Classification of Music Signals:
Feature extraction and selection

Statistical classification

- Is a statistical procedure in which an individual item is characterized into one of a number of categories or classes.
- In our case, the items are audio signals (songs, sound files), their characteristics are features extracted from the signals, and the classes depend on the problem definition.
- The issue of classification becomes finding (or learning) an appropriate mapping between the features and the classes.
- This can be thought of as dividing the feature space into regions, each corresponding to a given class.

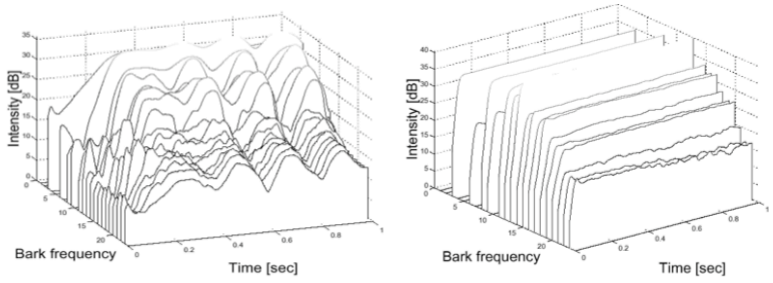
- We can distinguish two cases:
  - Supervised: when the mapping between features and a pre-defined taxonomy is learned by example.
  - Unsupervised: when the algorithm is left to find its own organization of the data (clustering) and the resulting classes are labeled a posteriori.

Classification of music signals

- Statistical classification is relevant to a number of MIR-related tasks:
  - Musical instrument identification
  - Artist identification
  - Genre classification
  - Music/Speech discrimination
  - Video segmentation based on audio
  - Transcription of percussive instruments
  - And more…
- Research on computer music classification largely re-implements knowledge from other closely-related fields, such as speech analysis.

An example: Instrument ID

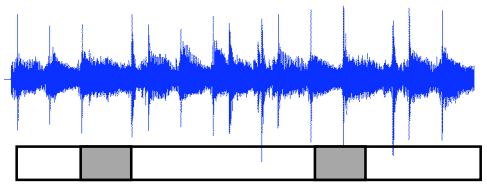
- Different properties of musical instruments make them recognizable.
- Humans are very good at focusing on them and on identifying families in complex situations (superimpositions). However, trained listeners are usually better at recognizing them
- Instrumental sounds change considerably within families (see flute vs. clarinet below) and even within the same instrument
- What are the *invariants* that define an instrument's sound?,



A music classifier

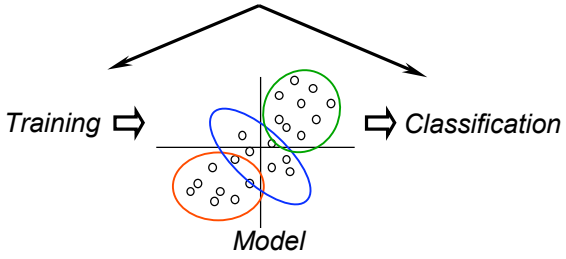
- Feature extraction:

1. Segmentation (Optional)
2. Calculation of feature vectors



- Classification:

1. Learn models of feature distribution from examples
2. Classification of new events based on these models



Feature extraction revisited (1)

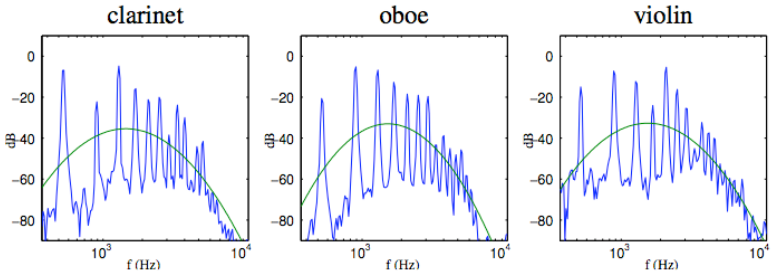
- Feature extraction is necessary as audio signals carry too much information, and that information is mostly irrelevant to our analysis
- Good features are a must for classification
- They should emphasize the differences between classes and the similarities within
- They can be estimated on a frame by frame basis or within segments, sounds or songs.
- As we have seen they are many possible features: spectral, temporal, pitch, chroma, etc.

Feature extraction revisited (2)

- Spectral features, include the low order spectral moments: centroid and spread:

$$SC = \frac{\sum_{k=0}^{N/2} f_k |X(k)|^2}{\sum_{k=0}^{N/2} |X(k)|^2} \qquad SS = \sqrt{\frac{\sum_{k=0}^{N/2} (f_k - SC)^2 |X(k)|^2}{\sum_{k=0}^{N/2} |X(k)|^2}}$$

- Their mean (μ) and variance (σ^2) as ways of characterizing a segment or sound



Feature extraction revisited (3)

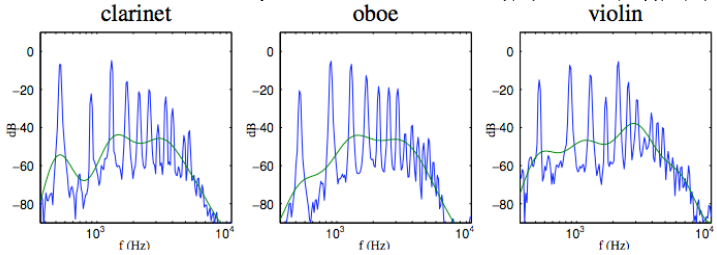
- The mean and variance of the spectral flatness:

$$SF_b = \frac{\sqrt[k]{\prod_k |X(k)|^2}}{\frac{1}{N_b} \sum_k |X(k)|^2}$$

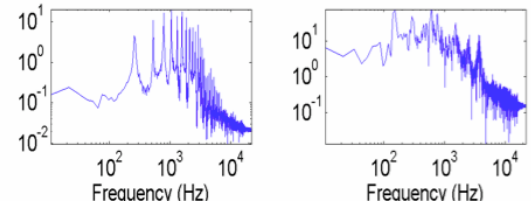
- (Log) Power of first few partials
- Harmonicity, Harmonic spectral deviation
- Relative energy in odd and even partials
- Spectral Smoothness
- …

Feature extraction revisited (4)

- μ and σ^2 of the first cepstral coefficients: $c_r(n)=IFFT(\log|X(n,k)|)$

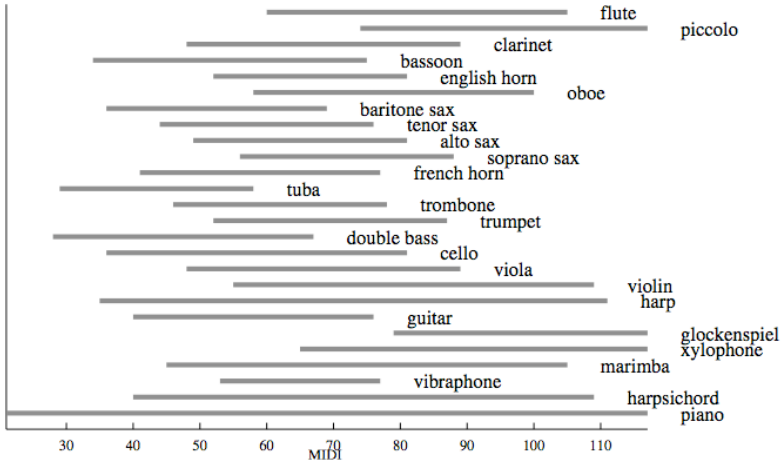


- And of their log-amplitude, log-frequency and uncorrelated version, the MFCCs (+ other spectral envelope features):



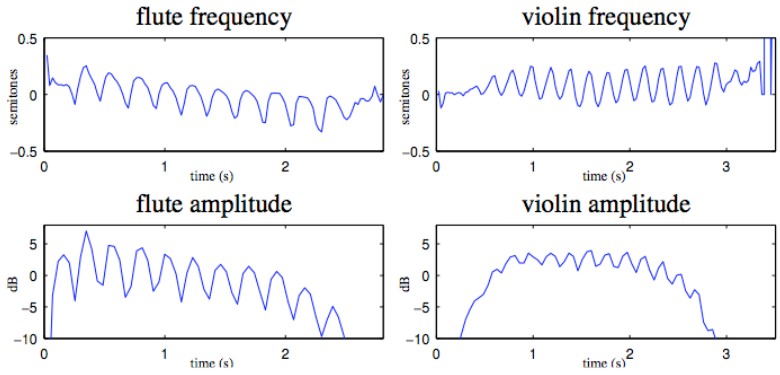
Feature extraction revisited (5)

- Pitch can also be used as a feature for classification:



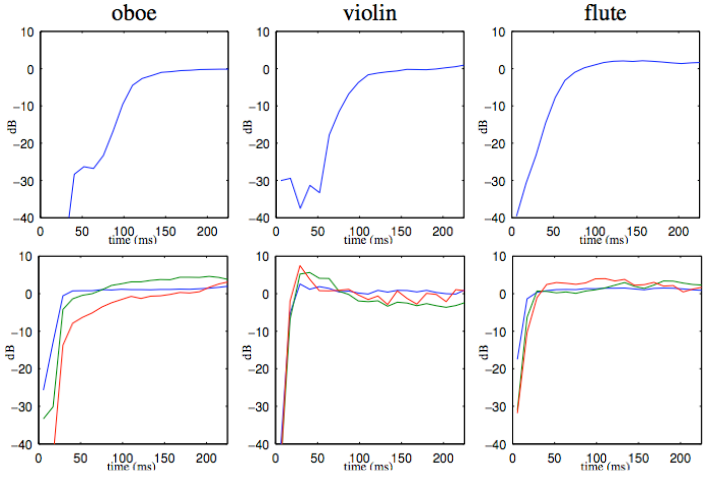
Feature extraction revisited (6)

- Rate and depth of amplitude and frequency modulations.
- Also applies to modulations of other features (e.g. centroid)



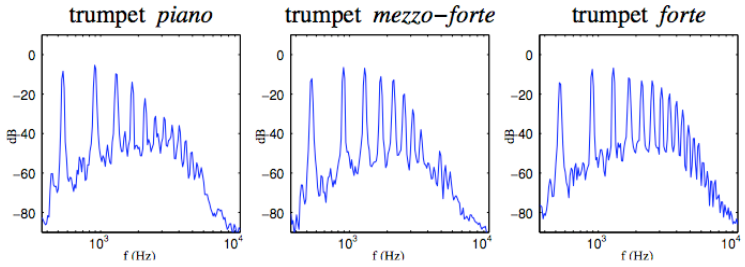
Feature extraction revisited (7)

- Temporal features include the shape of the onset detection function, rise time (time between onset and attack maximum), onset asynchrony (between subbands, partials), feature flux (Δfeatures), ZCR, etc



Feature variations

- Feature changes within classes for a number of reasons: pitch, loudness, playing style, articulation, recording conditions, the instrument itself, ambient noise, etc

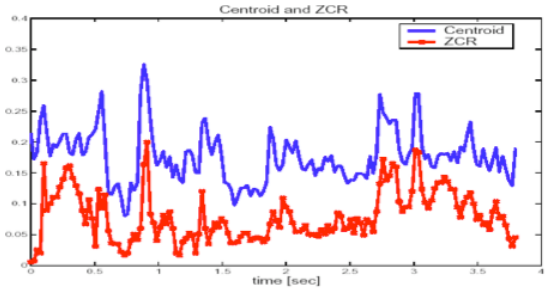


- For example, spectral centroid changes on the violin sample according to loudness and pitch.
- The classifier must be trained on many sounds/songs to account for these variations (selection of an appropriate training database)
- The class is not defined by a point in the feature space but by a distribution with a certain mean and a certain variance

Feature independence

- The features are not necessarily independent from each other (strong correlation), e.g. centroid and zero-crossing rate (ZCR)

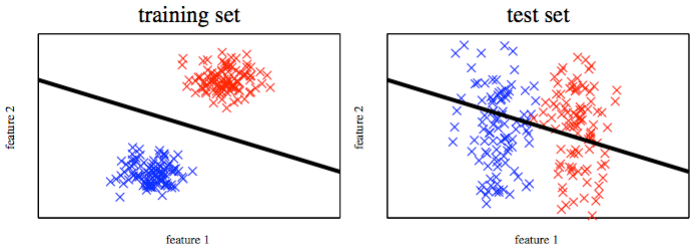
$$ZCR = \frac{1}{N}\sum_{n=2}^{N}\left|sign(x(n)) - sign(x(n-1))\right| \qquad sign(x) = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases}$$



- This matters because the quantity of training data increases exponentially with the number of parameters (+ complex model)

Feature reliability

- Good features result in large distances between classes and small distances within classes
- Features might be unreliable because they are irrelevant (e.g. chroma for instrument recognition), because they are difficult to calculate (e.g. onset rise times in polyphonies) or because they are not well represented on the feature set.
- All this leads to overfitting: the feature only covers a sub-region of its natural range on the training set, thus underperforming on the test set.

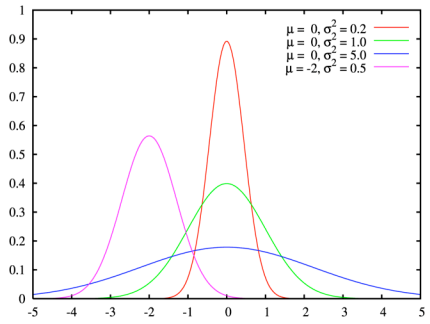


- We want to select the smallest reliable feature set, but how?

Feature distribution

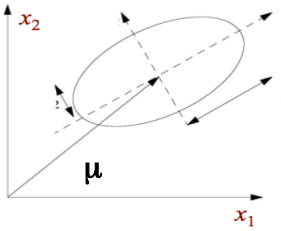
- A Normal or Gaussian distribution is a bell-shaped probability density function defined by two parameters, its mean (μ) and variance (σ^2):

$$N(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)} \qquad \mu = \frac{1}{N}\sum_{n=1}^{N} x(n) \qquad \sigma^2 = \frac{1}{N}\sum_{n=1}^{N} (x(n)-\mu)^2$$



Feature distribution

- In D-dimensions, every pair of features generates a 2-d space where the distribution becomes an ellipse:



- The mean becomes a D-dimensional vector μ, and the variance becomes a DxD co-variance matrix defined as:

$$C_x = \frac{1}{N}\sum_{n=1}^{N}(x-\mu)(x-\mu)^T$$

- Strong correlation between features results on a narrow ellipse

Feature selection (1)

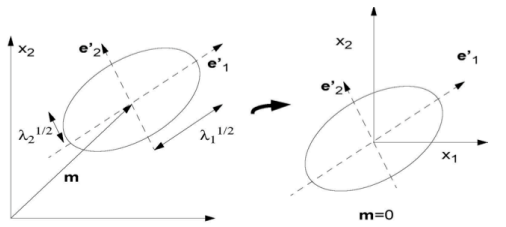
- To avoid bias towards larger features, we can normalize them to have zero mean and unit variance:

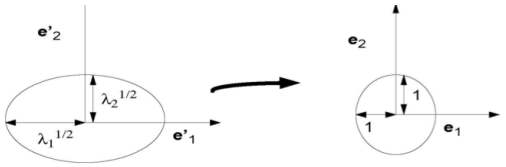
$$\hat{x} = (x-\mu)/\sigma$$

- Alternatively we can apply Principal Component Analysis (PCA):

$$\hat{x} = C_x^{-1/2}(x-\mu)$$

- The resulting features are normalized and uncorrelated





Feature selection (2)

- Furthermore, PCA can be used to reduce the number of features:
- The covariance matrix C_x can be re-ordered such that the first row corresponds to the feature with the highest variance (largest eigenvalue λ_i) and the last row to the feature showing the least variance (smallest eigenvalue).
- 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 a few directions that maximize the variance.
- We do not need to choose between correlating (redundant) features, PCA chooses for us.
- It can also be used for projecting onto low-dimensional spaces (e.g. for visualization).

Classification

- What we (should) have:
  - a taxonomy;
  - a training dataset consisting of labeled, representative samples of the sounds to be classified and the classes themselves;
  - and the optimal set of parameters (that guarantees the best discrimination).
- What now? The goal is to classify new instances from the knowledge gathered during training
- We can use algorithms where data is clustered into a group of unknown classes which are latter tagged (unsupervised).
- We will concentrate instead on the case when the classes of training instances are given (supervised training)

Distance-based classification (1)

- As with segmentation, we can calculate the distance between samples in the feature space
- This is at the core of most simple classification schemes

- Minimum distance classification:
  - Calculates the distance (e.g. Euclidean, cosine) between the new sample and all samples in the training set
  - Selects the class of the closest training sample

- k-nearest neighbor (k-NN) classifier:
  - Calculates the distance between the new sample and all samples in the training set
  - Picks the k nearest neighbors
  - Selects the class that was more often picked.

Distance-based classification (2)

- In both these cases, training is trivial (just store the labeled training instances for comparison)

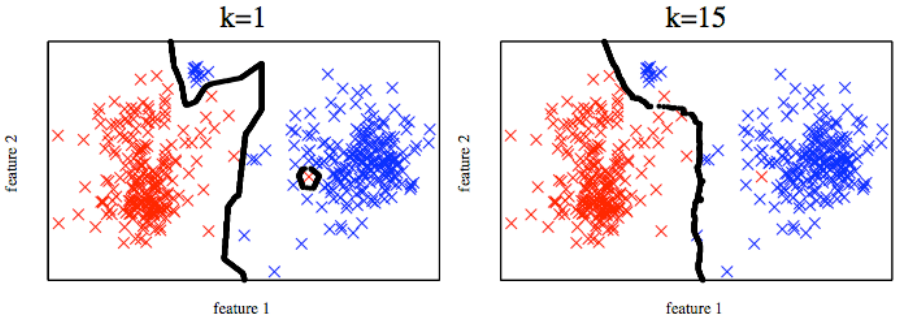
- These classifiers are known as "lazy" classifiers as they do not entice any parameterization of the problem

- The complexity of the classification and the computational cost increases as the number of training instances increase (because we need to measure the distance from the new instance to all existing ones).

- We can improve the computational complexity by storing only a few class prototypes/models (e.g. class centroids)

Distance-based classification (3)

- k-nearest neighbor results on good discrimination between classes with a very simple implementation (at the expense of computational cost)
- It depends on the choice of distance metric (Manhattan, Euclidean, etc).
- A good metric is the Mahalanobis distance (normalizes and decorrelates features): $d_M = (x-y)^T C_x^{-1} (x-y)$
- Also, we need to choose k to avoid overfitting. A standard approximation is k = √N where N is the number of training samples



Probabilistic Classifiers (1)

- The idea is to interpret the feature vector x as a random variable whose probability distribution depends on the current class, thus the class can be inferred from the observation of the feature vector.
- Probabilistic classifiers are defined in terms of Bayes' rule:

Posterior: Probability of class i given a feature vector x

Likelihood: Probability of a feature vector x, given class i (can be estimated from training)

Prior Probability of class i

$$P(class_i \mid x) = \frac{P(x \mid class_i) P(class_i)}{P(x)}$$

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

$$P(x) = \sum_i P(x \mid class_i) P(class_i)$$

Probabilistic Classifiers (2)

- Note that classification becomes the task of finding the class with the highest probability of occurring given the observation x
- Thus, what we want to know (but don't know) is the posterior probability $P(class_i|x)$ for every class i, so that we can pick the maximum.
- Furthermore, since P(x) is the same for all classes, maximizing $P(class_i|x)$ is equivalent to maximizing the numerator of the rule: $P(x|class_i)P(class_i)$
- This is known as Maximum A Posteriori (MAP) classification:

$$class_i = \arg\max_i \{P(x \mid class_i)P(class_i)\}$$

- Happily, from the training data we can learn the likelihood $P(x|class_i)$ and define (or learn) the prior $P(class_i)$
- But how?

Gaussian Mixture Model (1)

- We can model (parameterize) the likelihood using a Gaussian Mixture Model (GMM)
- The model approximates $P(x|class_i)$ as the weighted sum of K multidimensional Gaussian distributions:

$$P(x \mid class_i) = \sum_{k=1}^{K} w_{ik} N(x; \mu_{ik}, C_{ik})$$

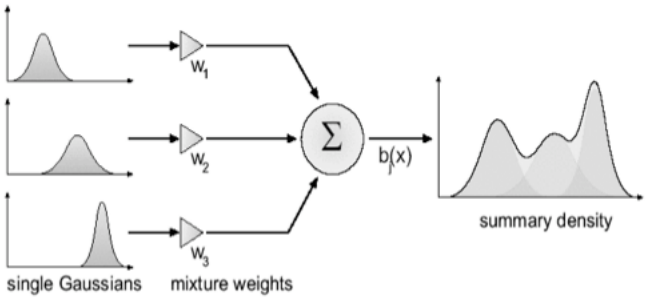
- Where w_{ik} are the weights and $N(x;\mu,C_x)$ is a Gaussian distribution

$$\mathrm{N}(x; \mu, C_x) = \frac{1}{\sqrt{(2\pi)^D \mid C_x \mid}} e^{\left(-\frac{1}{2}(x-\mu)C_x^{-1}(x-\mu)^T\right)}$$

$$C_x = \frac{1}{N}\sum_{n=1}^{N}(x-\mu)(x-\mu)^T$$

Gaussian Mixture Model (2)

- 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 (3)

- The model is parametric, consisting of K weights, mean vectors and covariance matrices for every class.
- If features are uncorrelated, 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 algorithm (Dempster et al, 1977).
- EM is an iterative algorithm whose objective is to find the set of parameters that maximizes the likelihood $P(x|class_i)$ for a class. It is also able to learn the prior.
- Alternatively, parameters for all classes can be estimated simultaneously using a Maximum discrimination objective using a standard non-linear optimization algorithm

MAP classification

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

$$class_i = \arg\max_i \{P(x \mid class_i)P(class_i)\}$$

- Commonly we won't have a single feature vector x describing a new instance, but a sequence of vectors $x_{1..T}$ such that the probabilities get multiplied:

$$class_i = \arg\max_i \left\{ \prod_{t=1}^{T} P(x_t \mid class_i)P(class_i) \right\}$$

- Because the log is order preserving, this maximization is equivalent to:

$$class_i = \arg\max_i \left\{ \log\left(\prod_{t=1}^{T} P(x_t \mid class_i)P(class_i) \right) \right\}$$

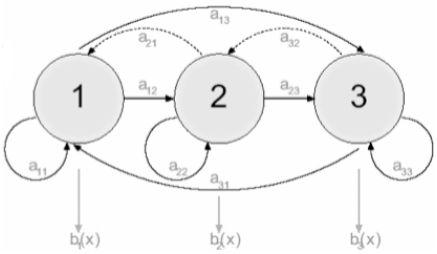
$$class_i = \arg\max_i \left\{ \sum_{t=1}^{T} \log\left(P(x_t \mid class_i)P(class_i) \right) \right\}$$

HMM (1)

- When the temporal evolution of features is relevant for the classification it can be convenient to use a Hidden Markov Model
- For a given state, the HMM models the probability of observing a feature vector x as a Gaussian Mixture Model.
- However, the HMM also takes into consideration the transition probabilities between states $a_{ij} = P(S_t = i | S_{t-1} = j)$
- Thus, we can train a HMM for each class, such that for a feature vector sequence $x_{1..T}$ we can obtain:

$$P(class_i \mid x_{1..T}) = \frac{P(x_{1..T} \mid class_i)P(class_i)}{P(x_{1..T})}$$

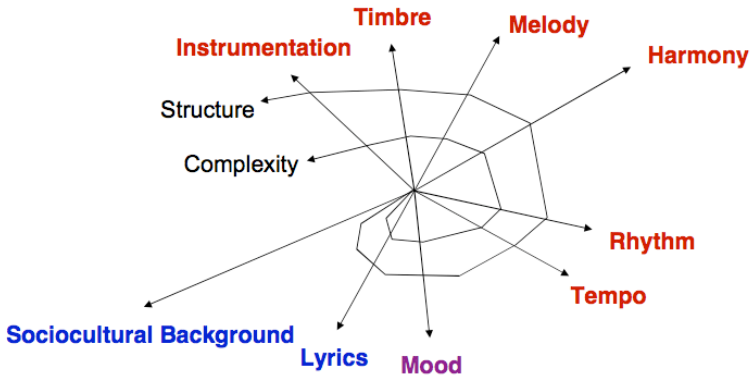
HMM (2)



- HMMs can also be trained using the Expectation-Maximization algorithm (in its Baum-Welch implementation)
- The hidden states are usually not important for classification
- HMMs are a generalization of GMMs for more than one state
- It is only useful to use HMMs if the temporal structure can be learned by the state-transition probability matrix
- If not, HMMs do not perform better classification than GMMs.

An end note about similarity (1)

- Perception of similarity is subjective and context-dependent and can happen on a number of different dimensions:



An end note about similarity (2)

- The distance-based organization of data models on a feature space is at the core of music similarity (and visualization) tools
- Similarity is not necessarily attached to labels, thus supervised training is not a requirement. Instead what we want is to discover the hidden structure of sound organization
- See Elias Pampalk's work (http://www.ofai.at/~elias.pampalk/)