Music Classification

Juan Pablo Bello
MPATE-GE 2623 Music Information Retrieval
New York University

Classification

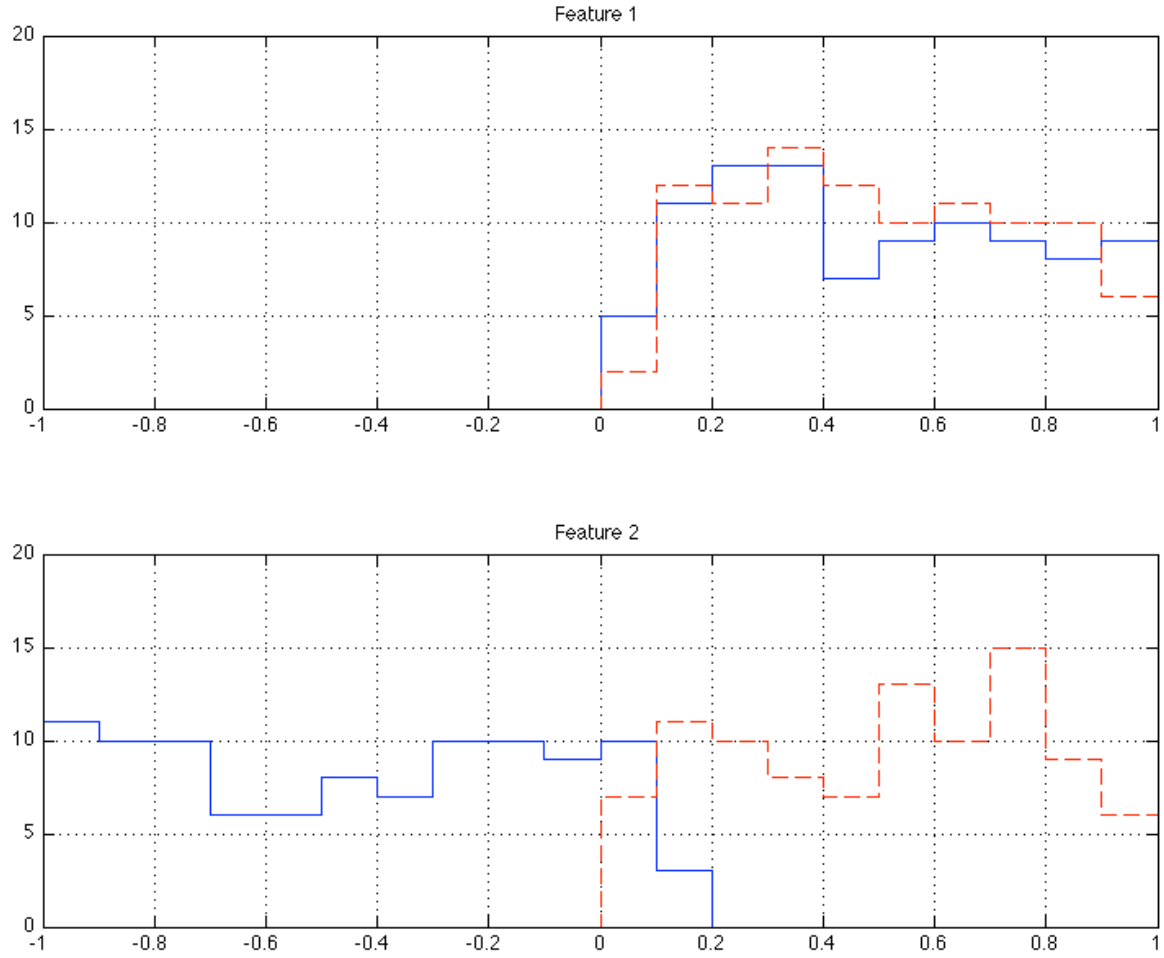
- It is the process by which we automatically assign an individual item to one of a number of categories or classes, based on its characteristics.

- In our case:
  - (1) the items are audio signals (e.g. sounds, songs, excerpts);
  - (2) their characteristics are the features we extract from them (MFCC, chroma, centroid);
  - (3) the classes (e.g. instruments, genres, chords) fit the problem definition

- The complexity lies in finding an appropriate relationship between features and classes

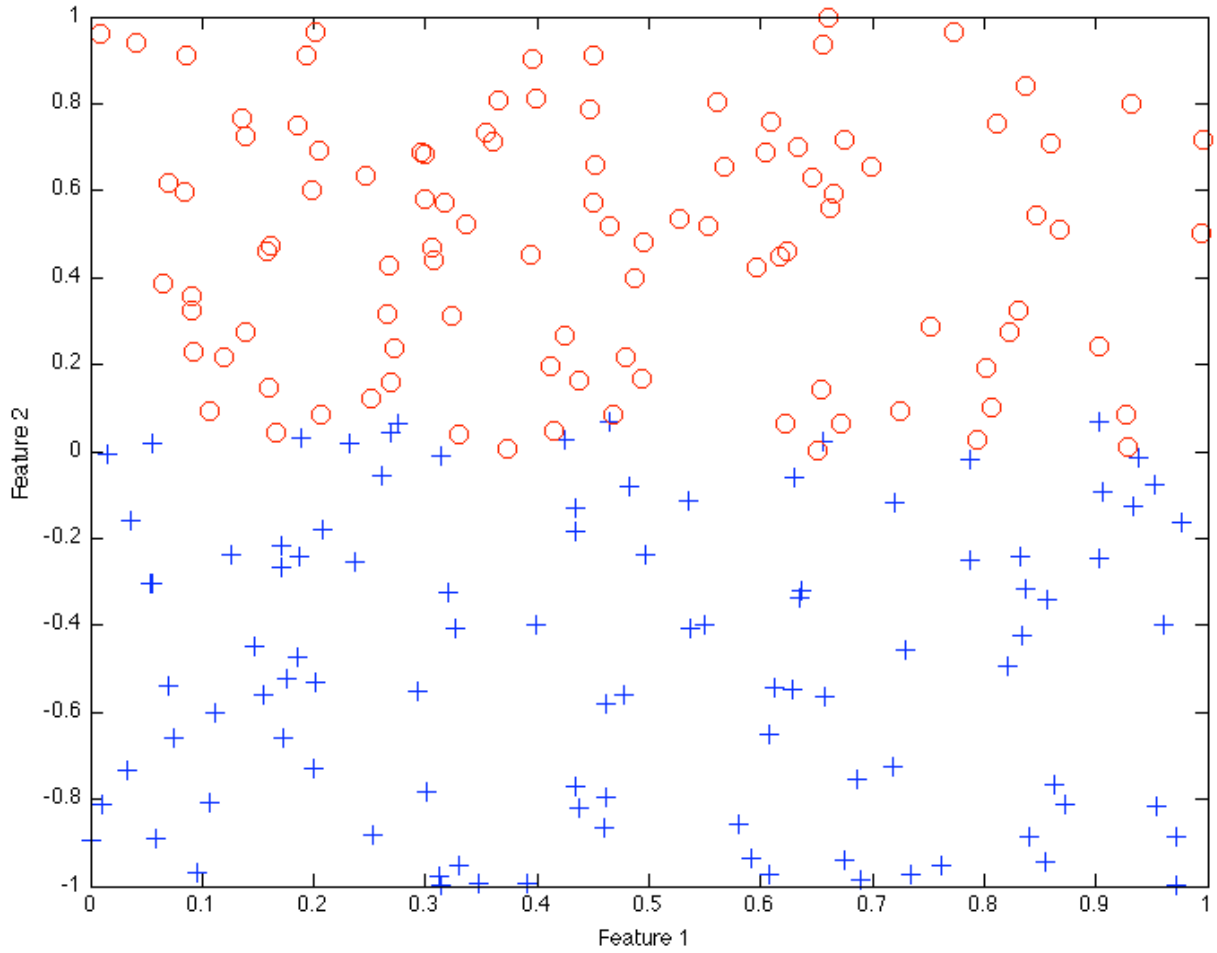
Example

- 200 sounds of 2 different kinds (red and blue); 2 features extracted per sound



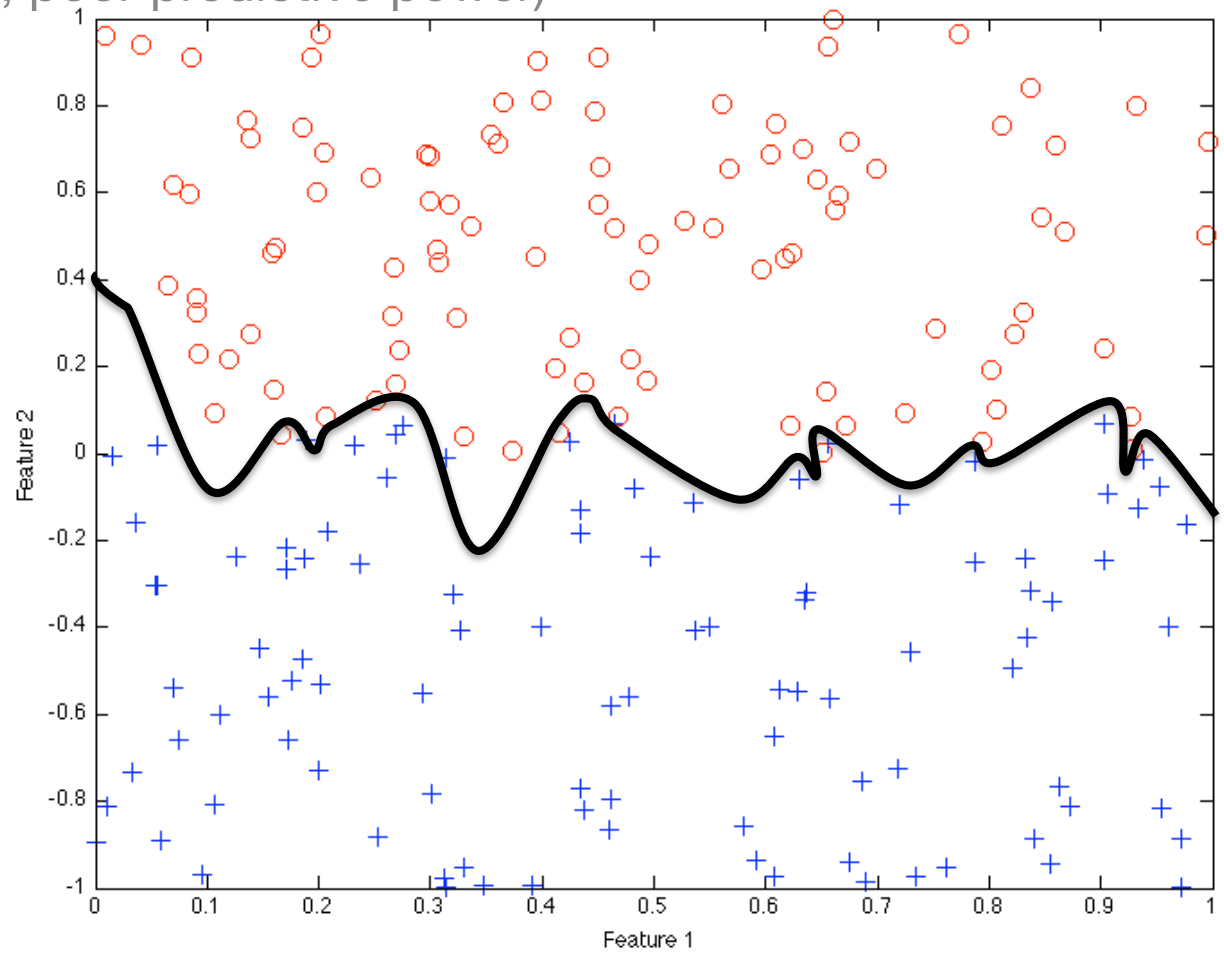
Example

- The 200 items in the 2-D feature space



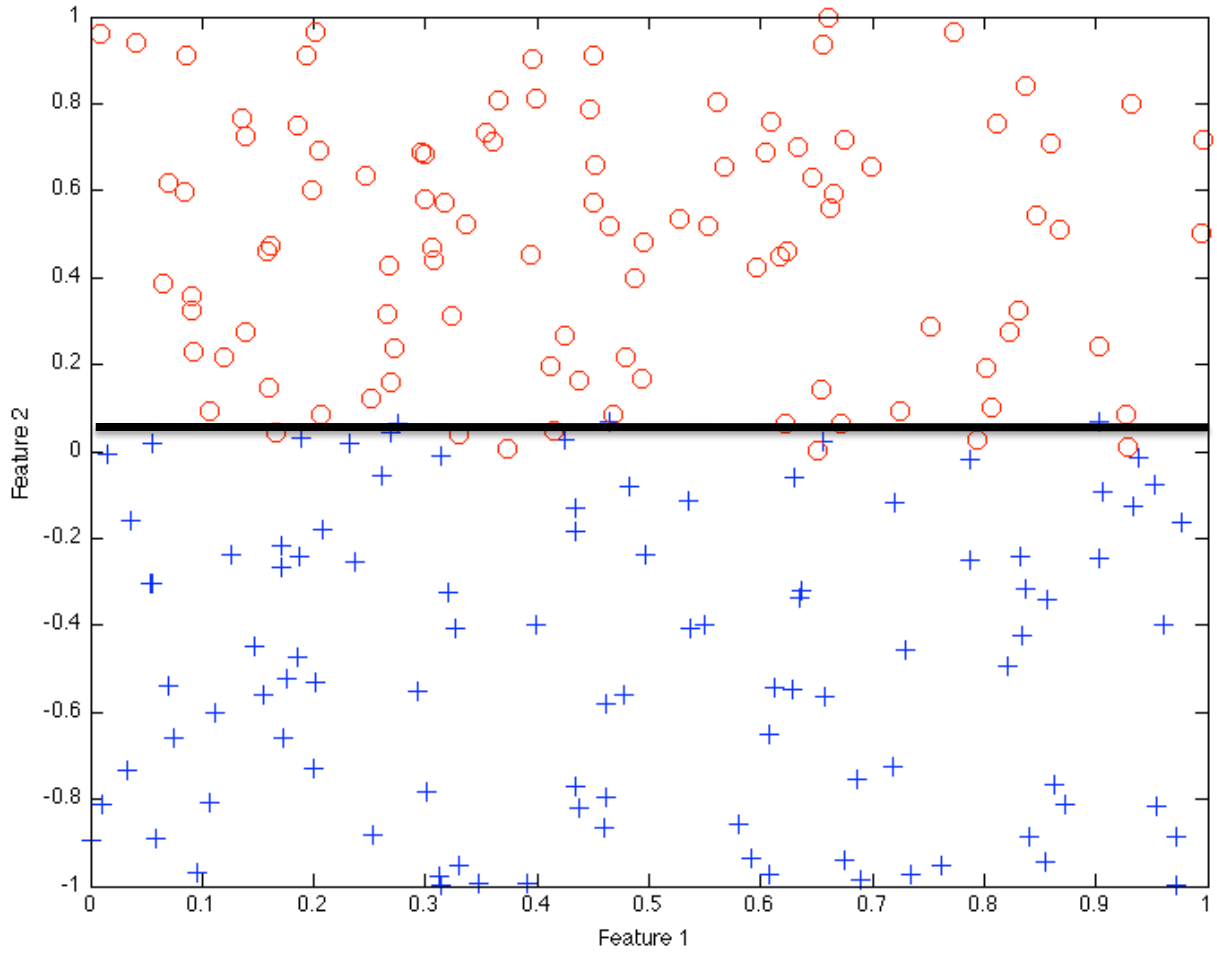
Example

- Boundary that optimizes performance -> risk of overfitting (excessive complexity, poor predictive power)



Example

- Generalization -> Able to correctly classify novel input

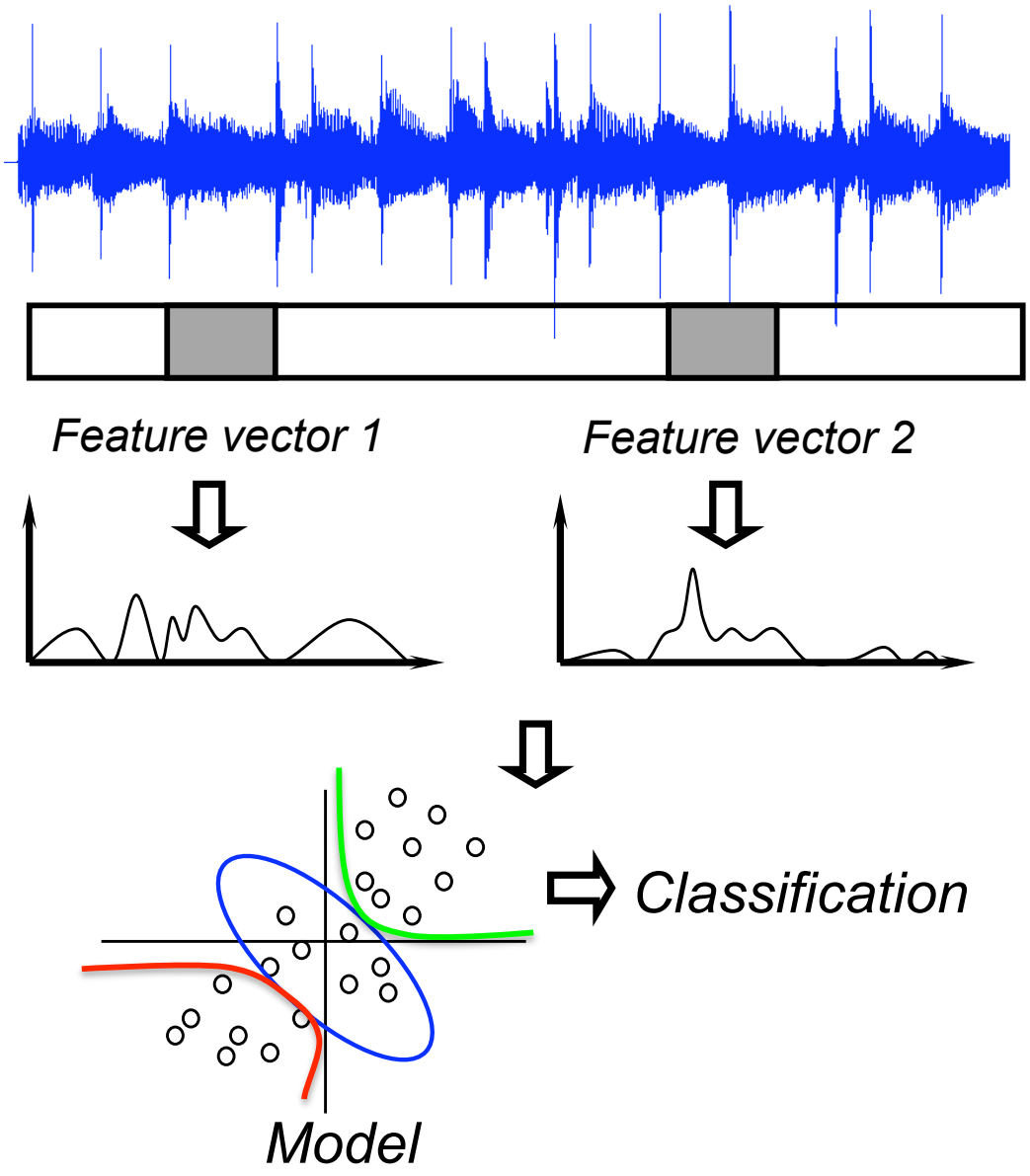


Classification of music signals

- A number of relevant MIR tasks:
    - Music Instrument Identification
    - Artist ID
    - Genre Classification
    - Music/Speech Segmentation
    - Music Emotion Recognition
    - Transcription of percussive instruments
    - Chord recognition
- Re-purposing of machine learning methods that have been successfully used in related fields (e.g. speech, image processing)

A music classifier

- Feature extraction: (1) feature computation; (2) summarization
- Pre-processing: (1) normalization; (2) feature selection
- Classification: (1) use sample data to estimate boundaries, distributions or class-membership; (2) classify new data based on these estimations



Feature set (recap)

- Feature extraction is necessary as audio signals carry too much redundant and/or irrelevant information
- They can be estimated on a frame by frame basis or within segments, sounds or songs.
- Many possible features: spectral, temporal, pitch-based, etc.
- A good feature set is a must for classification
- What should we look for in a feature set?

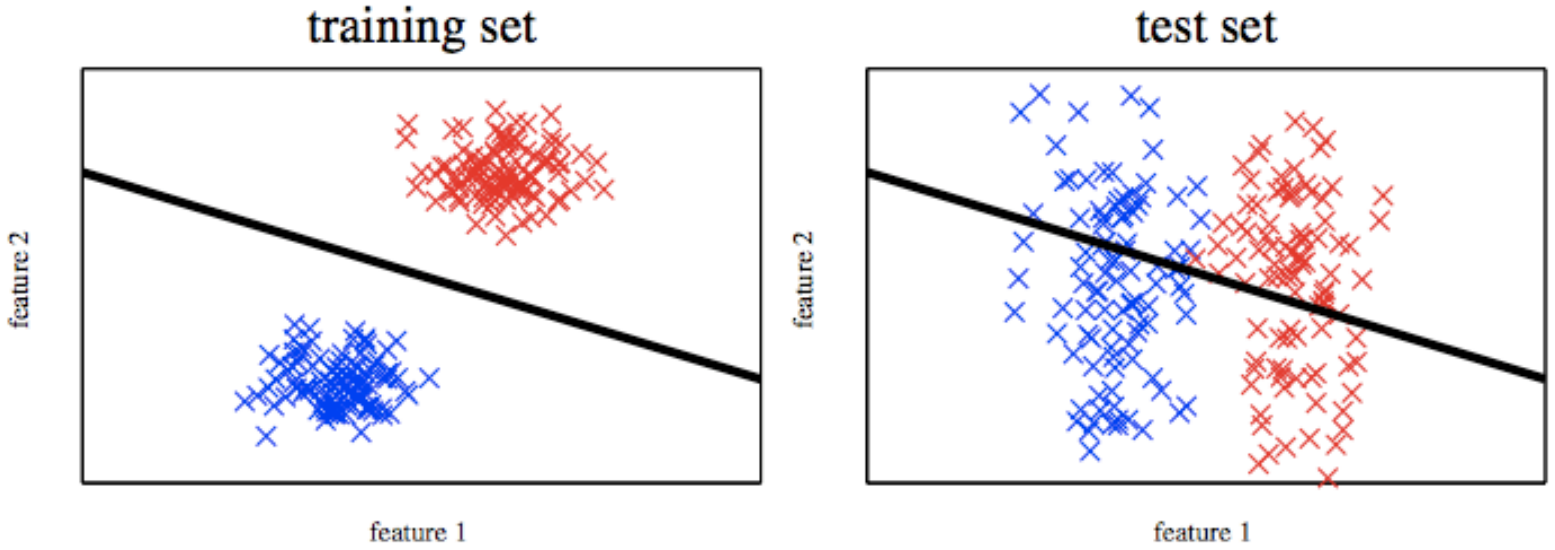
Feature set (what to look for?)

- A few issues of feature design/choice:
- (1) can be robustly estimated from audio (e.g. spectral envelope vs onset rise times in polyphonies)
- (2) relevant to classification task (e.g. MFCC vs chroma for instrument ID) -> noisy features make classification more difficult!
- Classes are never fully described by a point in the feature space but by the distribution of a sample population

Features and training

- Class models must be learned on many sounds/songs to properly account for between/within class variations

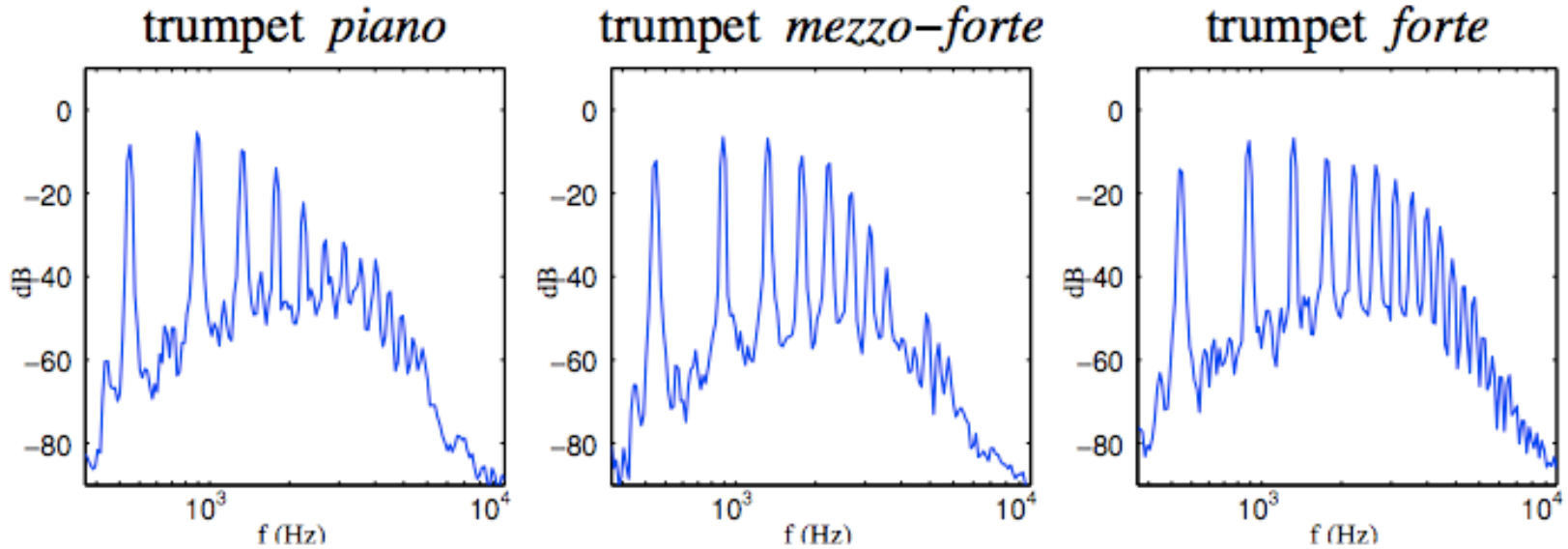
- The natural range of features must be well represented on the sample population



- Failure to do so leads to overfitting: training data only covers a sub-region of its natural range and class models are inadequate for new data.

Feature set (what to look for?)

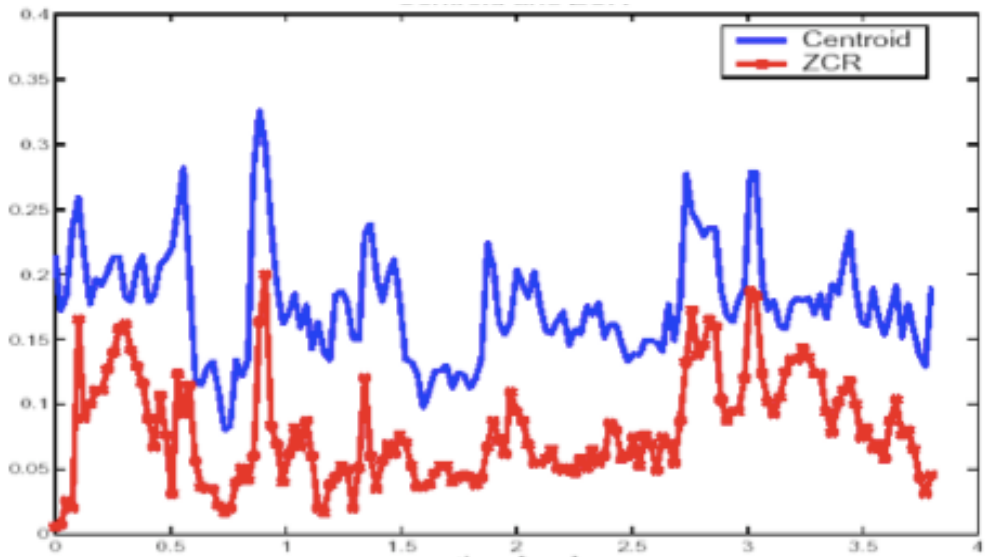
- We expect variability within sound classes
- For example: trumpet sounds change considerably between, e.g. different loudness levels, pitches, instruments, playing style or recording conditions



- (3) feature set should be as invariant as possible to those changes

Feature set (what to look for?)

- (4) low(er) dimensional feature space -> Classification becomes more difficult as the dimensionality of the feature space increases.

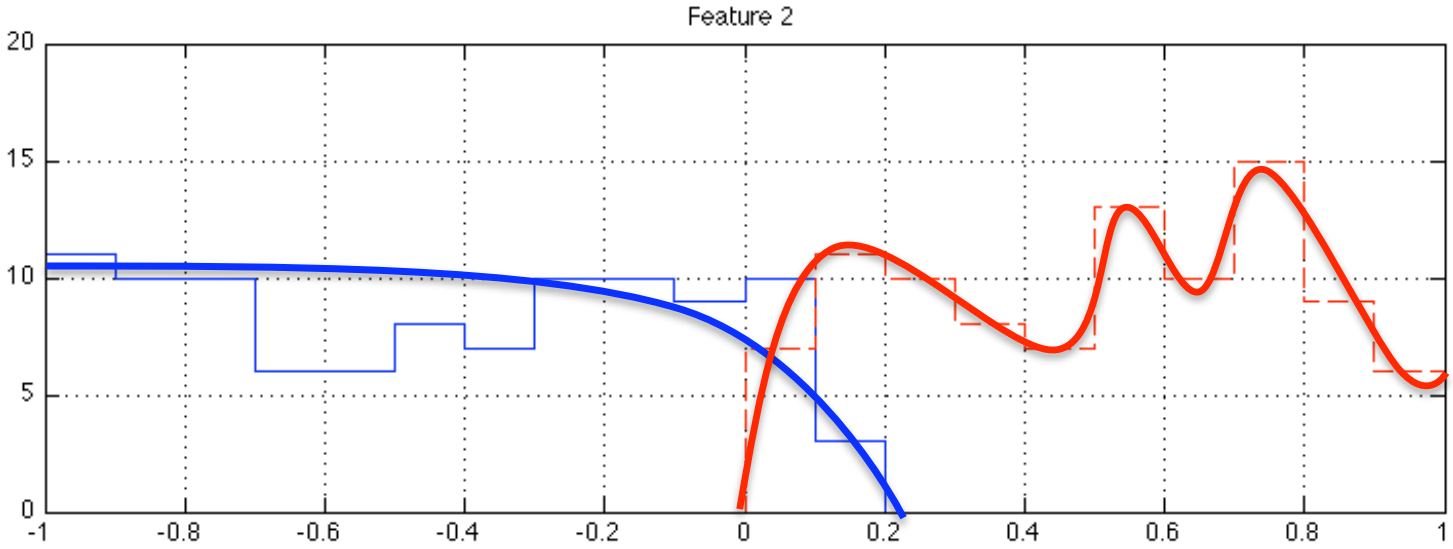


- (5) as free from redundancies (strongly correlated features) as possible

- (6) discriminative power: good features result in separation between classes and grouping within classes

Feature distribution

- Remember the histograms of our example. They describe the behavior of features across our sample population.



- It is desirable to parameterize this behavior

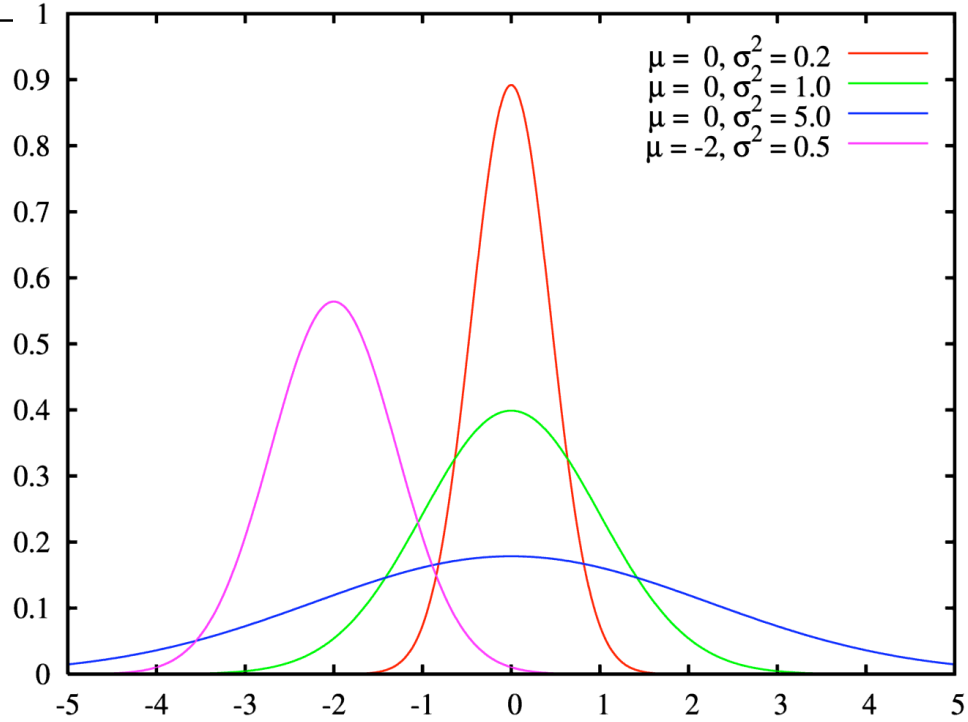
Feature distribution

- A Normal or Gaussian distribution is a bell-shaped probability density function defined by two parameters, its mean (μ) and variance (σ^2):

$$\mathcal{N}(x_l; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x_l-\mu)^2}{2\sigma^2}}$$

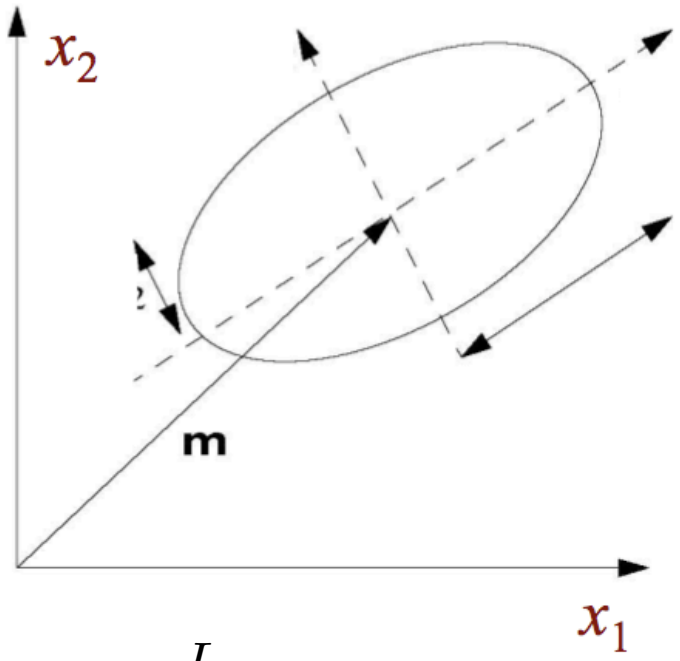
$$\mu = \frac{1}{L}\sum_{l=1}^{L} x_l$$

$$\sigma^2 = \frac{1}{L}\sum_{l=1}^{L}(x_l - \mu)^2$$



Feature distribution

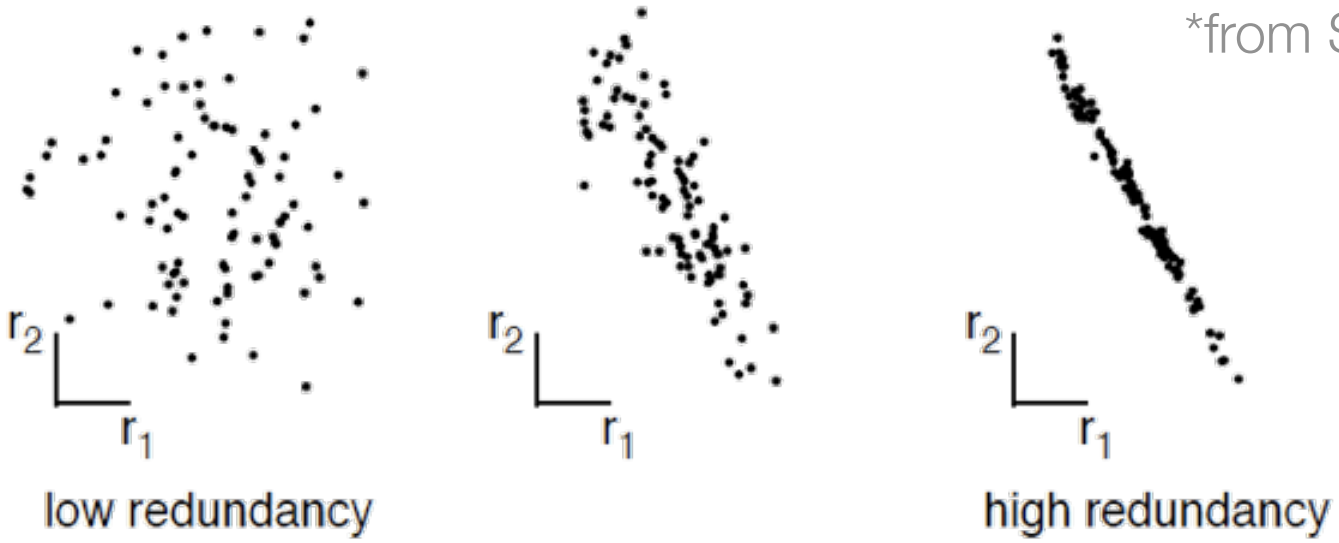
- In D-dimensions, the distribution becomes an ellipsoid defined by a D-dimensional mean vector and a DxD covariance matrix:



$$C_x = \frac{1}{L}\sum_{l=1}^{L}(\mathbf{x}_l - \mu)(\mathbf{x}_l - \mu)^T$$

Feature distribution

- Cx is a square symmetric DxD matrix: diagonal components are the feature variances; off-diagonal terms are their co-variances



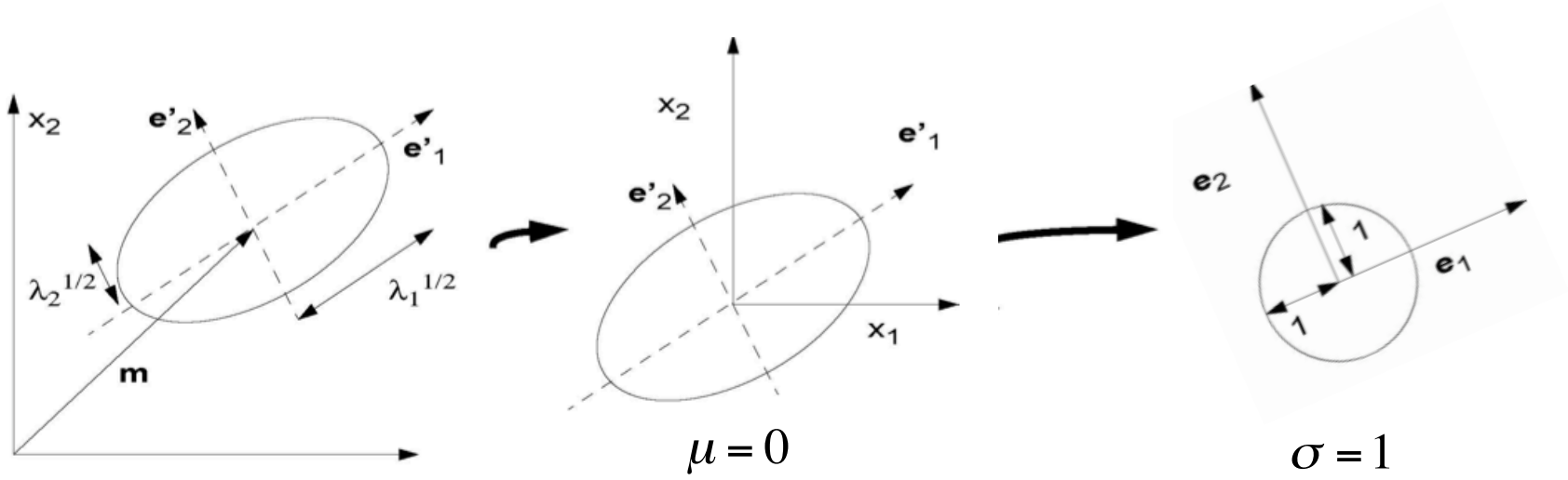
*from Shlens, 2009

- High covariance between features shows as a narrow ellipsoid (high redundancy)

Data normalization

- To avoid bias towards features with wider range, we can normalize all to have zero mean and unit variance:

$$\hat{x} = (x - \mu)/\sigma$$



PCA

- Complementarily we can minimize redundancies by applying Principal Component Analysis (PCA)

- Let us assume that there is a linear transformation A, such that:

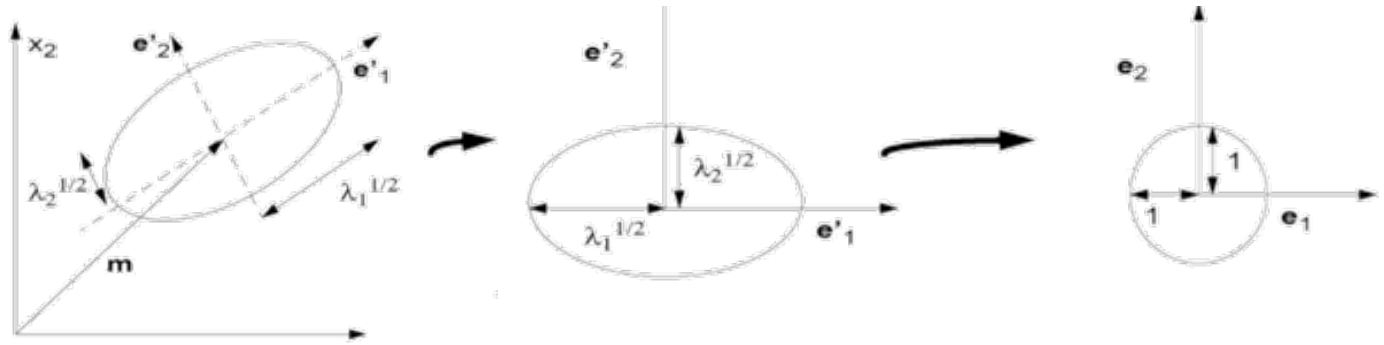
$$Y = AX$$

$$\begin{bmatrix} \mathbf{a}_1 \cdot \mathbf{x}_1 & \cdots & \mathbf{a}_1 \cdot \mathbf{x}_L \\ \vdots & \ddots & \vdots \\ \mathbf{a}_D \cdot \mathbf{x}_1 & \cdots & \mathbf{a}_D \cdot \mathbf{x}_L \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_D \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_L \end{bmatrix}$$

- Where $\mathbf{x}_I$ are the D-dimensional feature vectors (after mean removal) such that: $C_x = XX^T/L$

PCA

- What do we want from Y:
    - Decorrelated: All off-diagonal elements of C_y should be zero
    - Rank-ordered: according to variance
    - Unit variance
- A -> orthonormal matrix; rows = principal components of X



PCA

- How to choose A?

$$C_y = \frac{1}{L} Y Y^T = \frac{1}{L} (AX)(AX)^T = A \left(\frac{1}{L} X X^T \right) A^T = A C_x A^T$$

- Any symmetric matrix (such as C_x) is diagonalized by an orthogonal matrix E of its eigenvectors

- For a linear transformation Z, an eigenvector e_i is any non-zero vector that satisfies:

$$Z e_i = \lambda_i e_i$$

- Where λ_i is a scalar known as the eigenvalue

- PCA chooses $A = E^T$, a matrix where each row is an eigenvector of C_x

PCA

- In MATLAB:

```
function [signals,PC,V] = pca1(data)
% PCA1: Perform PCA using covariance.
%     data - MxN matrix of input data
%            (M dimensions, N trials)
%  signals - MxN matrix of projected data
%       PC - each column is a PC
%        V - Mx1 matrix of variances

[M,N] = size(data);

% subtract off the mean for each dimension
mn =  mean(data,2);
data = data - repmat(mn,1,N);

% calculate the covariance matrix
covariance = 1 / (N-1) * data * data';

% find the eigenvectors and eigenvalues
[PC, V] = eig(covariance);

% extract diagonal of matrix as vector
V = diag(V);

% sort the variances in decreasing order
[junk, rindices] = sort(-1*V);
V  = V(rindices);
PC = PC(:,rindices);

% project the original data set
signals = PC' * data;
```

From http://www.snl.salk.edu/~shlens/pub/notes/pca.pdf

Dimensionality reduction

- Furthermore, PCA can be used to reduce the number of features:
- Since A is ordered according to eigenvalue λ_i from high to low
- We can then use an MxD subset of this reordered matrix for PCA, such that the result corresponds to an approximation using the M most relevant feature vectors
- This is equivalent to projecting the data into the few directions that maximize variance
- We do not need to choose between correlating (redundant) features, PCA chooses for us.
- Can be used,e.g., to visualize high-dimensional spaces

Discrimination

- Let us define:

Proportion of occurrences of class k in the sample

$$\underbrace{S_w}_{\text{Within-class scatter matrix}} = \sum_{k=1}^{K} \overbrace{(L_k/L)}^{} \underbrace{C_k}_{\text{Covariance matrix for class k}}$$

global mean

$$\underbrace{S_b}_{\text{Between-class scatter matrix}} = \sum_{k=1}^{K} (L_k/L)(\mu_k - \overbrace{\mu}^{})(\underbrace{\mu_k}_{\text{Mean of class k}} - \mu)^T$$

Discrimination

- Trace{U} is the sum of all diagonal elements of U, s.t.:
- Trace{S_w} measures average variance of features across all classes
- Trace{S_b} measures average distance between class means and global mean across all classes
- The discriminative power of a feature set can be measured as:

$$J_0 = \frac{\mathrm{trace}\{S_b\}}{\mathrm{trace}\{S_w\}}$$

- High when samples from a class are well clustered around their mean (small trace{S_w}), and/or when different classes are well separated (large trace{S_b}).

Feature selection

- But how to select an optimal subset of M features from our D-dimensional space that maximizes class separability?
- We can try all possible M-long feature combinations and select the one that maximizes J_0 (or any other class separability measure)
- In practice this is unfeasible as there are too many possible combinations
- We need either a technique to scan through a subset of possible combinations, or a transformation that re-arranges features according to their discriminative properties

Feature selection

- Sequential backward selection (SBS):

    1. Start with F = D features.
    2. For each combination of F-1 features compute J_0
    3. Select the combination that maximizes J_0
    4. Repeat steps 2 and 3 until F = M

- Good for eliminating bad features; nothing guarantees that the optimal (F-1)-dimensional vector has to originate from the optimal F-dimensional one.

- Nesting: once a feature has been discarded it cannot be reconsidered

Feature selection

- Sequential forward selection (SFS):

    1. Select the individual feature (F = 1) that maximizes J_0
    2. Create all combinations of F+1 features including the previous winner and compute J_0
    3. Select the combination that maximizes J_0
    4. Repeat steps 2 and 3 until F = M

- Nesting: once a feature has been selected it cannot be discarded

LDA

- An alternative way to select features with high discriminative power is to use linear discriminant analysis (LDA)
- LDA is similar to PCA, but the eigenanalysis is performed on the matrix $S_w^{-1}S_b$ instead of C_x
- Like in PCA, the transformation matrix A is re-ordered according to the eigenvalues λ_i from high to low
- Then we can use only the top M rows of A, where M < rank of $S_w^{-1}S_b$
- LDA projects the data into a few directions maximizing class separability

Classification

- We have:
    - A taxonomy of classes
    - A representative sample of the signals to be classified
    - An optimal set of features
- Goals:
    - Learn class models from the data
    - Classify new instances using these models
- Strategies:
    - Supervised: models learned by example
    - Unsupervised: models are uncovered from unlabeled data

Instance-based learning

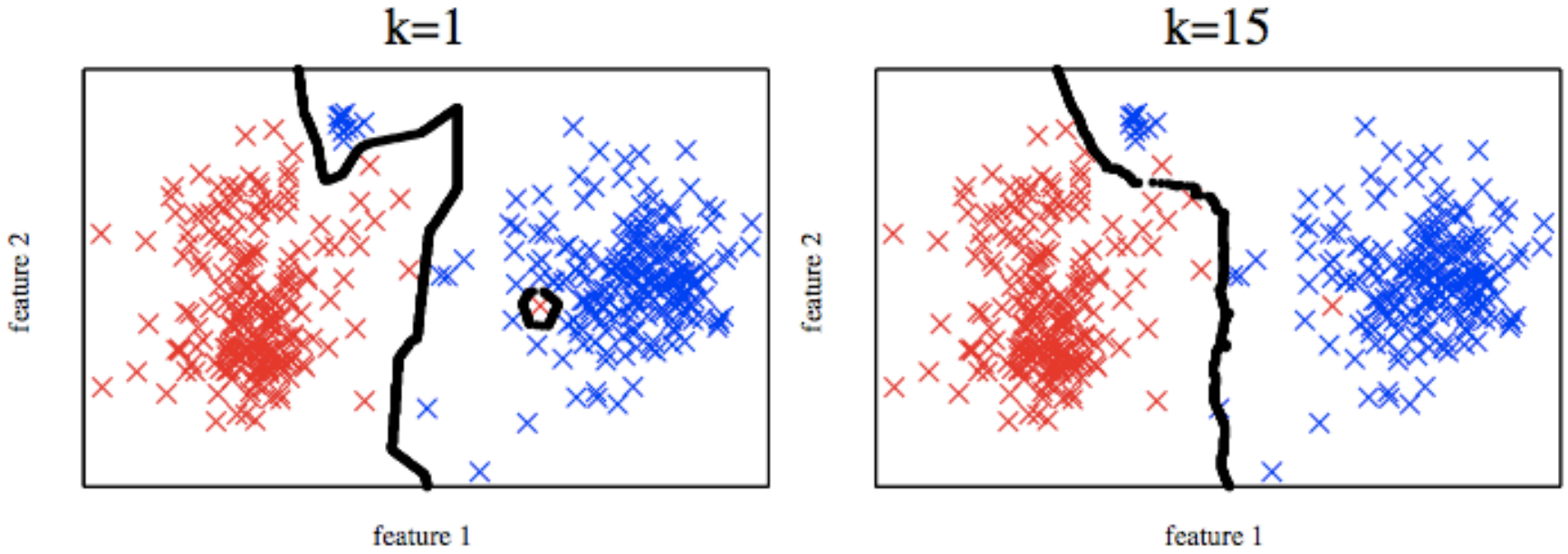
- Simple classification can be performed by measuring the distance between instances.
- Nearest-neighbor classification:
  - Measures distance between new sample and all samples in the training set
  - Selects the class of the closest training sample
- k-nearest neighbors (k-NN) classifier:
  - Measures distance between new sample and all samples in the training set
  - Identifies the k nearest neighbors
  - Selects the class that was more often picked.

Instance-based learning

- In both these cases, training is reduced to storing the labeled training instances for comparison
- Known as “lazy” or “memory-based” learning.
- All computations are performed during classification
- Complexity increases with number of training instances.
- Alternatively, we can store only a few class prototypes/models (e.g. class centroids)

Instance-based learning

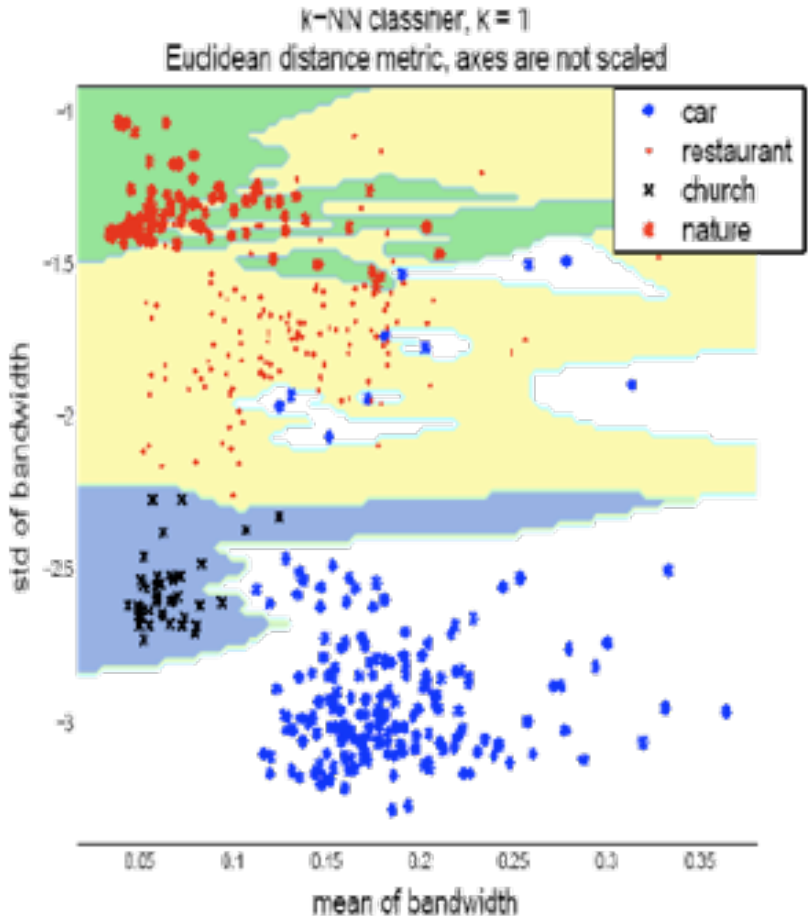
- We need to choose k to avoid overfitting, e.g., k = √L where L is the number of training samples

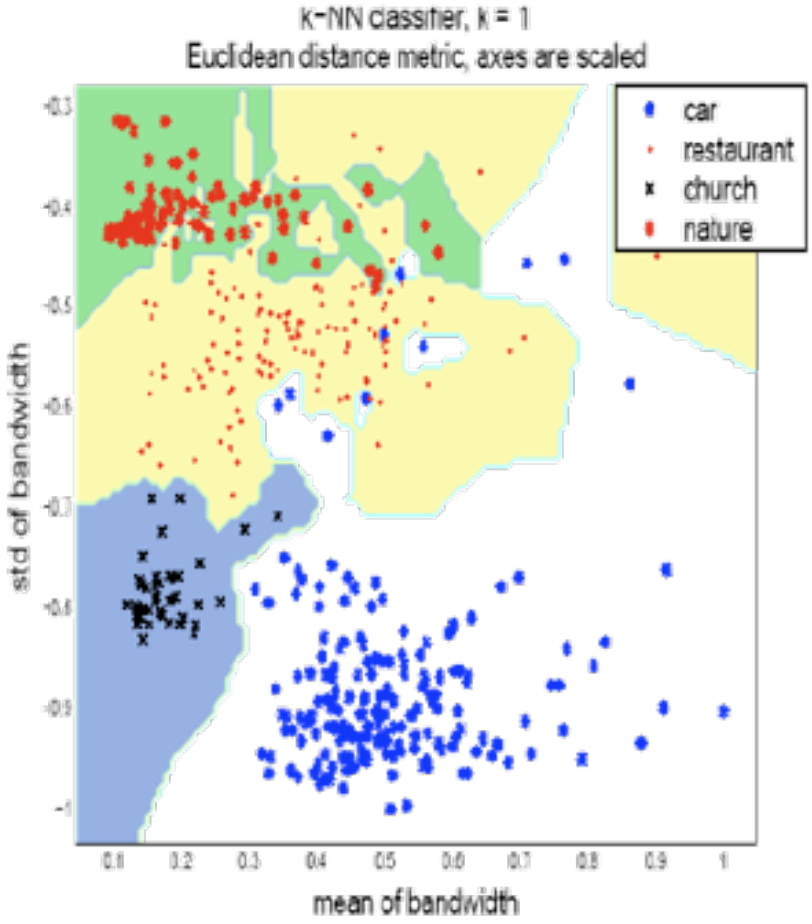


- Works well for well-separated classes and an appropriate distance metric and/or pre-processing of features

Instance-based learning

- The effect of standardization (from Peltonen's MSc thesis, 2001)

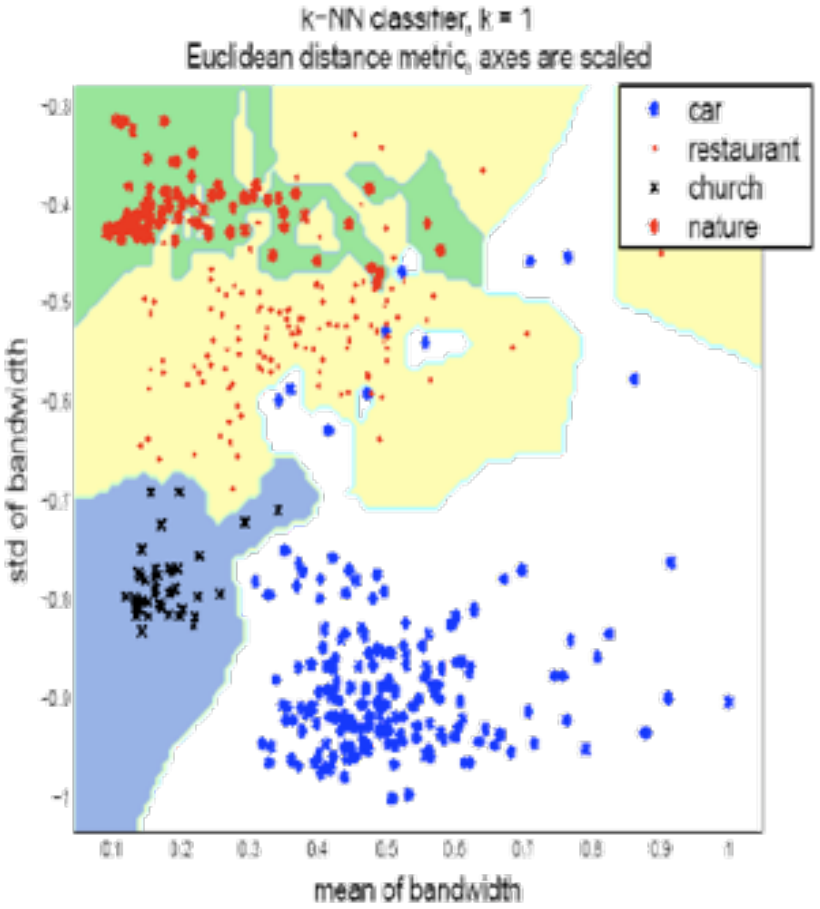


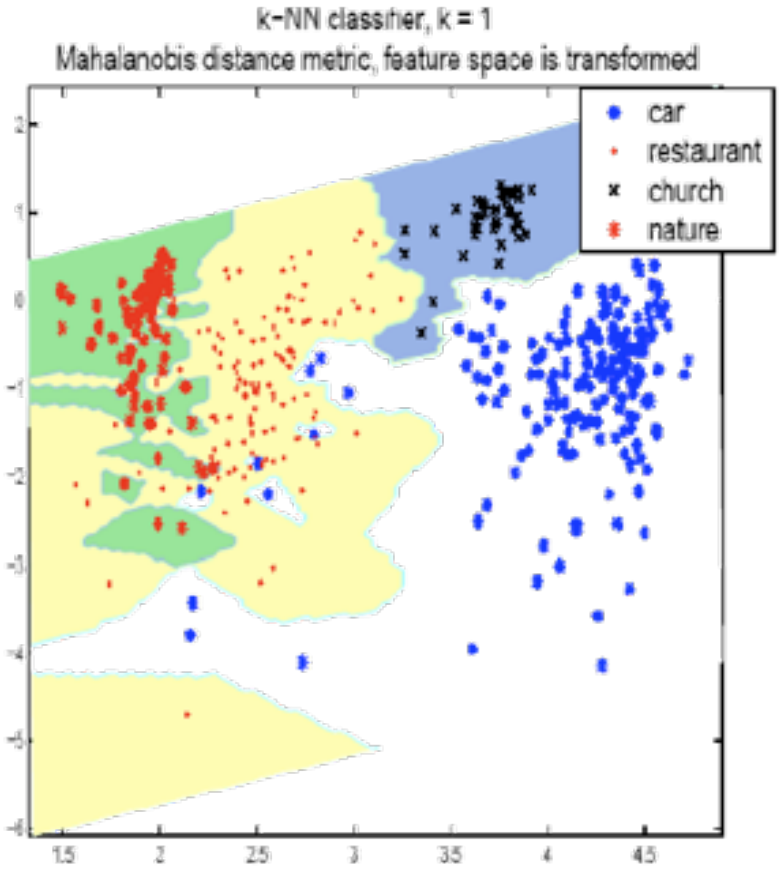


Instance-based learning

- Mahalanobis distance: considers the underlying distribution

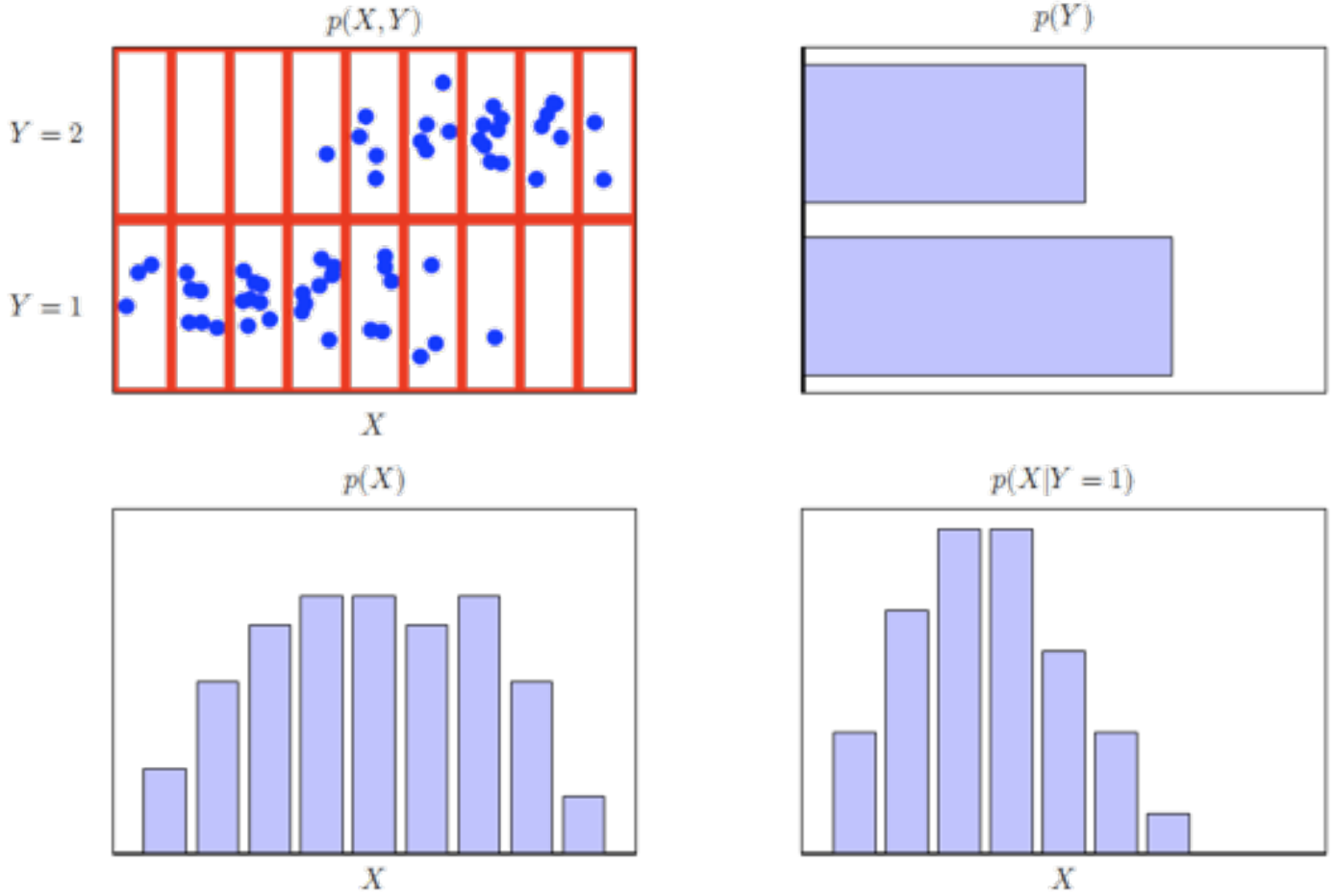
$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T C^{-1} (\mathbf{x} - \mathbf{y})}$$





Probability

- Let us assume that the observations X and classes Y are random variables



*From Bishop's Machine Learning book, 2007

Probability

$L = \text{total number of blue dots}$
$c_i = \text{number of dots in column } i$
$r_j = \text{number of dots in row } j$
$n_{ij} = \text{number of dots in cell } ij$

Joint Probability: $P(X, Y) = P(Y, X) = \frac{n_{ij}}{L}$ — Symmetry rule

Marginal Probability: $P(X) = \frac{c_i}{L} = \sum_j P(X, Y)$ — Sum rule

Conditional Probability: $P(Y|X) = \frac{n_{ij}}{c_i}$

$$P(X, Y) = \frac{n_{ij}}{L} = \frac{n_{ij}}{c_i}\frac{c_i}{L} = \underbrace{P(Y|X)P(X)}$$

Product rule

Probability

- Thus we can derive Bayes' theorem as:

Posterior: probability of class i given an observation x

Likelihood: Probability of observation x, given class i

$$P(\text{class}_i|x) = \frac{P(x|\text{class}_i)P(\text{class}_i)}{P(x)}$$

Prior: Probability of class i

Marginal Likelihood: Normalizing constant (same for all classes) that ensures posterior adds to 1

$$P(x) = \sum_i P(x|\text{class}_i)P(\text{class}_i)$$

Probabilistic Classifiers

- Classification: finding the class with the highest probability given the observation x

- Find i that maximizes the posterior probability P(class_i|x) -> Maximum A Posteriori (MAP)

- Since P(x) is the same for all classes, this is equivalent to:

$$\underset{i}{\text{argmax}}[P(x|\text{class}_i)P(\text{class}_i)]$$

- From the training data we can learn the likelihood P(x|class_i) and the prior P(class_i)

Gaussian Mixture Model

- We can model (parameterize) the likelihood using a Gaussian Mixture Model (GMM) -> the weighted sum of K multidimensional Gaussian distributions:

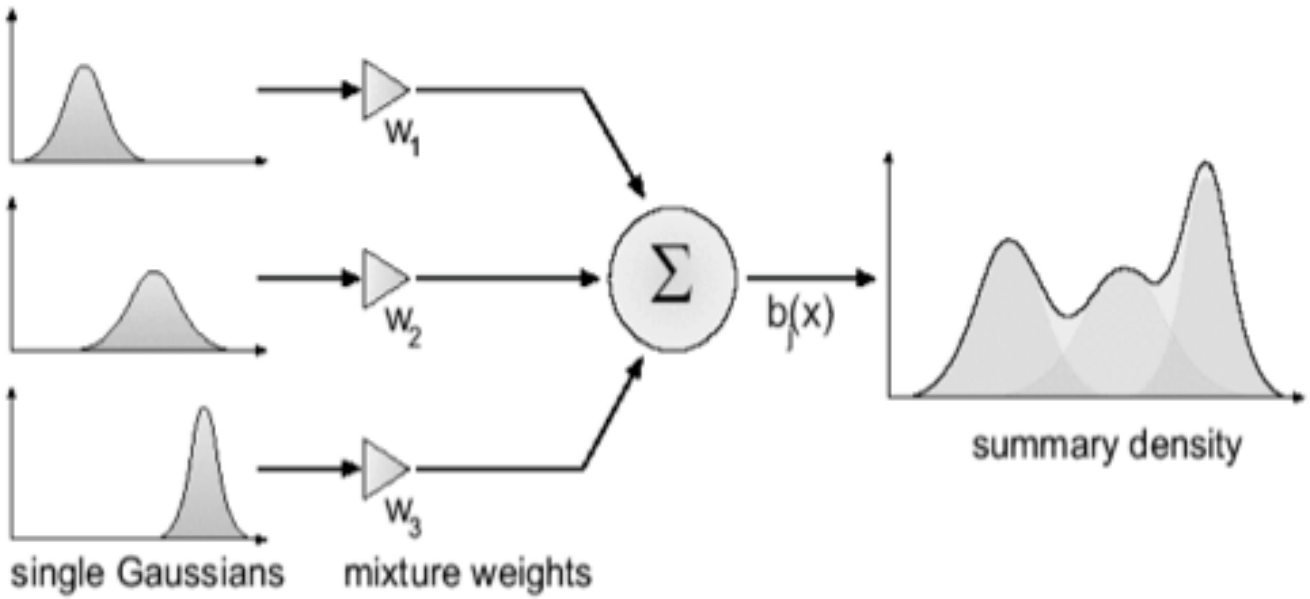
$$P(x|\text{class}_i) = \sum_{k=1}^{K} w_{ik}\mathcal{N}(x; \mu_{ik}, C_{ik})$$

- Where w_{ik} are the mixing weights and:

$$\mathcal{N}(\mathbf{x}; \mu, \mathbf{C_x}) = \frac{1}{(2\pi)^{D/2}|\mathbf{C_x}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu)^T \mathbf{C_x}^{-1}(\mathbf{x}-\mu)}$$

Gaussian Mixture Model

- 1-D GMM (Heittola, 2004)



- With a sufficiently large K a GMM can approximate any distribution
- However, increasing K increases the complexity of the model and compromises its ability to generalize

Gaussian Mixture Model

- The model is parametric, consisting of K weights, mean vectors and covariance matrices for every class.

- If features are decorrelated, then we can use diagonal covariance matrices, thus considerably reducing the number of parameters to be estimated (common approximation)

- Parameters can be estimated using the Expectation-Maximization (EM) algorithm (Dempster et al, 1977).

- EM is an iterative algorithm whose objective is to find the set of parameters that maximizes the likelihood.

K-means

- Dataset: L observations of a D-dimensional variable x

- Goal: find the partition into K clusters, each represented by a prototype μ_k, that minimizes the distortion:

$$J = \sum_{l=1}^{L} \sum_{k=1}^{K} r_{lk} \|\mathbf{x}_l - \mu_k\|_2^2$$

- where the “responsibility” function $r_{lk} = 1$ if the l^{th} observation is assigned to cluster k, 0 otherwise

- We don’t know the optimal r_{lk} and μ_k

K-means

1. Choose initial values for μ_k

2. E (expectation)-step: keeping μ_k fixed, minimize J with respect to r_{lk}

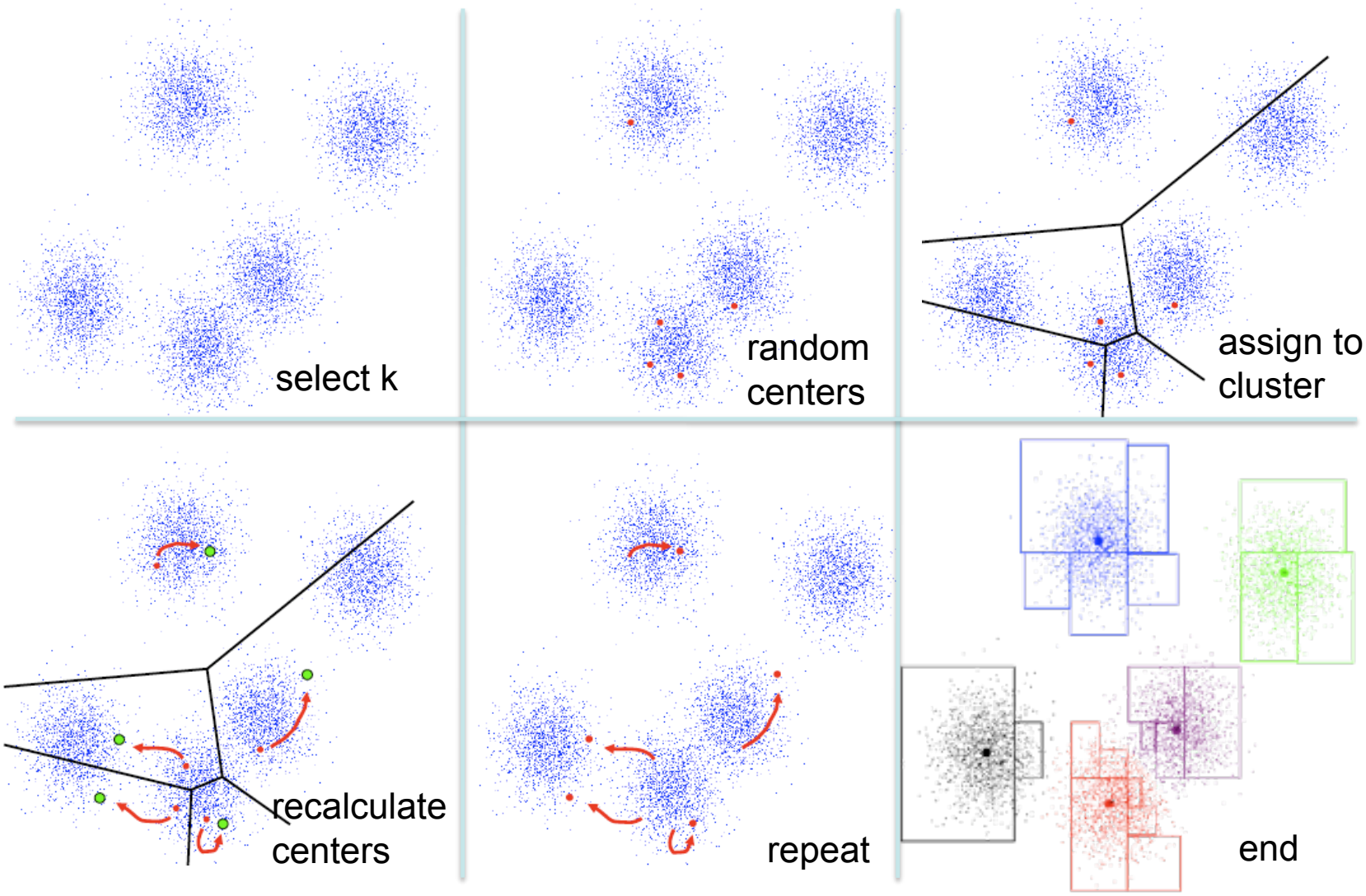
$$r_{lk} = \begin{cases} 1 & \text{if } \; k = \operatorname{argmin}_j \|\mathbf{x}_l - \mu_k\|_2^2 \\ 0 & \text{otherwise} \end{cases}$$

3. M (maximization)-step: keeping r_{lk} fixed, minimize J with respect to μ_k

$$\mu_k = \frac{\sum_l r_{lk}\mathbf{x}_l}{\sum_l r_{lk}}$$

4. repeat 2 and 3 until J or the parameters stop changing

K-means



*from http://www.autonlab.org/tutorials/kmeans11.pdf

K-means

- Many possible improvements (see, e.g. Dan Pelleg and Andrew Moore's work)
  - does not always converge to the optimal solution -> run k-means multiple times with different random initializations
  - sensitive to initial centers -> start with random datapoint as center; next center is farthest datapoint from closest center
  - sensitive to choice of K -> find the K that minimizes the Schwarz criterion (see Moore's tutorial):

$$\sum_{l}^{L} \|\mathbf{x}_l^{(k)} - \mu_k\|_2^2 + \lambda(DK)logL$$

EM Algorithm

- GMM: each cluster corresponds to a weighted Gaussian

- Soft responsibility function: conditional probability of belonging to Gaussian k given observation x_l

$$\gamma_{lk} = \frac{w_k \mathcal{N}(\mathbf{x}_l; \mu_k, \mathbf{C_k})}{\sum_{j=1}^{K} w_j \mathcal{N}(\mathbf{x}_l; \mu_j, \mathbf{C_j})}$$

- Goal: find the parameters that maximize

$$\log\{p(X|\mu, \mathbf{C}, \mathbf{w})\} = \sum_{l=1}^{L} \log \left\{ \sum_{k=1}^{K} w_k \mathcal{N}(\mathbf{x}_l; \mu_k, \mathbf{C_k}) \right\}$$

EM Algorithm

1. Initialize μ_k, C_k and w_k

2. E-step: evaluate responsibilities Υ_{lk} using current parameters

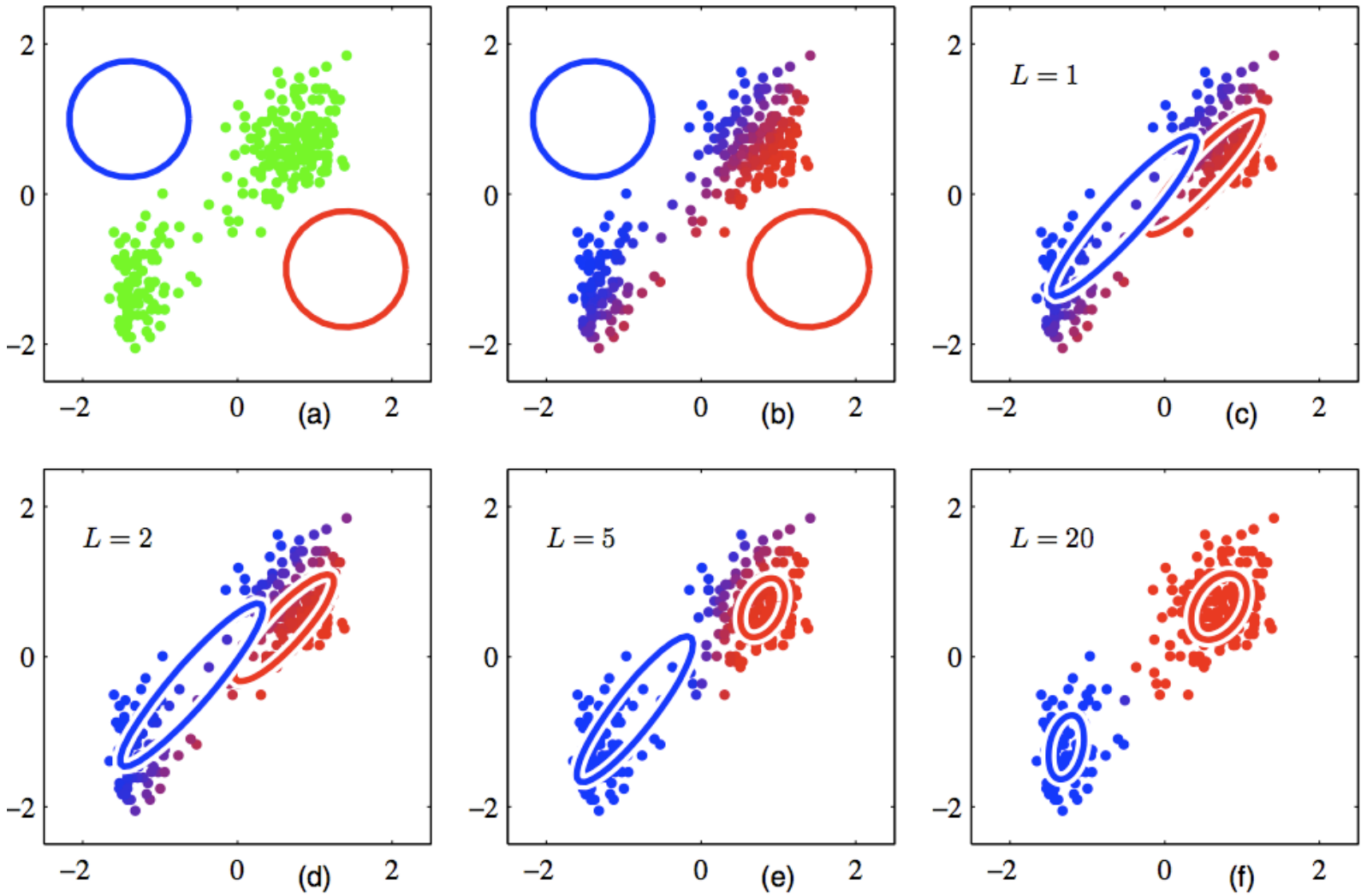
3. M-step: re-estimate parameters using current responsibilities

$$\mu_k^{\text{new}} = \frac{\sum_l \gamma_{lk} \mathbf{x}_l}{\sum_l \gamma_{lk}} \qquad w_k^{\text{new}} = \frac{\sum_l \gamma_{lk}}{L}$$

$$\mathbf{C}_k^{\text{new}} = \frac{\sum_l \gamma_{lk} (\mathbf{x}_l - \mu_k^{\text{new}})(\mathbf{x}_l - \mu_k^{\text{new}})^T}{\sum_l \gamma_{lk}}$$

4. repeat 2 and 3 until the log likelihood or the parameters stop changing

EM Algorithm



*From Bishop's Machine Learning book, 2007

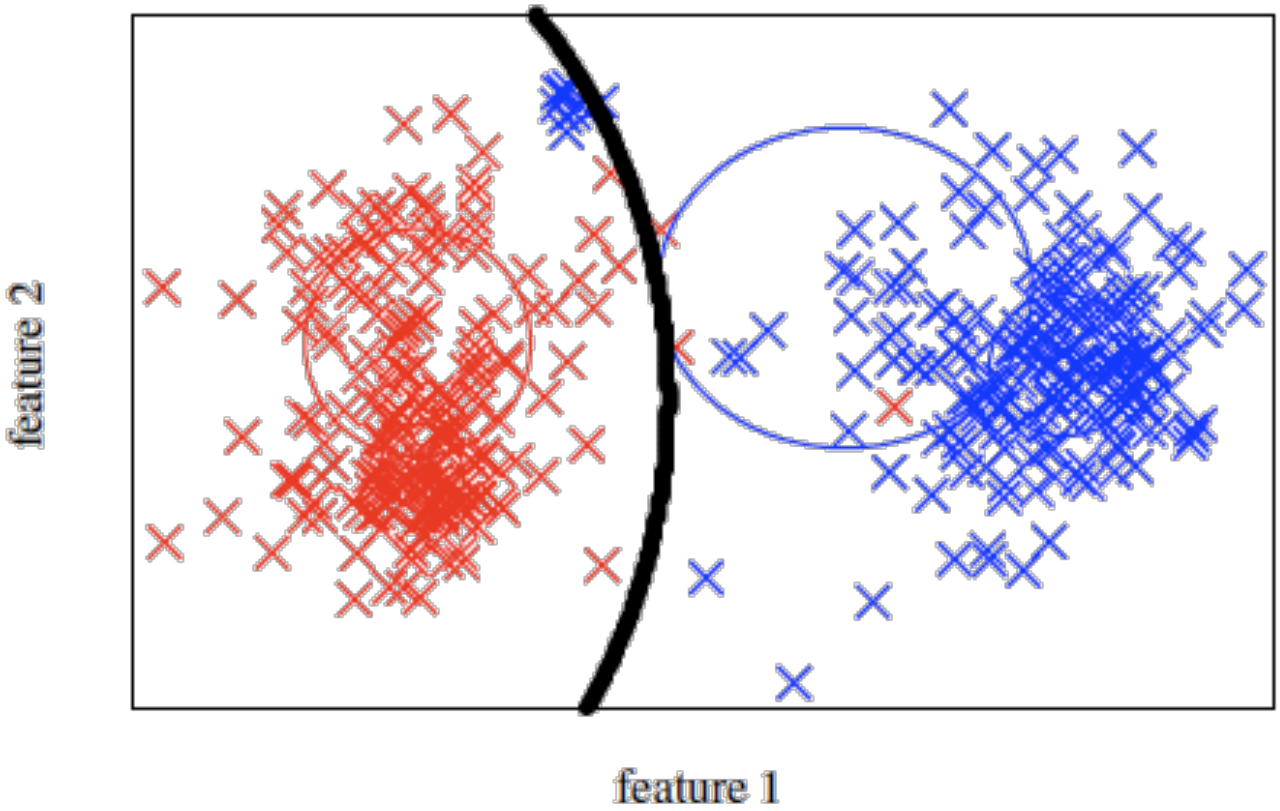
EM Algorithm

- EM is both more expensive and slower to converge than K-means
- Common trick: run K-means to initialize EM
    - Find cluster centers (means)
    - Compute sample covariances of the found clusters
    - Mixing weights -> fraction of L assigned to each cluster

MAP Classification

- After learning the likelihood and the prior during training, we can classify new instances based on MAP classification:

$$\underset{i}{\operatorname{argmax}}[P(x|\text{class}_i)P(\text{class}_i)]$$



References

- This lecture borrows heavily from Emmanuel Vincent's lecture notes on instrument classification (QMUL - Music Analysis and Synthesis) and from Anssi Klapuri's lecture notes on Audio Signal Classification (ISMIR 2004 Graduate School: http://ismir2004.ismir.net/graduate.html)
- Bishop, C.M. Pattern Recognition and Machine Learning. Springer (2007)
- Duda, R.O., Hart, P.E. and Stork, D.G. Pattern Classification (2nd Ed). John Wiley & Sons (2000)
- Witten, I. and Frank, E. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann (2005)
- Shlens, J. A Tutorial on Principal Component Analysis, Version 3.01 (2009): http://www.snl.salk.edu/~shlens/pca.pdf
- Moore, A. Statistical Data Mining Tutorials: http://www.autonlab.org/tutorials/