Reconeixement

Matching

- Consisteix en localitzar objectes (sub-imatges) en imatges
-Es busca el ‘best match’ basat en algun criteri d’optimitat

Template Matching

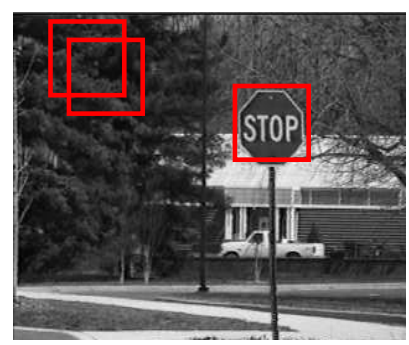
Objects can be represented by storing sample images or "templates"

Stop sign template

PENNSTATE

Hypotheses fromTemplate Matching

•Place the template at every location on the given image.



- •Compare the pixel values in the template with the pixel values in the underlying region of the image.
- •If a "good" match is found, announce that the object is present in the image.

•Possible measures are: SSD, SAD, Cross-correlation, Normalized Cross-correlation, max difference, etc.

PENNSTATE

Match metrics

MATCH METRIC	DEFINITION
Normalized Cross-Correlation (NCC)	$\frac{\sum_{u,v}\left(I_1(u,v)-\bar{I}_1\right)\cdot\left(I_2(u+d,v)-\bar{I}_2\right)}{\sqrt{\sum_{u,v}\left(I_1(u,v)-\bar{I}_1\right)^2\cdot\left(I_2(u+d,v)-\bar{I}_2\right)^2}}$
Sum of Squared Differences (SSD)	$\sum_{u,v}\left(I_1(u,v)-I_2(u+d,v)\right)^2$
Normalized SSD	$\sum_{u,v}\left(\frac{\left(I_1(u,v)-\bar{I}_1\right)}{\sqrt{\sum_{u,v}\left(I_1(u,v)-\bar{I}_1\right)^2}}-\frac{\left(I_2(u+d,v)-\bar{I}_2\right)}{\sqrt{\sum_{u,v}\left(I_2(u+d,v)-\bar{I}_2\right)^2}}\right)^2$
Sum of Absolute Differences (SAD)	$\sum_{u,v}\left\|I_1(u,v)-I_2(u+d,v)\right\|$

Better and faster results on gradient image/laplacian
Ex: NCC on gradient sign (-1,0,1)

Matching

Cerca de la cadira en la imatge

Resultat

UPC

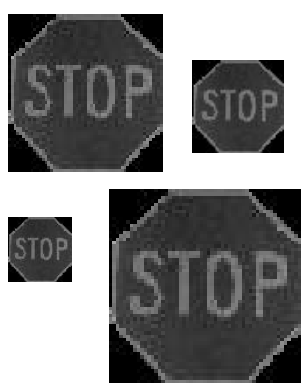
Limitations of Template Matching

- If the object appears scaled, rotated, or skewed on the image, the match will not be good.

PENNSTATE

Solution:

- Search for the template and possible transformations of the template:



Not very efficient! (but doable …)

PENNSTATE

Limitations of Template Matching

- It uses *global* information: it is sensitive to occlusion.

PENNSTATE

Limitations of Template Matching

- It uses pixel values: it is illumination and sensor dependent.

PENNSTATE

Problems of matching

Illumination

UPC

Problems of matching

Illumination

Scale

UPC

Problems of matching

Illumination

Scale

Rotation

UPC

Problems of matching

Illumination

Scale

Rotation

Affine

UPC

Problems of matching

Illumination

Scale

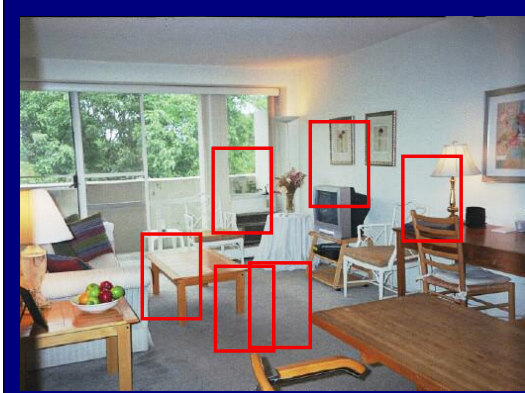
Rotation

Affine

Full Perspective

UPC

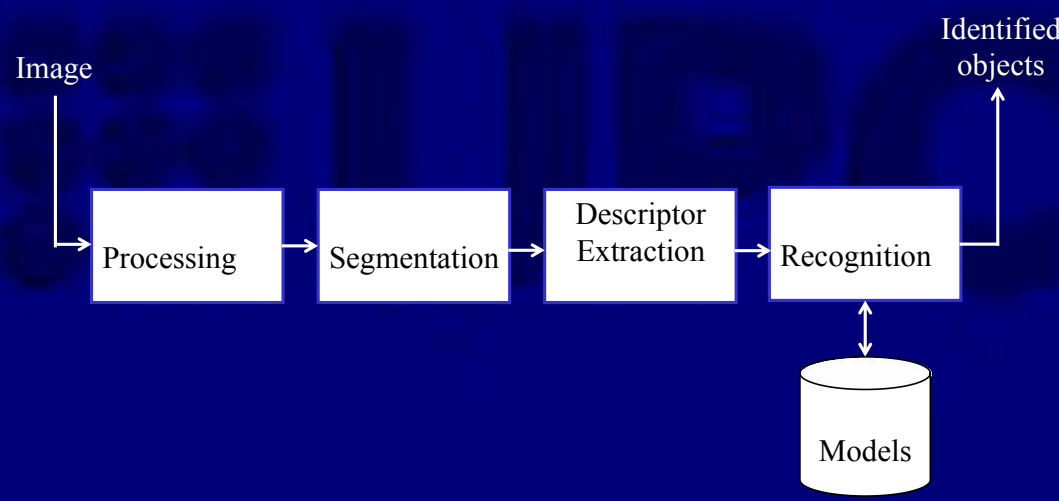
Matching



Resultat: Massa basura

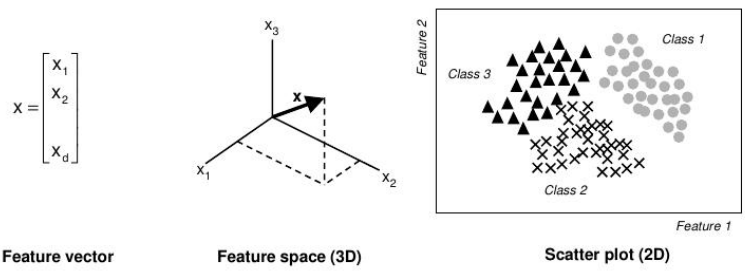
UPC

A computer vision system



Features and patterns

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

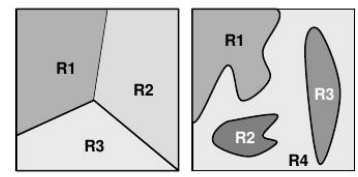


Feature vector **Feature space (3D)** **Scatter plot (2D)**

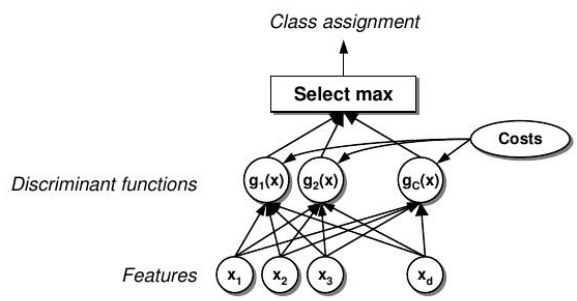
- **Pattern**
  - Pattern is a composite of traits or features characteristic of an individual
  - For our purposes, a pattern is a pair of variables $\{x, \omega\}$ where
    - $\boldsymbol{x}$ is a collection of observations or features (feature vector)
    - ω is the concept behind the observation (label)

What is a classifier ?

- **The task of a classifier is to partition feature space into class-labeled decision regions**
  - Borders between decision regions are called **decision boundaries**
  - The classification of feature vector ***x*** consists of determining which decision region it belongs to, and assign ***x*** to this class



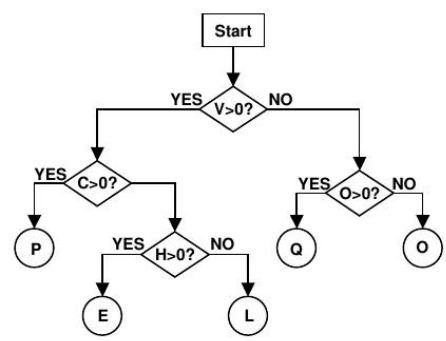
- **A classifier can be represented as a set of discriminant functions**
  - The classifier assigns a feature vector ***x*** to class ω_i if $g_i(\boldsymbol{x}) > g_j(\boldsymbol{x}) \quad \forall j \neq i$



A simple classifier

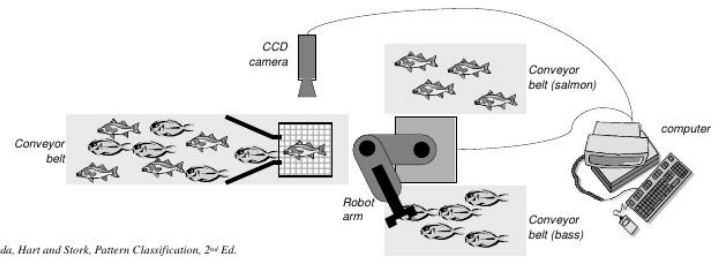
- **Consider the problem of recognizing the letters L,P,O,E,Q**
  - Determine a sufficient set of features
  - Design a tree-structured classifier

Character	Features			
	Vertical straight lines	*Horizontal straight lines*	*Oblique straight lines*	*Curved lines*
L	*1*	*1*	*0*	*0*
P	*1*	*0*	*0*	*1*
O	*0*	*0*	*0*	*1*
E	*1*	*3*	*0*	*0*
Q	*0*	*0*	*1*	*1*



A more realistic example

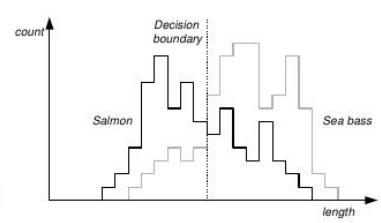
- **Consider the following scenario***
  - A fish processing plan wants to automate the process of sorting incoming fish according to species (salmon or sea bass)
  - The automation system consists of
    - a conveyor belt for incoming products
    - two conveyor belts for sorted products
    - a pick-and-place robotic arm
    - a vision system with an overhead CCD camera
    - a computer to analyze images and control the robot arm



*Adapted from Duda, Hart and Stork, Pattern Classification, 2nd Ed.

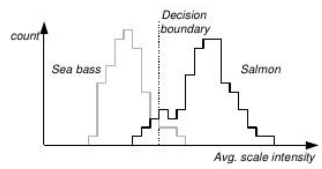
A more realistic example

- **Sensor**
  - The vision system captures an image as a new fish enters the sorting area
- **Preprocessing**
  - Image processing algorithms
    - adjustments for average intensity levels
    - segmentation to separate fish from background
- **Feature Extraction**
  - Suppose we know that, on the average, sea bass is larger than salmon
    - From the segmented image we estimate the length of the fish
- **Classification**
  - Collect a set of examples from both species
  - Compute the distribution of lengths for both classes
  - Determine a decision boundary (threshold) that minimizes the classification error
  - We estimate the classifier's probability of error and obtain a discouraging result of 40%
  - **What do we do now?**

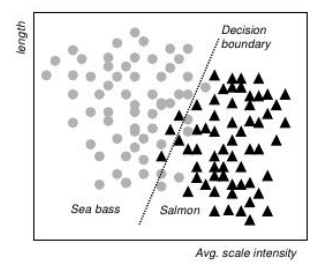


A more realistic example

- **Improving the performance of our PR system**
  - Determined to achieve a recognition rate of 95%, we try a number of features
    - Width, Area, Position of the eyes w.r.t. mouth...
    - only to find out that these features contain no discriminatory information
  - Finally we find a "good" feature: average intensity of the scales



  - We combine "*length*" and "*average intensity of the scales*" to improve class separability
  - We compute a linear discriminant function to separate the two classes, and obtain a classification rate of 95.7%



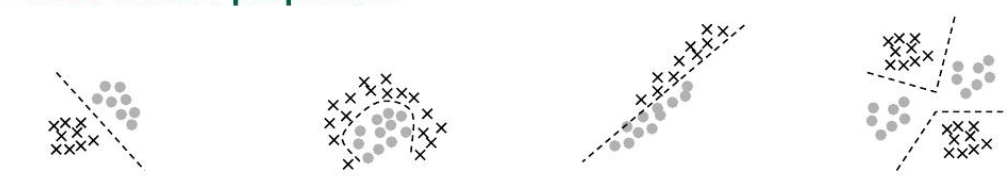
What is a 'good' feature?

What is a 'good' feature?

- **What makes a "good" feature vector?**
  - The quality of a feature vector is related to its ability to discriminate examples from different classes
    - Examples from the same class should have similar feature values
    - Examples from different classes have different feature values

"Good" features *"Bad" features*

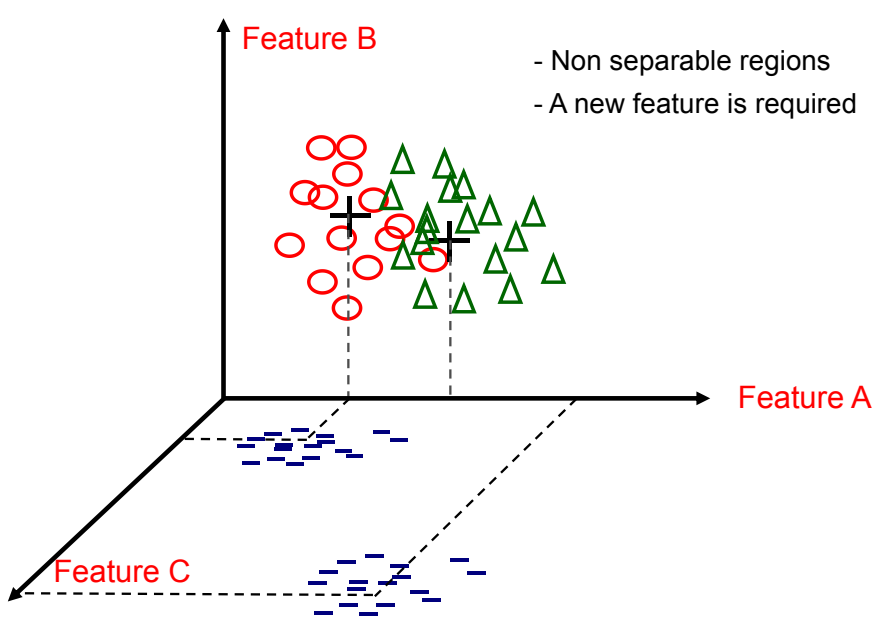
- **More feature properties**



Linear separability *Non-linear separability* *Highly correlated features* *Multi-modal*

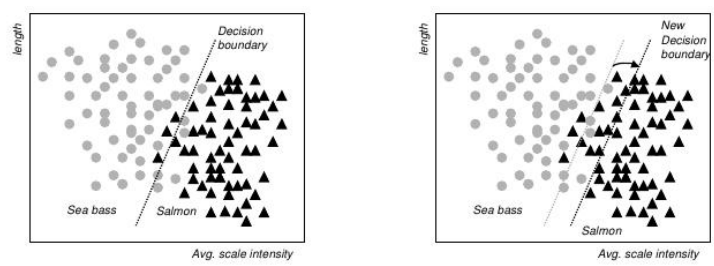
UPC

Class Separability



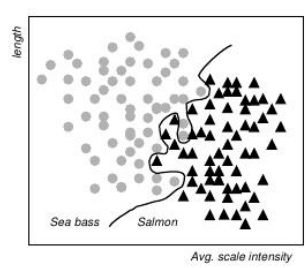
What is the cost?

- **Cost Versus Classification rate**
  - Our linear classifier was designed to minimize the overall misclassification rate
  - Is this the best objective function for our fish processing plant?
    - The **cost** of misclassifying salmon as sea bass is that the end customer will occasionally find a tasty piece of salmon when he purchases sea bass
    - The **cost** of misclassifying sea bass as salmon is an end customer upset when he finds a piece of sea bass purchased at the price of salmon
  - Intuitively, we could adjust the decision boundary to minimize this cost function



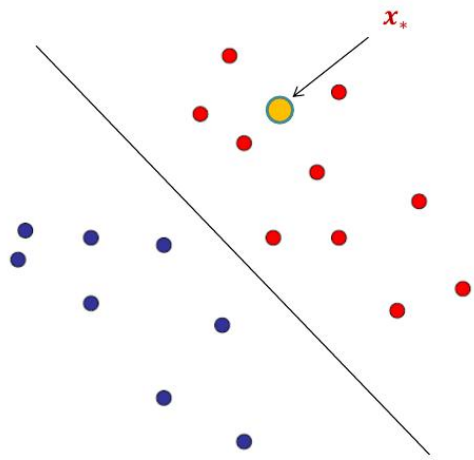
Overfitting

- **The issue of generalization**
  - The recognition rate of our linear classifier (95.7%) met the design specs, but we still think we can improve the performance of the system
    - We then design an artificial neural network with five hidden layers, a combination of logistic and hyperbolic tangent activation functions, train it with the Levenberg-Marquardt algorithm and obtain an impressive classification rate of 99.9975% with the following decision boundary



  - Satisfied with our classifier, we integrate the system and deploy it to the fish processing plant
    - After a few days, the plant manager calls to complain that the system is misclassifying an average of 25% of the fish
    - What went wrong?

Linear classifiers



Once we have learned w and b, we can do classification on any given test point x_*. This is known as the **testing stage**.

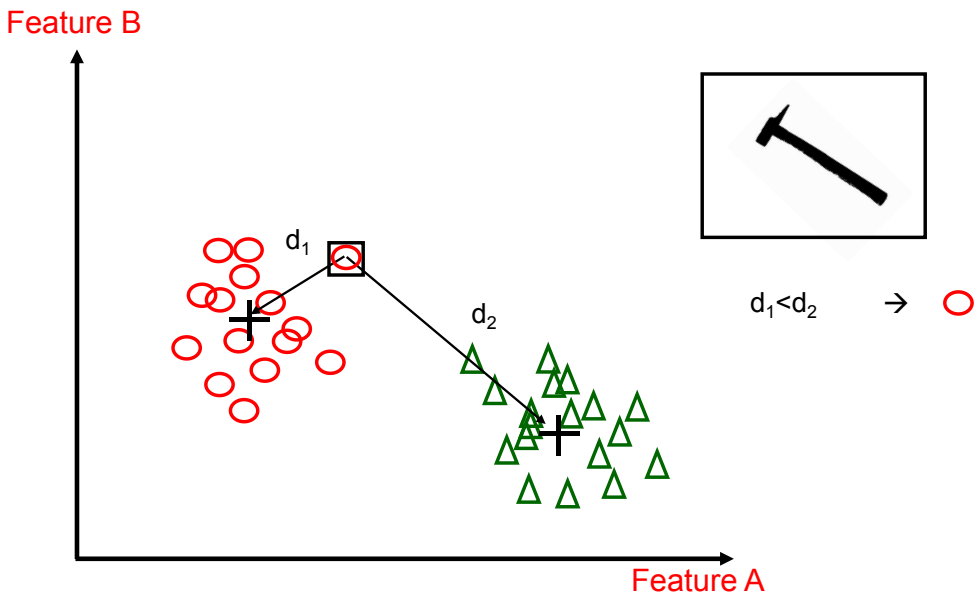
If $w^T x_* + b \geq +1$ then
  classify x_* into class 1
else
  classify x_* into class 2

UPC

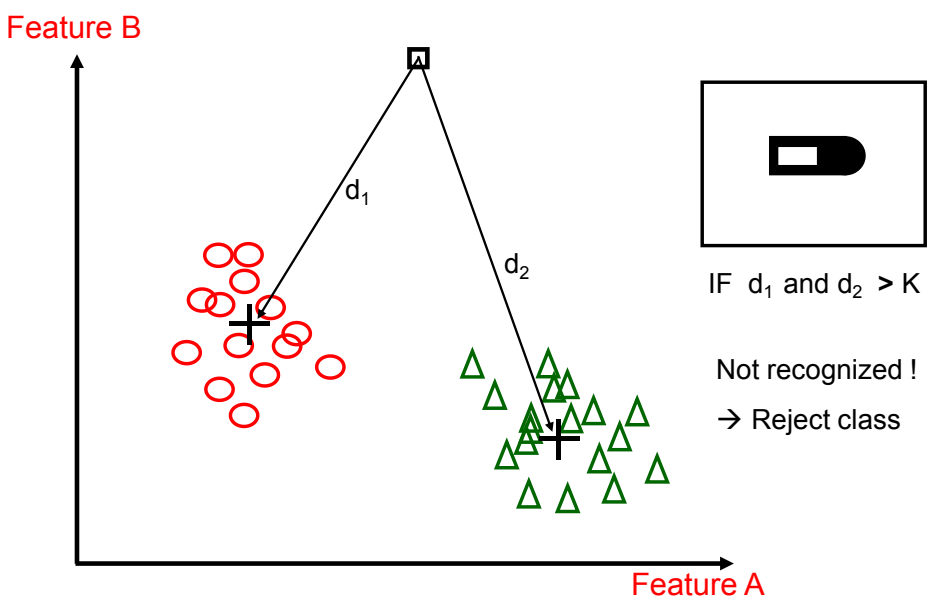
Simple N-N clasifier

- Using the nearest class mean:
  - Each class is represented by a mean feature vector
  - The input feature vector is labeled according to the closest class mean
- Using the nearest neighbor:
  - Each class is represented by a set of feature vectors
  - The input feature vector is labeled according to the closest example vector

Distance: Nearest class mean



Distance: Nearest class mean



Distance ?

- **a)Measuring distances:**
- L1 Norm (Manhattan distance): $|x_1 - x_2| = \sum_{i=1,N} |x_1[i] - x_2[i]|$
- L2 Norm (Euclidean distance): $\|x_1 - x_2\| = \sqrt{\sum_{i=1,N} (x_1[i] - x_2[i])^2}$
- Scaled Euclidean distance: $\|x_1 \widehat{-} x_2\| = \sqrt{\sum_{i=1,N} \frac{(x_1[i] - x_2[i])^2}{\sigma_i^2}}$

Distance ?

- Maximum Scaled distance: $MSD(x_1, x_2) = \max_{i=1,N} \left(\frac{(x_1[i] - x_2[i])^2}{\sigma_i^2} \right)$
- Mahalanobis distance:

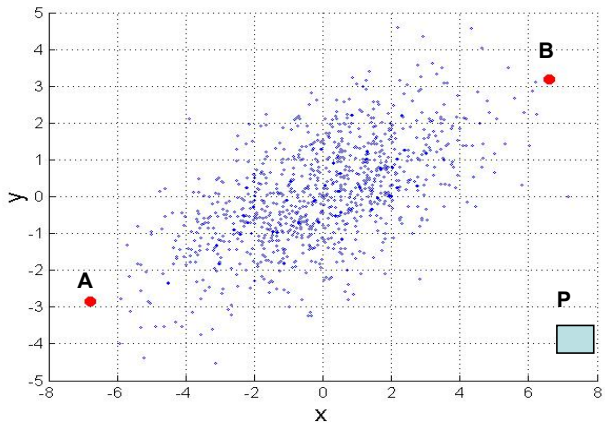
$$MH(x_1, x_2) = (x_1 - x_2)^t \Sigma^{-1} (x_1 - x_2)$$

  - Σ is the covariance matrix, which must be invertible
  - $(x_1\text{-}x_2)$ is a column vector, which is transposed on the left side
  - The end result is the most statistically pleasing if the variances and covariances are good estimates for the data set.

- Discrete vector distance: count up the number of mismatched elements
  - Works with vectors of binary values (hamming distance).
  - Works with vectors of discrete values that don't have meaningful distances.

Mahalanobis Distance

$$mahalanobis(p,q) = (p-q)\Sigma^{-1}(p-q)^T$$



Σ is the covariance matrix of the input data *X*

$$\Sigma_{j,k} = \frac{1}{n-1}\sum_{i=1}^{n}(X_{ij} - \overline{X}_j)(X_{ik} - \overline{X}_k)$$

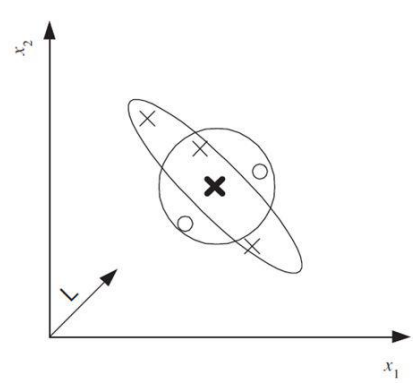
When the covariance matrix is identity Matrix, the mahalanobis distance is the same as the Euclidean distance.

Useful for detecting outliers.

Q: what is the shape of data when covariance matrix is identity?
Q: A is closer to P or B?

For red points, the Euclidean distance is 14.7, Mahalanobis distance is 6.

Mahalanobis Distance



Euclidean distance (circle) is not suitable,
Mahalanobis distance using an **M** (ellipse) is suitable.
After the data is projected along **L**, Euclidean distance can be used.

K-NN classifier

- Using the k-nearest neighbors:
  - Each class is represented by a set of feature vectors
  - Find the k-nearest neighbors out of all the example feature vectors

    Then if all of them belong to the same class assign this class to the input vector else chose one of the following:

    a) use the nearest neighbor class

    b) use the class of the majority of the k-nearest neighbors

    c) use the reject class.

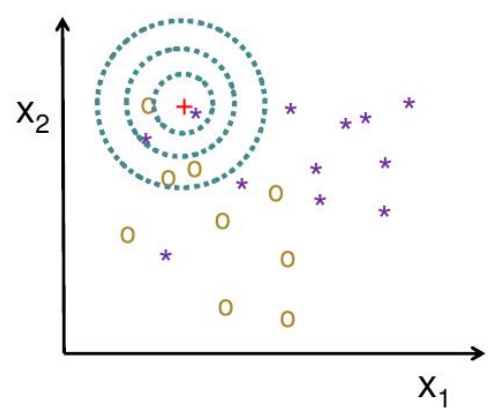
  - Alternatively, find the k-nearest neighbors in every example feature vector class. Then classify the input feature vector according to the lowest average distance

K-nearest-neighbour

- Distance measure – Euclidean

$$D(X,Y) = \sqrt{\sum_{i=1}^{D}(x_i, y_i)^2}$$

- 1-nearest-neighbour
  $f(+) = *$
- 3-nearest-neighbour
  $f(+) = *$
- 5-nearest-neighbour
  $f(+) = o$



K-NN Practical Matters

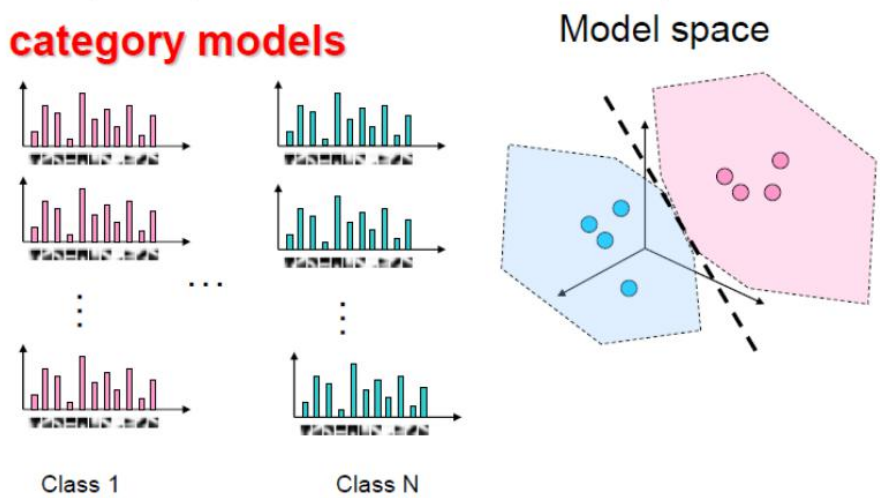
- Choosing the value of k
  - If too small, sensitive to noise points
  - If too large, neighbourhood may include points from other classes
  - Solution: cross-validation
- Can produce counter-intuitive results
  - Each feature may have a different scale
  - Solution: normalize each feature to zero mean, unit variance
- Curse of dimensionality
  - Solution: no good solution exists so far

- This classifier works well provided there are **lots of training data** and the **distance function is good**.

UPC

Support vector machines

- Support Vector Machines: find the hyper-planes (if the features are linearly separable) that separate these classes in the model space

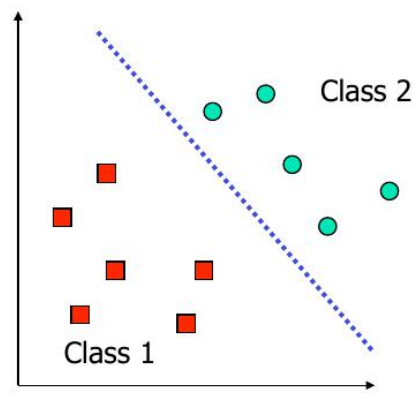


UPC

Support vector machines

What is a good Decision Boundary?

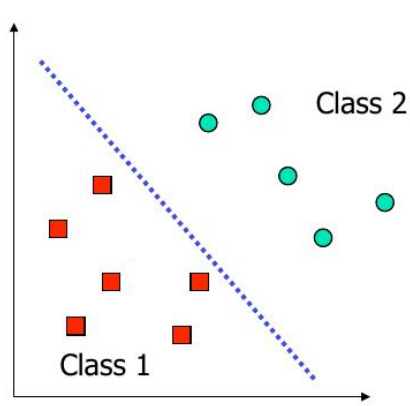
- Consider a two-class, linearly separable classification problem
- Many decision boundaries!
  - The Perceptron algorithm can be used to find such a boundary
  - Different algorithms have been proposed (DHS ch. 5)
- Are all decision boundaries equally good?

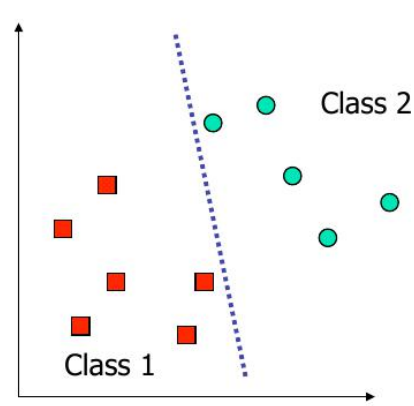


UPC

Support vector machines

Examples of Bad Decision Boundaries



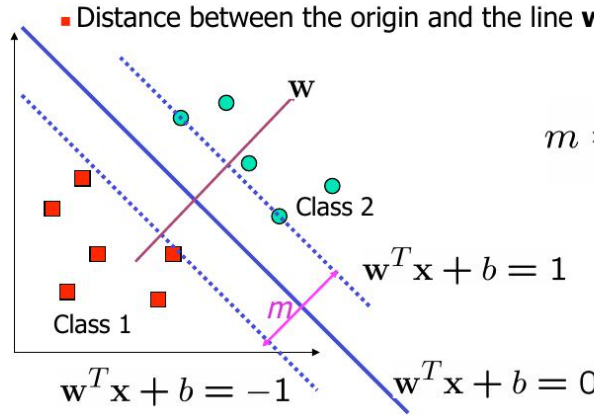


UPC

Support vector machines

Large-margin Decision Boundary

- The decision boundary should be as far away from the data of both classes as possible
  - We should maximize the margin, *m*
  - Distance between the origin and the line $\mathbf{w}^t\mathbf{x}=k$ is $k/||\mathbf{w}||$



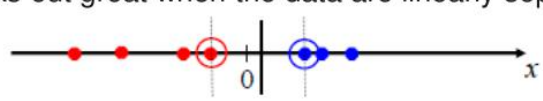
$$m = \frac{2}{||\mathbf{w}||}$$

UPC

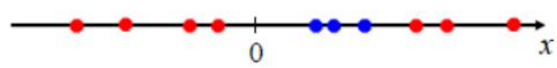
Support vector machines

Nonlinear SVMs

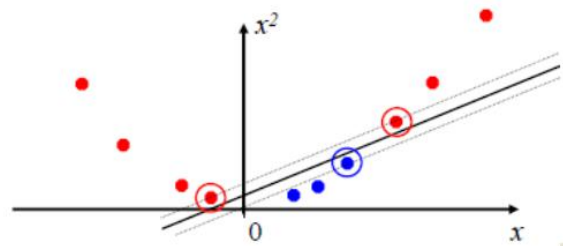
- The linear SVM works out great when the data are linearly separable. E.g. the 1D case below:



- But what is the data are more complicated? Like



- We can map the data to a higher-dimensional space:

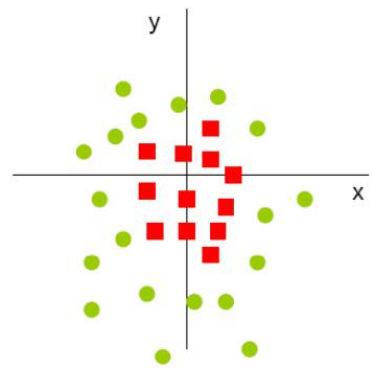


UPC

Support vector machines

Non Linear separating plane

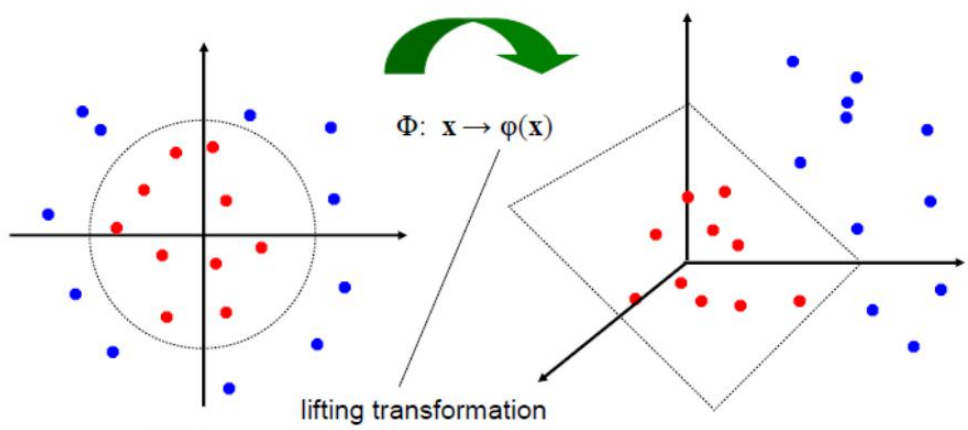
- Add distance to origin $(x^2+y^2)^{1/2}$ as a third feature
- Data now lives on a parabolic surface in 3D
- Linear separation in 3D
- In original feature space, boundary is an ellipse



UPC

Support vector machines

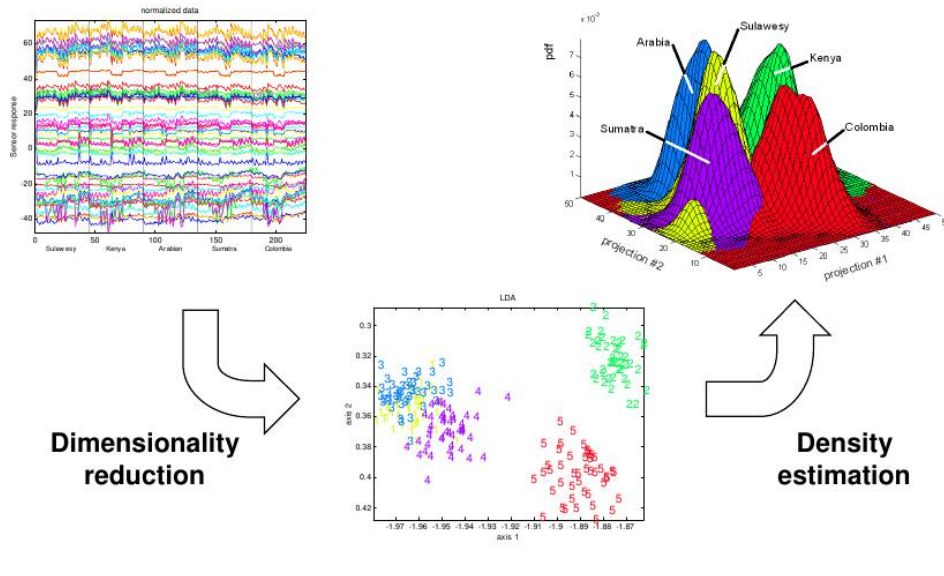
Nonlinear SVMs



↘ We use a **lifting transformation** Φ to transform the feature vectors to a higher dimensional space.

UPC

Statistical pattern recognition



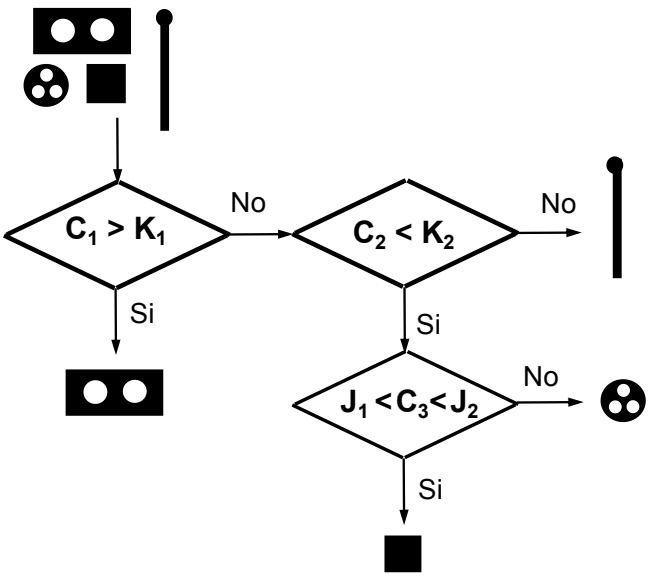
UPC

Decision trees

- Using decision trees:
  - Make decisions about each feature in a specific order
  - Each node in the tree involves a decision using one feature
  - The leaves of the tree are all individual classes
  - The same class may appear on multiple leaves
  - It is possible to learn optimal decision trees from the data

Ordering features: Decision tree

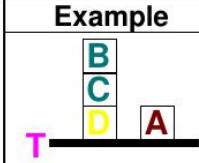
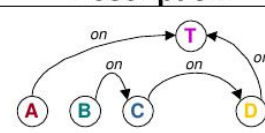
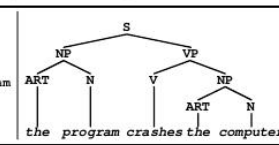
Objects to be classified:



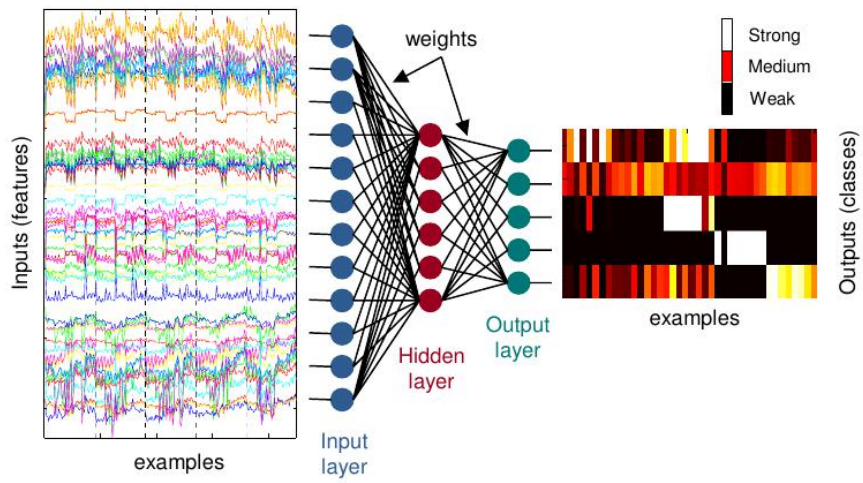
C_1 : (Area)
C_2 : (Perimeter)
C_3 : Vertex number

K_i, J_j, functions of C_i^j

Structural or syntactic pattern recognition

Example	Model	Description
	Directed Graph	
"the program crashes the computer"	Grammar	P = { SENTENCE → NP + VP NOUN PHRASE → ART + N VERB PHRASE → V + NP NOUN → computer|program VERB → crashes ARTICLE → the } 

Neural neworks



UPC

What are Neural Networks?

- Models of the brain and nervous system
- Highly parallel
  - Process information much more like the brain than a serial computer
- Learning

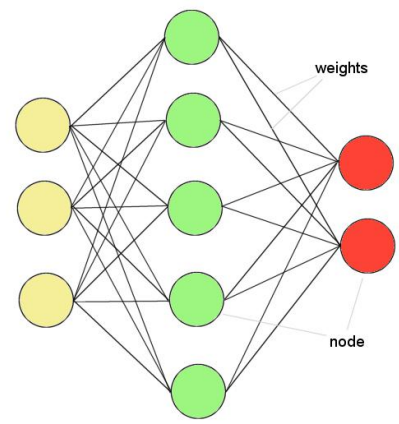
- Very simple principles
- Very complex behaviours

- Applications
  - As powerful problem solvers
  - As biological models

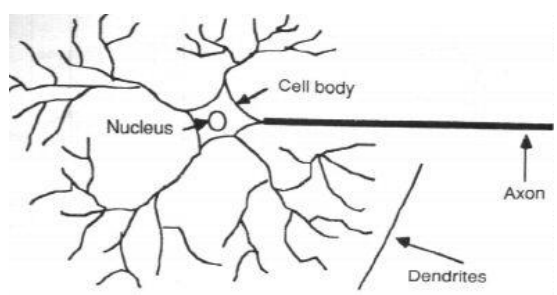
ANNs – The basics

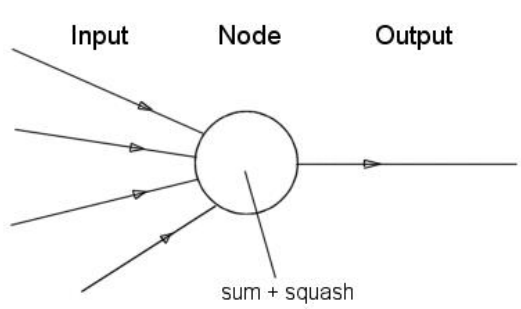
- ANNs incorporate the two fundamental components of biological neural nets:

1. Neurones (nodes)
2. Synapses (weights)

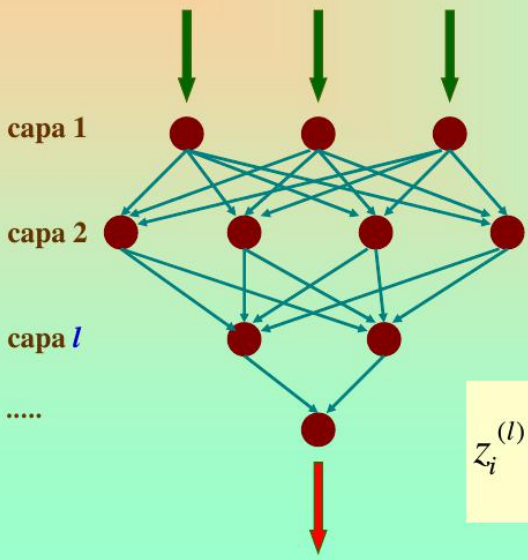


- Neurone vs. Node





How does a neural network work ?



multilayer

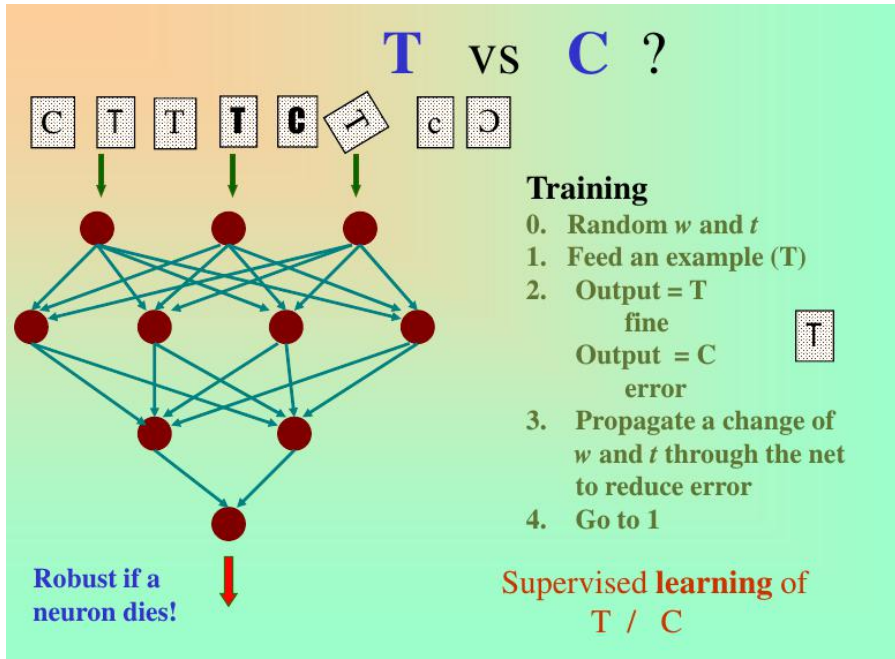
feedforward

Neural Network

$$z_i^{(l)} = f\left(\sum_{j=1}^{n(l-1)} w_{ij}^{(l)} z_j^{(l-1)} + t_i^{(l)}\right)$$

- Weight settings determine the behaviour of a network

  → How can we find the right weights?



Evaluating a classifier

- Independent test data:
  - Good independent test data (data not used during the classifier's training/settlement) must be labeled with its true class label. It must include representatives of all of the classes, including the reject class.
- Classification error:
  - The classifier makes an error when it labels the input feature vector as a class that is not its true class.

- Empirical error rate:
  - The number of classification errors made on independent test data divided by the number of classifications attempted.
- Empirical reject rate:
  - The number of rejects made on independent test data divided by the number of classifications attempted.
- False positives:
  - Occur in two-class or detection problems when detection occurs but it shouldn't.
- False negatives:
  - Occur in two-class or detection problems when detection should occur but it doesn't.

Precision and recall

$$\text{precision} = \frac{t_p}{t_p + f_p}$$

$$\text{recall} = \frac{t_p}{t_p + f_n}$$

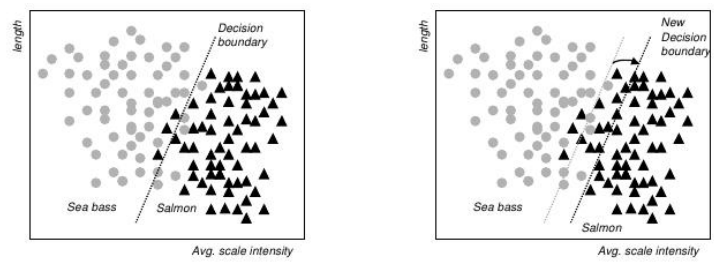
- Una precision = 1 vol dir que tots els peixos classificats com a salmons, eren salmons. Ara bé, no dona informació dels salmons classificats com a llobarros.

- Un recall = 1 vol dir que tots els salmons que hi havia han estat classificats com a salmons. Ara bé, no dona informació dels llobarros clasificats com a salmons.

$$F_{score} = \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

What is the cost?

- **Cost Versus Classification rate**
  - Our linear classifier was designed to minimize the overall misclassification rate
  - Is this the best objective function for our fish processing plant?
    - The **cost** of misclassifying salmon as sea bass is that the end customer will occasionally find a tasty piece of salmon when he purchases sea bass
    - The **cost** of misclassifying sea bass as salmon is an end customer upset when he finds a piece of sea bass purchased at the price of salmon
  - Intuitively, we could adjust the decision boundary to minimize this cost function

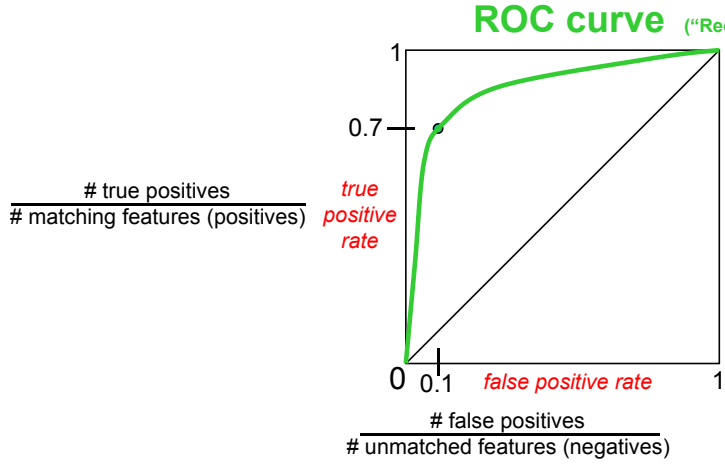


UPC

The ROC curve

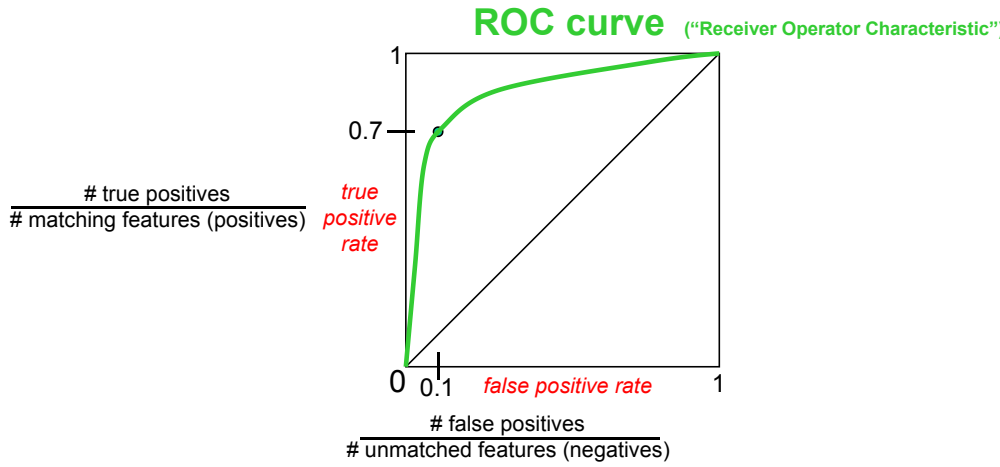
- When there is a tradeoff of error types, a single performance number is not the best solution to represent the capabilities of a system.
- A receiver operating characteristic, or simply ROC curve, is a graphical plot which illustrates the performance of a binary classifier system as its discrimination threshold is varied.
- It is a plot of the true positive rate against the false positive rate, then, the tradeoff between sensitivity and specificity

ROC curve ("Receiver Operator Characteristic")



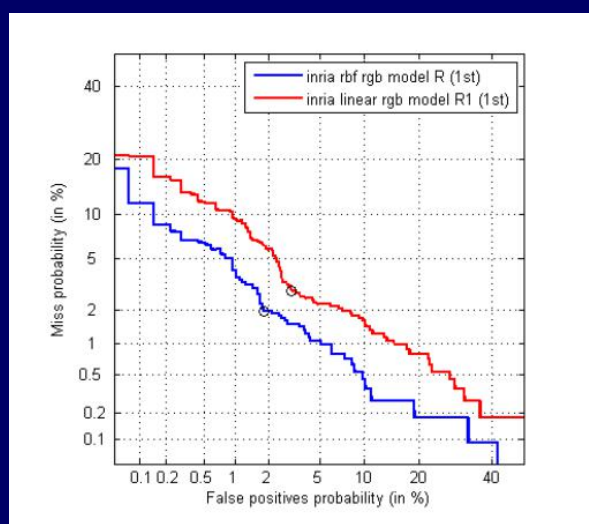
The ROC curve

• The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test.
• The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test.
• The slope of the tangent line at a point gives the likelihood ratio for that value of the test.
• The area under the curve is a measure of accuracy. An area of 1 represents a perfect test; an area of .5 represents a worthless test.



The DET curve

Detection Error Trade-off

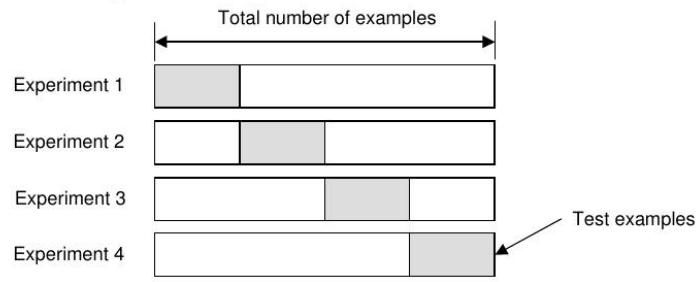


- Confusion matrix:
  - List the classes (including the reject class) along both rows and columns of the matrix.
  - The rows represent the true label of the object
  - The columns represent the label given by the classifier
  - A perfect classifier has a diagonal confusion matrix
  - In a two-class problem the off-diagonals are the numbers of false positives (upper right) and false negatives (lower left).

	zero	un	dos	tres	quatre	cinc	sis	set	buit	nou	cap
zero	6	0	0	0	0	0	2	0	0	0	5
un	0	9	0	0	0	0	0	0	0	0	7
dos	0	0	8	0	0	0	0	1	0	0	4
tres	0	0	7	1	0	0	0	1	0	0	5
quatre	0	0	0	0	8	0	0	2	0	0	2
cinc	0	0	4	1	0	4	0	2	0	0	4
sis	0	0	0	0	0	0	3	0	1	5	3
set	0	0	1	0	0	0	0	9	0	0	4
buit	0	0	0	0	0	0	0	0	9	0	5
nou	2	0	0	0	0	0	1	0	0	7	3
cap	0	0	0	0	0	0	0	0	0	0	0

K-Fold Cross-validation

- **Create a K-fold partition of the the dataset**
  - For each of K experiments, use K-1 folds for training and the remaining one for testing



- **K-Fold Cross validation is similar to Random Subsampling**
  - The advantage of K-Fold Cross validation is that all the examples in the dataset are eventually used for both training and testing
- **As before, the true error is estimated as the average error rate**

$$E = \frac{1}{K}\sum_{i=1}^{K} E_i$$

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 a small number of folds**
  - \+ The number of experiments and, therefore, 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
- **A common choice for K-Fold Cross Validation is K=10**

Reducció de la dimensionalitat

- Feature selection
- Principal Componet Analysis

UPC

Efficient Image Storage

Toy Example: Images with 3 pixels

Consider the following 3x1 templates:

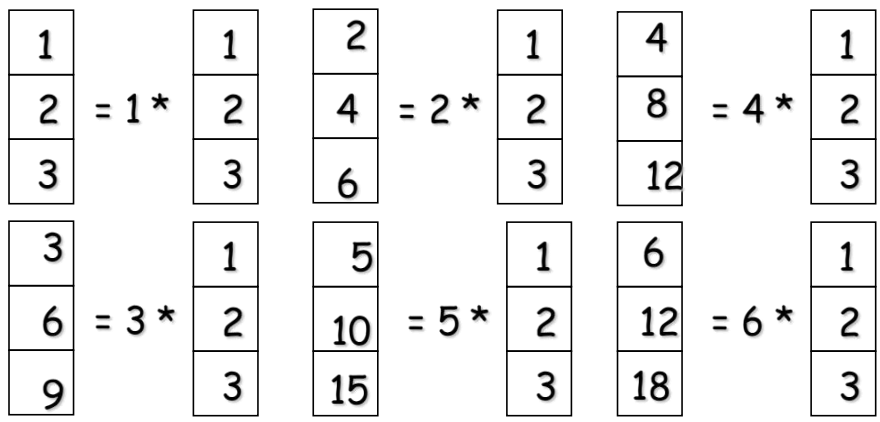
1	2	4	3	5	6
2	4	8	6	10	12
3	6	12	9	15	18

If each pixel is stored in a byte, we need 18 = 3 x 6 bytes

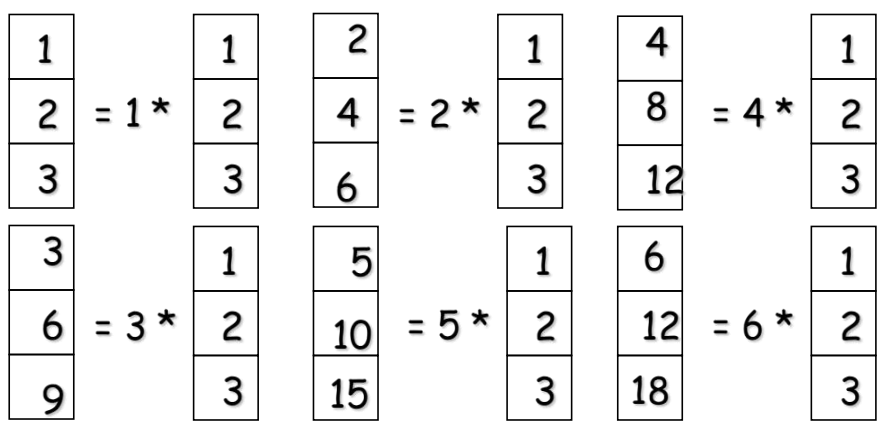
PENNSTATE

Efficient Image Storage

Looking closer, we can see that all the images are very similar to each other: they are all the same image, scaled by a factor:



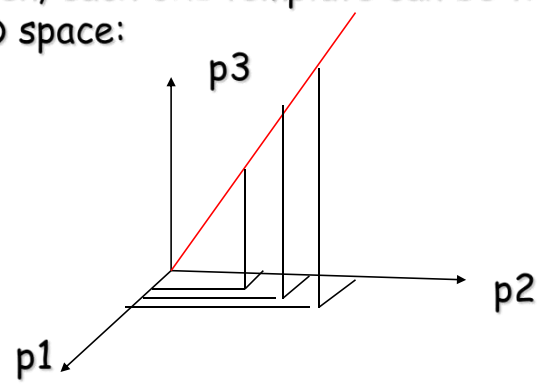
Efficient Image Storage



They can be stored using only 9 bytes (50% savings!):
Store one image (3 bytes) + the multiplying constants (6 bytes)

Geometrical Interpretation:

Consider each pixel in the image as a coordinate in a vector space. Then, each 3x1 template can be thought of as a point in a 3D space:

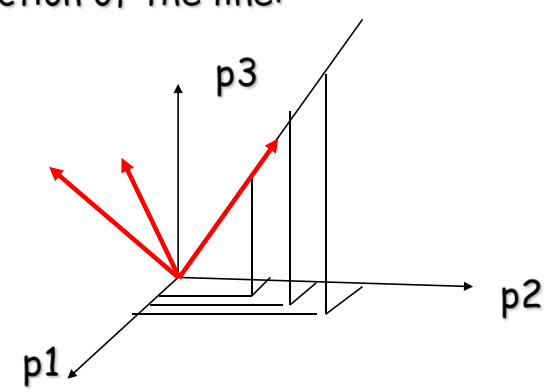


But in this example, all the points happen to belong to a line: a 1D subspace of the original 3D space.

PENNSTATE

Geometrical Interpretation:

Consider a new coordinate system where one of the axes is along the direction of the line:



In this coordinate system, every image has only one non-zero coordinate: we *only* need to store the direction of the line (a 3 bytes image) and the non-zero coordinate for each of the images (6 bytes).

PENNSTATE

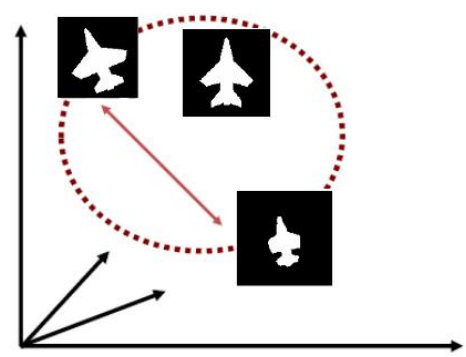
Principal Component Analysis (PCA)

- Given a set of templates, how do we know if they can be compressed like in the previous example?
  - The answer is to look into the correlation between the templates
  - The tool for doing this is called PCA

PENNSTATE

Images as Data Points

- A N x N pixel image represented as a vector occupies a single point in N^2-dimensional image space.
- Images of particular objects being similar in overall configuration, will not be randomly distributed in this huge image space, but will form *clusters*.
- Therefore, they can be compactly represented and modelled in a low dimensional subspace.



UPC

PCA Theorem

Let x_1 x_2 … x_n be a set of n N^2 x 1 vectors and let $\bar{x}$ be their average:

$$\mathbf{x}_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{iN^2} \end{bmatrix} \qquad \bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^{i=n} \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{iN^2} \end{bmatrix}$$

Note: Each N x N image template can be represented as a N^2 x 1 vector whose elements are the template pixel values.

PENNSTATE

PCA Theorem

Let X be the N^2 x n matrix with columns $x_1 - \bar{x}$, $x_2 - \bar{x}$,… $x_n - \bar{x}$:

$$X = \begin{bmatrix} \mathbf{x}_1 - \bar{\mathbf{x}} & \mathbf{x}_2 - \bar{\mathbf{x}} & \cdots & \mathbf{x}_n - \bar{\mathbf{x}} \end{bmatrix}$$

Note: subtracting the mean is equivalent to translating the coordinate system to the location of the mean.

PENNSTATE

PCA Theorem

Let Q = X X^T be the N^2 x N^2 matrix:

$$Q = XX^T = \begin{bmatrix} \mathbf{x}_1 - \bar{\mathbf{x}} & \mathbf{x}_2 - \bar{\mathbf{x}} & \cdots & \mathbf{x}_n - \bar{\mathbf{x}} \end{bmatrix} \begin{bmatrix} (\mathbf{x}_1 - \bar{\mathbf{x}})^T \\ (\mathbf{x}_2 - \bar{\mathbf{x}})^T \\ \vdots \\ (\mathbf{x}_n - \bar{\mathbf{x}})^T \end{bmatrix}$$

Notes:

1. Q is square
2. Q is symmetric
3. Q is the *covariance* matrix
4. Q can be very large (remember that N^2 is the number of pixels in the template)

PENNSTATE

PCA Theorem

Theorem:

Each x_j can be written as: $$\mathbf{x}_j = \bar{\mathbf{x}} + \sum_{i=1}^{i=n} g_{ji}\mathbf{e}_i$$

where e_i are the n eigenvectors of Q with non-zero eigenvalues.

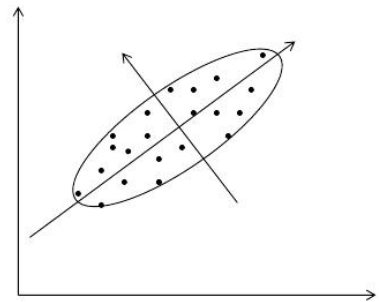
Notes:

1. The eigenvectors e_1 e_2 … e_n span an ***eigenspace***
2. e_1 e_2 … e_n are N^2 x 1 orthonormal vectors (N x N images).
3. The scalars g_{ji} are the coordinates of x_j in the space.
4. $$g_{ji} = (\mathbf{x}_j - \bar{\mathbf{x}}).\mathbf{e}_i$$

PENNSTATE

Principal Component Analysis (PCA)

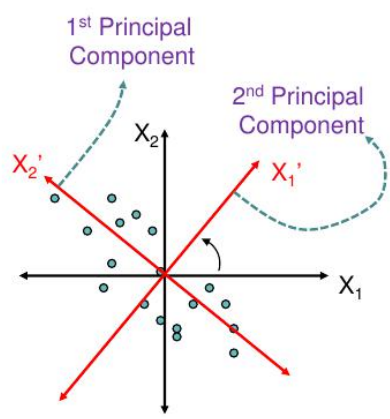
- Calculate vectors that account for the maximum variance of data
  - These vectors are called *Eigen Vectors*.
- Eigen Vectors show the direction of axes a fitted ellipsoid
- Eigen Values show the significance of the corresponding axis.
  - Large value -> more variance
- For high dimensional data, only a few of the Eigen values are significant



UPC

Principal Component Analysis (PCA)

- Find Eigen Values and Eigen Vectors
- Chose the highest P Eigen Values
- Form a new coordinate system defined by the significant Eigen vectors
- Project data to the new space (rotate the basis)
- Compressed Data



UPC

Using PCA to Compress Data

- Expressing x in terms of $e_1 \ldots e_n$ has not changed the size of the data
- However, if the templates are highly correlated many of the coordinates of x will be zero or closed to zero.

Using PCA to Compress Data

- Sort the eigenvectors e_i according to their eigenvalue:

$$\lambda_1 \geq \lambda_2 \geq \ldots \lambda_n$$

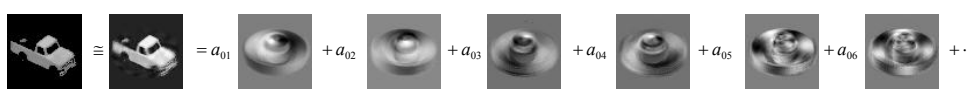
- Assuming that $\lambda_i \approx 0$ if $i > k$

- Then

$$\mathbf{x}_j \approx \bar{\mathbf{x}} + \sum_{i=1}^{i=k} g_{ji}\mathbf{e}_i$$

Eigenspaces: Efficient Image Storage

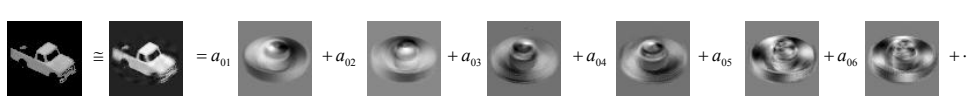
- Use PCA to compress the data:
  - each image is stored as a k-dimensional vector
  - Need to store k N x N eigenvectors
- $k << n << N^2$



PENNSTATE

Eigenspaces: Efficient Image Comparison

- Use the same procedure to compress the given image to a k-dimensional vector.
- Compare the compressed vectors:
  - Dot product of k-dimensional vectors
  - $k << n << N^2$



PENNSTATE

Algorithm EIGENSPACE_LEARN

Assumptions:

1. Each image contains one object only.
2. Objects are imaged by a fixed camera .
3. Images are normalized in size N x N:
   - The image frame is the minimum rectangle enclosing the object.
4. Energy of pixels values is normalized to 1:
   - $\sum_i \sum_j I(i,j)^2 = 1$
5. The object is completely visible and unoccluded in all images.

PENNSTATE

Case Study – Face Recognition

↘ Milestone methods in face detection / recognition

1. **PCA and Eigenfaces** (Turk & Pentland, 1991)
2. LDA and Fisherfaces (Bellumeur et al. 1997)
3. AdaBoost (Viola & Jones, 2001)
4. Local Binary Patterns (Ahonen et al. 2004)
5. DeepFace (Facebook, 2014)

UPC

PCA and Eigenfaces – Training

1. Align training images $x_1, x_2, \ldots, x_N$

2. Compute average face $\mu = \frac{1}{N}\Sigma x_i$ 

3. Compute PCA of the covariance matrices of the difference images
4. Compute training projections $a_1, a_2, \ldots, a_N$

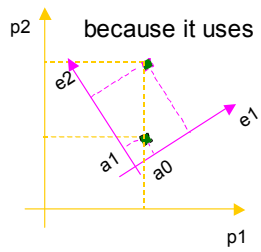
PCA and Eigenfaces – Testing

Visalization of Eigenfaces

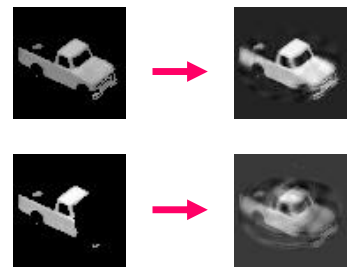
These are the first 4 eigenfaces (eigenvectors) from a training set of 400 images

1. Take query image y
2. Project y into the Eigenface space $\omega = U_p^T(y - \mu)$
3. Compare projection ω with all training projection a_i
4. Identity of the query image X is chosen as that of the nearest image (i.e. the one with the lowest $||w - a_i||$

PCA has problems with occlusion



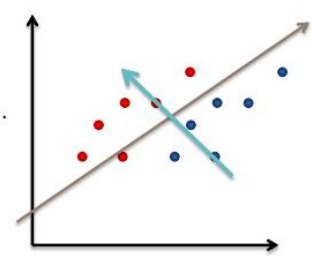
because it uses global information



PENNSTATE

PCA Final Note

- PCA finds directions of maximum variance of the data.
- This may not separate classes at all.
- Basic PCA is also sensitive to noise and outliers (read other variants e.g. Robust PCA).
- Linear Discriminant Analysis LDA finds the direction along which between class distance is maximum.



- Sometimes PCA is followed by LDA to combine the advantages of both.
- Eigen eyes, eigen nose, eigen X your imagination is the only limination.

UPC


UPC
UPC