Université de Montréal

Department of Computer Science

# Part 1: A Frame-Level Speech/Music Discrimination using AdaBoost

Norman Casagrande

Douglas Eck

Balázs Kégl

MITACS 2005

Université de Montréal

Department of Computer Science

# Part 2: Audio Genre Recognition – the winner of the MIREX 2005 competition

James Bergstra

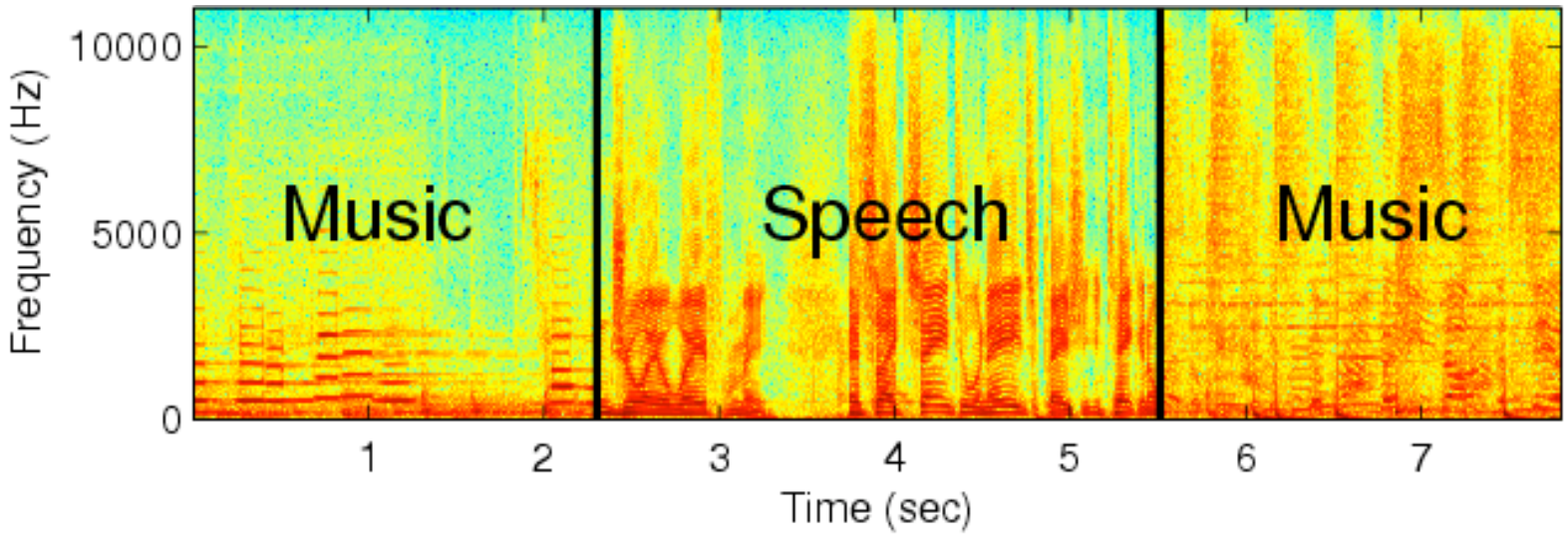
Norman Casagrande

Douglas Eck

Balázs Kégl

# Introduction

- Does voice have a visible distinguishable pattern?

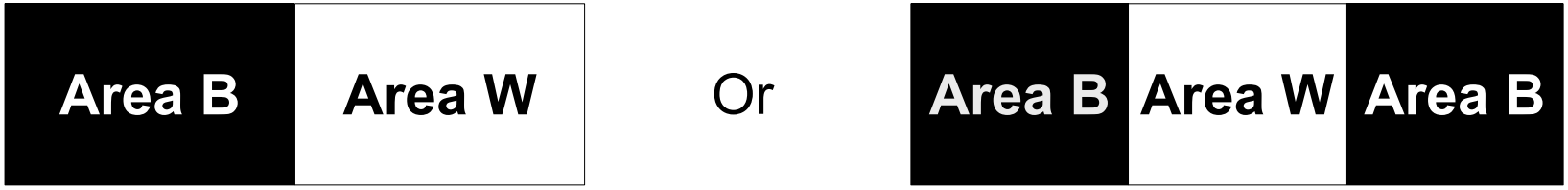


# The Features

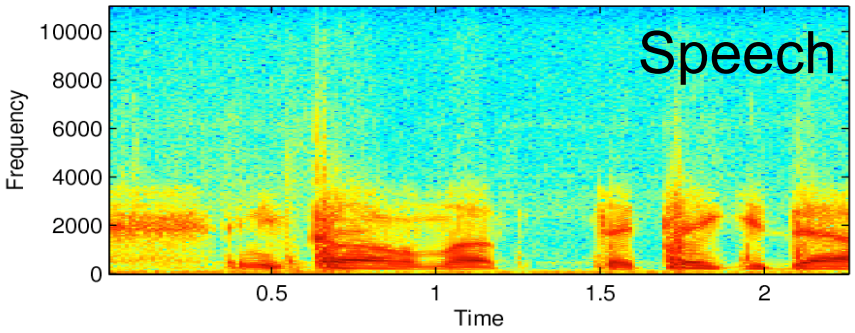
- Using a robust image classifier by Viola & Jones (2001) we can quickly classify music against speech on real world data

# The Features

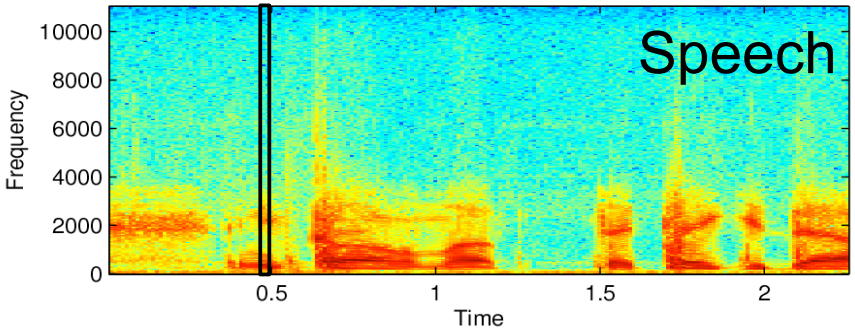
- Using a robust image classifier by Viola & Jones (2001) we can quickly classify music against speech on real world data
  - Find simple basic features that computes the difference of amount of energy



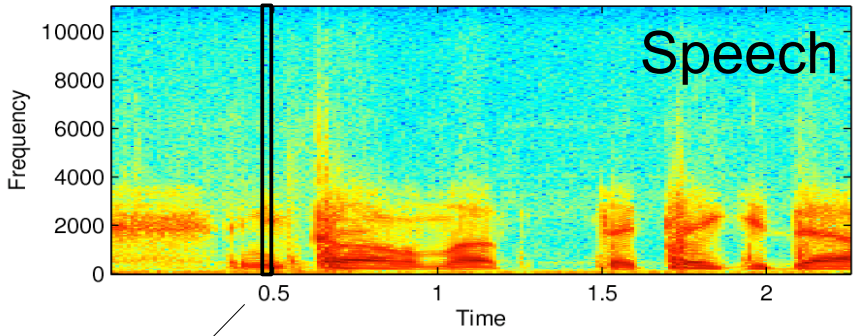
# The Features



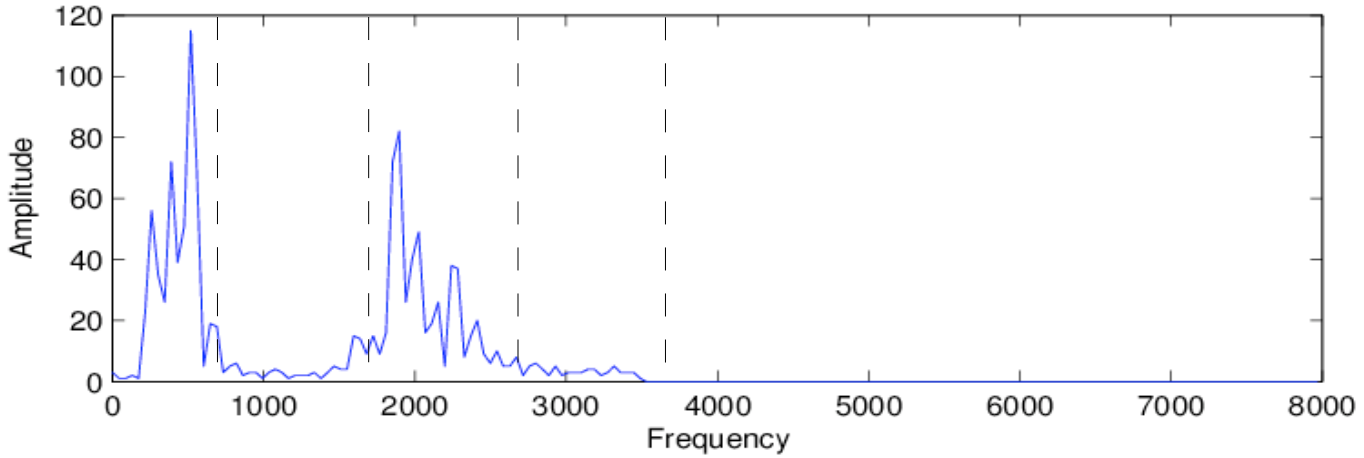
# The Features



# The Features



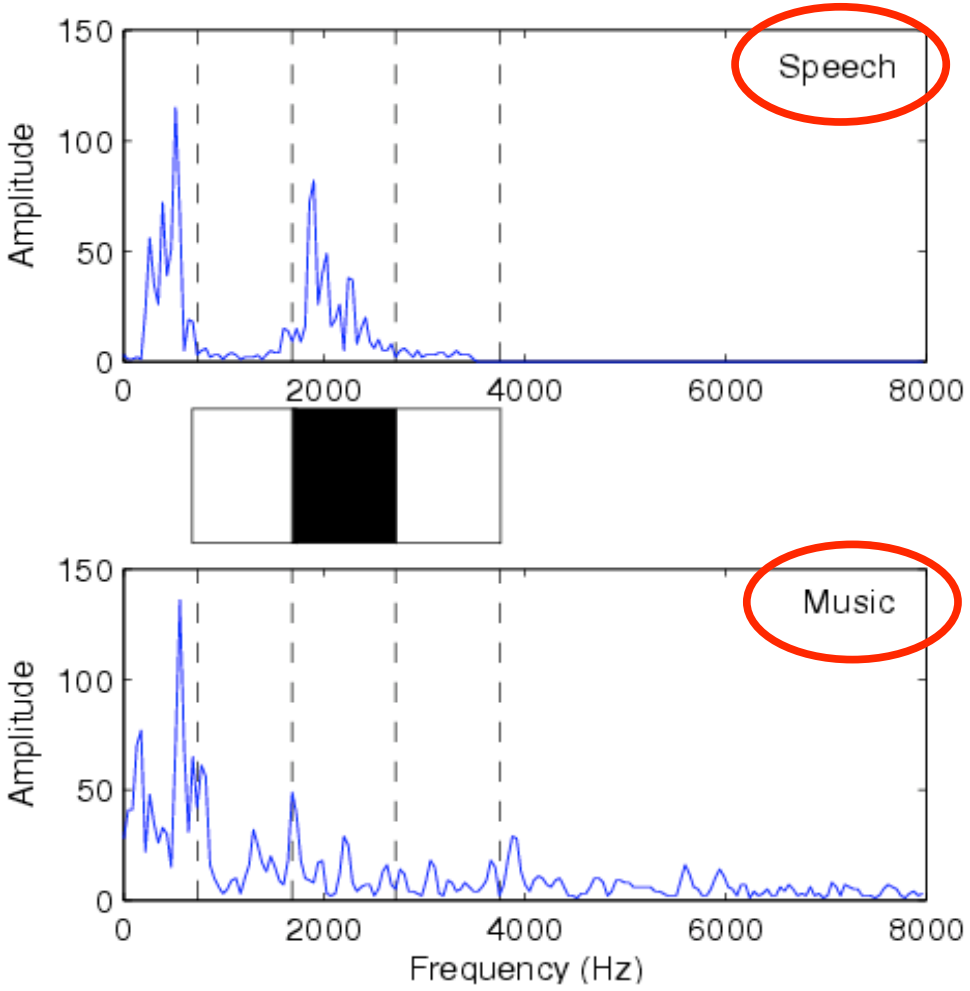

Single frame




$$B_1 - W + B_2 = x_i$$

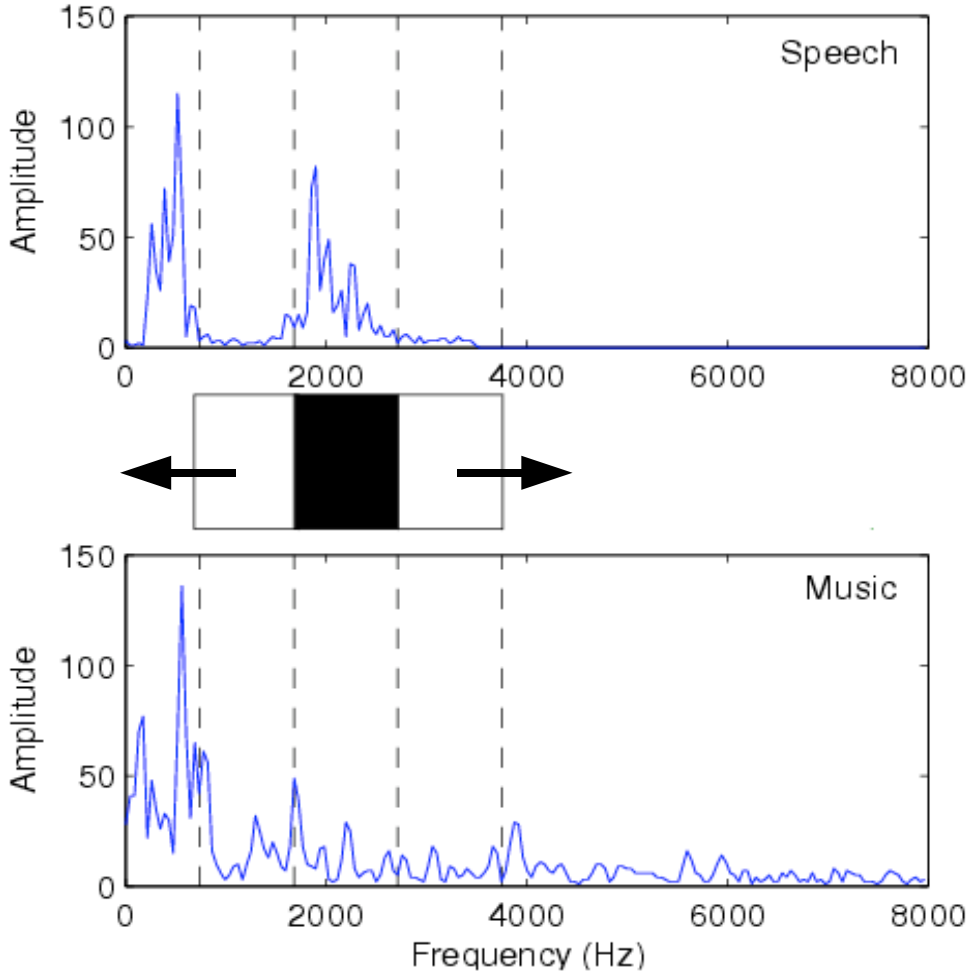
# The Features

- How do we find the position of the filter?



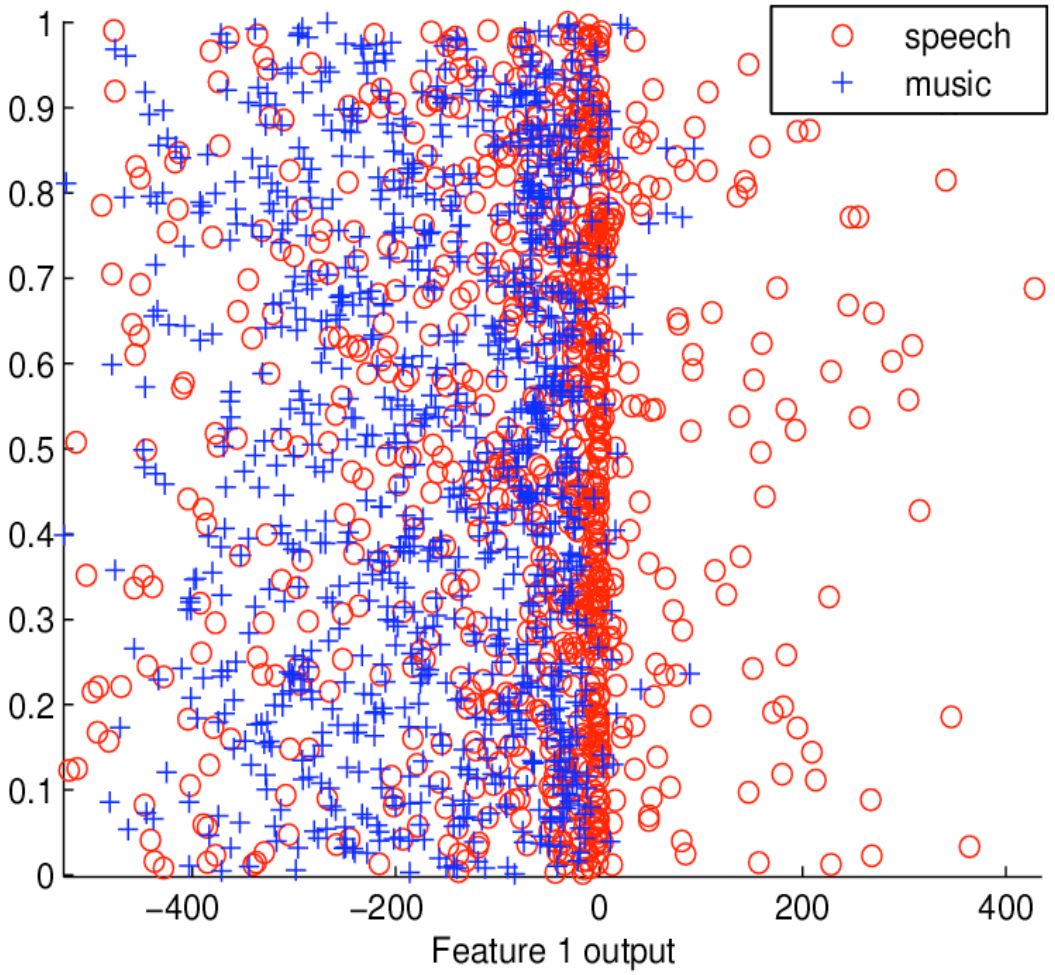
# The Features

- How do we find the position of the filter?
- The filters have also different sizes!



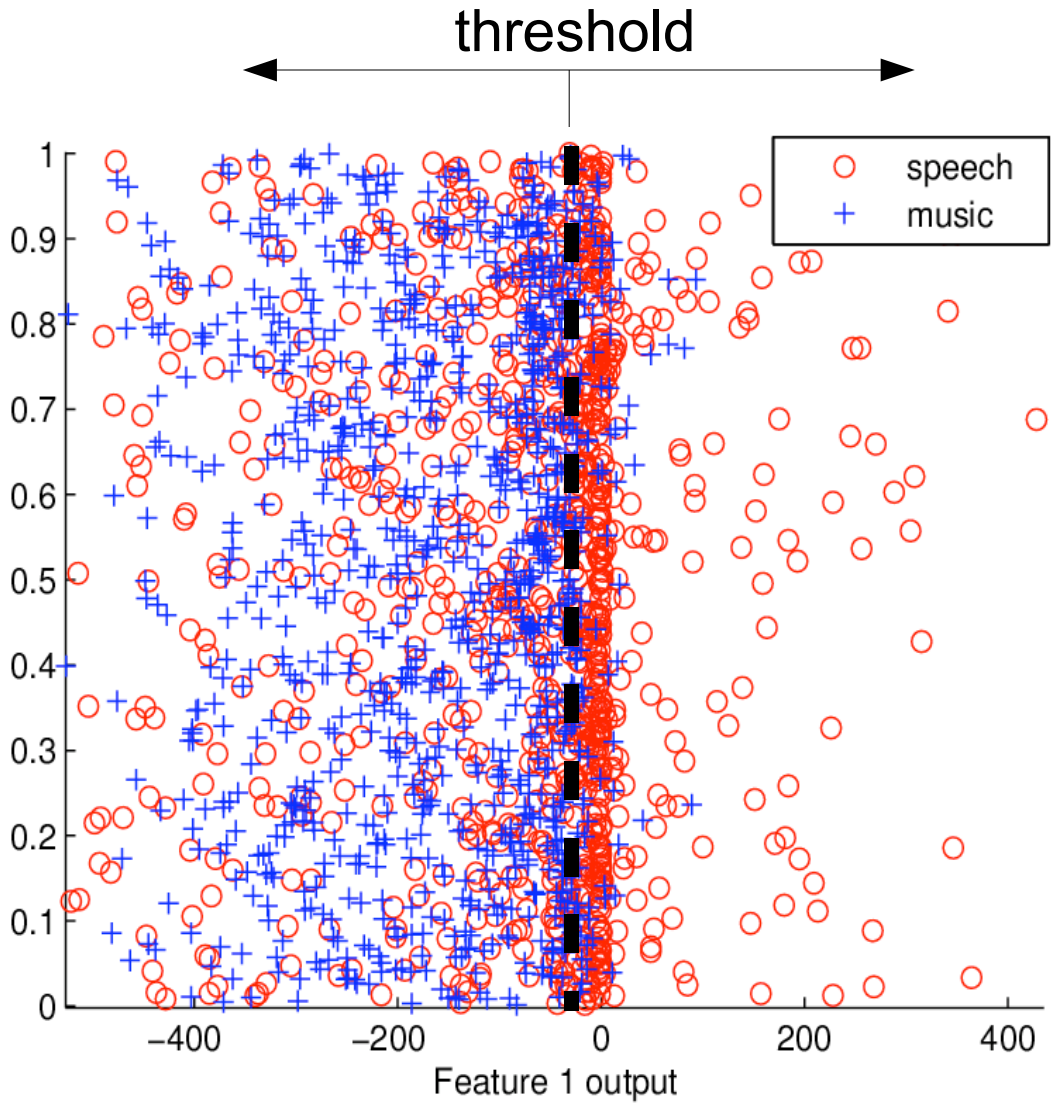
# The Features

- The output of this simple filter is already revealing a separation between speech and music



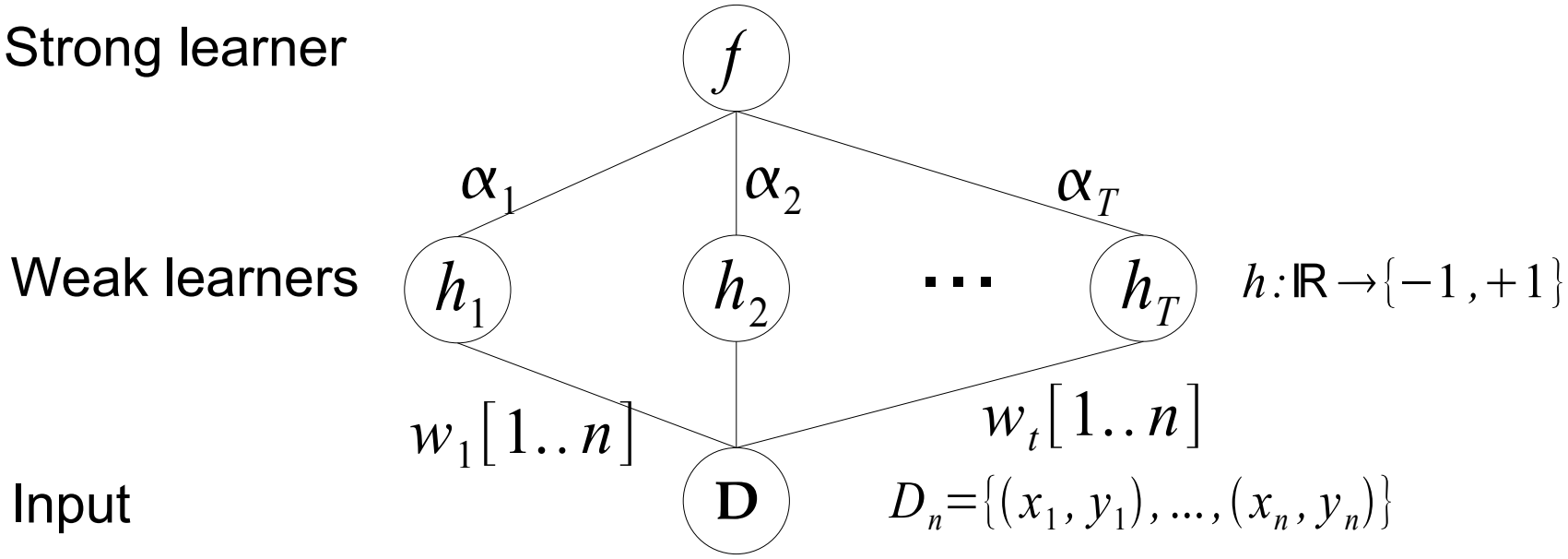
# The Features

- The output of this simple filter is already revealing a separation between speech and music
- Accuracy ~68%
- Still too weak alone



# AdaBoost

- Successful general purpose learning method
- At each iteration $t$ a weak learner $h$ returns a binary prediction with error epsilon.



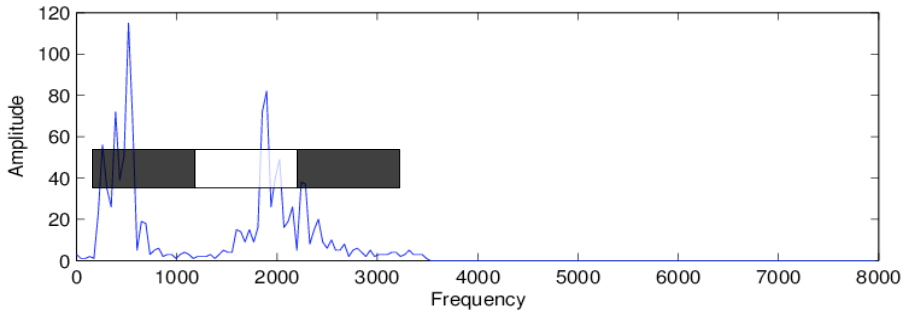
# AdaBoost

- Set a weight distribution $\mathbf{w}^t = (w_1^t, \ldots, w_n^t)$ over the data points

# AdaBoost

- Set a weight distribution $\mathbf{w}^t = (w_1^t, \ldots, w_n^t)$ over the data points
- For 1 ... T
  - Find $h^t$ by minimizing the weighted error

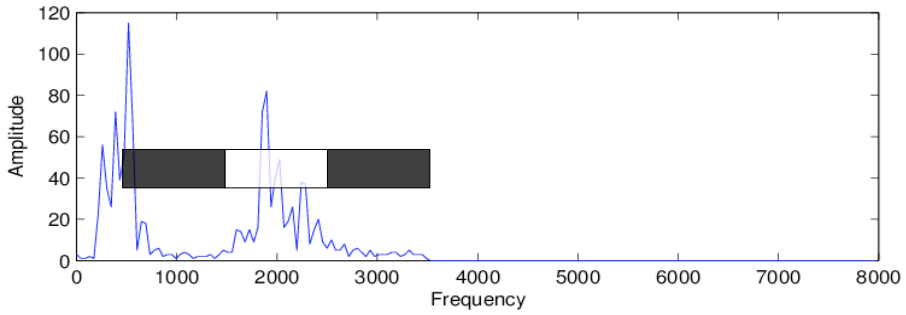
    $\epsilon = \sum_{i=1}^{n} I_{\{h^t(x_i) \neq y_i\}} w_i^t$ over the features parameters



# AdaBoost

- Set a weight distribution $\mathbf{w}^t = (w_1^t, \ldots, w_n^t)$ over the data points
- For 1 ... T
  - Find $h^t$ by minimizing the weighted error

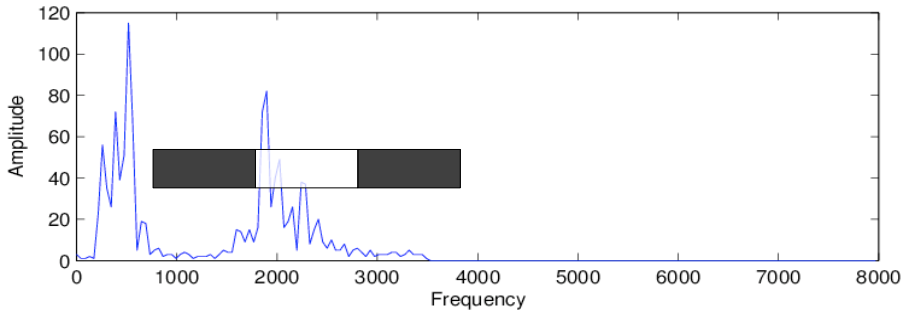
    $\epsilon = \sum_{i=1}^{n} I_{\{h^t(x_i) \neq y_i\}} w_i^t$ over the features parameters



# AdaBoost

- Set a weight distribution $\mathbf{w}^t = (w_1^t, \ldots, w_n^t)$ over the data points
- For 1 ... T
  - Find $h^t$ by minimizing the weighted error

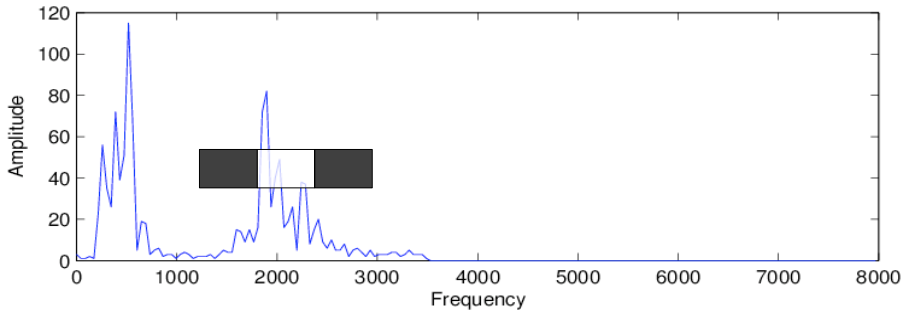
    $\epsilon = \sum_{i=1}^{n} I_{\{h^t(x_i) \neq y_i\}} w_i^t$ over the features parameters



# AdaBoost

- Set a weight distribution $\mathbf{w}^t = (w_1^t, \ldots, w_n^t)$ over the data points
- For 1 ... T
  - Find $h^t$ by minimizing the weighted error

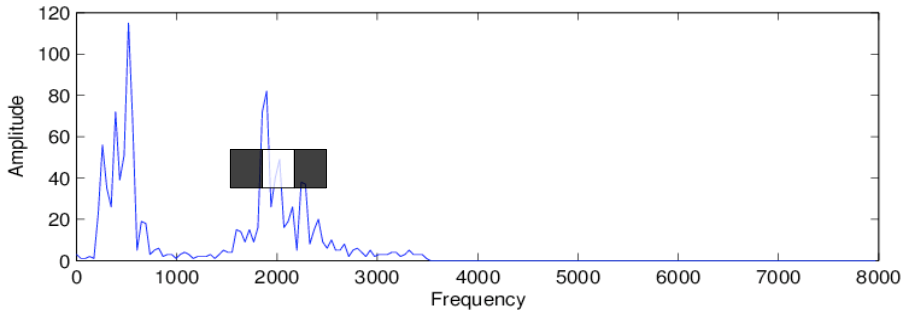
    $\epsilon = \sum_{i=1}^{n} I_{\{h^t(x_i) \neq y_i\}} w_i^t$ over the features parameters



# AdaBoost

- Set a weight distribution $\mathbf{w}^t = (w_1^t, \ldots, w_n^t)$ over the data points
- For 1 ... T
  - Find $h^t$ by minimizing the weighted error

    $\epsilon = \sum_{i=1}^{n} I_{\{h^t(x_i) \neq y_i\}} w_i^t$ over the features parameters



# AdaBoost

- Set a weight distribution $\mathbf{w}^t = (w_1^t, \ldots, w_n^t)$ over the data points
- For 1 ... T
  - Find $h^t$ by minimizing the weighted error $\epsilon = \sum_{i=1}^{n} I_{\{h^t(x_i) \neq y_i\}} w_i^t$ over the features parameters
  - Compute the confidence $\alpha^t = \frac{1}{2} \ln\left(\frac{1-\epsilon^t}{\epsilon^t}\right)$

# AdaBoost

- Set a weight distribution $\mathbf{w}^t = (w_1^t, \ldots, w_n^t)$ over the data points
- For 1 ... T
  - Find $h^t$ by minimizing the weighted error $\epsilon = \sum_{i=1}^{n} I_{\{h^t(x_i) \neq y_i\}} w_i^t$ over the features parameters
  - Compute the confidence $\alpha^t = \frac{1}{2} \ln\left(\frac{1-\epsilon^t}{\epsilon^t}\right)$
  - Update weight vector $\mathbf{w}$

$$w_i^{t+1} = w_i^t \times \begin{cases} \frac{1}{2(1-\epsilon^t)} & \text{if } h^t(x_i) = y_i \\ \frac{1}{2\epsilon^t} & \text{if } h^t(x_i) \neq y_i \end{cases}$$

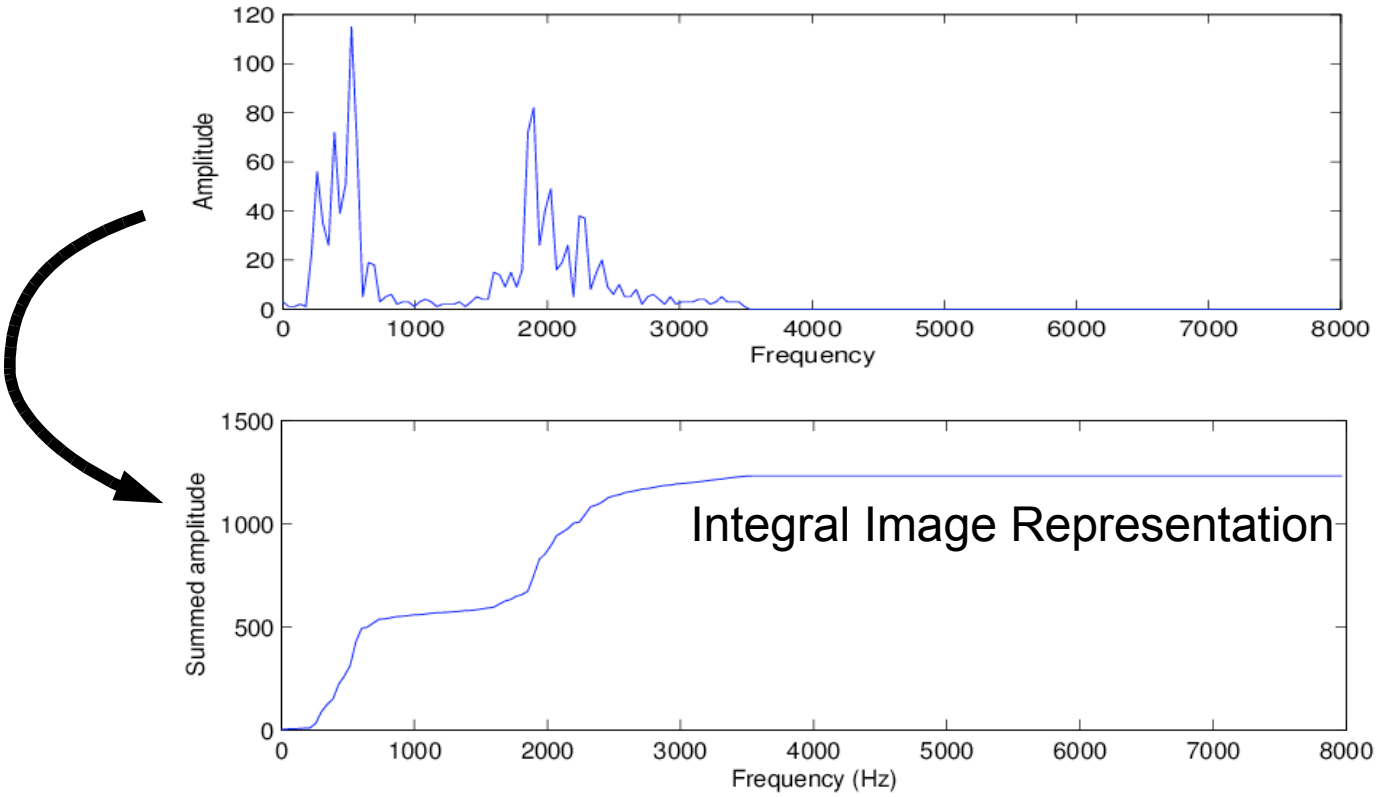
# AdaBoost

- Set a weight distribution $\mathbf{w}^t = (w_1^t, \ldots, w_n^t)$ over the data points
- For 1 ... T
  - Find $h^t$ by minimizing the weighted error $\epsilon = \sum_{i=1}^{n} I_{\{h^t(x_i) \neq y_i\}} w_i^t$ over the features parameters
  - Compute the confidence $\alpha^t = \frac{1}{2} \ln\left(\frac{1-\epsilon^t}{\epsilon^t}\right)$
  - Update weight vector $\mathbf{w}$
- Output the final strong learner

$$f(x) = sign\left(\sum_{t=1}^{T} \alpha^t h^t(x)\right)$$

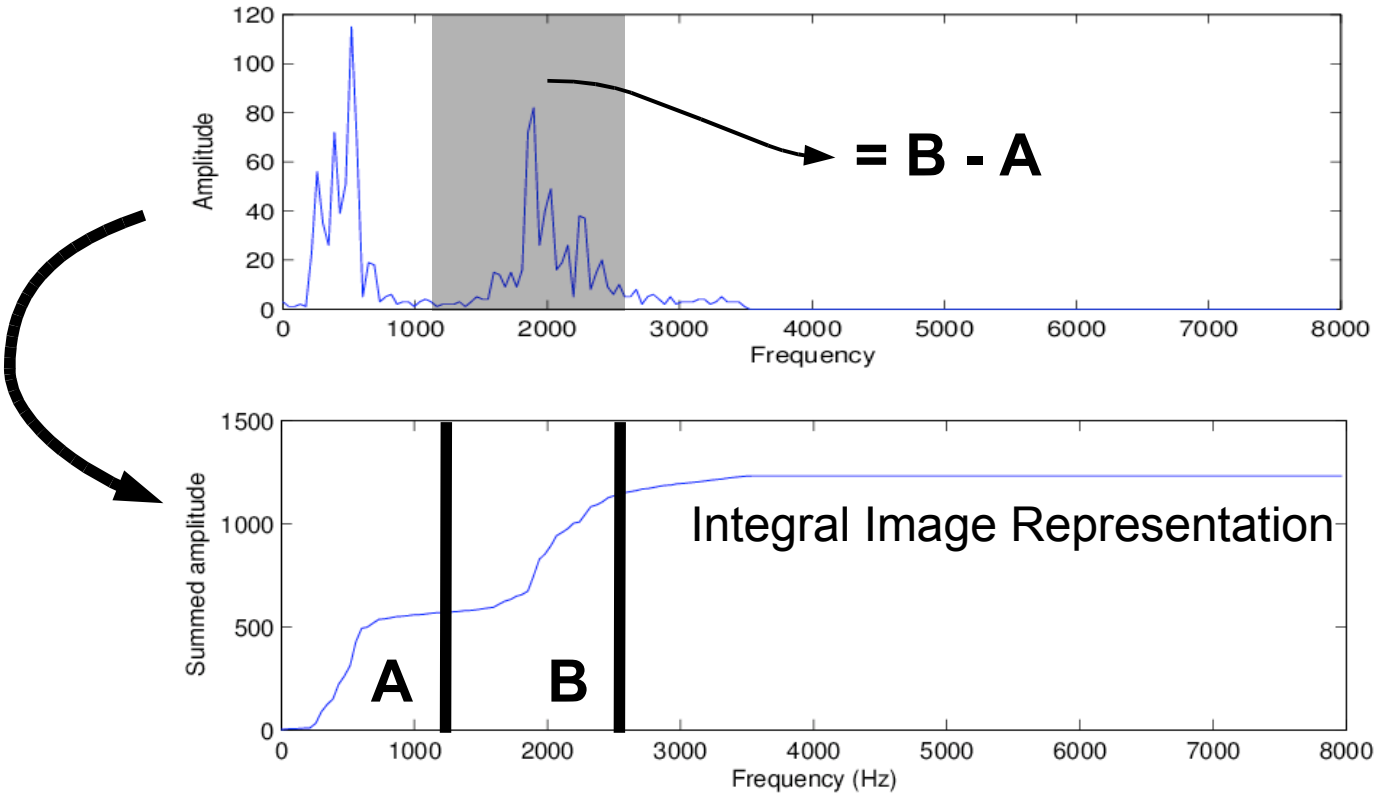
# Optimization

- The amount of energy of the filter can be computed in constant time



# Optimization

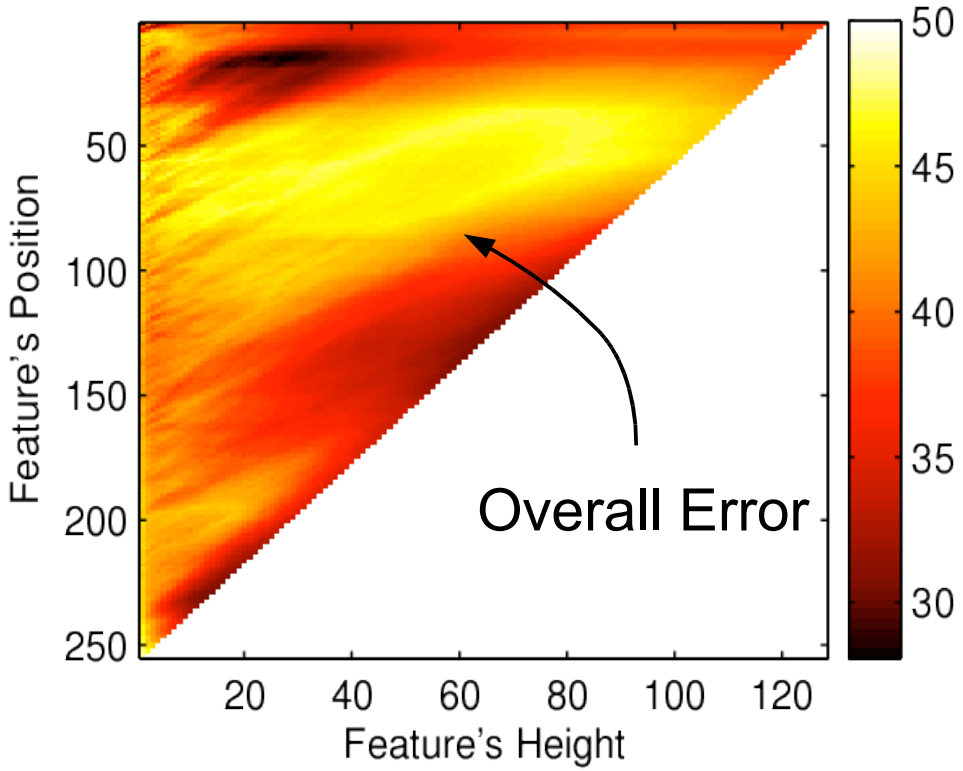
- The convolution of the filter can be computed in constant time



# Optimization

- A comprehensive search in the parameter space can be avoided

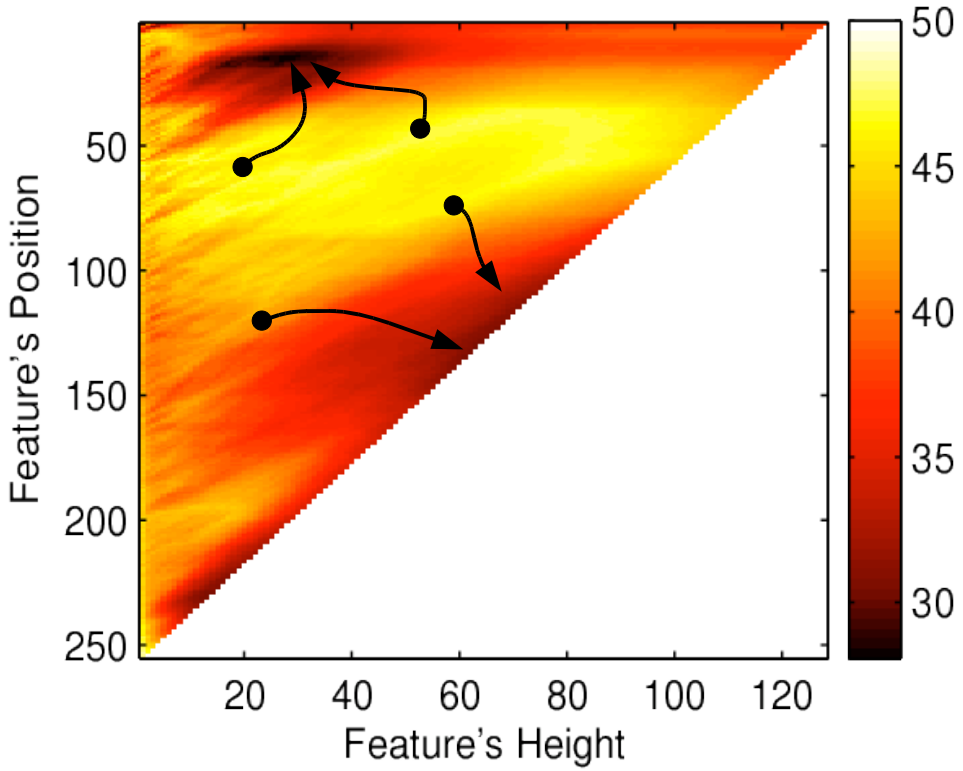
- **Observation**: Slight change in the parameters do not change the error significantly



# Optimization

- A set of random starting point is chosen, then a discrete gradient descent is performed.

- The overall performance is equivalent at the cost of few more iterations.



# Optimization

- The performance can be increased by a simple smoothing on the output of the previous $k$ frames
- **Observation**:
  If in the last $k$ frames there has been speech (or music), it is highly probable that current frame $\tau$ will be speech (or music) too.

$$g(x_\tau) = \frac{\sum_{i=\tau-k}^{\tau} a^{\tau-i} f(x_i)}{\sum_{j=\tau-k}^{\tau} a^{\tau-j}}$$

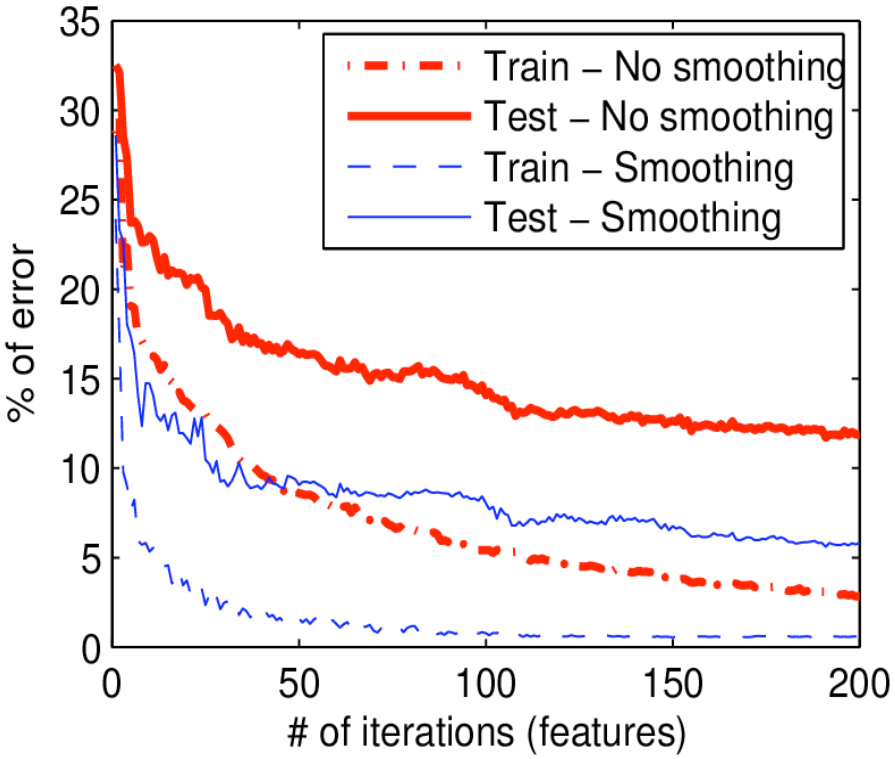
# Results

- Dataset of real world radio transmission with music, talks, jingles, etc..
  used by Scheirer and Slaney (1997)

- 11200 frames of normalized and processed with 20 ms STFT at a resolution of 256 points.

# Results

- The error reaches a plateau after ~150 iterations/filters
- At frame level already the error get to ~12%
- With the smoothing the error drops to 6.7%



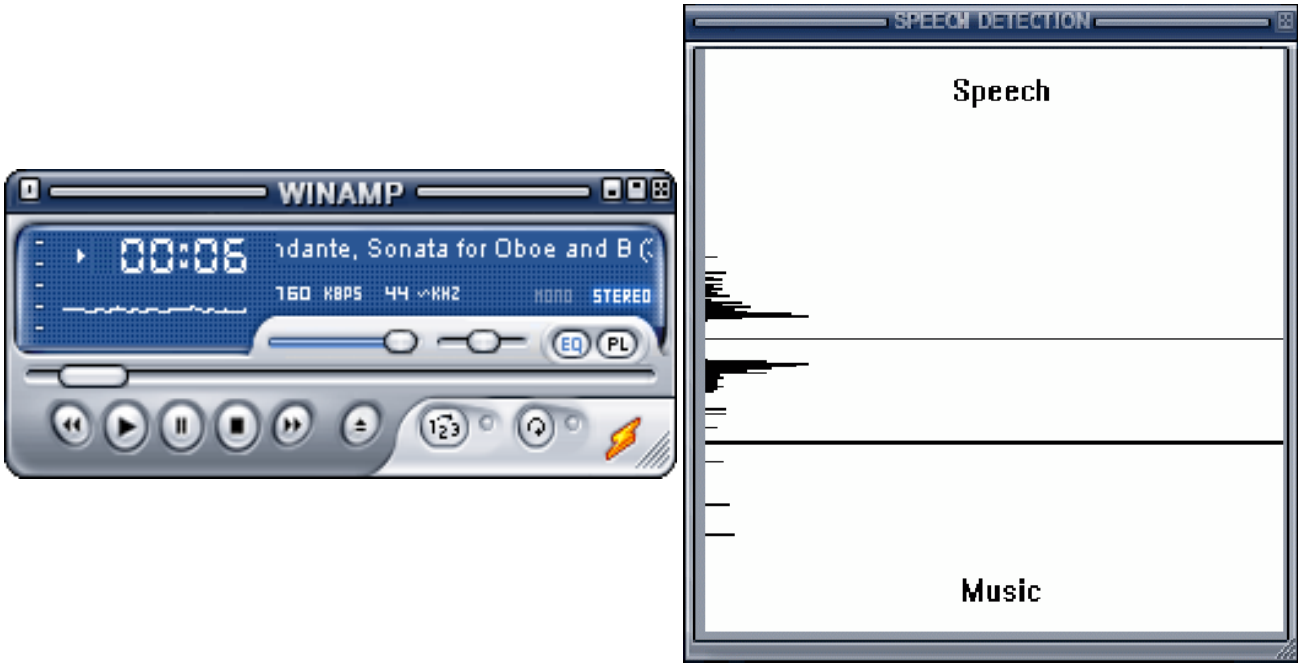
# Results

- Better than typical frame level features

| Feature | CPU | Error |
|---|---|---|
| Rolloff | 17% | ~46 ±3 % |
| Spec. Cent. | 17% | ~39 ±8 % |
| Spec. Flux | 17% | ~39 ±1 % |
| ZCR | 0% | ~38 ±4 % |
| Ceps Resid | 46% | ~37 ±7 % |
| **Proposed** | **<1%** | **~12 ±2 %** |

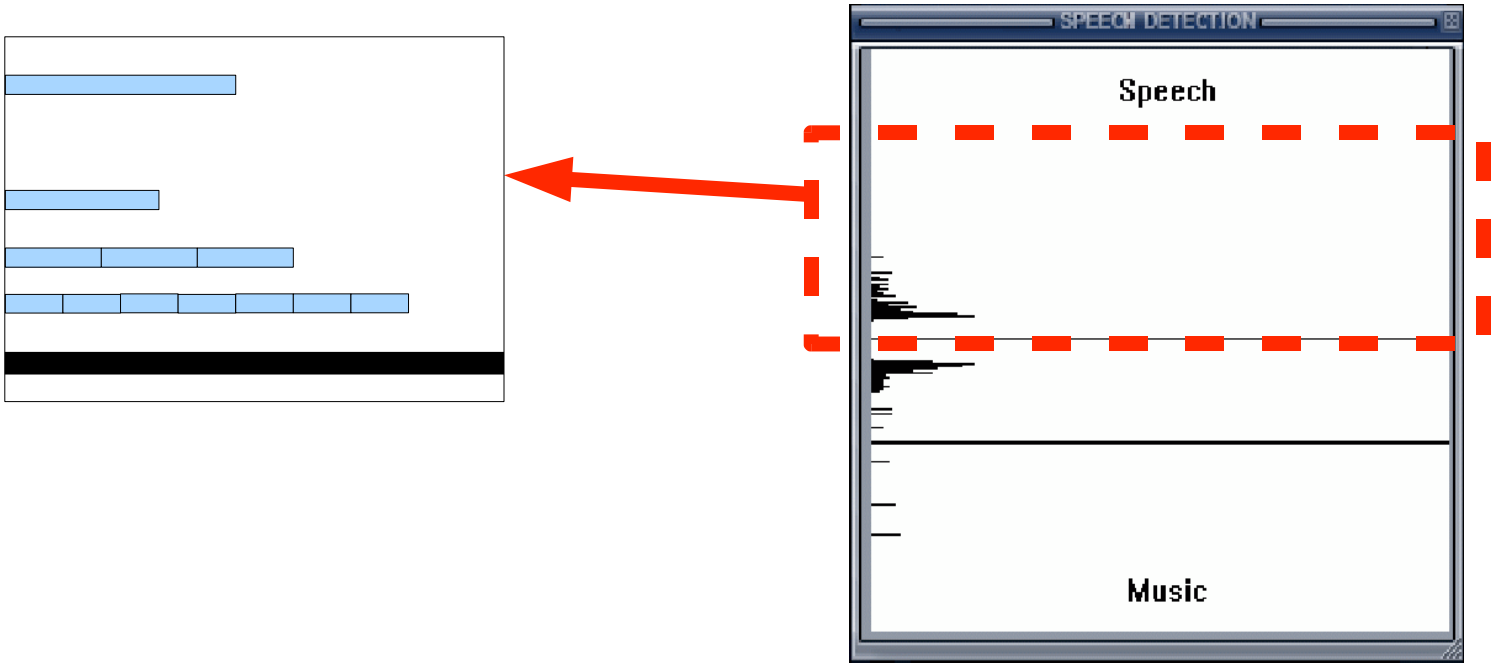
# Demo

- Real-time implementation as Winamp plugin available at: www.iro.umontreal.ca/~casagran/winamp



# Demo

- Real-time implementation as Winamp plugin available at: www.iro.umontreal.ca/~casagran/winamp



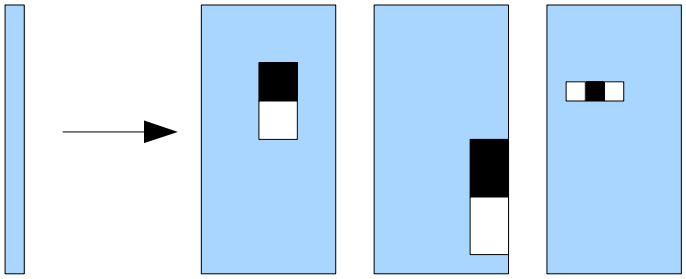
# Conclusions

- Best suited in an *ensemble* algorithm among other features
- Alone it is already capable of good performance
- Extremely fast during detection

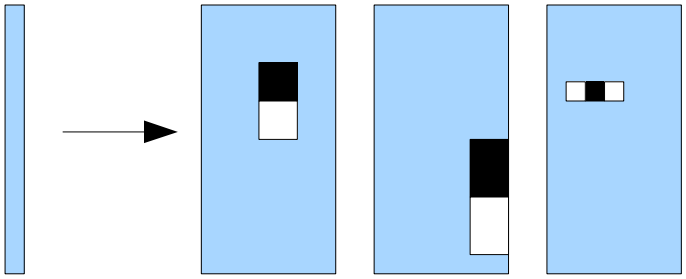
# Conclusions

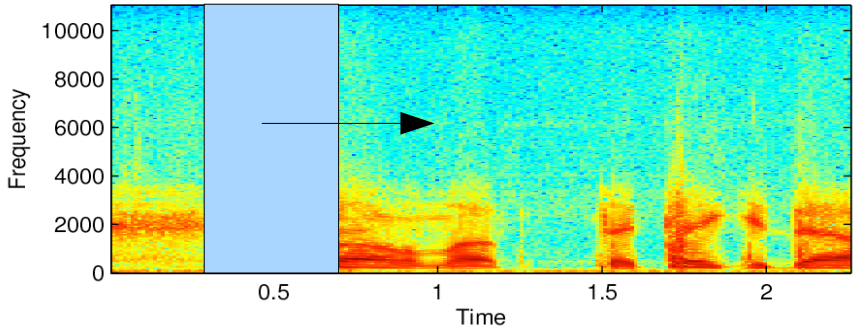
- Best suited in an *ensemble* algorithm among other features
- Alone it is already capable of good performance
- Extremely fast during detection
- Can learn any type of pattern
  - The patterns do not need to be limited to one frame!

# Conclusions



# Conclusions





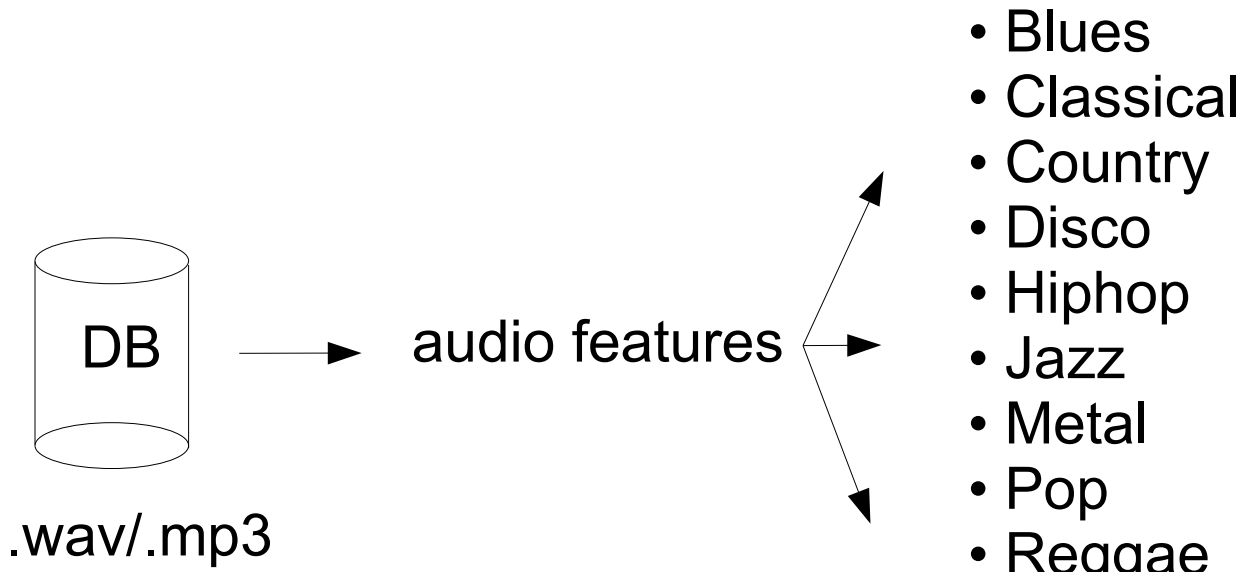
# Conclusions

- Best suited in an *ensemble* algorithm among other features
- Alone it is already capable of good performance
- Extremely fast during detection
- Can learn any type of pattern
  - The patterns do not need to be limited to one frame!
- Simple and easy implementation

# End Part 1

- Thank you!
  - Questions?

# Part 2 – Audio Genre Recognition



# Audio Features : why?

- Signal audio is a real-valued vector
- Why not classify it directly?
  - Very high-dimensional vector

# Audio Features : why?

- Signal audio is a real-valued vector
- Why not classify it directly?
  - Very high-dimensional vector
  - classification should have shift invariance

# Audio Features : why?

- Signal audio is a real-valued vector
- Why not classify it directly?
  - Very high-dimensional vector
  - classification should have shift invariance
  - small differences in spectral magnitude are...
    - important for small magnitudes (quiet frequencies)
    - not important for large magnitudes (loud frequencies)
  - global frequency scaling is irrelevant
  - global frequency shifting is highly relevant

# Audio Features : essence of music?

- Timbre – almost-instant sound quality

- Rhythm – repeated sound structure over a few seconds

# Audio Features : essence of music?

- Timbre – almost-instant sound quality
  - cepstral coefficients: real(fft (log( T abs(fft(s)) ) ) )
    - rceps: T = I
    - mfcc: T implements a Mel-scale projection
    - other options for T are Bark-scale and log-scale.

- Rhythm – repeated sound structure over a few seconds

# Audio Features : essence of music?

- Timbre – almost-instant sound quality
  - cepstral coefficients: real(fft (log( T abs(fft(s)) ) ) )
    - rceps: T = I
    - mfcc: T implements a Mel-scale projection
    - other options for T are Bark-scale and log-scale.
  - zero-crossing rate

- Rhythm – repeated sound structure over a few seconds

# Audio Features : essence of music?

- Timbre – almost-instant sound quality
  - cepstral coefficients: real(fft (log( T abs(fft(s)) ) ) )
    - rceps: T = I
    - mfcc: T implements a Mel-scale projection
    - other options for T are Bark-scale and log-scale.
  - zero-crossing rate
  - spectral centroid and flatness
    - E[] and Var[] of normalized FFT

- Rhythm – repeated sound structure over a few seconds

# Audio Features : essence of music?

- Timbre – almost-instant sound quality
  - cepstral coefficients: real(fft (log( T abs(fft(s)) ) ) )
    - rceps: T = I
    - mfcc: T implements a Mel-scale projection
    - other options for T are Bark-scale and log-scale.
  - zero-crossing rate
  - spectral centroid and flatness
    - E[] and Var[] of normalized FFT
  - linear predictive coefficients & reconstruction error
- Rhythm – repeated sound structure over a few seconds

# Audio Features : essence of music?

- Timbre – almost-instant sound quality
  - cepstral coefficients: real(fft (log( T abs(fft(s)) ) ) )
    - rceps: T = I
    - mfcc: T implements a Mel-scale projection
    - other options for T are Bark-scale and log-scale.
  - zero-crossing rate
  - spectral centroid and flatness
    - E[] and Var[] of normalized FFT
  - linear predictive coefficients & reconstruction error
- Rhythm – repeated sound structure over a few seconds
  - important, interesting, but not used at MIREX

# Audio Features : for AdaBoost

- Segment Feature Extraction

# Audio Features : for AdaBoost

- Segment Feature Extraction
  - input: a signal of (1024 * c) samples
  - define s(j) to be the sub-signal of length 1024 starting at j*1024
  - define r(j) = ( mfcc(s(j)), rceps(s(j)), lpc(s(j)), zcr(s(j)), ro(s(j)), fftc(s(j)) )

MFCC FFTC

$$\vec{r}(j) = x_0, x_1, \ldots, x_{62}, x_{63}, \ldots, x_{369}, x_{370}, x_{371}, \ldots, x_{401}$$

# Audio Features : for AdaBoost

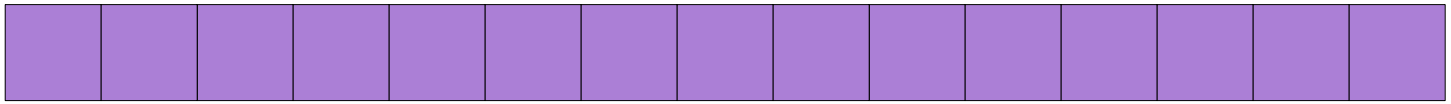
- Segment Feature Extraction
  - input: a signal of (1024 * c) samples
  - define s(j) to be the sub-signal of length 1024 starting at j*1024
  - define r(j) = ( mfcc(s(j)), rceps(s(j)), lpc(s(j)), zcr(s(j)), ro(s(j)), fftc(s(j)) )
  - return ( E[r(J)], Var[ r(J) ] ) for J uniform on [0,c-1]
    - relative ordering of sub-signals is ignored

MFCC FFTC

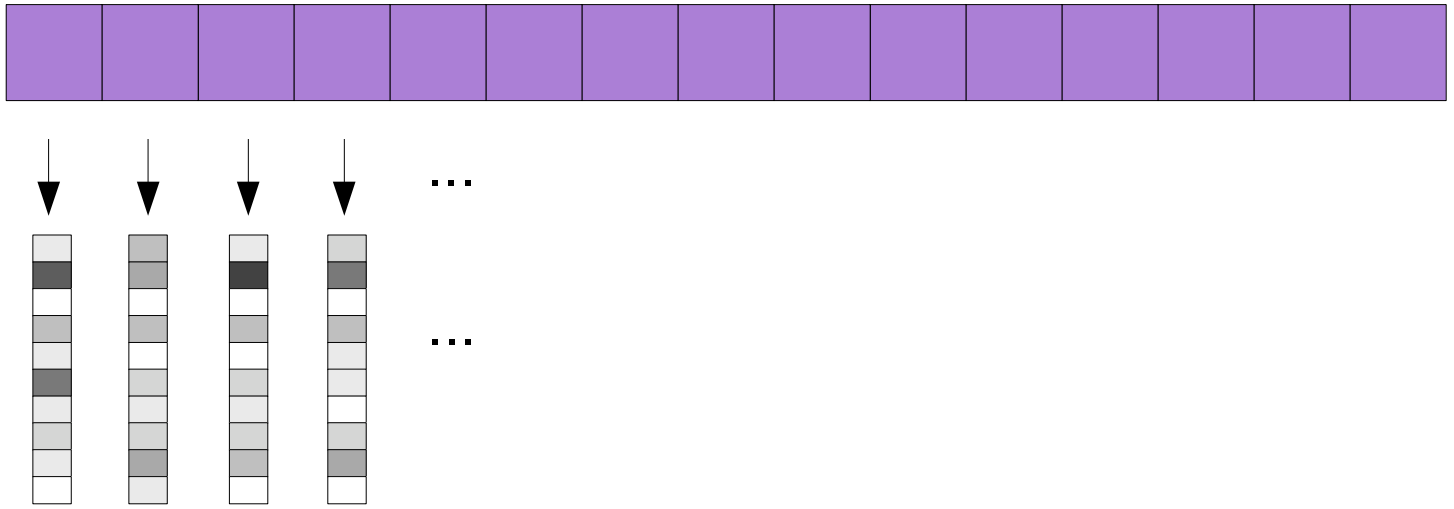
$$\vec{r}(j) = x_0, x_1, \ldots, x_{62}, x_{63}, \ldots, x_{369}, x_{370}, x_{371}, \ldots, x_{401}$$

# Data

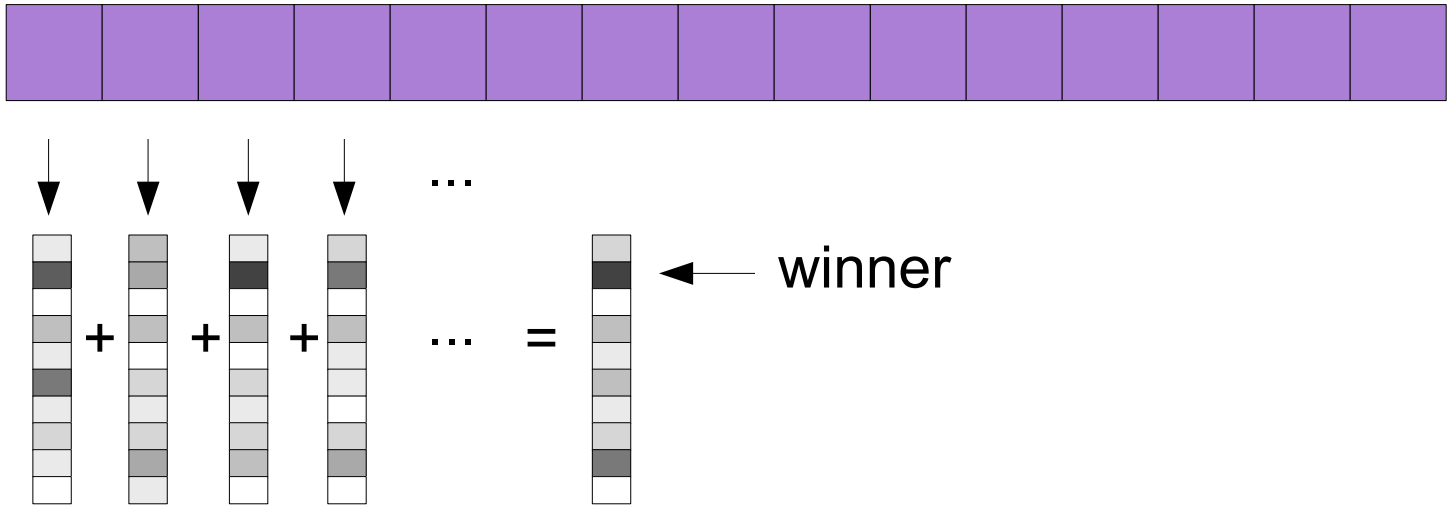
# Data



# Data



# Data



# Algorithm

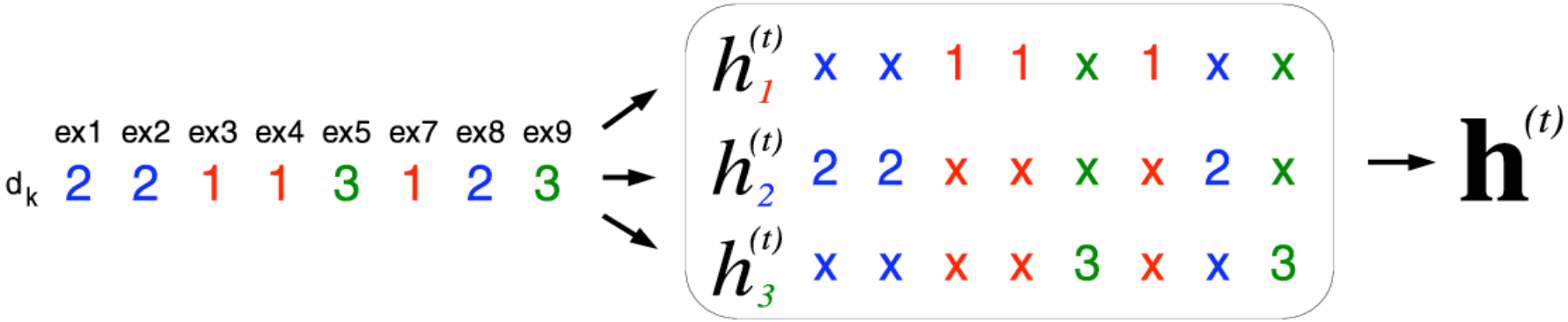
- We used AdaBoost.MH to classify each window

# Algorithm

- We used AdaBoost.MH (Schapire & Singer 1998) to classify each window
  - Weight distribution over examples and classes $w_{i,l}^{t}$

# Algorithm

- We used AdaBoost.MH (Schapire & Singer 1998) to classify each window
  - Weight distribution over examples and classes $w_{i,l}^{t}$
  - Find the dimension and threshold that minimizes the weighted error on one-vs-all binary classifiers



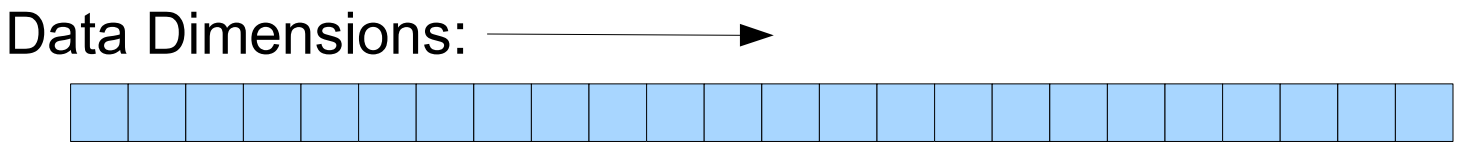
# Evaluation

- Tzanetakis Database
  - 1100 files
  - 10 classes:
    - blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, rock

| | Correct Rate |
|---|---|
| G. Tzanetakis (2002) | 61% |
| T. Li (2003) | 79% |
| Our Approach (MIREX) | **83%** |
| With Autocorrelation | 86% |

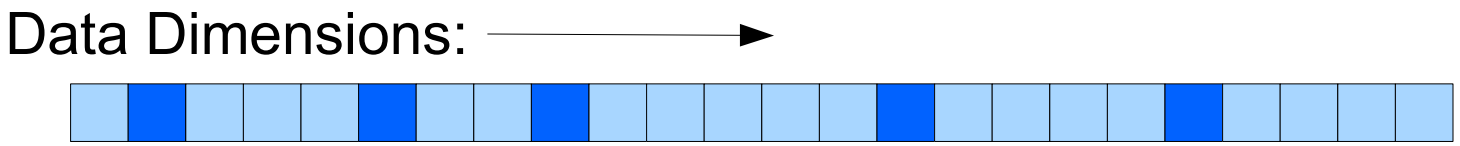
# Evaluation

- The *weak* map



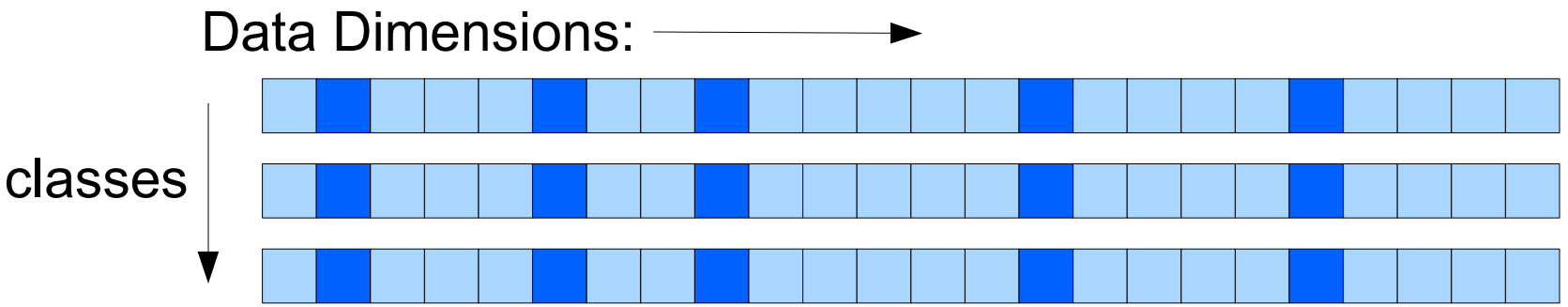
# Evaluation

- The *weak* map



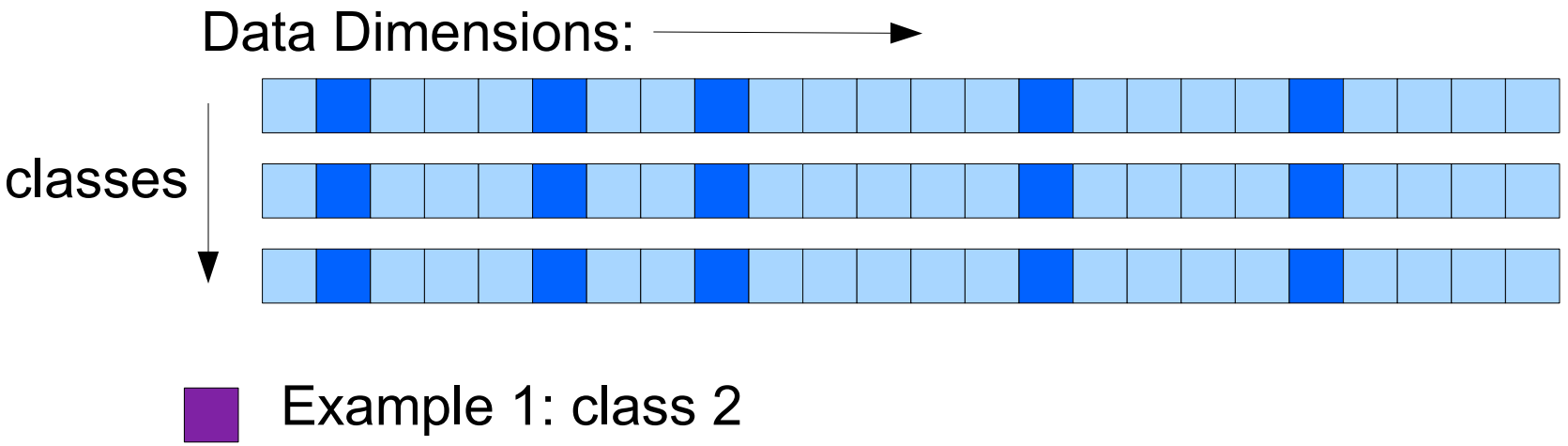
# Evaluation

- The *weak* map



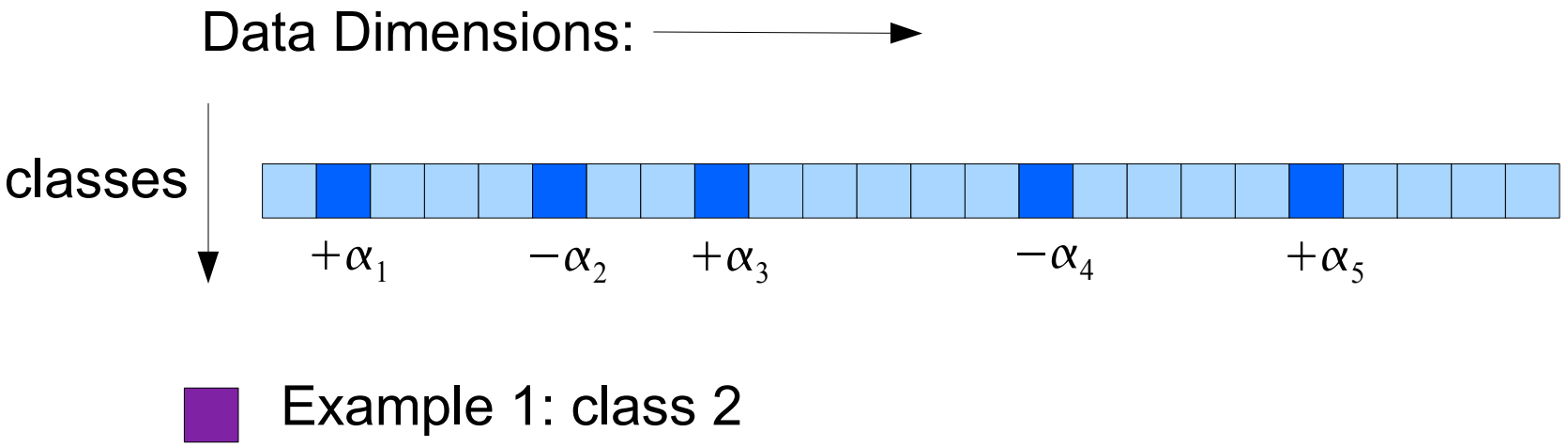
# Evaluation

- The *weak* map



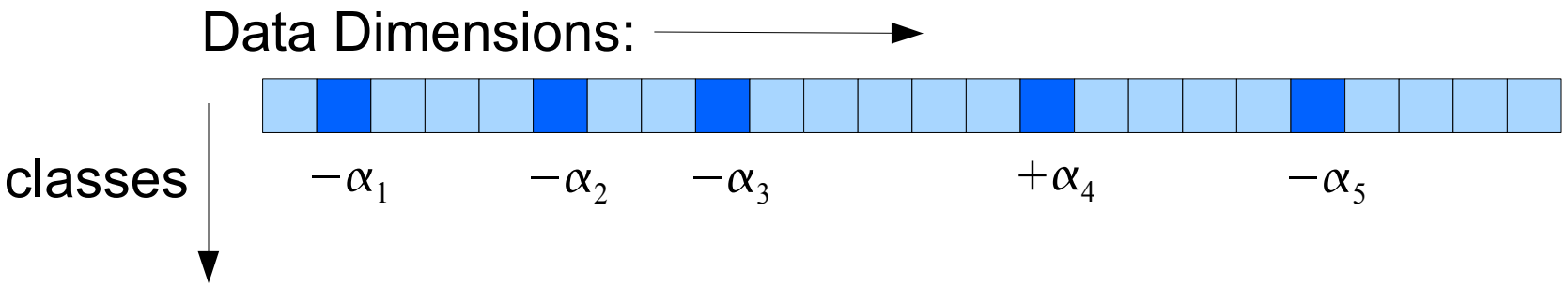
# Evaluation

- The *weak* map



# Evaluation

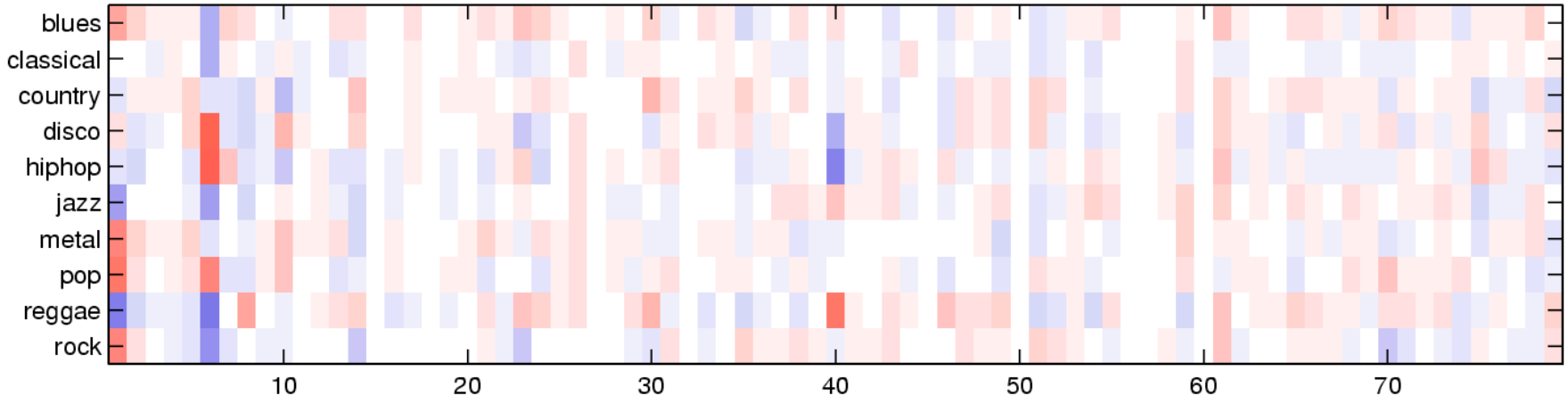
- The *weak* map




Example 2: class 1

# Evaluation

- The *weak* map
  - RCEPS



# Mirex Competition

- Two Databases
  - Magnatune – 10 classes
    - ambient, blues, classical, electronic, ethnic, folk, jazz, new age, punk, rock.
  - USPOP – 6 classes
    - country, electronica & dance, new age, rap & hiphop ,reggae, rock.

# Mirex Competition Results - USPS

| Rank | Participant | USPOP Raw Classification Accuracy |
|---|---|---|
| 1 | Bergstra, Casagrande, Eck & Kégl (1) | **86.29%** |
| 2 | Mandel & Ellis | 85.65% |
| 3 | Pampalk, E. | 80.38% |
| 4 | Lidy & Rauber (SSD+RH) | 79.75% |
| 5 | West, K. | 78.90% |
| 6 | Lidy & Rauber (RP+SSD) | 78.48% |
| 6 | Ahrendt, P. | 78.48% |
| 8 | Lidy & Rauber (RP+SSD+RH) | 78.27% |
| 9 | Scaringella, N. | 75.74% |
| 10 | Soares, V. | 66.67% |
| 11 | Tzanetakis, G. | 63.29% |
| 12 | Burred, J. | 47.68% |
| 13 | Chen & Gao | 22.93% |
| 14 | Li, M. | TO * |

# Mirex Competition Results - Magna

| Rank | Participant | Magnatune Hierarchical Classification Accuracy |
|---|---|---|
| 1 | Bergstra, Casagrande, Eck & Kégl (1) | **77.25%** |
| 2 | Mandel & Ellis | 71.96% |
| 3 | West, K. | 71.67% |
| 4 | Lidy & Rauber (RP+SSD) | 71.08% |
| 5 | Lidy & Rauber (RP+SSD+RH) | 70.88% |
| 6 | Lidy & Rauber (SSD+RH) | 70.78% |
| 7 | Scaringella, N. | 70.47% |
| 8 | Pampalk, E. | 69.90% |
| 9 | Ahrendt, P. | 64.61% |
| 10 | Burred, J. | 59.22% |
| 11 | Tzanetakis, G. | 58.14% |
| 12 | Soares, V. | 55.29% |
| 13 | Li, M. | TO * |
| 13 | Chen & Gao | DNC * |

# Mirex Competition Results - Overall

| Rank | Participant | Mean of Magnatune Hierarchical Classification Accuracy and USPOP Raw Classification Accuracy |
|---|---|---|
| 1 | Bergstra, Casagrande, Eck & Kégl (1) | **81.77%** |
| 2 | Mandel & Ellis | 78.81% |
| 3 | West, K. | 75.29% |
| 4 | Lidy & Rauber (SSD+RH) | 75.27% |
| 5 | Pampalk, E. | 75.14% |
| 6 | Lidy & Rauber (RP+SSD) | 74.78% |
| 7 | Lidy & Rauber (RP+SSD+RH) | 74.58% |
| 8 | Scaringella. N. | 73.11% |
| 9 | Ahrendt, P. | 71.55% |
| 10 | Soares, V. | 60.98% |
| 11 | Tzanetakis, G. | 60.72% |
| 12 | Burred, J. | 53.45% |

# The End - Again

- Thank you!
  - Questions?