DAY 5

Intelligent Audio Systems:
A review of the foundations and applications of semantic audio analysis and music information retrieval

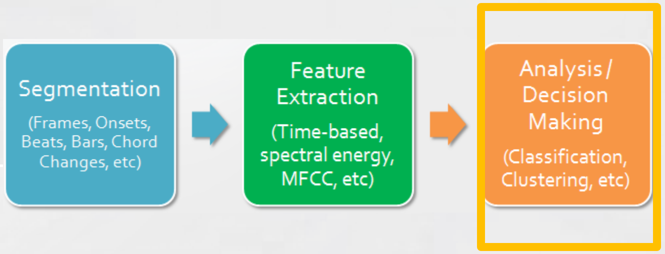
CCRMA
Center for Computer Research in Music and Acoustics

June 2012

Details

Send me your contact info: jay@izotope.com

CCRMA Tour (?)



ANALYSIS AND DECISION MAKING: GMMS

Mixture Models (GMM)

- K-means = hard clusters.
- GMM = soft clusters.

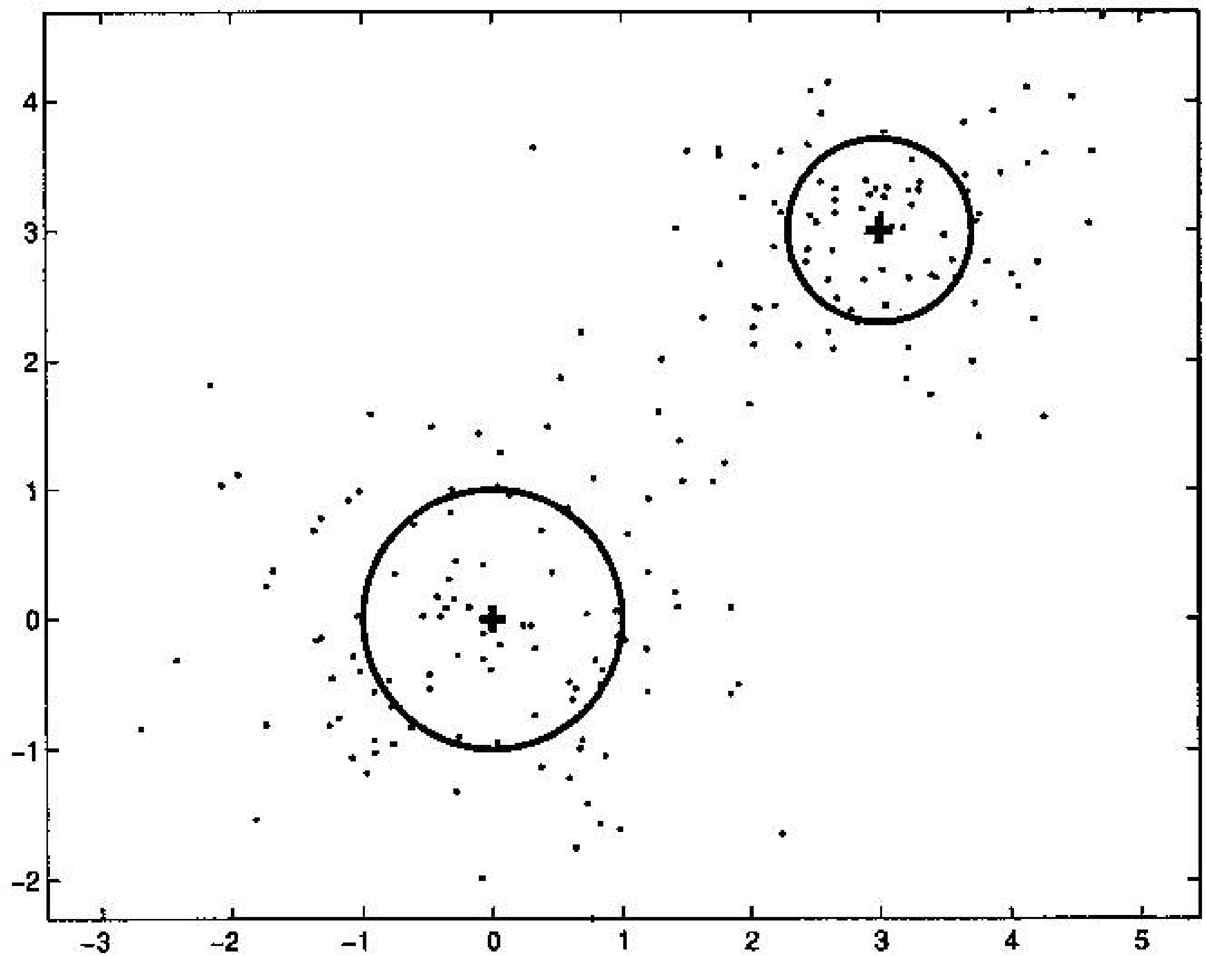


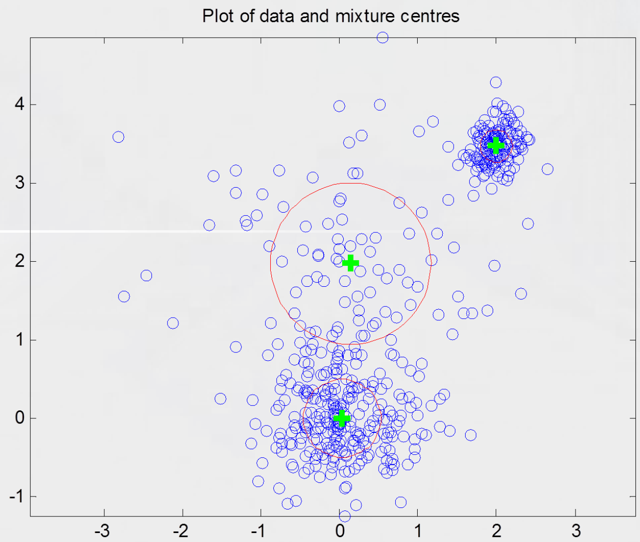
Fig. 3.1. Spherical covariance mixture model. Sampled data (*dots*), **centres** (*crosses*) and one standard deviation error bars (*lines*).

Mixture Models (GMM)

- GMM is good because:
  1. Can approximate any pdf with enough components
  2. EM makes it easy to find components parameters
     - EM - the means and variances adapt to fit the data as well as possible
  3. Compresses data considerably

- Can make softer decisions (decide further downstream given additional information)

GMM Parameters



Input

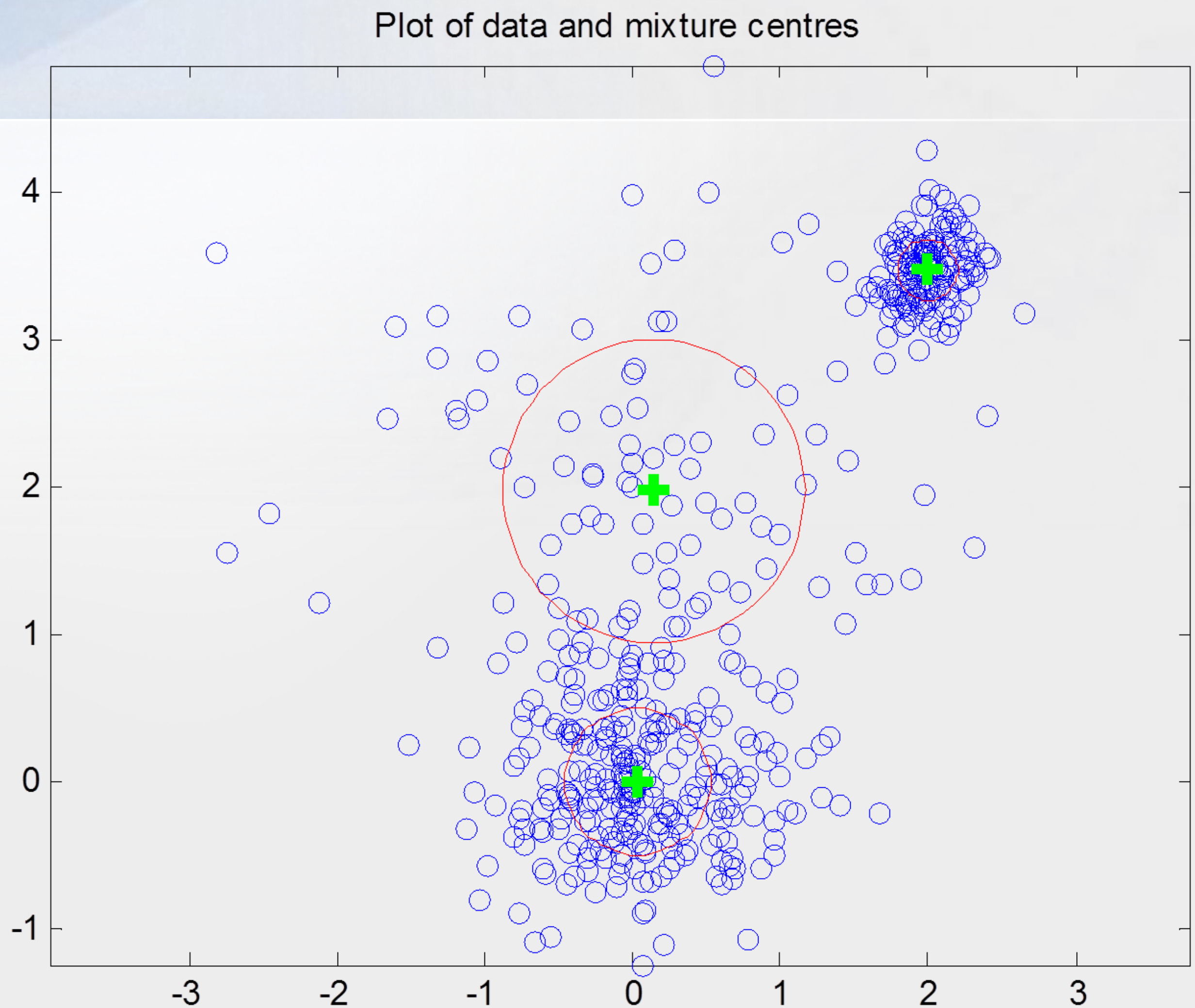
- Number of components (Gaussians)
  - e.g., 3
- Mixture coefficients (sum = 1)
  - e.g., [0.5 0.2 0.3]
  - "Priors" or "Prior probabilities"
  - Priors are "the ***original*** probability that each point came from a given mixture."
  - "A prior is often the purely subjective assessment of an experienced expert."
- Initialized centers, means, variances. (optional)

Output

- Component centers/means, variances, and mixture coeff.
- Posterior probabilities
  - "Posterior probabilities are the responsibilities which the Gaussian components have for each of the data points."

Query

- Obtain similarity via Likelihood


Plot of data and mixture centres
4
3
2
1
0
-1
-3
-2
-1
0
1
2
3

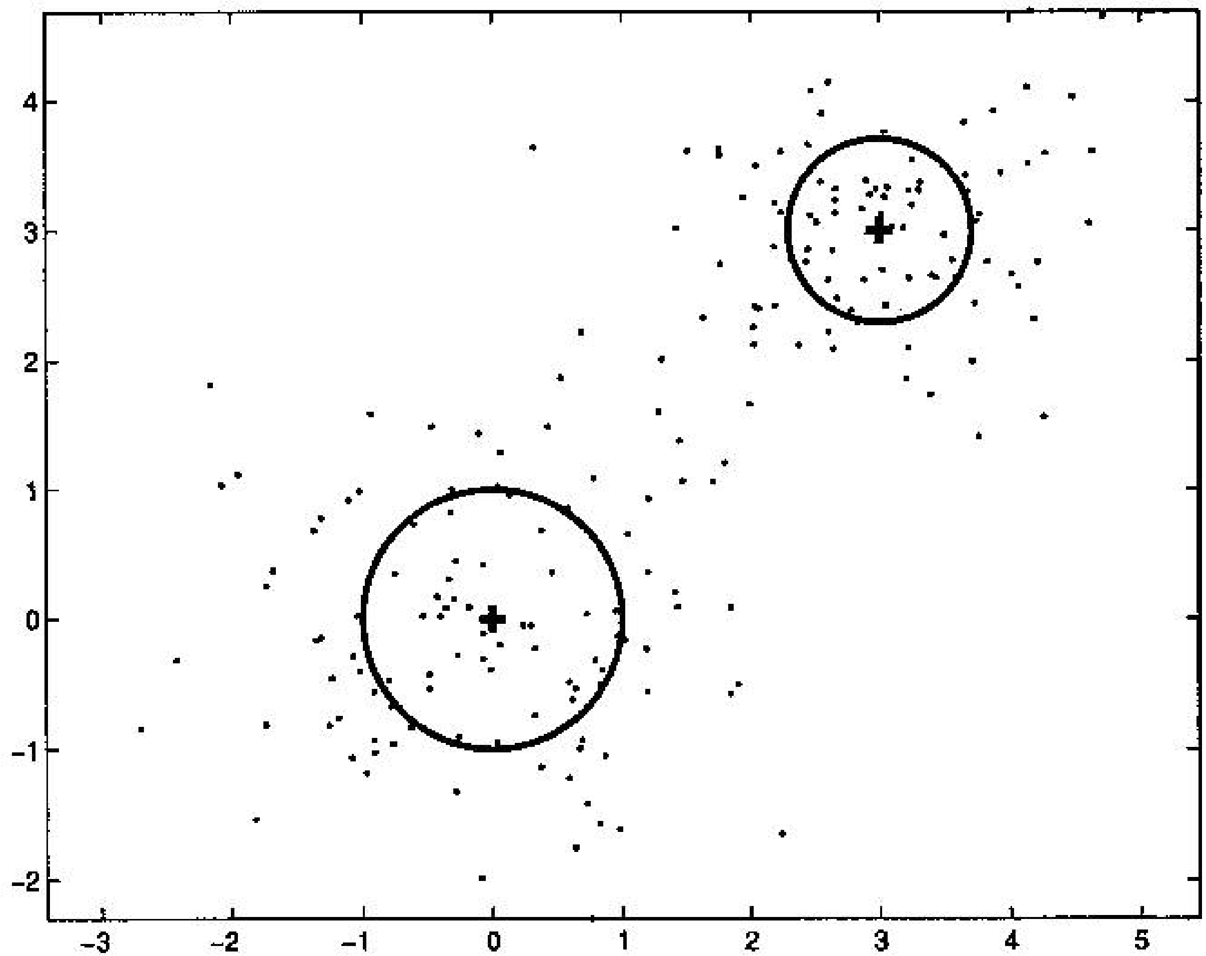
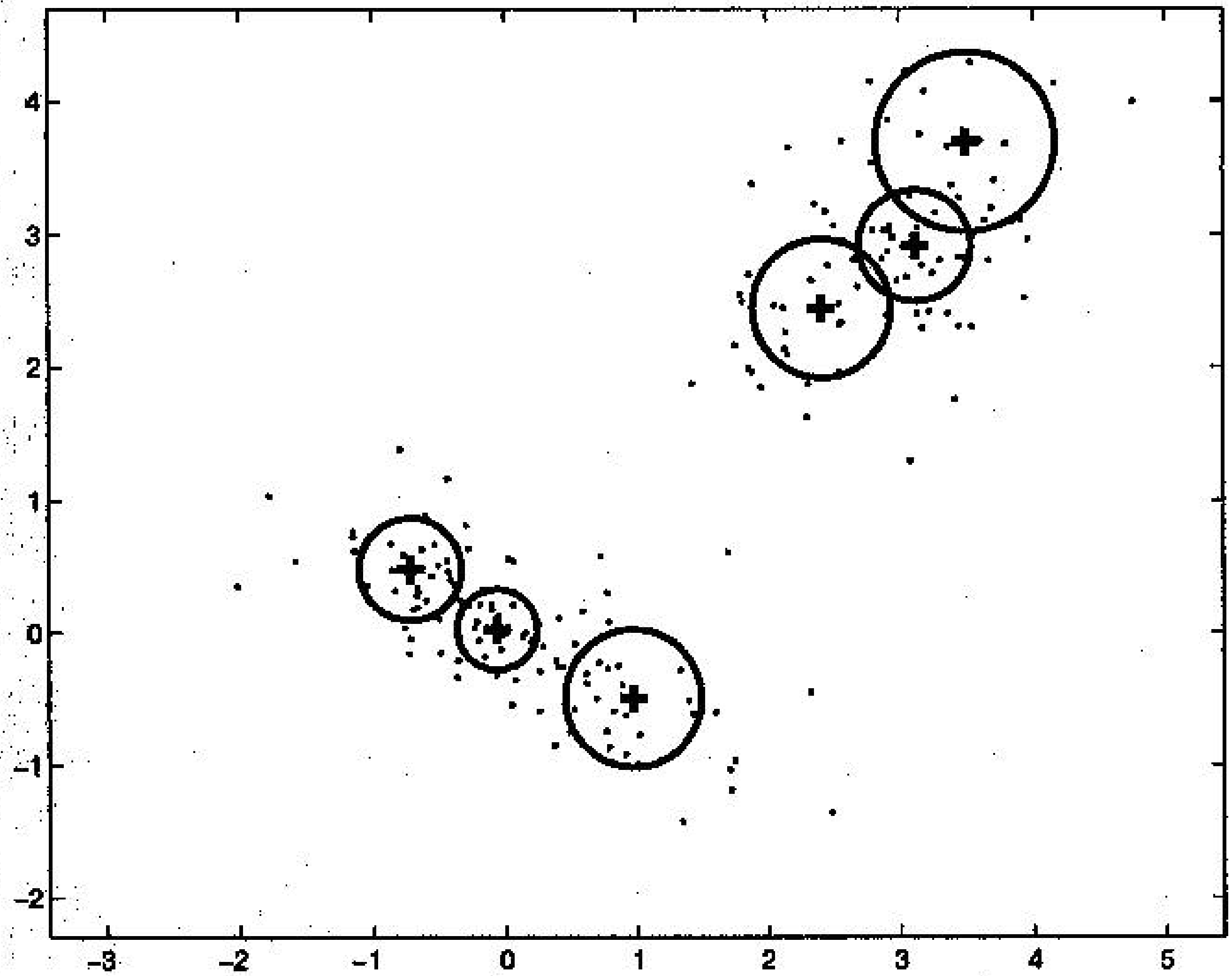


Fig. 3.1. Spherical covariance mixture model. Sampled data (*dots*), **centres** (*crosses*) and one standard deviation error bars (*lines*).



4. Spherical covariance mixture model with six components fitted to the mpled from the full covariance two-component model in Fig. 3.3. Sampled ots), centres (*crosses*) and one standard deviation error bars (*lines*).

- From Netlab (p82-83)

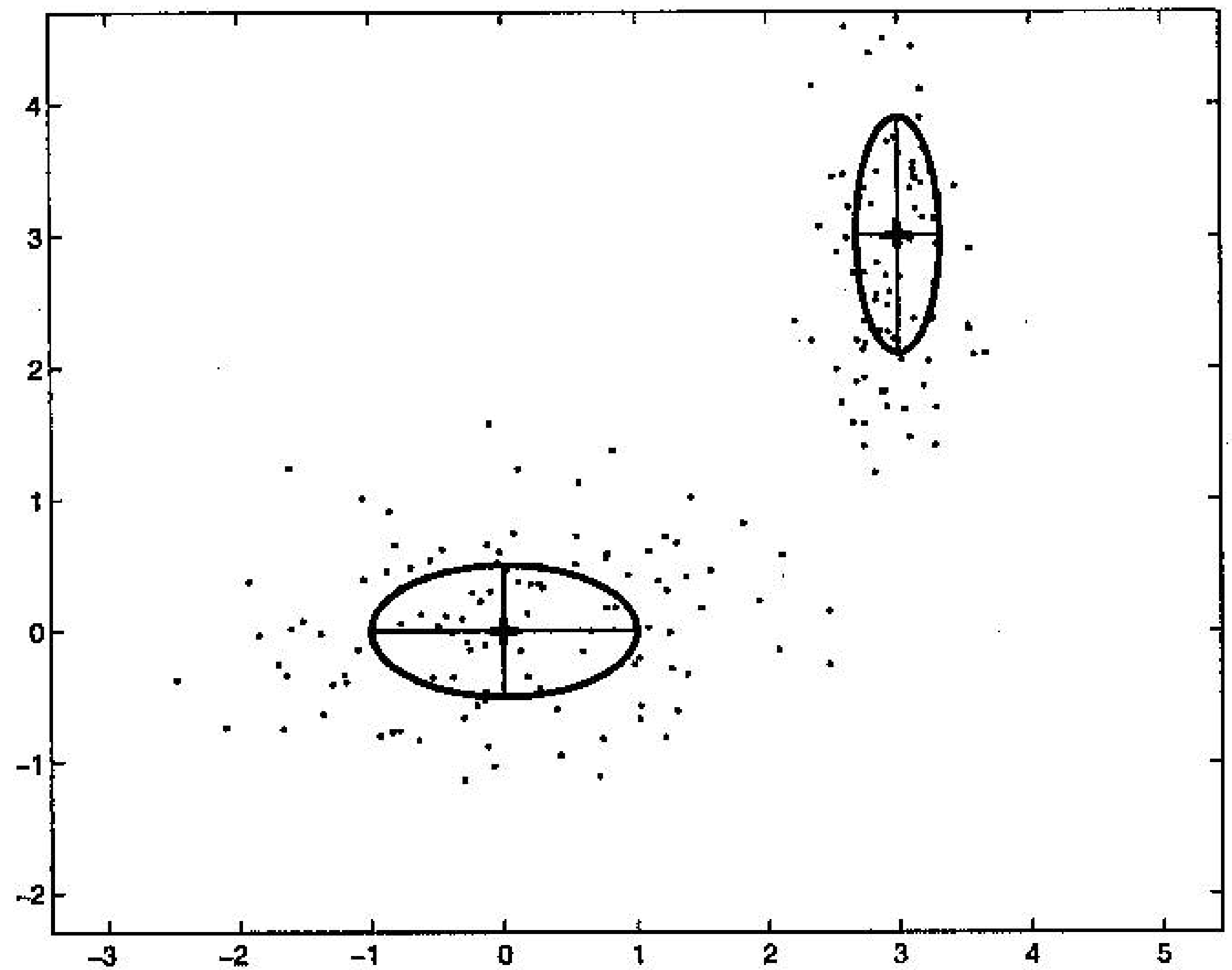
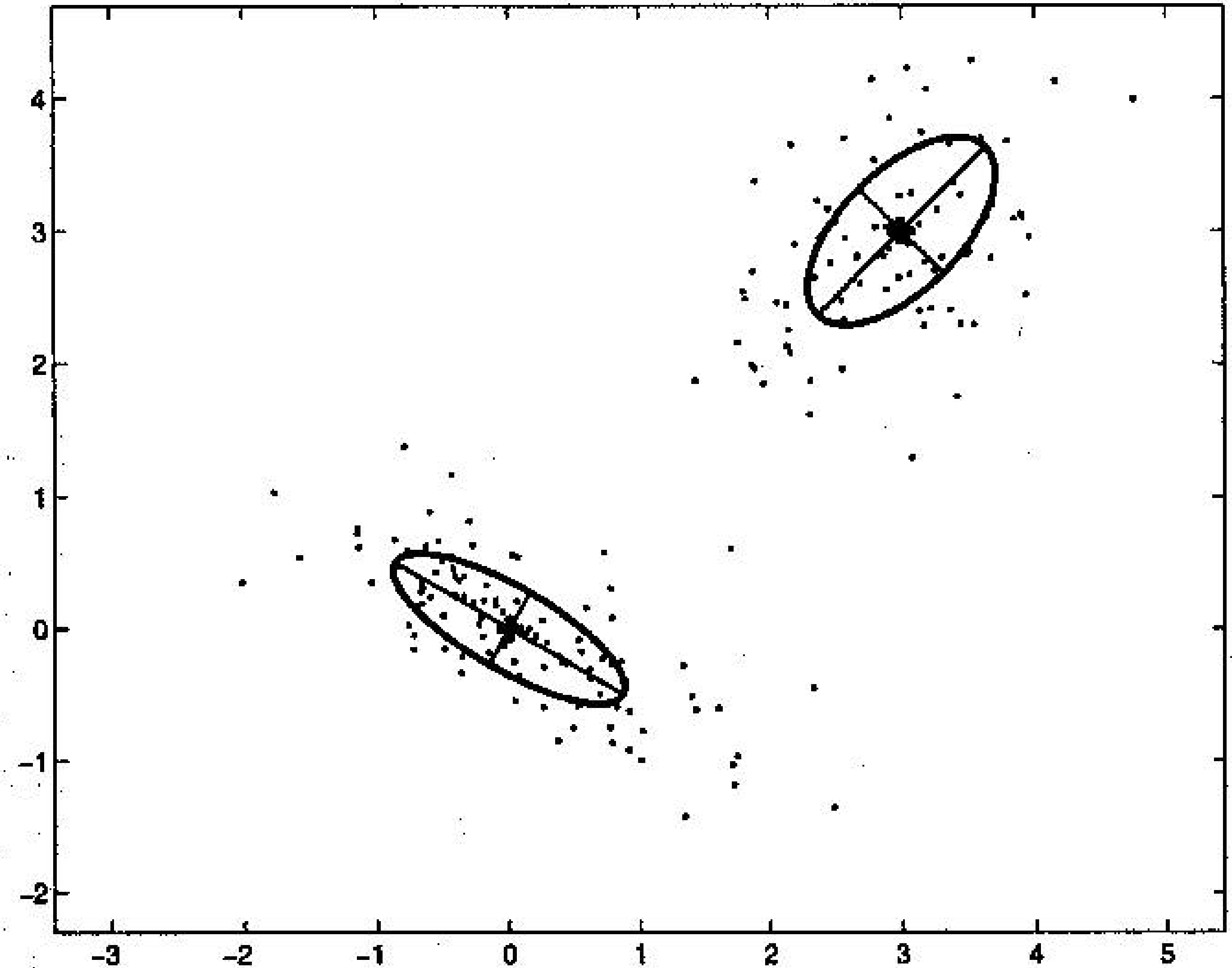


Fig. 3.2. Diagonal covariance mixture model. Sampled data (*dots*), centres (*crosses*), covariance axes (*thin lines*) and one standard deviation error bars (*thick lines*).



3. Full covariance mixture model. Sampled data (*dots*), centres (*crosses*), nce axes (*thin lines*) and one standard deviation error bars (*thick lines*).

GMM: Likelihood

1. Evaluate the probability of that mixture modeling your point.

```
likelihoodgm1 = gmmprob(gm1,testing_features)
likelihoodgm2 = gmmprob(gm2,testing_features);
loglikelihood  = log(likelihoodKick ./likelihoodSnare )
```

- Log-function is “order-preserving” – maximizing a function vs. maximizing its log gives same results

Minimization Problems

>Demgmm1

- EM is gradient-based – it does not find the global maximum in the general case, unless properly initialized in the general region of interest.

- Error wants to be –inf, which occurs when Gaussian is fit for each data point. (mean = data point and variance = 0)

- “There are often a large number of local minima which correspond to poor models. Solution is to build models from many different initialization points and take the best model.”

GMM

- “Pooled covariance" - using a single covariance to describe all clusters (saves on parameter computation)

EXAMPLE OF GMMS: GENRE CLASSIFICATION

Genre

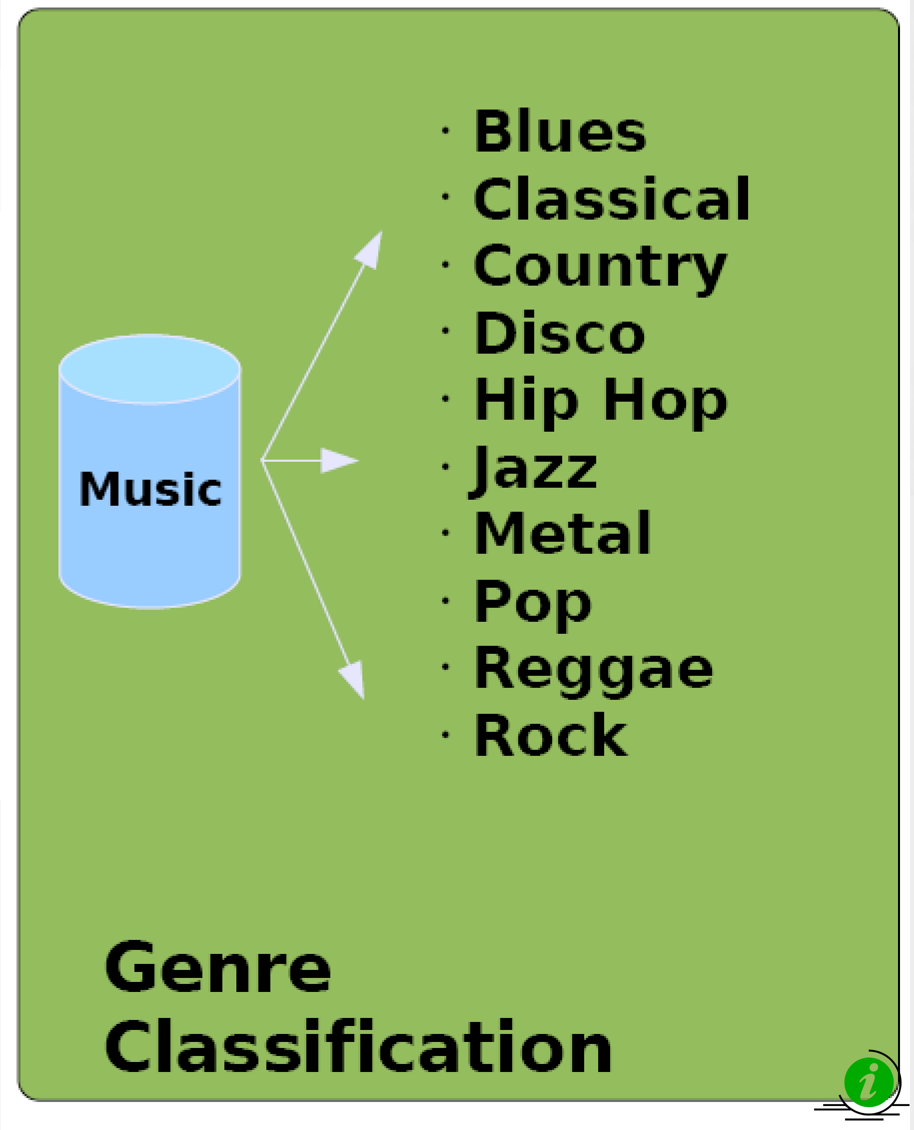
“Because feature vectors are computed from short segments of audio, an entire song induces a cloud of points in feature space.”

“The cloud can be thought of as samples from a distribution that characterizes the song, and we can model that distribution using statistical techniques. Extending this idea, we can conceive of a distribution in feature space that characterizes the entire repertoire of each artist.”

A. Berenzweig, B. Logan, D. Ellis, and B. Whitman. A large-scale evalutation of acoustic and subjective music similarity measures. In Proceedings of 4th International Symposium on Music Information Retrieval, Baltimore, Maryland, 2003.

- **Genre Classification**:
  - Manual : (Perrot/Gjerdigen) 72%
  - Automated (2002) (Tzanetakis) 60%
  - Automated (2005) (Bergstra/Casagrande/Eck) 82%
  - Automated (2007) 76%

From ISMIR 2007 Music Recommender Tutorial (Lamere & Celma)



How?

- Version 1 - One feature vector per song
  - High-level features extracted from data
    - Timbral (MFCCs, etc), Rhythmic content (beat histogram, autocor, tempos), Pitch info
    - Sampling of the frames in the song
  - Statistics of features extracted from a piece (includes means, weights, etc)
  - Representative of MFCC spectral shape
  - Could further use “Anchor space” where classifiers are training to represent musically meaningful classifiers. (Euclidean distance between anchor space)
- Version 2 - Cloud of points
  - Extract audio every *N* frames
  - K-Means or GMM representing a “cloud of points” for song
    - Clusters: mean, covariance and weight of each cluster = signature for song/artist/genre

Training and test data

- An overfit model matches every training example (Now it's "overtrained.")
- Training Error AKA "Class Loss"
- Generalization
  - The goal is to classify new, unseen data.
  - The goal is NOT to fit the training data perfectly.
- An overfit model will not be well-generalized, and ***will*** make errors.
- Rule of thumb: favor simple solutions and more "general" solutions.

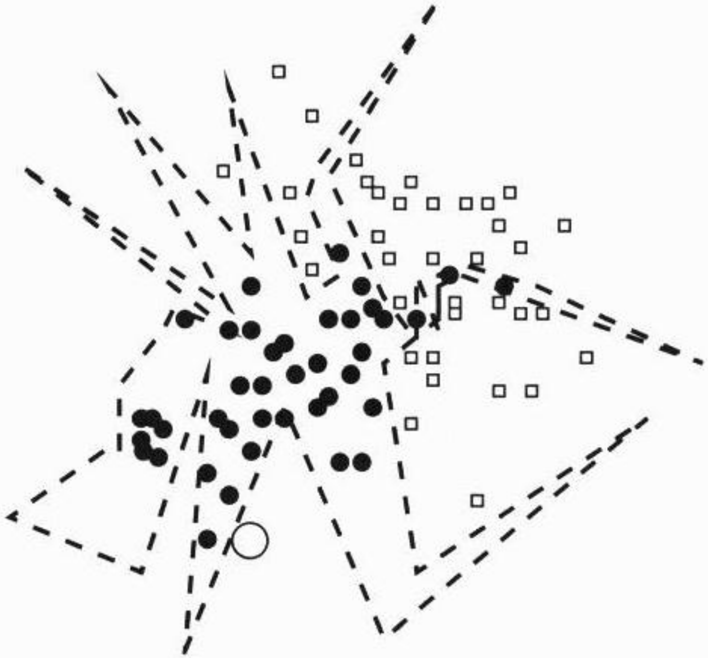


Fig. 2.13. Supervised classification into two classes with 2-dimensional data. In the training set (X, Y), data with label $\mathsf{y} = -1$ are represented with dots, whereas data with label $\mathsf{y} = 1$ are represented with squares. The dotted line is a classification function F such that $\mathsf{R}^{\mathrm{emp}}_{(\mathsf{X},\mathsf{Y})}[\mathsf{F}] = 0$. Though it achieves zero empirical risk, F is not a good classification function, as it makes an error for a new datum which is not in the training set (circle at the bottom, with the true label $\mathsf{y} = -1$).

Evaluation Measures

True+	correct	Classifier correctly predicted something in it's list of known positives
False-	absent	Classifier did not hit, for a known positive result.
False+	incorrect	Classifier said that something was positive when it's actually negative

Evaluation Measures

“Accuracy”

↑ is good

Precision - “Positive Predictive Value”

↓ = high F+ rate, the classifier is hitting all the time

↑ = low F+ rate, no extraneous hits

Recall – “Missed Hits”

↓ = high F- rate, the classifier is missing good hits

↑ = low F- rate, great at negative discrimination – always returns a negative when it should

F-Measure – a blend of precision and recall (harmonic-weighted mean)

↑

Evaluate Measures

P = T+ / (T+ + F+)	[0…1]
R = T+ / (T+ + F -)	[0…1]
F = 2*P*R/(P+R)	[0…1]

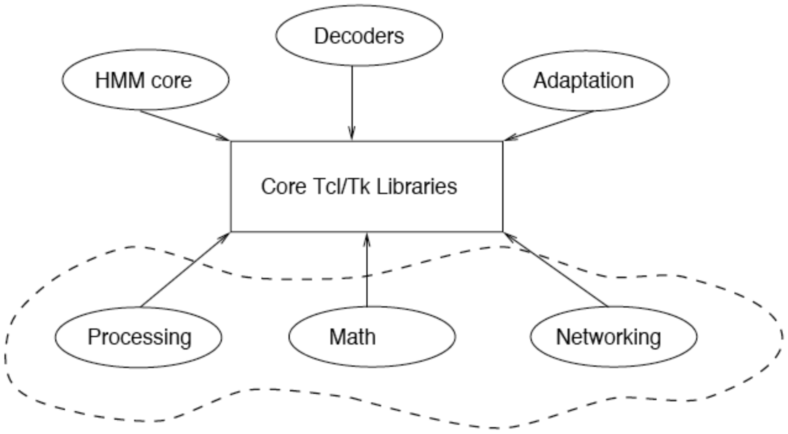
Training and test data

- Cross-validation
- Training, Validation, and Test set
  - Partition randomly to ensure that relative proportion of files in each category was preserved for each set
    - Weka or Netlab has sampling code
- Warnings:
  - Don't test (or optimize, at least) with training data
  - Don't train on test data (no!)

Data preparation

- Examine your data at every chance. (means, max, min, std, “NaN”, “Infs”)
- Try to visualize data when possible to see patterns and see if it makes. Incredible sanity check.
- Eliminate noisy data
- Data preparation
  - Cleaning
    - Open up and examine
    - Handle missing values
  - Relevance / Feature analysis
    - Remove irrelevant or redundant attributes
  - Data Transformation
    - Generalize or normalize data

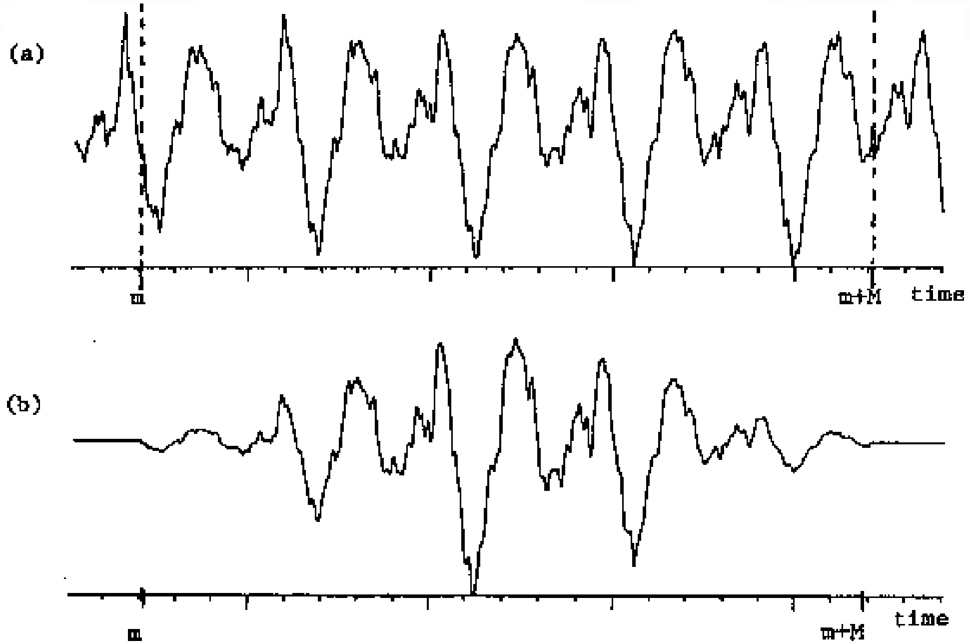
APIs for MIR Tools



- Marsyas: G. Tzanetakis (11), flexible tool set, scripting language, segmentation and classification
- LibOFA: Holm/Pope (00), simple FV for unique ID comparing to a large pre-analyzed database
- D2K/M2K: West/MIREX (06), Java-based GUI related to D2K, many apps.
- LibTSP: P. Kabal (00), C routines for DASP & IO
- CSL: STP/MAT (05), C++ class library for DASP, synthesis, control, spatialization and MIR

APIs - 2

- Libxtract
- Aubio
- SonicVisualizer plug-ins
- Loris
- SPEAR
- CSL
- LibTSP
- Mirtoolbox
- Echonest



Spectral Tools

- SPEAR
- Loris
- Marsyas
- Sonic visualizer

Code Examples

- Buffer, Window classes (see CSL)
- Analyzer class (Marsyas)
- aubio, libxtract
- IO libraries (libSndFile, PortAudio)
- DASP libraries (libTSP, etc.)

Using FFT APIs

- Simple FFT
  - See MAT240B (http://HeavenEverywhere.com/TheBigMATBook)
  - See F. R. Moore’s Elements of Computer Music
- FFTW
  - FFTW data types
  - FFTW plans
  - See CSL Spectral class