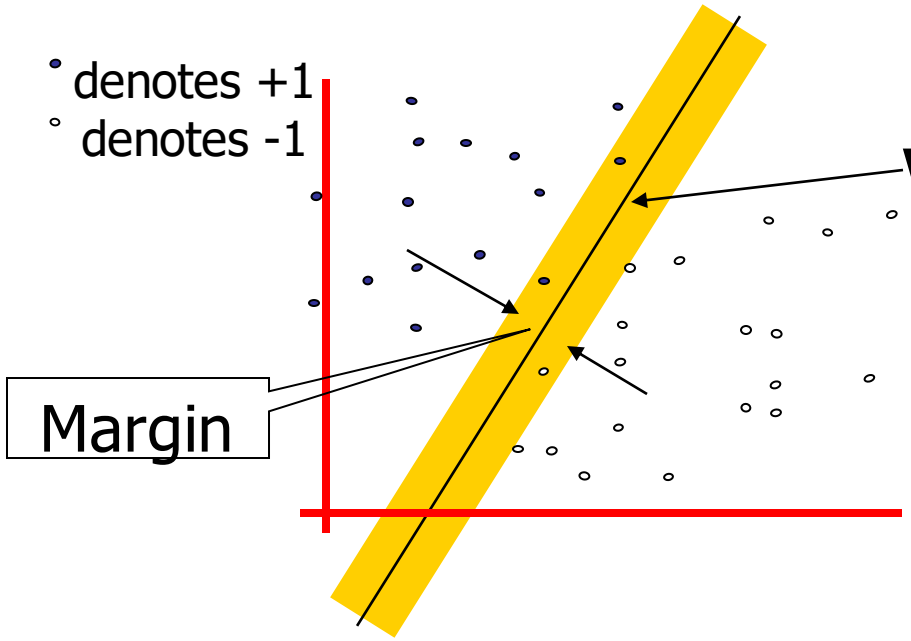


$$\underset{\mathbf{w},b}{\operatorname{argmax}} \min_{\mathbf{x}_i \in D} \frac{\left| b + \mathbf{x}_i \cdot \mathbf{w} \right|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

$$\text{subject to } \forall \mathbf{x}_i \in D : y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b \right) > 0$$

- **Min-max problem → game problem**

Support Vector Machine (SVM) Maximize Margin



$$\underset{\mathbf{w},b}{\operatorname{argmax}} \min_{\mathbf{x}_i \in D} \frac{\left| b + \mathbf{x}_i \cdot \mathbf{w} \right|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

subject to $\forall \mathbf{x}_i \in D : y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b \right) \geq 0$

Strategy:

$$\forall \mathbf{x}_i \in D : \left| b + \mathbf{x}_i \cdot \mathbf{w} \right| \geq 1$$

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{d} w_i^2$$

subject to $\forall \mathbf{x}_i \in D : y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b \right) \geq 1$

Support Vector Machine (SVM) Maximum Margin Linear Classifier

$$\{\vec{w}^*, b^*\} = \underset{\vec{w},b}{\operatorname{argmin}} \sum_{k=1}^{d} w_k^2$$

subject to

$$y_1\left(\vec{w}\cdot\vec{x}_1 + b\right) \geq 1$$

$$y_2\left(\vec{w}\cdot\vec{x}_2 + b\right) \geq 1$$

....

$$y_N\left(\vec{w}\cdot\vec{x}_N + b\right) \geq 1$$

- **quadratic programming is a well-studied class of optimization algorithms can be used to maximize a quadratic function of some real-valued variables subject to linear constraints.**

Support Vector Machine (SVM) Quadratic Programming

Find $$\arg\min_{\mathbf{u}} \; c + \mathbf{d}^T\mathbf{u} + \frac{\mathbf{u}^T R\mathbf{u}}{2}$$ ← Quadratic criterion

Subject to

$$\begin{aligned} a_{11}u_1 + a_{12}u_2 + \ldots + a_{1m}u_m &\le b_1 \\ a_{21}u_1 + a_{22}u_2 + \ldots + a_{2m}u_m &\le b_2 \\ &\vdots \\ a_{n1}u_1 + a_{n2}u_2 + \ldots + a_{nm}u_m &\le b_n \end{aligned}$$

n additional linear inequality constraints

And subject to

$$\begin{aligned} a_{(n+1)1}u_1 + a_{(n+1)2}u_2 + \ldots + a_{(n+1)m}u_m &= b_{(n+1)} \\ a_{(n+2)1}u_1 + a_{(n+2)2}u_2 + \ldots + a_{(n+2)m}u_m &= b_{(n+2)} \\ &\vdots \\ a_{(n+e)1}u_1 + a_{(n+e)2}u_2 + \ldots + a_{(n+e)m}u_m &= b_{(n+e)} \end{aligned}$$

e additional linear equality constraints

Support Vector Machine (SVM) Quadratic Programming

$$\{\vec{w}^*, b^*\} = \min_{\vec{w},b} \sum_i w_i^2$$

subject to $y_i\left(\vec{w}\cdot\vec{x}_i + b\right) \geq 1$ for all training data $(\vec{x}_i, y_i)$

$$\{\vec{w}^*, b^*\} = \operatorname{argmax}_{\vec{w},b} \left\{0 + \vec{0}\cdot\vec{w} - \vec{w}^T \mathbf{I_n} \vec{w}\right\}$$

$$\left.\begin{array}{l} y_1\left(\vec{w}\cdot\vec{x}_1 + b\right) \geq 1 \\ y_2\left(\vec{w}\cdot\vec{x}_2 + b\right) \geq 1 \\ .... \\ y_N\left(\vec{w}\cdot\vec{x}_N + b\right) \geq 1 \end{array}\right\} \text{inequality constraints}$$

Support Vector Machine (SVM) The Dual Form of QP

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2} \sum_{k=1}^{R} \sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\mathbf{x}_k \cdot \mathbf{x}_l)$

Subject to these constraints: $0 \le \alpha_k \le C \quad \forall k$ $\sum_{k=1}^{R} \alpha_k y_k = 0$

Then define:

$$\mathbf{w} = \sum_{k=1}^{R} \alpha_k y_k \mathbf{x}_k$$

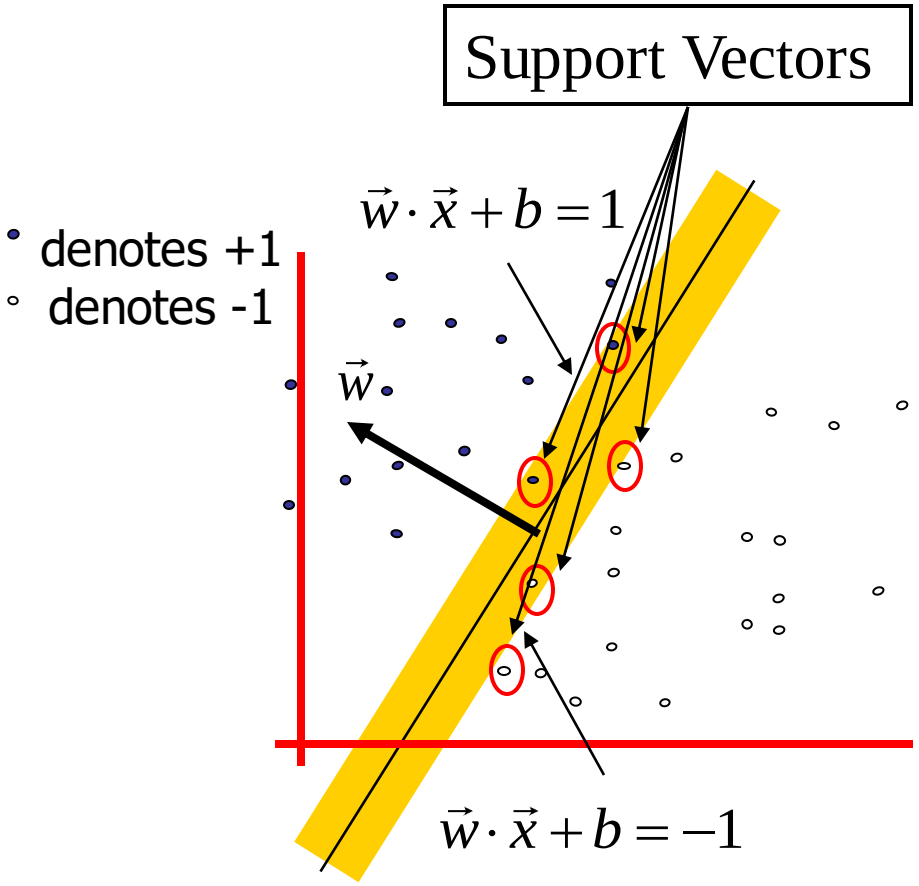
Datapoints with $\alpha_k > 0$ will be the support vectors

..so this sum only needs to be over the support vectors.

Then classify with:

$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w} \cdot \mathbf{x} - b)$

Support Vector Machine (SVM)
Support Vectors



$$\forall i : \alpha_i \left(y_i \left(\vec{w} \cdot \vec{x}_i + b \right) - \left(1 - \varepsilon_i \right) \right) = 0$$

$\alpha_i = 0$ for non-support vectors

$\alpha_i \neq 0$ for support vectors

$$\mathbf{w} = \sum_{k=1}^{R} \alpha_k y_k \mathbf{x}_k$$

Decision boundary is determined only by those support vectors !

Support Vector Machine (SVM) Kernel Functions

- $K(a,b)=(a \cdot b+1)^d$ is an example of an SVM Kernel Function
- Beyond polynomials there are other very high dimensional basis functions that can be made practical by finding the right Kernel Function
  - Radial-Basis-style Kernel Function: $K(\mathbf{a},\mathbf{b}) = \exp\left(-\frac{(\mathbf{a}-\mathbf{b})^2}{2\sigma^2}\right)$
  - Neural-net-style Kernel Function: $K(\mathbf{a},\mathbf{b}) = \tanh(\kappa \mathbf{a}.\mathbf{b} - \delta)$
- Not all functions are kernel functions
  - Need to be decomposable
    - $K(a,b) = \phi(a) \cdot \phi(b)$

Support Vector Machine (SVM) Kernel Tricks

- **Mercer's condition**
  - **To expand Kernel function K(x,y) into a dot product, i.e. $K(x,y)=\Phi(x)\cdot\Phi(y)$, K(x, y) has to be positive semi-definite function, i.e., for any function f(x) whose $\int f^2(x)dx$ is finite, the following inequality holds**

$$\int dxdyf(x)K(x,y)f(y) \geq 0$$

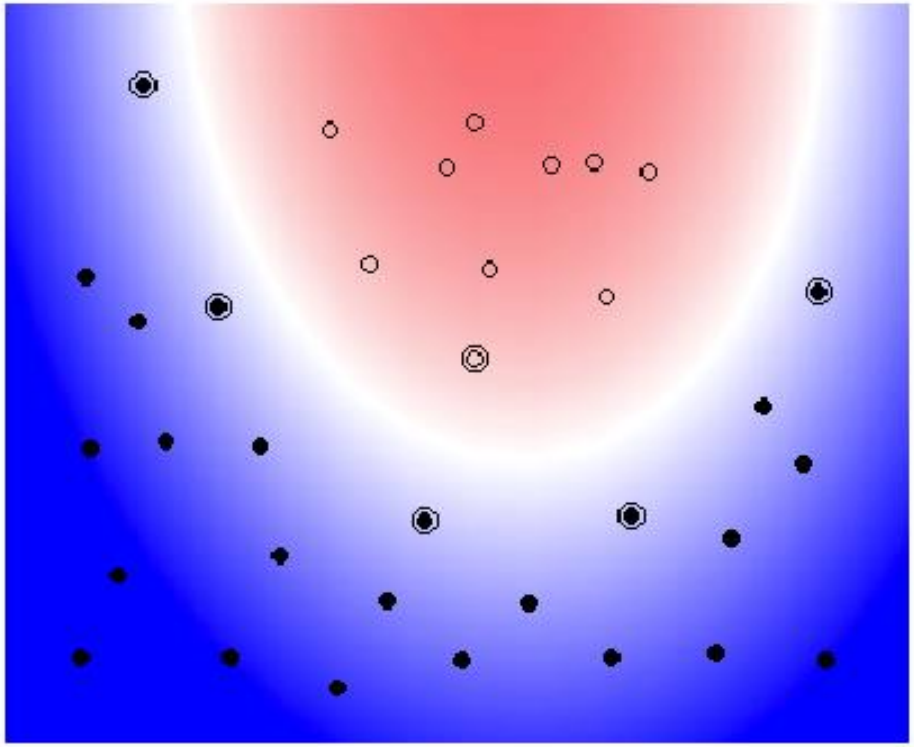
  - **Could $K(a,b) = (a-b)^3$ be a kernel function ?**
  - **Could $K(a,b) = (a-b)^4 - (a+b)^2$ be a kernel function?**
  - **Could $K(\vec{x},\vec{y}) = \left(\sum_i x_i y_i\right)^p$ be a kernel function?**
- **Pro**
  - **Introducing nonlinearity into the model**
  - **Computational cheap**
- **Con**
  - **Still have potential overfitting problems**

Support Vector Machine (SVM) Nonlinear Kernel (I)

Example: SVM with Polynomial of Degree 2

Kernel: $K(\vec{x}_i,\vec{x}_j) = [\vec{x}_i \cdot \vec{x}_j + 1]^2$

plot by Bell SVM applet

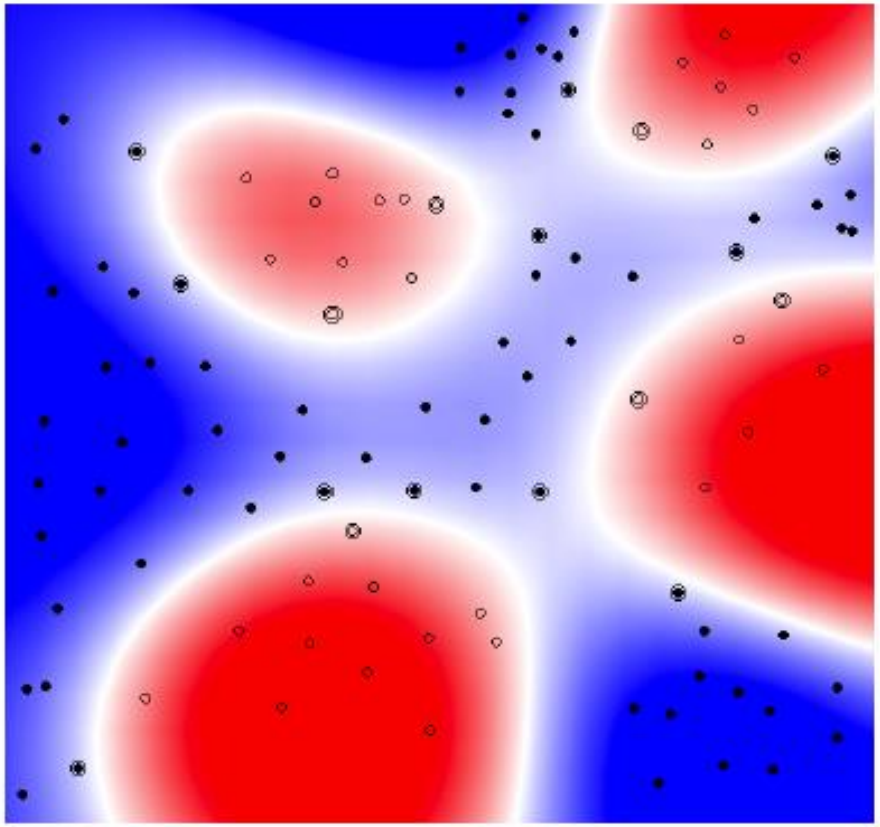


Support Vector Machine (SVM) Nonlinear Kernel (II)

Example: SVM with RBF-Kernel

Kernel: $K(\vec{x}_i,\vec{x}_j) = \exp(-|\vec{x}_i - \vec{x}_j|^2/\sigma^2)$

plot by Bell SVM applet



Support Vector Machine (SVM) Kernelize Logistic Regression

$$p(y \mid \vec{x}) = \frac{1}{1+\exp(-y\vec{x}\cdot\vec{w})}$$

$$l_{reg}(\vec{\alpha}) = \sum_{i=1}^{N} \log \frac{1}{1+\exp(-y\vec{x}\cdot\vec{w})} - c\sum_{k=1}^{N} w_k^2$$

How can we introduce the nonlinearity into the logistic regression?

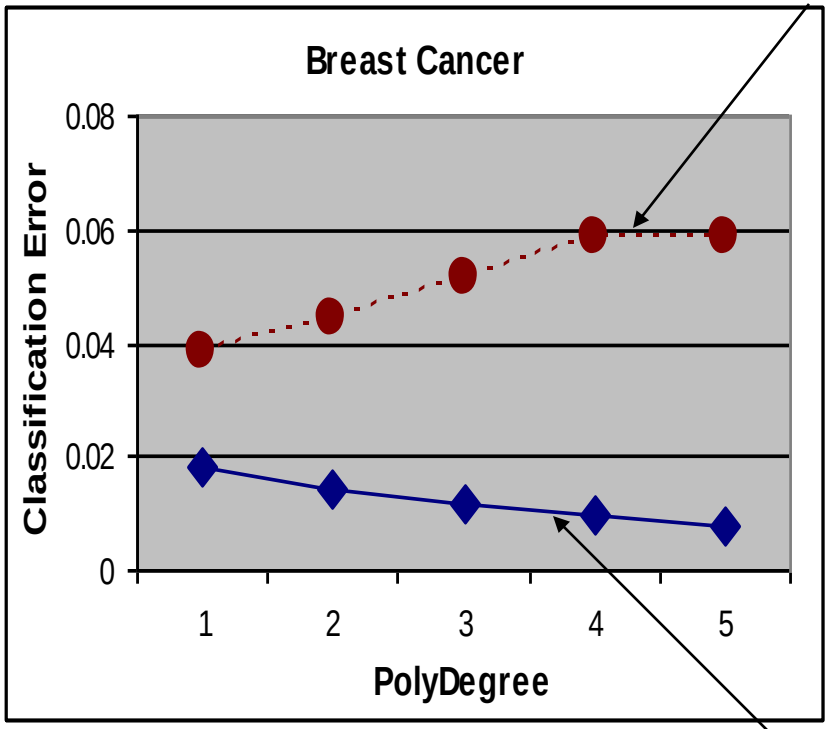
$$\vec{x} \to \vec{\phi}(\vec{x}),\ \vec{w} = \sum_{i=1}^{N} \alpha_i \vec{\phi}(\vec{x}_i) \quad , \quad K(\vec{w},\vec{x}) = \sum_{i=1}^{N} \alpha_i K(\vec{x}_i,\vec{x})$$

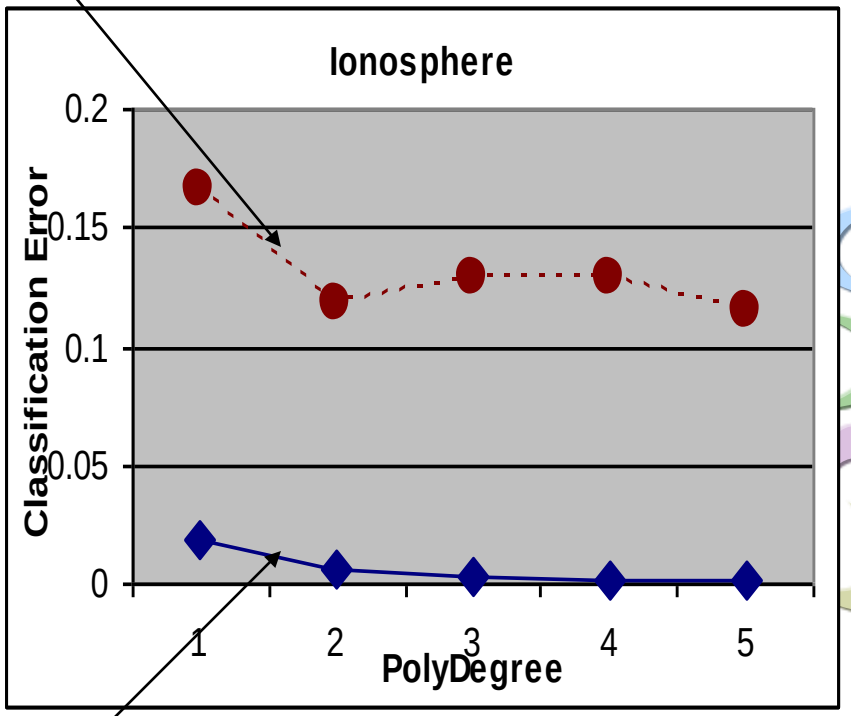
$$p(y \mid \vec{x}) = \frac{1}{1+\exp(-yK(\vec{x},\vec{w}))} = \frac{1}{1+\exp\left(-y\sum_{i=1}^{N} \alpha_i K(\vec{x}_i,\vec{x})\right)}$$

$$l_{reg}(\vec{\alpha}) = \sum_{i=1}^{N} \log \frac{1}{1+\exp\left(-y_i\sum_{j=1}^{N} \alpha_j K(\vec{x}_j,\vec{x}_i)\right)} - c\sum_{i,j=1}^{N} \alpha_i\alpha_j K(\vec{x}_i,\vec{x}_j)$$

Support Vector Machine (SVM)
Overfitting in SVM

Testing Error





Training Error

Support Vector Machine (SVM) Diffusion Kernel

- **Kernel function describes the correlation or similarity between two data points**

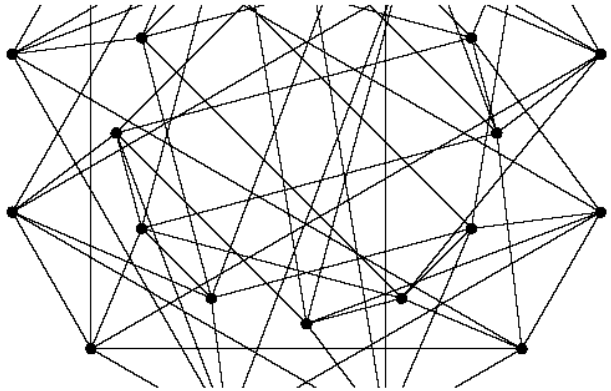
- **Given that I have a function s(x,y) that describes the similarity between two data points. Assume that it is a non-negative and symmetric function. How can we generate a kernel function based on this similarity function?**

- **A graph theory approach …**

Support Vector Machine (SVM) Diffusion Kernel

- **Create a graph for the data points**
  - **Each vertex corresponds to a data point**
  - **The weight of each edge is the similarity s(x,y)**
- **Graph Laplacian**

$$L_{i,j} = \begin{cases} s(x_i, x_j) & i \neq j \\ -\sum_{k \neq i} s(x_i, x_k) & i = j \end{cases}$$



- **Properties of Laplacian**
  - **Negative semi-definite**

Support Vector Machine (SVM) Diffusion Kernel

- **Consider a simple Laplacian**

$$L_{i,j} = \begin{cases} 1 & x_i \text{ and } x_j \text{ are connected} \\ -\sum_{x_k \in N(x_i)} 1 & i = j \end{cases}$$

- **Consider** $L^2, L^4, \ldots$
  - **What do these matrixes represent?**

- **A diffusion kernel**

$$K_\beta = e^{\beta L} = \lim_{n\to\infty} \left(I + \frac{\beta}{n} L\right)^n$$

Support Vector Machine (SVM) Diffusion Kernel: Properties

- **Positive definite**
- **Local relationships L induce global relationships**

$$K_\beta = e^{\beta L}, \text{ or } \frac{d}{d\beta} K_\beta = L K_\beta$$

- **Works for undirected weighted graphs with similarities**

$$s(x_i, x_j) \neq s(x_j, x_i)$$

- **How to compute the diffusion kernel** $e^{\beta L}$

Support Vector Machine (SVM) Computing Diffusion Kernel

- **Singular value decomposition of Laplacian *L***

$$\mathbf{L} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^T = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m) \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_m \end{pmatrix} (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m)^T$$

$$= \sum_{i=1}^{m} \lambda_i \mathbf{u}_i \mathbf{u}_i^T$$

$$\forall i, j : \mathbf{u}_i^T \mathbf{u}_j = \delta_{i,j}$$

- **What is $\mathbf{L}^2$?**

$$\mathbf{L}^2 = \left(\sum_{i=1}^{m} \lambda_i \mathbf{u}_i \mathbf{u}_i^T \right)^2$$

$$= \sum_{i,j=1}^{m} \lambda_i \lambda_j \mathbf{u}_i \mathbf{u}_i^T \mathbf{u}_j \mathbf{u}_j^T = \sum_{i,j=1}^{m} \lambda_i \lambda_j \mathbf{u}_i \mathbf{u}_j^T \delta_{i,j}$$

$$= \sum_{i=1}^{m} \lambda_i^2 \mathbf{u}_i \mathbf{u}_i^T$$

Support Vector Machine (SVM) Computing Diffusion Kernel

- **What about L^n ?** $\mathbf{L}^2 = \sum_{i=1}^{m} \lambda_i^n \mathbf{u}_i \mathbf{u}_i^T$

- **Compute diffusion kernel** $e^{\beta L}$

$$e^{\beta L} = \sum_{n=1}^{\infty} \frac{\beta^n L^n}{n!} = \sum_{n=1}^{\infty} \frac{\beta^n}{n!} \left(\sum_{i=1}^{m} \lambda_i^n \mathbf{u}_i \mathbf{u}_i^T \right)$$

$$= \sum_{i=1}^{m} \mathbf{u}_i \mathbf{u}_i^T \left(\sum_{n=1}^{\infty} \frac{\lambda_i^n \beta^n}{n!} \right) = \sum_{i=1}^{m} \mathbf{u}_i \mathbf{u}_i^T e^{\beta \lambda_i}$$

Support Vector Machine (SVM) Doing multi-class classification

- SVMs can only handle two-class outputs (i.e. a categorical output variable with arity 2).
- What can be done?
- Answer: with output arity N, learn N SVM's
  - SVM 1 learns "Output==1" vs "Output != 1"
  - SVM 2 learns "Output==2" vs "Output != 2"
  - :
  - SVM N learns "Output==N" vs "Output != N"
- Then to predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.

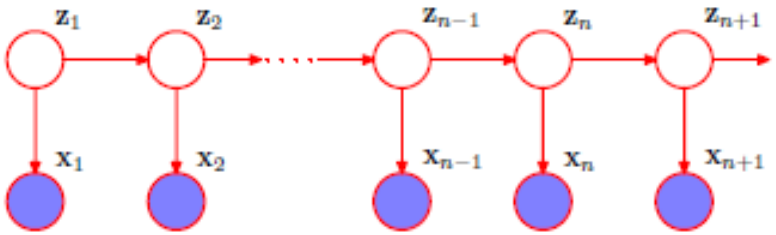
Probabilistic Graphical Models

Markov Models

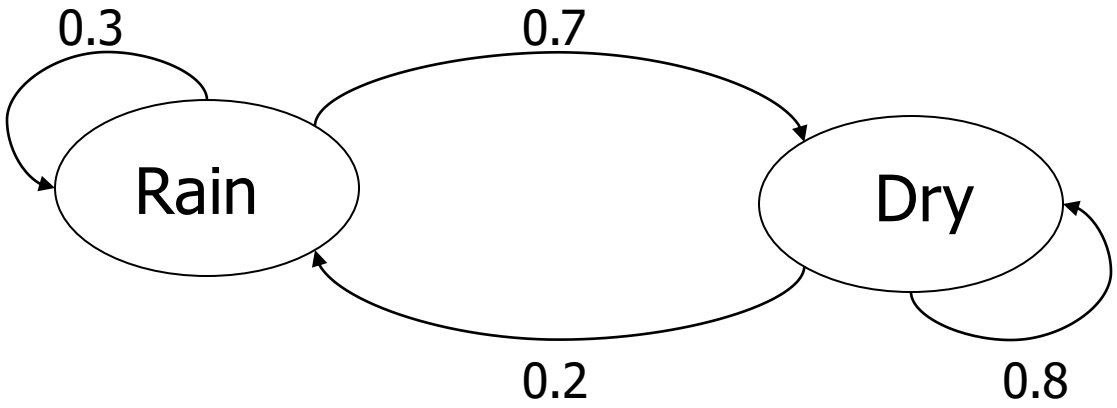
- **Real-world has structures and processes which have (or produce) observable outputs**
  - **Usually sequential (process unfolds over time)**
  - **Cannot see the event producing the output**
    - Example: speech signals
    - Example: Weather forecasting
- **Problem: how to construct a model of the structure or process given only observations?**
  - **Basic theory developed and published in 1960s and 70s**
  - **No widespread understanding and application until late 80s; Why?**
    - Theory published in mathematic journals which were not widely read by practicing engineers
    - Insufficient tutorial material for readers to understand and apply concepts

Markov Models

- **Markov random process**
  - **A random sequence has the Markov property if its distribution is determined solely by its current state. Any random process having this property is called a *Markov random process*.**
- **Markov chain**
  - **For observable state sequences (state is known from data), this leads to a *Markov chain* model.**
- **Hidden Markov Model**
  - **For non-observable states, this leads to a *Hidden Markov Model* (HMM).**
  - **It is widely used in sequential data applications such as speech recognition, handwriting recognition, DNA analysis, and other**



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 **.**

Markov Models

- **Some uses of HMM are:**
- **Speech recognition and processing**
  - Recognizing spoken words and phrases
  - Speech synthesis
  - Many applications in this field
- **Text processing**
  - Parsing raw records into structured records
  - Part-of-Speech Tagging
- **Bioinformatics**
  - Protein sequence prediction
- **Financial**
  - Stock market forecasts (price pattern prediction)
  - Comparison shopping services

Markov Models

- **Set of states:** $\{s_1, s_2, \ldots, s_N\}$
- **Process (i) 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** $\pi_i = P(s_i)$

Markov Models

- **Components**
  - **N – the number of states**
    - Q = {q1; q2; … ;qT} - set of states over time
  - **M – the number of symbols (observables)**
    - O = {o1; o2; … ;oT} - set of symbols over time
  - **A - the state transition probability matrix**
    - aij = P(qt+1 = j|qt = i)
  - **B- observation probability distribution**
    - bj(k) = P(ot = k|qt = j) 1 ≤ k ≤ M
  - π **- the initial state distribution**

- **Full HMM is thus specified as a triplet:**

$$\lambda = (A, B, \pi)$$

Central problems in HMM modelling

- **Problem 1: Evaluation**
  - Probability of occurrence of a particular observation sequence, O = {o1,…,ok}, given the model - P(O|λ)
  - Complicated – hidden states
  - Useful in sequence classification
- **Problem 2: Decoding**
  - Optimal state sequence to produce given observations, O = {o1,…,ok}, given model
  - Optimality criterion
  - Useful in recognition problems
- **Problem 3: Learning**
  - Determine optimum model, given a training set of observations
  - Find λ, such that P(O|λ) is maximal

HMM problem 1: Evaluation

- **Naïve solution :**

- **State sequence is** $Q = q_1, \ldots, q_t$
- **Assume independent observations:**

$$P(o|q,\lambda) = \prod_{t=1}^{T} P(o_t|q_t,\lambda) = b_{q_1}(o_1)b_{q_2}(o_2)\ldots b_{q_T}(o_T)$$

  - **Observations are mutually independent, given the hidden states. (Joint distribution of independent variables factorises into marginal distributions of the independent variables.)**

- **Observe that:** $P(o|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \ldots a_{q_{T-1} q_T}$

- **And finally:** $P(o|\lambda) = \sum_q P(o|q,\lambda)P(q|\lambda)$

HMM problem 1: Evaluation

- **Forward algorithm (Efficient solution):**
- **A Dynamic Programming approach**
- **Define auxiliary forward variable** α: $\boldsymbol{\alpha_t = P(o_1,\ldots, o_t|q_t = i,\lambda)}$
  - $\boldsymbol{\alpha_t(i)}$ is the probability of observing a partial sequence of observables $\boldsymbol{o_1,\ldots, o_t}$ such that at time t, state $\boldsymbol{q_t = i}$
- **Recursive algorithm:**
  - Initialise: $\boldsymbol{\alpha_t(i) = \pi_i b_i(o_1)}$
  - Calculate: $\boldsymbol{\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i) a_{ij}] b_j(o_{t+1})}$
  - Obtain: $\boldsymbol{P(o|\lambda) = \sum_{i=1}^{N} \alpha_r(i)}$

HMM problem 1: Evaluation

- **Backward algorithm (Alternative solution):**

- **Again Dynamic Programming**
- **Define auxiliary forward variable β:**

$$\boldsymbol{\beta_t(i) = P(o_{t+1}, \ldots, o_T | q_t = i, \lambda)}$$

  - $\beta_t(i)$: the probability of observing a sequence of observables $o_{t+1}, \ldots, o_T$ given state $q_t = i$ at time t, and λ
- **Recursive algorithm:**
  - Initialise: $\boldsymbol{\beta_T(j) = 1}$
  - Calculate: $\boldsymbol{\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(i) a_{ij} b_j(o_{t+1})}$
  - Obtain: $\boldsymbol{P(o|\lambda) = \sum_{i=1}^{N} \beta_1(i)}$

HMM problem 2: Decoding

- **Choose state sequence to maximise probability of observation sequence**

- **Viterbi algorithm - inductive algorithm that keeps the best state sequence at each instance**
  - **Utilizes dynamic programming**
  - **State sequence to maximise:** $P(q_1,\ldots, q_t|O,\lambda)$
  - **Define auxiliary variable δ :** $\delta_t(i) = \max_q P(q_1,\ldots, q_t = i, o_1,\ldots, o_t|\lambda)$
    - $\delta_t(i)$:the probability of the most probable path ending in state $q_t = i$

- **We used Recurrent property to solve it:**

$$\delta_{t+1}(j) = \max_i[(\delta_t(i)a_{ij})]b_j(o_{t+1})$$

HMM problem 2: Decoding

- **Algorithm:**
  - To get state sequence, need to keep track of the argument that maximises this, for each t and j. Done via the array $\psi_t(j)$.
  - **1. Initialise:** $\delta_1(i) = \pi_i b_i(o_1)\ , 1 \le i \le N$

    $$\psi_1(i) = o$$
  - **2. Recursion:** for $2 \le t \le T$ and $1 \le j \le N$

    $$\delta_t(j) = \max_{1 \le i \le N} (\delta_{t-1}(i) a_{ij}) b_j(o_t)$$

    $$\psi_t(i) = arg \max_{1 \le i \le N} (\delta_{t-1}(i) a_{ij})$$

  3. Terminate: $P^* = \max_{1 \le i \le N} \delta_T(i)$

  $$q_T^* = arg \max_{1 \le i \le N} \delta_T(i)$$

    - P* gives the state-optimised probability
    - Q* is the optimal state sequence ($Q^* = \{q_1^*, q_2^*, \dots, q_T^*\}$)
  - **4. Backtrack state sequence:** $q_t^* = \psi_{t+1}(q_{t+1}^*);\ t = T-1, T-2, \dots, 1$

HMM problem 3: Learning

- **Training HMM to encode observation sequence such that HMM should identify a similar observation sequence in future**

- **Find** $\lambda = (A, B, \pi)$, **maximising** $P(O|\lambda)$
- **General algorithm:**
  - **Initialise:** λ_0
  - **Compute new model** λ, **using** λ_0 **and observed sequence** O
  - **Then** $\lambda_0 \leftarrow \lambda$
  - **Repeat steps 2 and 3 until:** $\log P(O|\lambda) - \log P(O|\lambda_0) < d$

- **We don't cover the learning algorithms in this course.**

Learning in HMMs: E-M algorithm

- **In order to learn the parameters in an "empty" HMM, we need:**
  - **The topology of the HMM**
  - **Data - the more the better**

- **The learning algorithm is called "Estimate-Maximize" or E-M**
  - **Also called the Forward-Backward algorithm**
  - **Also called the Baum-Welch algorithm**

Combinational Classifiers

Combining Classifiers

- **Just like different features capturing different properties of a pattern, different classifiers also capture different structures and relationships of these patterns in the feature space.**
  - **An empirical comparison of different classifiers can help us choose one of them as the best classifier for the problem at hand.**
  - **However, although most of the classifiers may have similar error rates, sets of patterns misclassified by different classifiers do not necessarily overlap.**
  - **Not relying on a single decision but rather combining the advantages of different classifiers is intuitively promising to improve the overall accuracy of classification.**
- **Such combinations are variously called *combined classifiers*, *ensemble classifiers*, *mixture-of-expert models*, or *pooled classifiers*.**

Combining Classifiers

- **In summary, we may have different feature sets, training sets, classification methods, and training sessions, all resulting in a set of classifiers whose outputs may be combined.**

- **Combination architectures can be grouped as:**
  - ***Parallel*: all classifiers are invoked independently and then their results are combined by a combiner.**
  - ***Serial (cascading)*: individual classifiers are invoked in a linear sequence where the number of possible classes for a given pattern is gradually reduced.**
  - ***Hierarchical (tree)*: individual classifiers are combined into a structure, which is similar to that of a decision tree, where the nodes are associated with the classifiers.**

Combining Classifiers

- **Selecting and training of individual classifiers:**
  - **Combination of classifiers is especially useful if the individual classifiers are largely independent.**
  - **This can be explicitly forced by using different training sets, different features and different classifiers.**
- **classifier combination schemes :**
  - **Majority voting (each classifier makes a binary decision (vote) about each class and the final decision is made in favor of the class with the largest number of votes),**
  - **Sum, product, maximum, minimum and median of the posterior probabilities computed by individual classifiers,**
  - **Class ranking (each class receives m ranks from m classifiers, the highest (minimum) of these ranks is the final score for that class),**
  - **Weighted combination of classifiers.**

Combining Classifiers

- **Combining Classifiers**

The basic philosophy behind the combination of different classifiers lies in the fact that even the "best" classifier fails in some patterns that other classifiers may classify correctly. Combining classifiers aims at exploiting this complementary information residing in the various classifiers.

Thus, one designs different optimal classifiers and then combines the results with a specific rule.

- **Assume that each of the, say, L designed classifiers provides at its output the posterior probabilities:**

$$P(\omega_i \mid \underline{x}), i = 1, 2, \ldots, M$$

Combining Classifiers
Majority Voting Rule

- **Majority Voting Rule:** **Assign $\underline{x}$ to the class for which there is a consensus or when at least ℓ_c of the classifiers agree on the class label of $\underline{x}$ where:**

$$\ell_c = \begin{cases} \frac{L}{2}+1, \text{ L even} \\ \frac{L+1}{2}, \text{ L odd} \end{cases}$$

otherwise the decision is rejection, that is no decision is taken.

Thus, correct decision is made if the majority of the classifiers agree on the correct label, and wrong decision if the majority agrees in the wrong label.

Combining Classifiers
Product and Sum Rule

- **Product Rule: Assign $\underline{x}$ to the class ω_i :**

$$i = \arg\max_k \prod_{j=1}^{L} P_j(\omega_k \mid \underline{x})$$

where $P_j(\omega_k \mid \underline{x})$ is the respective posterior probability of the j^{th} classifier.

- **Sum Rule: Assign $\underline{x}$ to the class ω_i :**

$$i = \arg\max_k \sum_{j=1}^{L} P_j(\omega_k \mid \underline{x})$$

Combining Classifiers
Dependency

- **Dependent or not Dependent classifiers?**
  - **Although there are not general theoretical results, experimental evidence has shown that the more independent in their decision the classifiers are, the higher the expectation should be for obtaining improved results after combination. However, there is no guarantee that combining classifiers results in better performance compared to the "best" one among the classifiers.**
- **Towards Independence: A number of Scenarios.**
  - **Train the individual classifiers using different training data points. To this end, choose among a number of possibilities:**
    - **Bootstrapping: This is a popular technique to combine unstable classifiers such as decision trees (Bagging belongs to this category of combination).**
    - **Stacking: Train the combiner with data points that have been excluded from the set used to train the individual classifiers.**
    - **Use different subspaces to train individual classifiers: According to the method, each individual classifier operates in a different feature subspace. That is, use different features for each classifier.**

Combining Classifiers
Bagging

- ***Bagging (bootstrap aggregating)*** **uses multiple versions of the training set, each created by bootstrapping the original training data.**
  - **Each of these bootstrap data sets is used to train a different component classifier.**
  - **The final classification decision is based on the vote of each component classifier.**
    - **Traditionally, the component classifiers are of the same general form (e.g., all neural networks, all decision trees, etc.) where their differences are in the final parameter values due to their different sets of training patterns.**
      - **A classifier/learning algorithm is informally called unstable if small changes in the training data lead to significantly different classifiers and relatively large changes in accuracy (like decision trees and neural networks).**
  - **In general, bagging improves recognition for unstable classifiers.**

Combining Classifiers
boosting

- **In boosting, each training pattern receives a weight that determines its probability of being selected for the training set for an individual component classifier.**
  - **Adopt a weak classifier known as the base classifier.**
    - **Employing the base classifier, design a series of classifiers, in a hierarchical fashion, each time employing a different weighting of the training samples. Emphasis in the weighting is given on the hardest samples, i.e., the ones that keep "failing".**
  - **If a training pattern is accurately classified, its chance of being used again in a subsequent component classifier is reduced.**
  - **Conversely, if the pattern is not accurately classified, its chance of being used again is increased.**
  - **The final classification decision is based on the weighted sum of the votes of the component classifiers where the weight for each classifier is a function of its accuracy.**

Combining Classifiers
AdaBoost

- **The popular *AdaBoost (adaptive boosting)* algorithm allows continuous adding of classifiers until desired low training error has been achieved.**
  - **Let $a^t(x_i)$ denote the weight of pattern x_i at trial t, where $a^1(x_i) = 1/n$ for every x_i.**
  - **At each trial $t = 1, \ldots, T$, a classifier C^t is constructed from the given patterns under the distribution at where $a^t(x_i)$ reflects occurrence probability of x_i.**
    - The error ε^t of this classifier is also measured with respect to the weights, and consists of the sum of the weights of the patterns that it misclassifies.
      - **If ε^t is greater than 0.5, the trials are terminated and T is set to $t-1$.**
      - **Conversely, if C^t correctly classifies all patterns so that ε^t is zero, the trials are terminated and T becomes t.**
      - **Otherwise, the weights a^{t+1} for the next trial are generated by multiplying the weights of patterns that C^t classifies correctly by the factor $\beta^t = \varepsilon^t/(1-\varepsilon^t)$ and then are renormalized so that $\sum_{i=1}^{n} a^t(x_i) = 1$**
  - **The boosted classifier C^* is obtained by summing the votes of the classifiers C^1,..., C^T , where the vote for classifier C^t is also weighted by $log(1/\beta^t)$).**

Combining Classifiers
AdaBoost

- **Provided that ε^t is always less than 0.5, it was shown that the error rate of C^* on the given patterns under the original uniform distribution a^1 approaches zero exponentially quickly as T increases.**

- **A succession of weak classifiers $\{C^t\}$ can thus be boosted to a strong classifier C^* that is at least as accurate as, and usually much more accurate than, the best weak classifier on the training data.**

- **However, note that there is no guarantee of the generalization performance of a bagged or boosted classifier on unseen patterns.**

Clustering Methods

What is Cluster Analysis?

- Cluster: A collection of data objects
  - similar (or related) to one another within the same group
  - dissimilar (or unrelated) to the objects in other groups
- Cluster analysis (or *clustering*, *data segmentation*, …)
  - Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters
- Unsupervised learning: no predefined classes (i.e., *learning by observations* vs. learning by examples: supervised)
- Typical applications
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms

Applications of Cluster Analysis

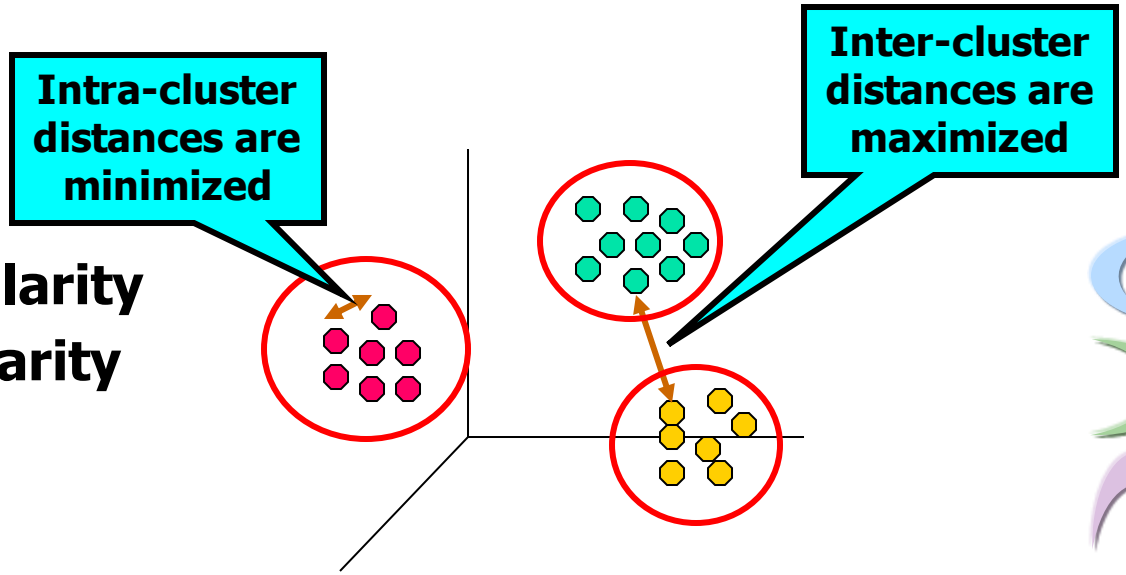
- Data reduction
  - Summarization: Preprocessing for regression, PCA, classification, and association analysis
  - Compression: Image processing(vector quantization)
- Hypothesis generation and testing
- Prediction based on groups
  - Cluster & find characteristics/patterns for each group
- Finding K-nearest Neighbors
  - Localizing search to one or a small number of clusters
- Outlier detection: Outliers are often viewed as those “far away” from any cluster

Clustering: Application Examples

- **Biology:** taxonomy of living things: kingdom, phylum, class, order, family, genus and species
- **Information retrieval:** document clustering
- **Land use:** Identification of areas of similar land use in an earth observation database
- **Marketing:** Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- **City-planning:** Identifying groups of houses according to their house type, value, and geographical location
- **Earth-quake studies:** Observed earth quake epicenters should be clustered along continent faults
- **Climate:** understanding earth climate, find patterns of atmospheric and ocean
- **Economic Science:** market resarch

Quality: What Is Good Clustering?

- A good clustering method will produce high quality clusters with
  - high intra-class similarity
  - low inter-class similarity



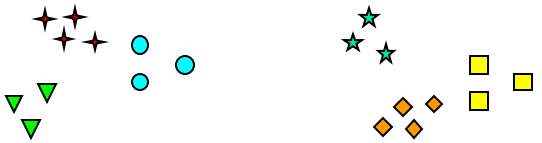
- The quality of a clustering result depends on both the similarity measure used by the method and its implementation
- The quality of a clustering method is also measured by its ability to discover some or all of the hidden patterns

Measure the Quality of Clustering

- **Dissimilarity/Similarity metric**: Similarity is expressed in terms of a distance function, typically metric: d(i, j)
- There is a separate "quality" function that measures the "goodness" of a cluster.
- The definitions of **distance functions** are **usually very different** for interval-scaled, boolean, categorical, ordinal ratio, and vector variables.
- **Weights** should be associated with different variables based on applications and data semantics.
- It is hard to define "**similar enough**" or "**good enough**"
  - the answer is typically highly subjective.

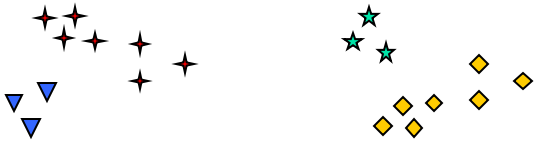
Notion of a Cluster can be Ambiguous

How many clusters?



Six Clusters

Two Clusters



Four Clusters

Basic Steps to Develop a Clustering Task

- **Feature selection**
  - Select info concerning the task of interest
  - Minimal information redundancy
- **Proximity measure**
  - Similarity of two feature vectors
- **Clustering criterion**
  - Expressed via a cost function or some rules
- **Clustering algorithms**
  - Choice of algorithms
- **Validation of the results**
  - Validation test (also, *clustering tendency* test)
- **Interpretation of the results**
  - Integration with applications

Attribute Types

- **Nominal:** categories, states, or "names of things"
  - *Hair_color* = {*auburn, black, blond, brown, grey, red, white*}
  - marital status, occupation, ID numbers, zip codes
- **Binary**
  - Nominal attribute with only 2 states (0 and 1)
  - Symmetric binary: both outcomes equally important
    - e.g., gender
  - Asymmetric binary: outcomes not equally important.
    - e.g., medical test (positive vs. negative)
    - Convention: assign 1 to most important outcome (e.g., HIV positive)
- **Ordinal**
  - Values have a meaningful order (ranking) but magnitude between successive values is not known.
  - *Size* = {*small, medium, large*}, grades, army rankings
- **Numeric**
  - Values have a meaningful order (ranking) and magnitude between successive values is known.

Similarity and Dissimilarity

- **Similarity**
  - Numerical measure of how alike two data objects are
  - Value is higher when objects are more alike
  - Often falls in the range [0,1]

- **Dissimilarity (e.g., distance)**
  - Numerical measure of how different two data objects are
  - Lower when objects are more alike
  - Minimum dissimilarity is often 0
  - Upper limit varies

- **Proximity refers to a similarity or dissimilarity**

Data Matrix and Dissimilarity Matrix

- **Data matrix**
  - n data points with p dimensions
  - Two modes

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}$$

- **Dissimilarity matrix**
  - n data points, but registers only the distance
  - A triangular matrix
  - Single mode

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

Type of data in clustering analysis
Interval-valued variables

- **Standardize data**
  - **Calculate the mean absolute deviation:**

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + \ldots + |x_{nf} - m_f|)$$

where $$m_f = \frac{1}{n}(x_{1f} + x_{2f} + \ldots + x_{nf})$$

  - **Calculate the standardized measurement (*z-score*)**

$$z_{if} = \frac{x_{if} - m_f}{s_f}$$

- **Using mean absolute deviation is more robust than using standard deviation**

Similarity and Dissimilarity Between Objects

- **Distances** are normally used to measure the **similarity** or **dissimilarity** between two data objects
- **Some popular ones include: *Minkowski distance***

$$d(i,j)=\sqrt[q]{(|x_{i_1}-x_{j_1}|^q+|x_{i_2}-x_{j_2}|^q+...+|x_{i_p}-x_{j_p}|^q)}$$

where $i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ and $j = (x_{j1}, x_{j2}, \ldots, x_{jp})$ are two p-dimensional data objects, and q is a positive integer

- **If $q = 1$, d is Manhattan distance**

$$d(i,j)=|x_{i_1}-x_{j_1}|+|x_{i_2}-x_{j_2}|+...+|x_{i_p}-x_{j_p}|$$

Similarity and Dissimilarity Between Objects

- ***If q = 2, d* is Euclidean distance**

$$d(i,j)=\sqrt{(|x_{i_1}-x_{j_1}|^2+|x_{i_2}-x_{j_2}|^2+...+|x_{i_p}-x_{j_p}|^2)}$$

  - **Properties**
    - $d(i,j) \geq 0$
    - $d(i,i) = 0$
    - $d(i,j) = d(j,i)$
    - $d(i,j) \leq d(i,k) + d(k,j)$

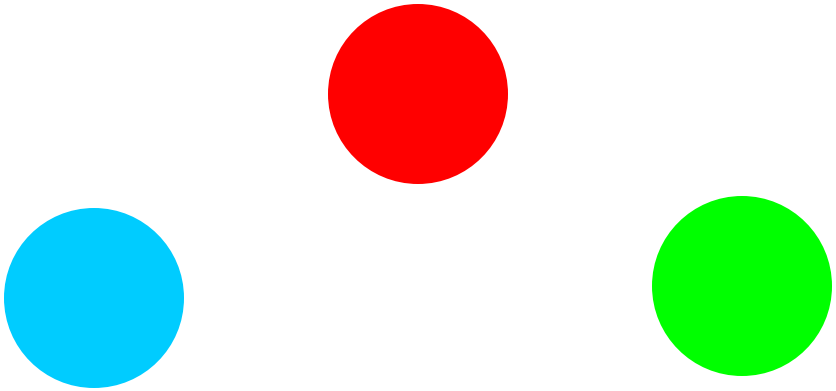
- **Also, one can use weighted distance, parametric Pearson product, moment correlation, or other dissimilarity measures**

Types of Clusters

- Well-separated clusters
- Center-based clusters
- Contiguous clusters
- Density-based clusters
- Property or Conceptual
- Described by an Objective Function

Types of Clusters: Well-Separated

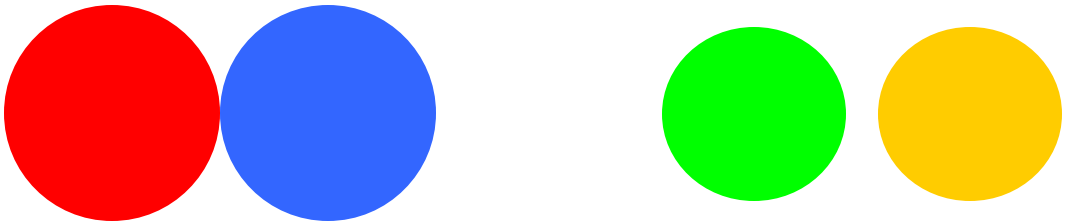
- **Well-Separated Clusters:**
  - **A cluster is a set of points such that any point in a cluster is closer (or more similar) to every other point in the cluster than to any point not in the cluster.**



3 well-separated clusters

Types of Clusters: Center-Based

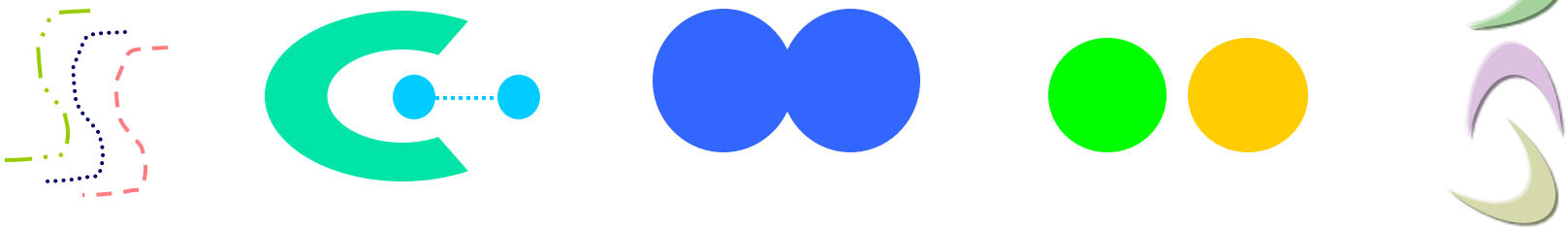
- **Center-based**
  - **A cluster is a set of objects such that an object in a cluster is closer (more similar) to the "center" of a cluster, than to the center of any other cluster**
  - **The center of a cluster is often a centroid, the average of all the points in the cluster, or a medoid, the most "representative" point of a cluster**



4 center-based clusters

Types of Clusters: Contiguity-Based

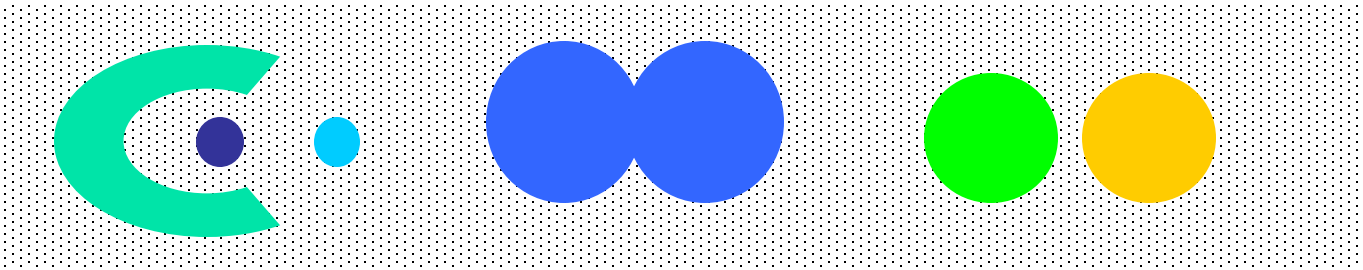
- **Contiguous Cluster (Nearest neighbor or Transitive)**
  - **A cluster is a set of points such that a point in a cluster is closer (or more similar) to one or more other points in the cluster than to any point not in the cluster.**



8 contiguous clusters

Types of Clusters: Density-Based

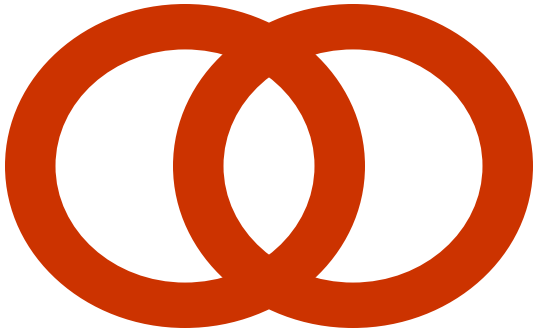
- **Density-based**
  - **A cluster is a dense region of points, which is separated by low-density regions, from other regions of high density.**
  - **Used when the clusters are irregular or intertwined, and when noise and outliers are present.**



6 density-based clusters

Types of Clusters: Conceptual Clusters

- **Shared Property or Conceptual Clusters**
  - **Finds clusters that share some common property or represent a particular concept.**



2 Overlapping Circles

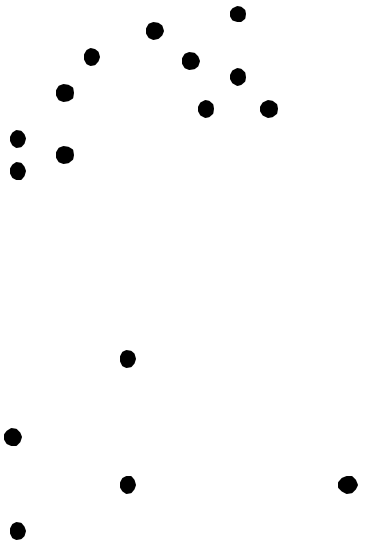
Types of Clusters: Objective Function

- **Clusters Defined by an Objective Function**
  - Finds clusters that minimize or maximize an objective function.
  - Enumerate all possible ways of dividing the points into clusters and evaluate the `goodness' of each potential set of clusters by using the given objective function. (NP Hard)
  - Can have global or local objectives.
    - Hierarchical clustering algorithms typically have local objectives
    - Partitional algorithms typically have global objectives
- **Map the clustering problem to a different domain and solve a related problem in that domain**
  - Proximity matrix defines a weighted graph, where the nodes are the points being clustered, and the weighted edges represent the proximities between points
  - Clustering is equivalent to breaking the graph into connected components, one for each cluster
  - Want to minimize the edge weight between clusters and maximize the edge weight within clusters

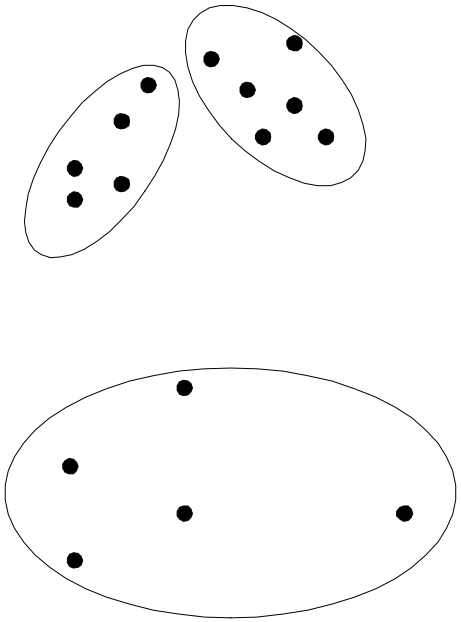
Types of Clustering Methods

- A clustering is a process to find a set of clusters
- Important distinction between hierarchical and partitional sets of clusters
- Partitional Clustering
  - A division data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset
- Hierarchical clustering
  - A set of nested clusters organized as a hierarchical tree
- Density based clustering
  - Discover clusters of arbitrary shape.
  - Clusters dense regions of objects separated by regions of low density

Partitional Clustering

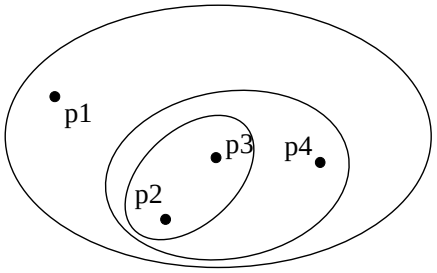


Original Points

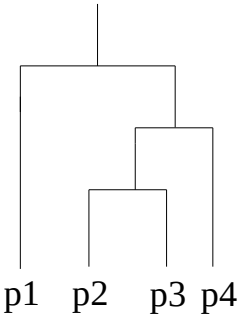


A Partitional Clustering

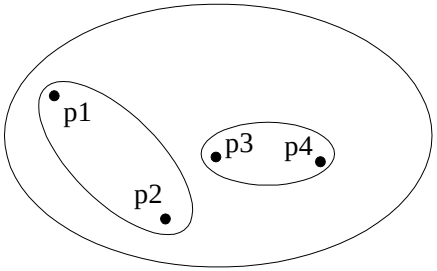
Hierarchical Clustering



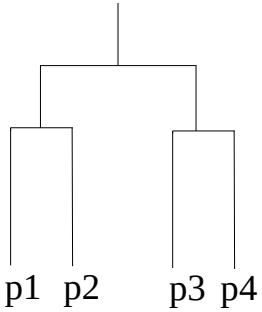
Traditional Hierarchical Clustering



Traditional Dendrogram



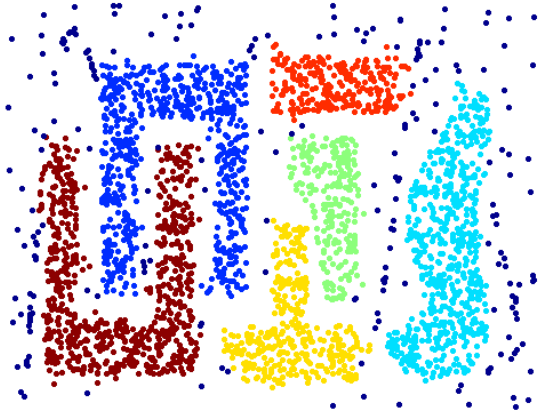
Non-traditional Hierarchical Clustering



Non-traditional Dendrogram

Density based Clustering

Original Points



Clusters

Partitional Clustering

- **Given**
  - **A data set of n objects**
  - **K the number of clusters to form**
- **Organize the objects into k partitions (k<=n) where each partition represents a cluster**
- **The clusters are formed to optimize an objective partitioning criterion**
  - **Objects within a cluster are similar**
  - **Objects of different clusters are dissimilar**

Partitional Clustering:
K-means Clustering

- **The basic algorithm is very simple**
- **Number of clusters, K, must be specified**
- **Each cluster is associated with a centroid (mean or center point)**
- **Each point is assigned to the cluster with the closest centroid**

1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

Partitional Clustering:
K-means Clustering

- **Initial centroids are often chosen randomly.**
  - **Clusters produced vary from one run to another.**
- **The centroid is (typically) the mean of the points in the cluster.**
- **'Closeness' is measured by Euclidean distance, cosine similarity, correlation, etc.**
- **K-means will converge for common similarity measures mentioned above.**
- **Most of the convergence happens in the first few iterations.**
  - **Often the stopping condition is changed to 'Until relatively few points change clusters' or some measure of clustering doesn't change.**
- **Complexity is O(n * K * I * d)**
  - **n = number of points, K = number of clusters, I = number of iterations, d = number of attributes**

Evaluating K-means Clusters

- **Most common measure is Sum of Squared Error (SSE)**
  - **For each point, the error is the distance to the nearest cluster**
  - **To get SSE, we square these errors and sum them.**

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

  - ***x*** **is a data point in cluster** C_i **and** m_i **is the representative point for cluster** C_i
    - **can show that** m_i **corresponds to the center (mean) of the cluster**
  - **Given two clusters, we can choose the one with the smallest error**

- **One easy way to reduce SSE is to increase K, i.e. the number of clusters**
  - **A good clustering with smaller K can have a lower SSE than a poor clustering with higher K**

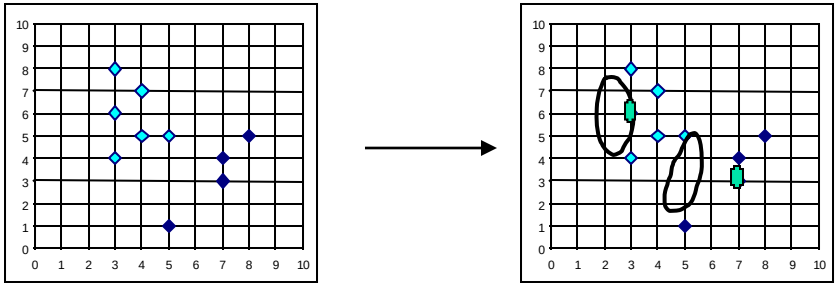
K-means Clusters
Solutions to Initial Centroids Problem

- **Multiple runs**
  - **Helps, but probability is not on your side**
- **Sample and use hierarchical clustering to determine initial centroids**
- **Select more than k initial centroids and then select among these initial centroids**
  - **Select most widely separated**
- **Pre-processing**
  - **Normalize the data**
  - **Eliminate outliers**
- **Post-processing**
  - **Eliminate small clusters that may represent outliers**
  - **Split 'loose' clusters, i.e., clusters with relatively high SSE**
  - **Merge clusters that are 'close' and that have relatively low SSE**
  - **Can use these steps during the clustering process**
    - **ISODATA**

Limitations of K-means

- **K-means has problems when clusters are of differing**
  - **Sizes**
  - **Densities**
  - **Non-globular shapes**

- **K-means has problems when the data contains outliers.**
  - **Since an object with an extremely large value may substantially distort the distribution of the data.**
  - **Solution: Instead of taking the mean value of the object in a cluster as a reference point, medoids can be used, which is the most centrally located object in a cluster.**



Partitional Clustering:
k-medoids Clustering

- **k-medoids**
  - **Instead of taking the mean value of the samples in a cluster as a reference point, medoids can be used**
  - **Note that choosing the new medoids is slightly different with choosing the new means in k-means algorithm**
  - **k-medoids is more robust than k-means in the presence of noise and outliers; and works effectively for small data sets.**
- **Algorithm k-medoids (k)**
  - **1. Select k representative samples arbitrarily**
  - **2. Associate each data point to the closest medoid**
  - **3. For each medoid** m **and data point** p
    - Swap m and p and compute the total cost of configuration
  - **4. Select the configuration with the lowest cost**
  - **5. repeat steps 2-5 until there is no change**

Partitional Clustering:
Fuzzy C-mean Clustering

- **The membership function μ_{il} expresses to what degree x_l belongs to class Ci.**
- **Crisp clustering: x_l can belong to one class only**

$$\mu_{il} = \begin{cases} 1 & if\ x_l \in C_i \\ 0 & if\ x_l \notin C_i \end{cases}$$

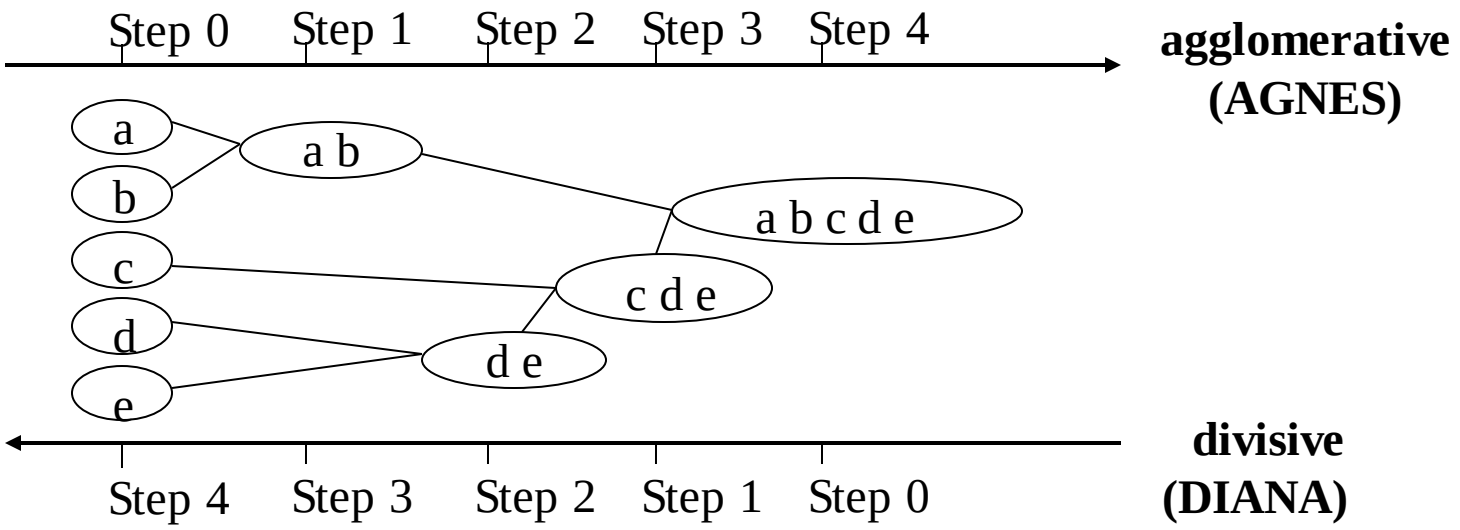
- **Fuzzy clustering: x_l belongs to all classes simultaneously with varying degrees of membership**

$$\mu_{il} = \left(\frac{1}{d(z_i^{(m)}, x_l)} \right)^{\frac{1}{q-1}} \Big/ \sum_{j=1}^{K} \left(\frac{1}{d(z_j^{(m)}, x_l)} \right)^{\frac{1}{q-1}}$$

  - **where $z^{(m)}$s are cluster means**
  - **q is a fuzziness index with 1<q<2**
    - Fuzzy clustering becomes crisp clustering when q→1
  - **Observe that $\sum_{i=1}^{K} \mu_{il} = 1,\ for\ l = 1, 2, \ldots, N$**
  - **C-mean minimizes $J_e = \sum_{i=1}^{K} J_i^f,\ J_i^f = \sum_{i=1}^{N} (\mu_{il})^q \left\| z_i^{(m)} - x_l \right\|^2$**

Hierarchical Methods:
Agglomerative and Divisive Clustering

- **Clusters have sub-clusters and sub-clusters can have sub-sub-clusters, …**
- **Use distance matrix as clustering criteria** (Merge nodes (clusters) that have the maximum similarity)



- **This method does not require the number of clusters k as an input, but needs a termination condition.**

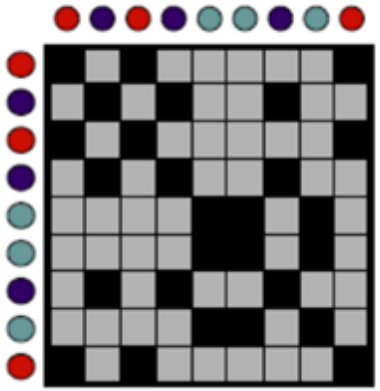
Hierarchical Methods:
Spectral Clustering

Goal: Given data points $X_1, \ldots, X_n$ and similarities $W(X_i,X_j)$, partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

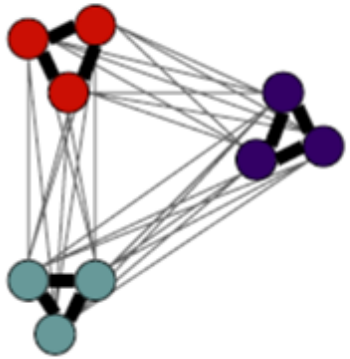
Similarity Graph: G(V,E,W)

V – Vertices (Data points)
E – Edge if similarity > 0
W - Edge weights (similarities)

Data



Similarities



Similarity graph

- $W = (w_{ij})$ adjacency matrix of the graph
- $d_i = \sum_j w_{ij}$ degree of a vertex
- $D = diag(d_1, \ldots, d_n)$ degree matrix

Hierarchical Methods:
Spectral Clustering

Input: Similarity matrix S, number k of clusters to construct

- Build similarity graph
- Compute the first k eigenvectors $v_1, \ldots, v_k$ of the matrix

$$\begin{cases} L = D - W & \text{for unnormalized spectral clustering} \\ L_{rw} = D^{-1}L & \text{for normalized spectral clustering} \end{cases}$$

- Build the matrix $V \in \mathbb{R}^{n \times k}$ with the eigenvectors as columns
- Interpret the rows of V as new data points $Z_i \in \mathbb{R}^k$

	v_1	v_2	v_3
Z_1	v_{11}	v_{12}	v_{13}
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Z_n	v_{n1}	v_{n2}	v_{n3}

- Cluster the points Z_i with the k-means algorithm in $\mathbb{R}^k$.

Determine the Number of Clusters

- **Empirical method**
  - **# of clusters: $k \approx \sqrt{n/2}$ for a dataset of n points, e.g., n = 200, k = 10**
- **Elbow method**
  - **Use the turning point in the curve of sum of within cluster variance w.r.t the # of clusters**
- **Cross validation method**
  - **Divide a given data set into *m* parts**
  - **Use *m* – 1 parts to obtain a clustering model**
  - **Use the remaining part to test the quality of the clustering**
    - E.g., For each point in the test set, find the closest centroid, and use the sum of squared distance between all points in the test set and the closest centroids to measure how well the model fits the test set
  - **For any k > 0, repeat it *m* times, compare the overall quality measure w.r.t. different *k's*, and find # of clusters that fits the data the best**

End

Your Presentations Scheduled From Next Week