DAY 2

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

Imagine Research

CCRMA
Center for Computer Research in Music and Acoustics

Jay LeBoeuf
Imagine Research
jay{at}imagine-research.com

Rebecca Fiebrink
Princeton University
rfiebrink{at}princeton.edu

July 2010

These lecture notes contain hyperlinks to the CCRMA Wiki.

On these pages, you can find supplemental material for lectures - providing extra tutorials, support, references for further reading, or demonstration code snippets for those interested in a given topic .

Click on the symbol on the lower-left corner of a slide to access additional resources.

WIKI REFERENCES…

Review from Day 1

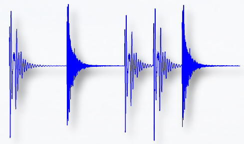
- What are the 3 major components of a MIR system?
- Name 3 ways of segmenting audio into frames

- What problems did you experience in the lab?
- Follow-up questions?
- Did you try other audio files?
- Did you do the simple instrument recognition?

FEATURE DEMOS

- Simple re-ordering or slices:
  - Slice up loop into segments and sort via features
  - Play audio
  - Play whole song snippet

Basic system overview



Segmentation

(Frames, Onsets, Beats, Bars, Chord Changes, etc)

→

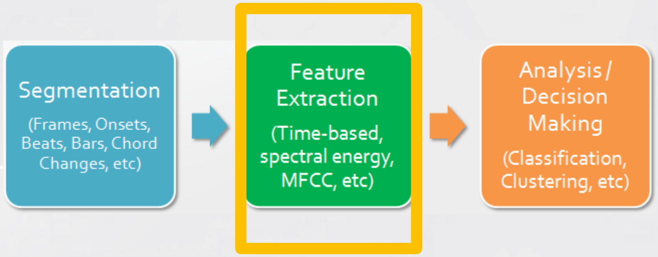
Feature Extraction

(Time-based, spectral energy, MFCC, etc)

→

Analysis / Decision Making

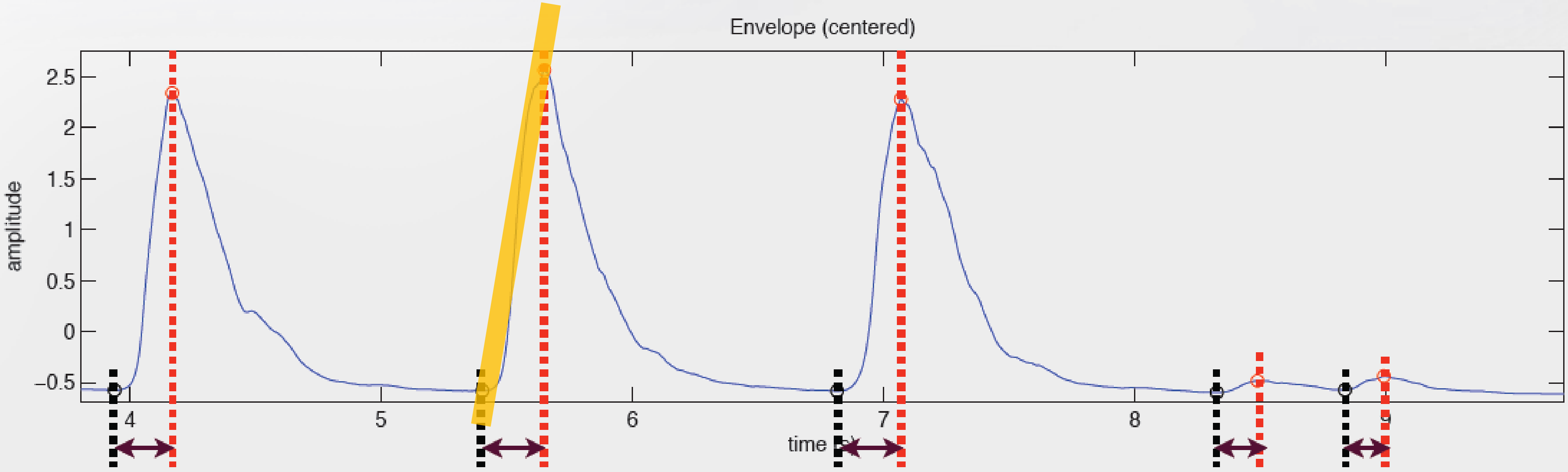
(Classification, Clustering, etc)



FEATURE EXTRACTION

Temporal Information

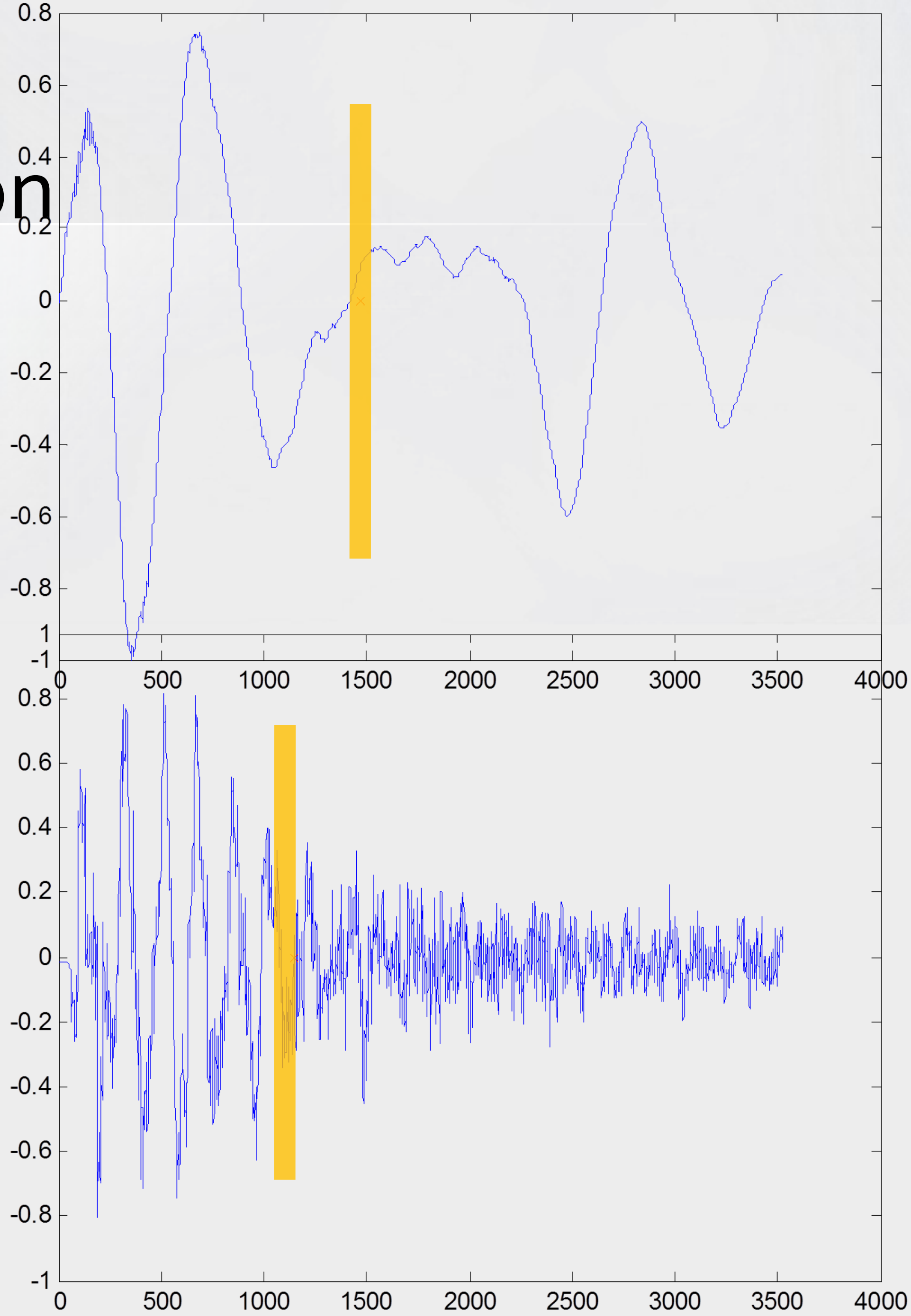
- Rise time or Attack time- time interval between the onset and instant of maximal amplitude
- Attack slope



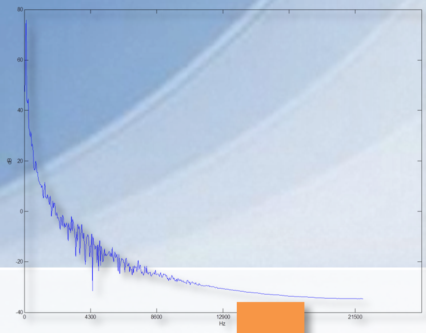
Picture courtesy: Olivier Lartillot

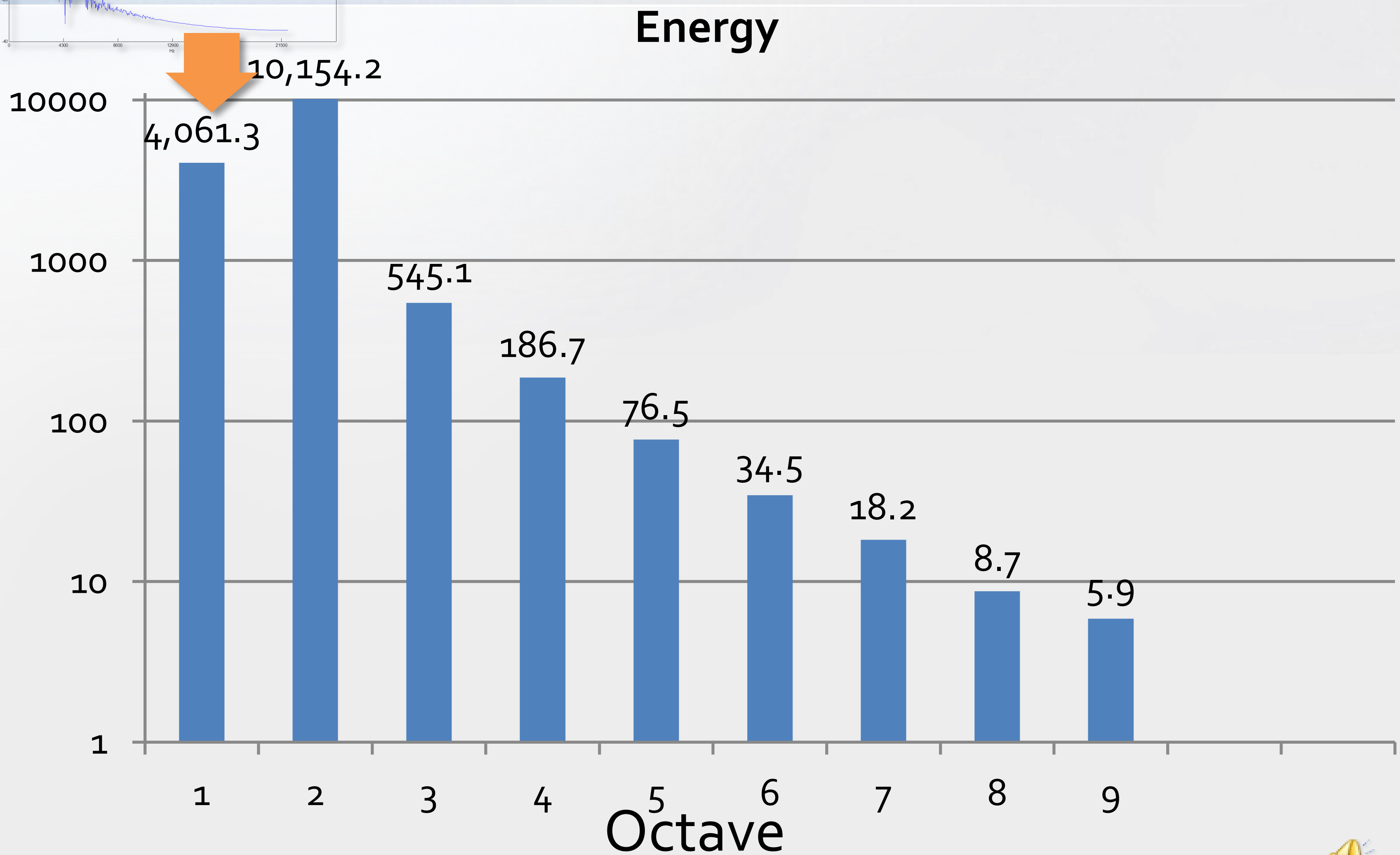
Temporal Information

- Temporal Centroid



Frame 1

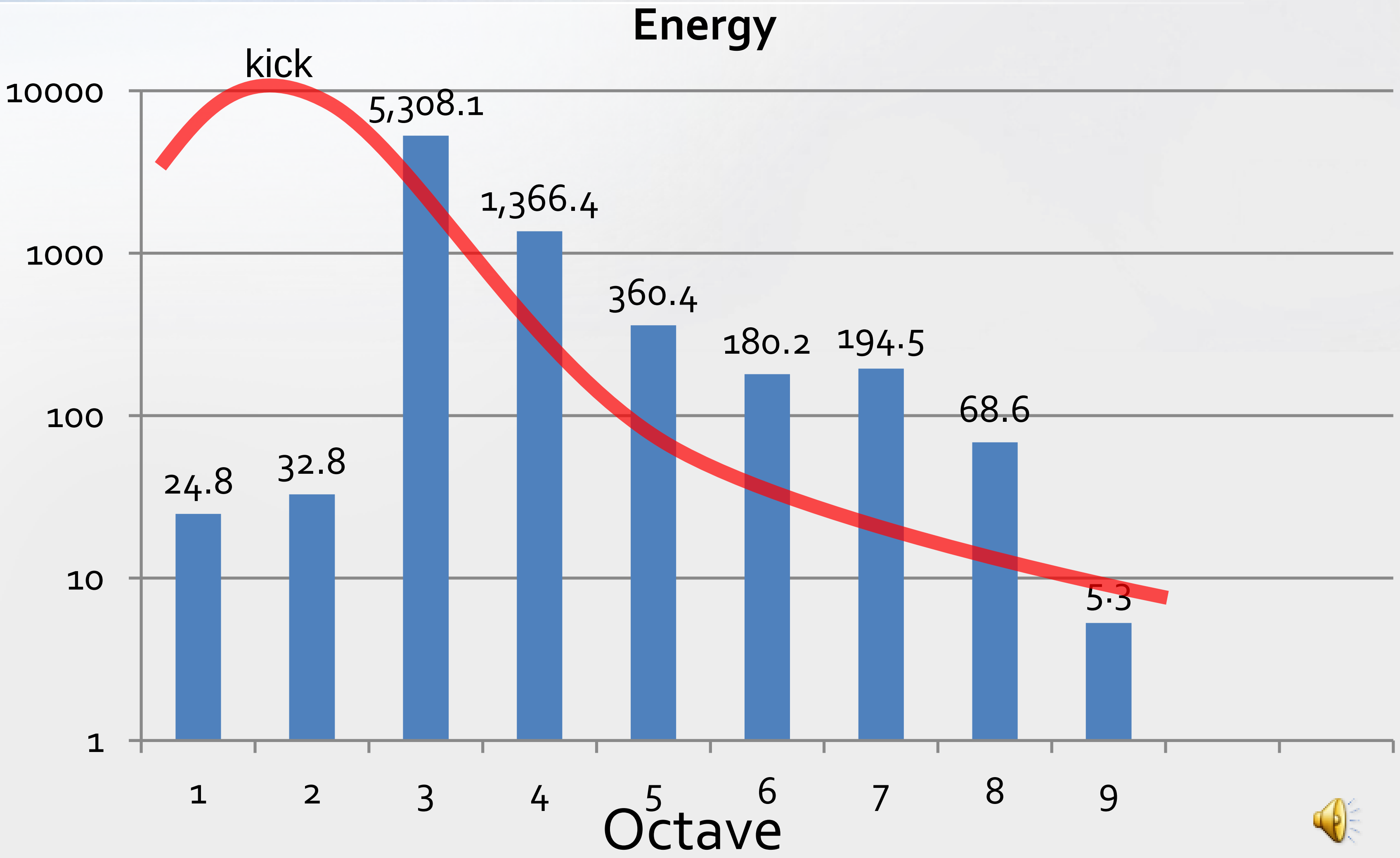



Energy
10000
1000
100
10
1
4,061.3
10,154.2
545.1
186.7
76.5
34.5
18.2
8.7
5.9
1
2
3
4
5
6
7
8
9
Octave

Features – Frame 1

Frame	ZC R	Centroid	BW	Skew	Kurtosis	E1	E2	E3	E4	E5	E6	E7	E8	E9
1	9	2.8kHz	5kHz	2.2	6.7	4000	10100	545	187	77	35	18	9	6

Frame 2



Energy
kick
10000
1000
100
10
1
24.8
32.8
5,308.1
1,366.4
360.4
180.2
194.5
68.6
5.3
1
2
3
4
5
6
7
8
9
Octave

Features : SimpleLoop.wav

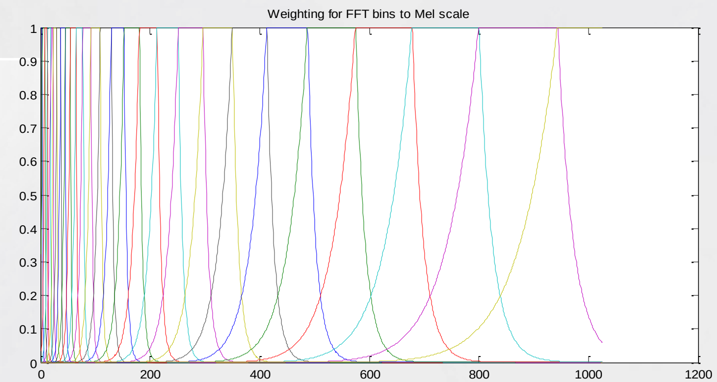
Frame	ZC R	Centroid	BW	Skew	Kurtosis	E1	E2	E3	E4	E5	E6	E7	E8
1	9	2.8kHz	5kHz	2.2	6.7	4000	10100	545	187	77	35	18	9
2	423	3.1kHz	4kHz	2	7.2	24	33	5300	1366	360	180	194	68

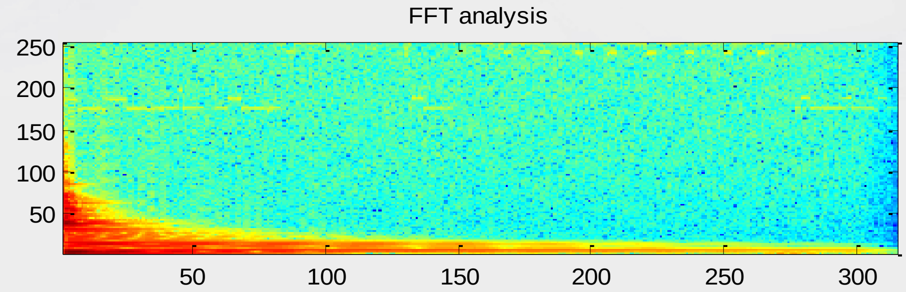
MFCCs

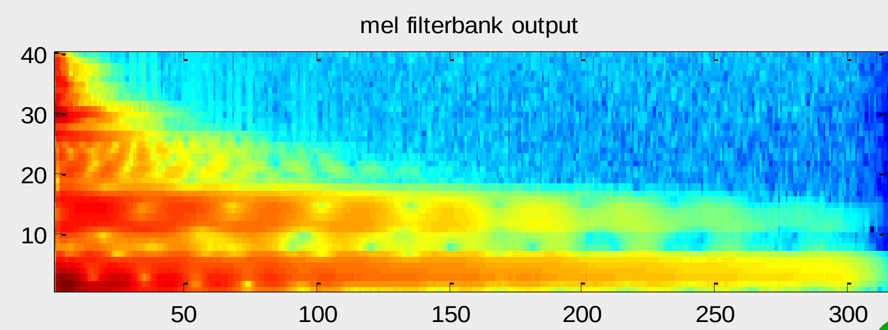
The idea of MFCCs is to capture spectrum in accordance with human perception.

1. STFT
2. log(STFT)
3. Perform mel-scaling to group and smooth coefficients. (perceptual weighting)
4. Decorrelate with DCT

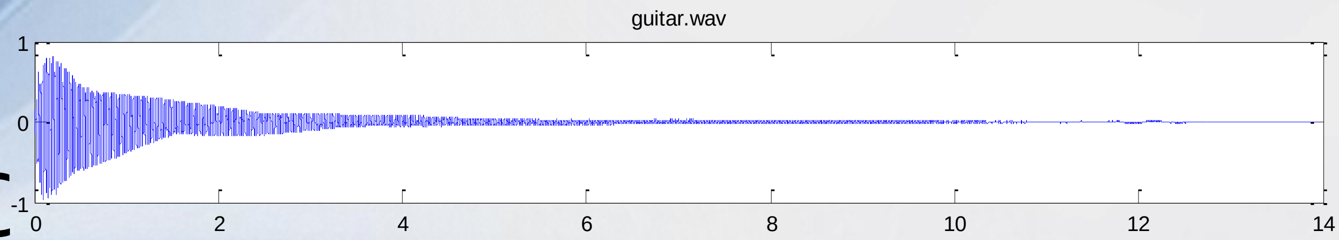
[...continued...]





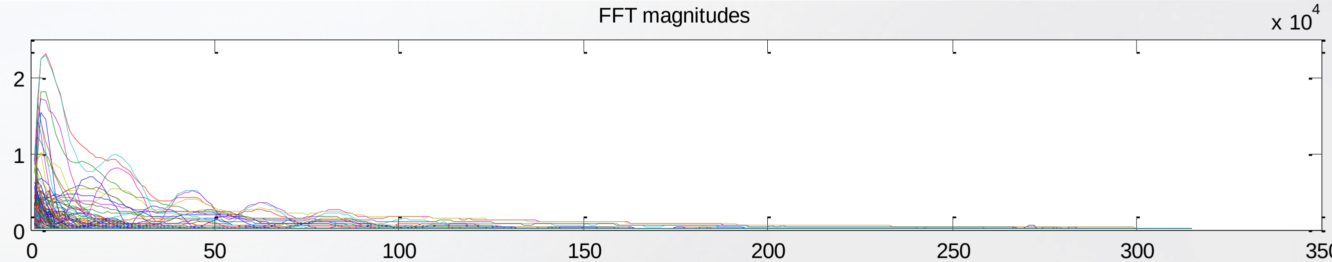


MFCC



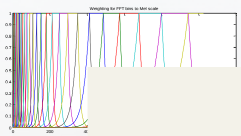
guitar.wav
1
0
-1
0
2
4
6
8
10
12
14
x 10^4

1



FFT magnitudes
2
1
0
0
50
100
150
200
250
300
350

2

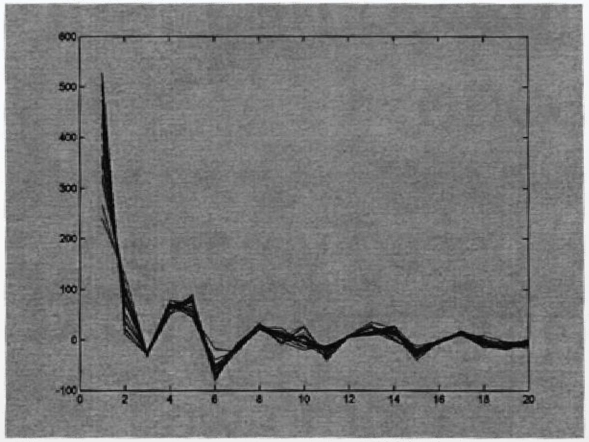


3

4

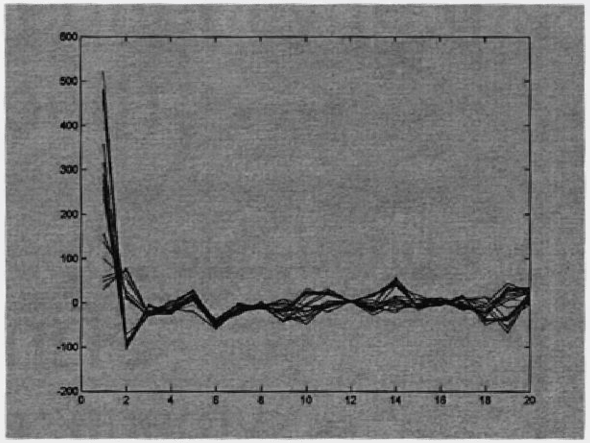
MFCC of Music

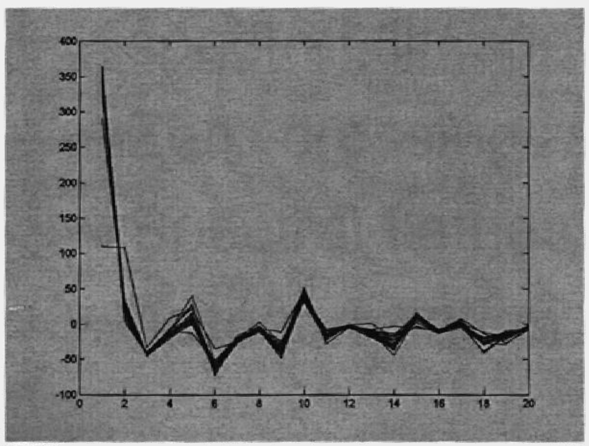
(Petruncio, 2003)



Piano

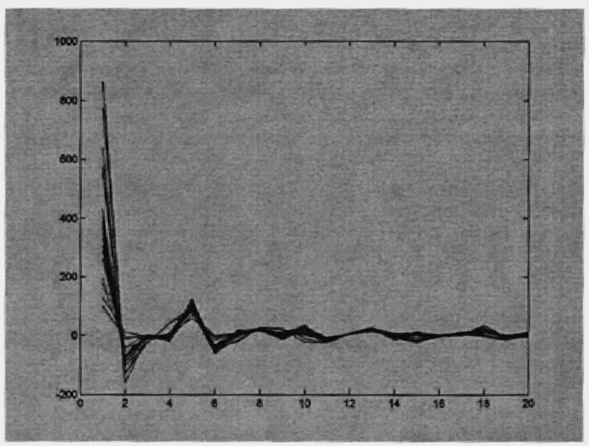
Saxophone





Tenor
Opera
Singer

Drums



Features: Measuring changes

- Δ and Δ Δ
  - Change between frames
  - How quickly the change is occurring

- Spectral flux is the distance between the spectrum of successive frames

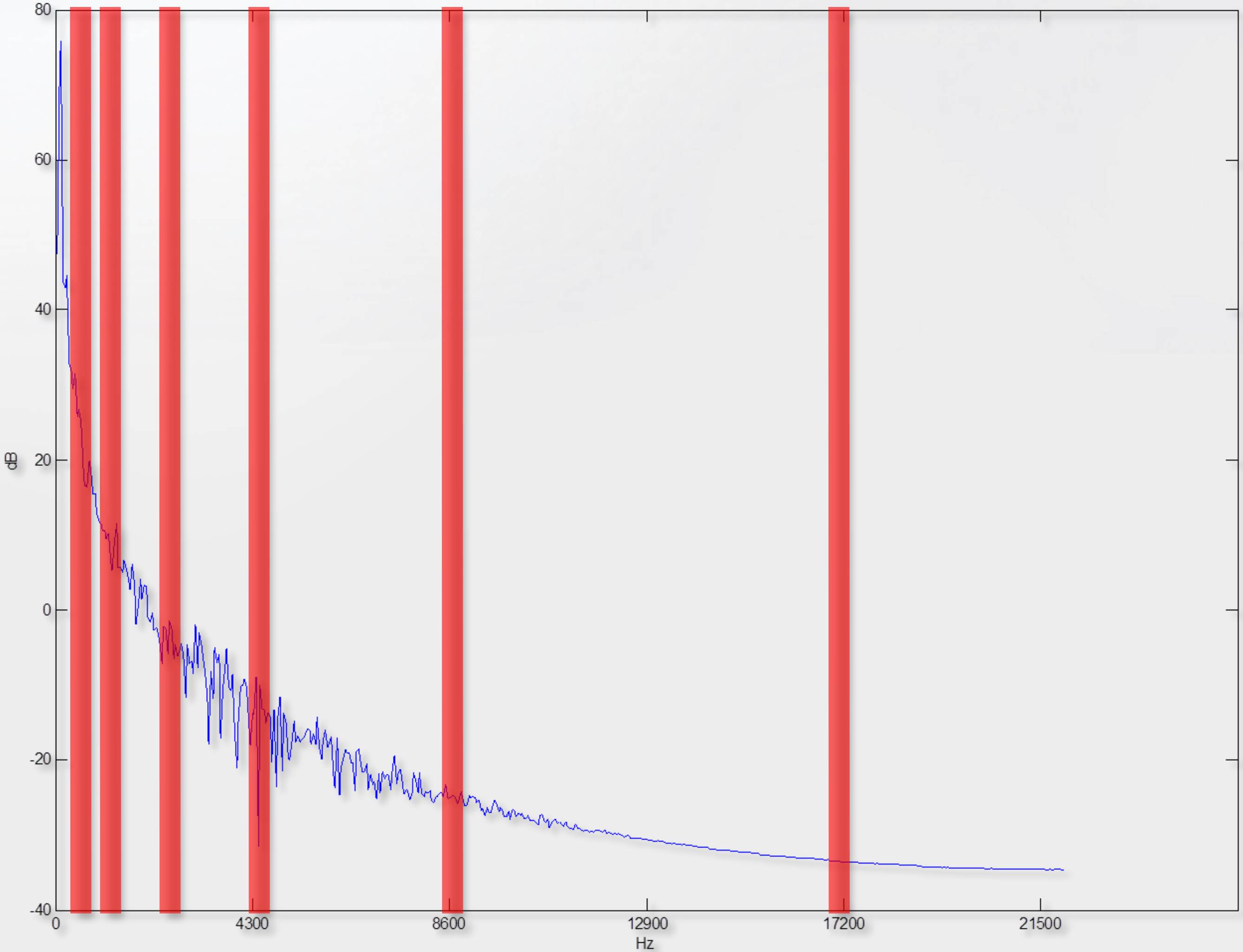
Spectral Features

- Spectral Flatness Measure
- Spectral Crest Factor
- Spectral Flux

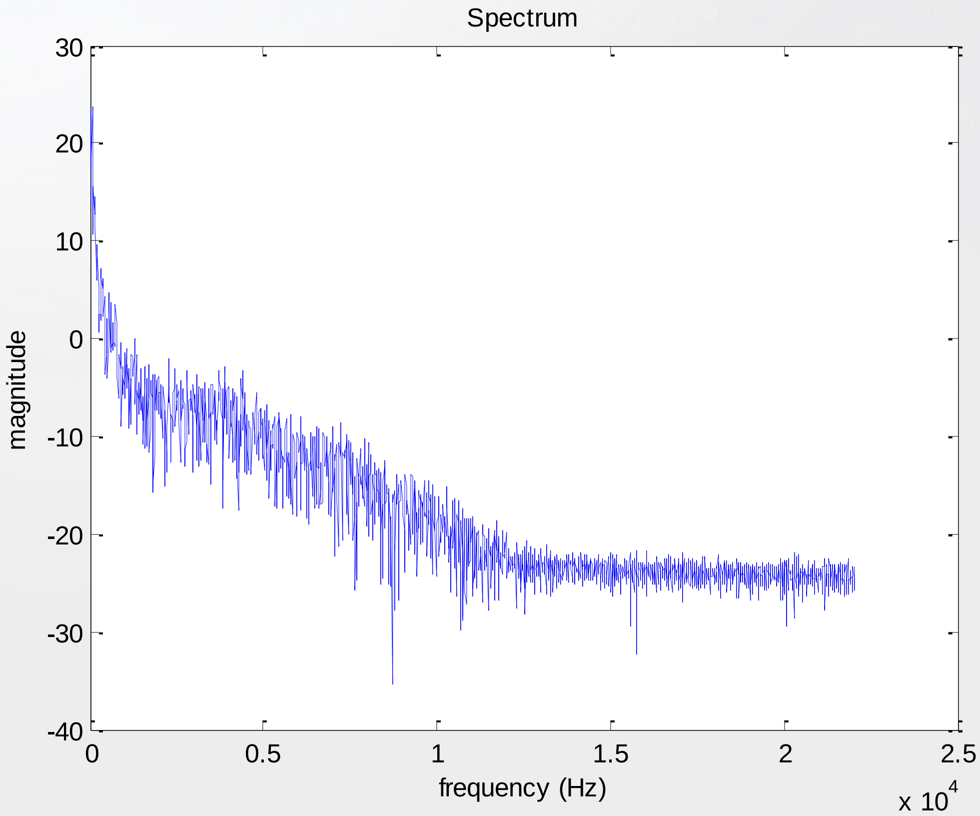
Feature extraction

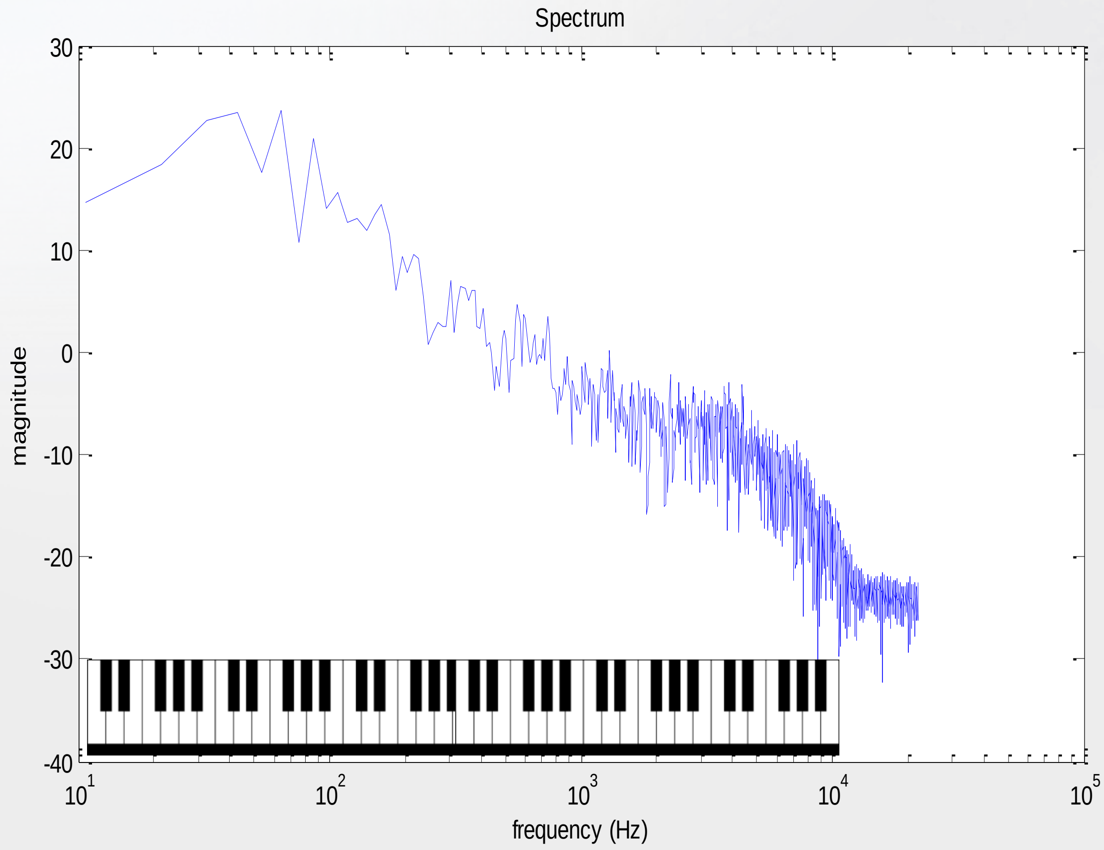
- Feature design and creation uses one’s domain knowledge.
- Choosing discriminating features is critical
- Smaller feature space yields smaller, simpler models, faster training, often less training data needed

Spectral Bands



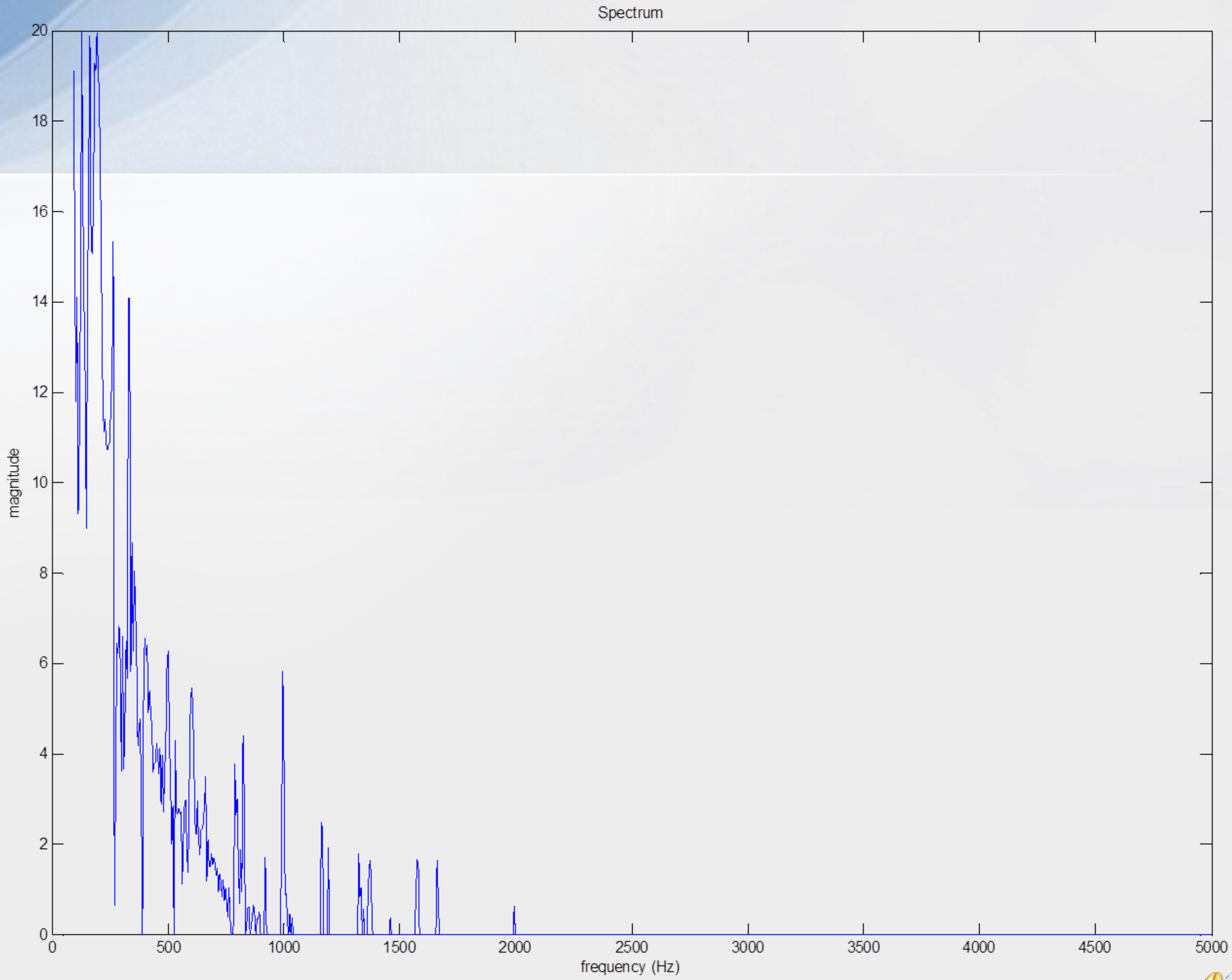
Log Spectrogram

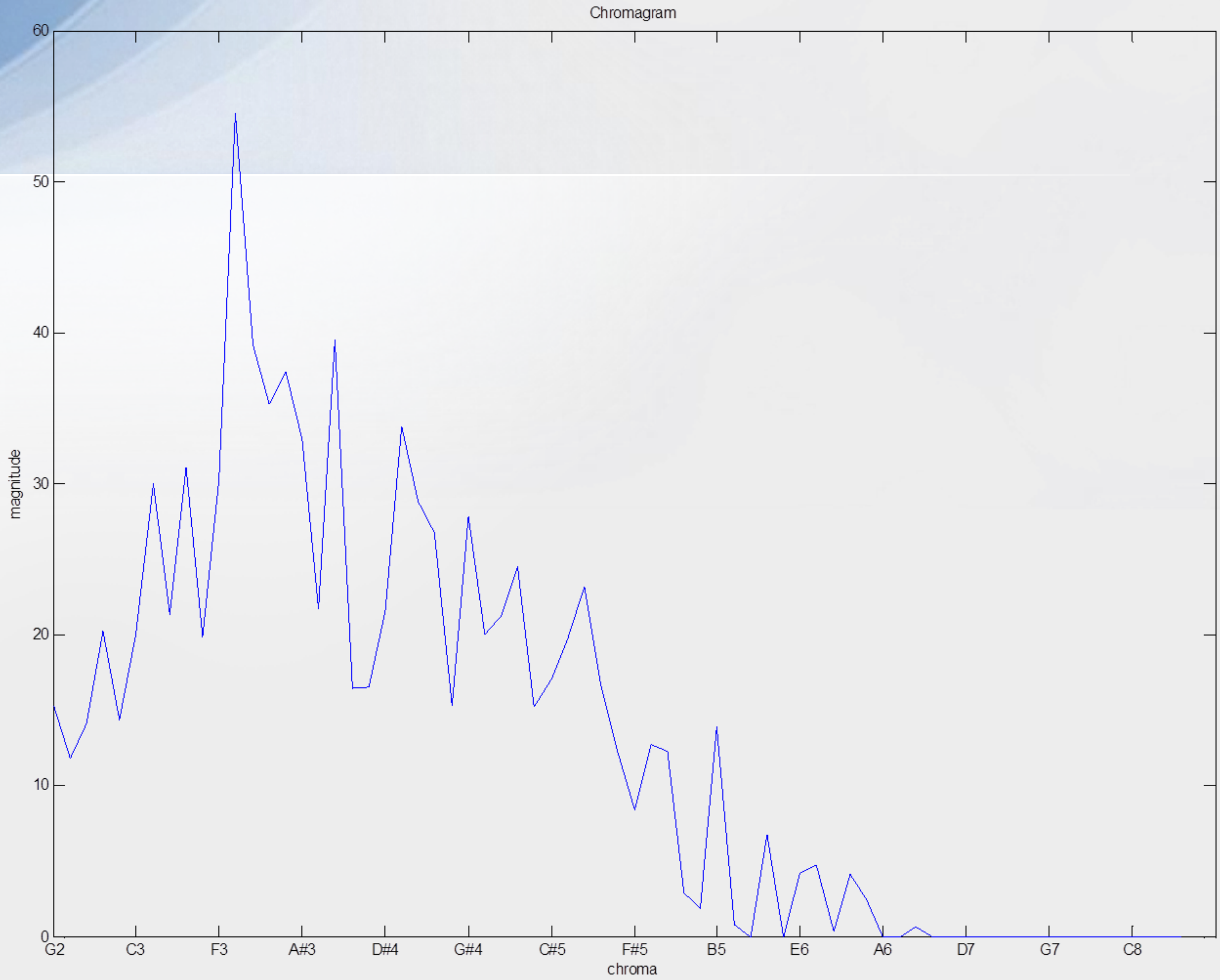


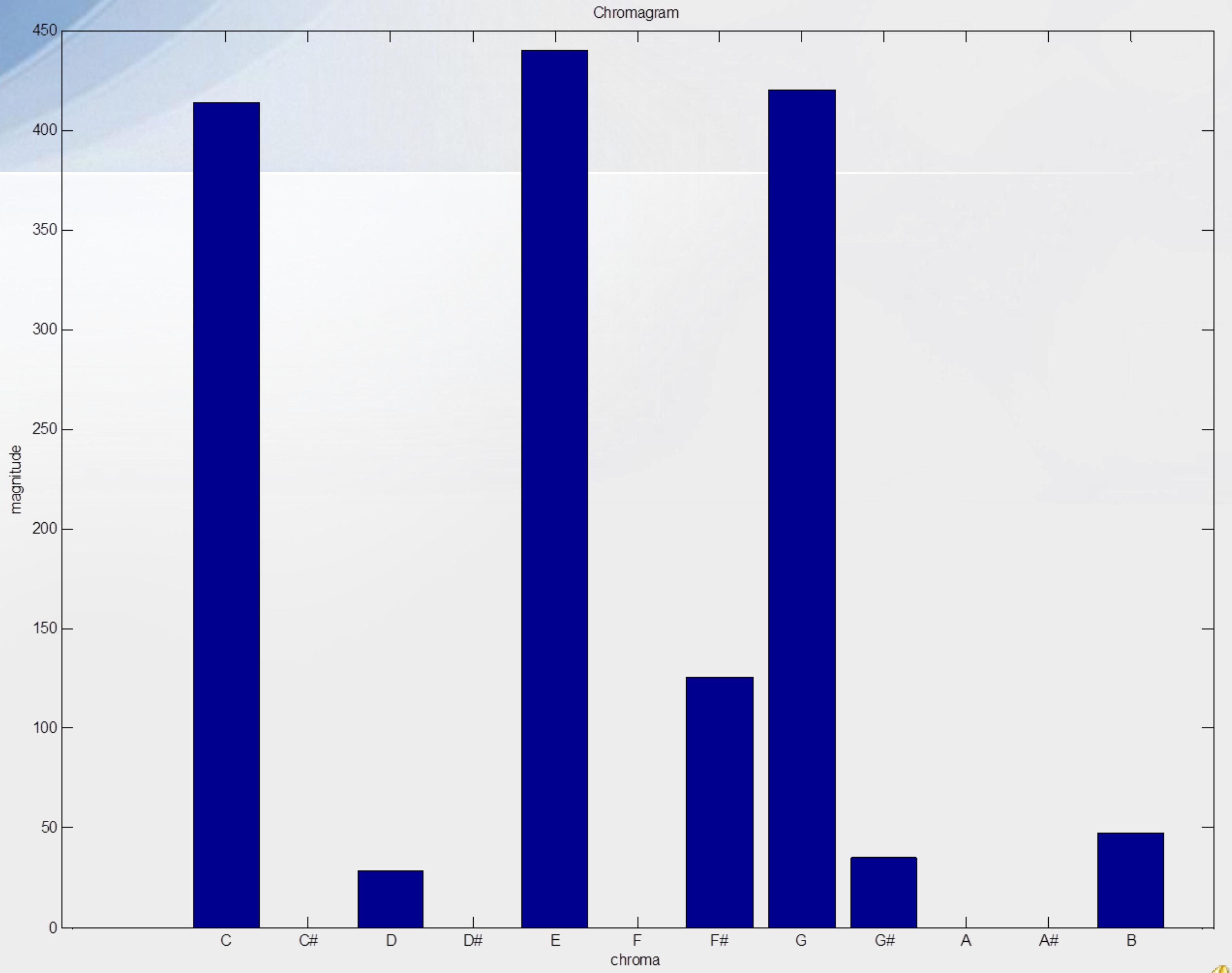


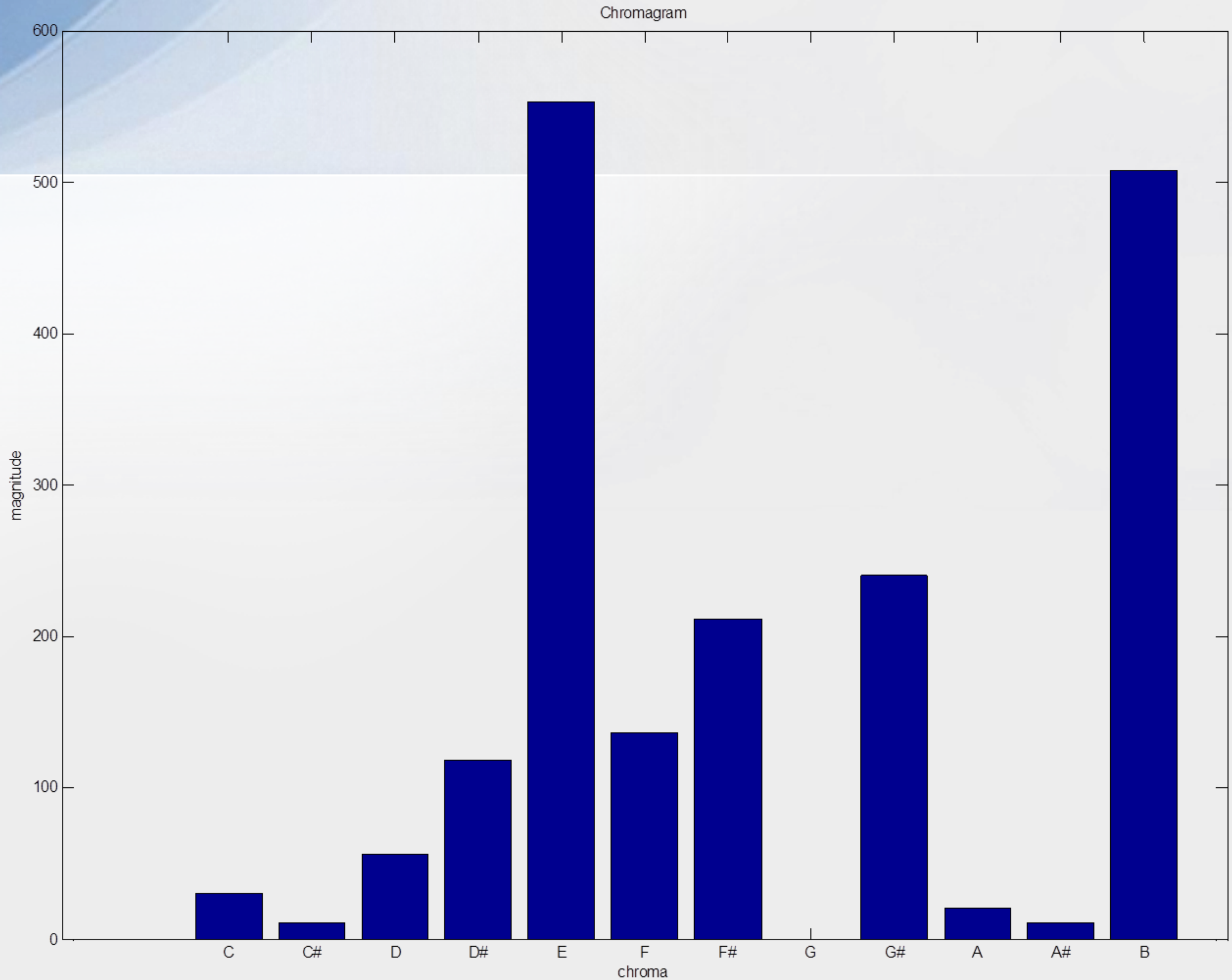
Spectrum
30
20
10
0
-10
-20
-30
-40
magnitude
10^1
10^2
10^3
10^4
10^5
frequency (Hz)

Chroma Bins

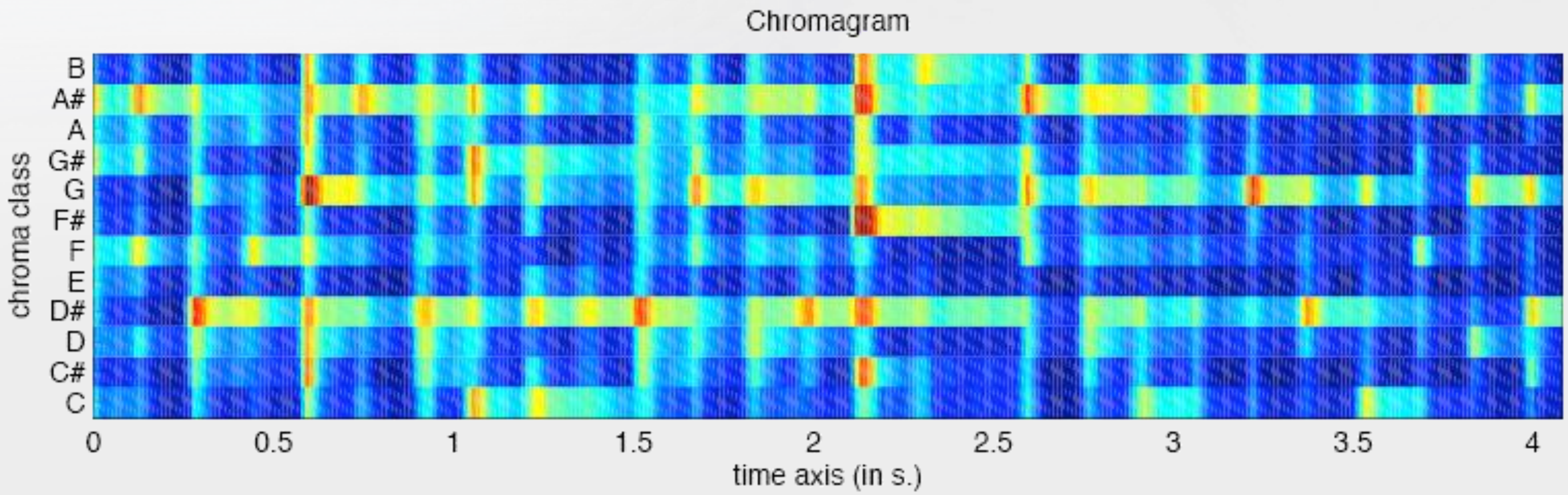

Spectrum
magnitude
frequency (Hz)
0
2
4
6
8
10
12
14
16
18
20
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000


Chromagram
magnitude
chroma
0
10
20
30
40
50
60
G2
C3
F3
A#3
D#4
G#4
C#5
F#5
B5
E6
A6
D7
G7
C8

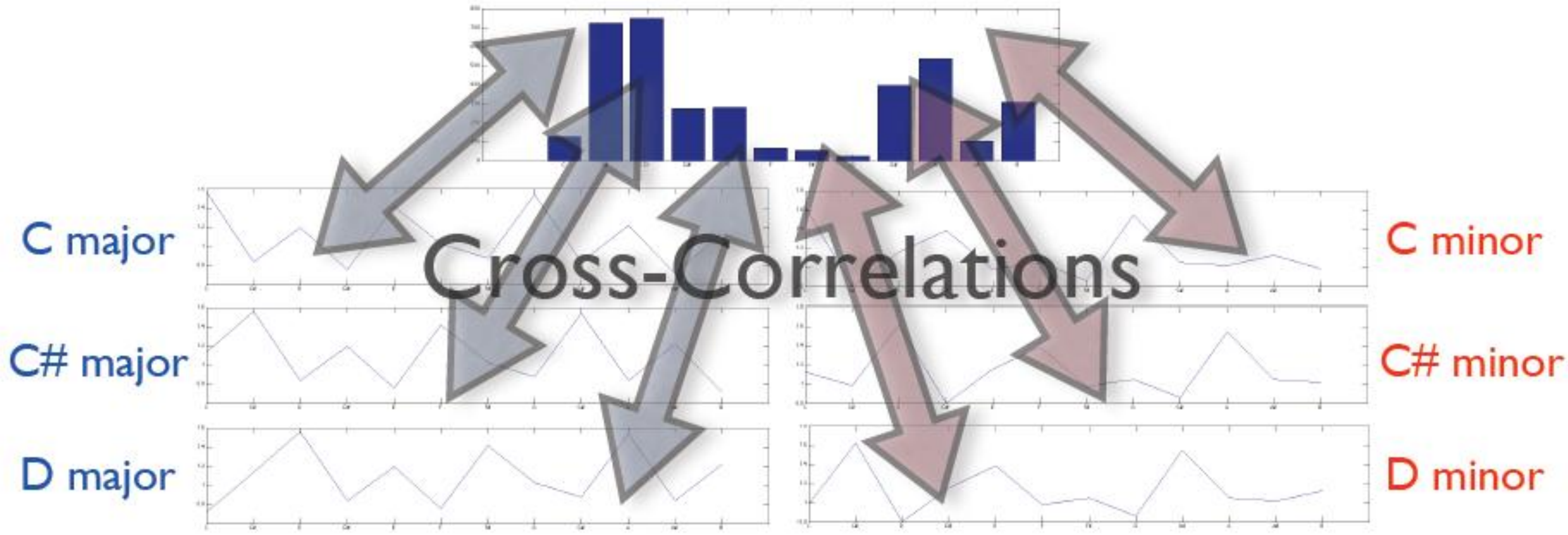

Chromagram
magnitude
chroma
0
50
100
150
200
250
300
350
400
450
C
C#
D
D#
E
F
F#
G
G#
A
A#
B


Chromagram
600
500
400
300
200
100
0
magnitude
C
C#
D
D#
E
F
F#
G
G#
A
A#
B
chroma

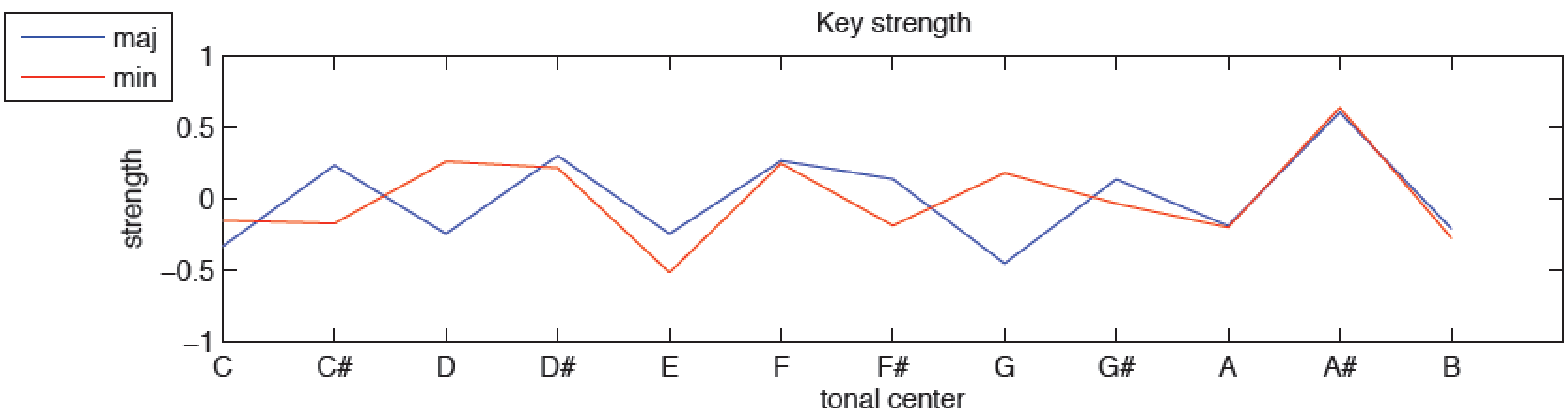
EXAMPLE



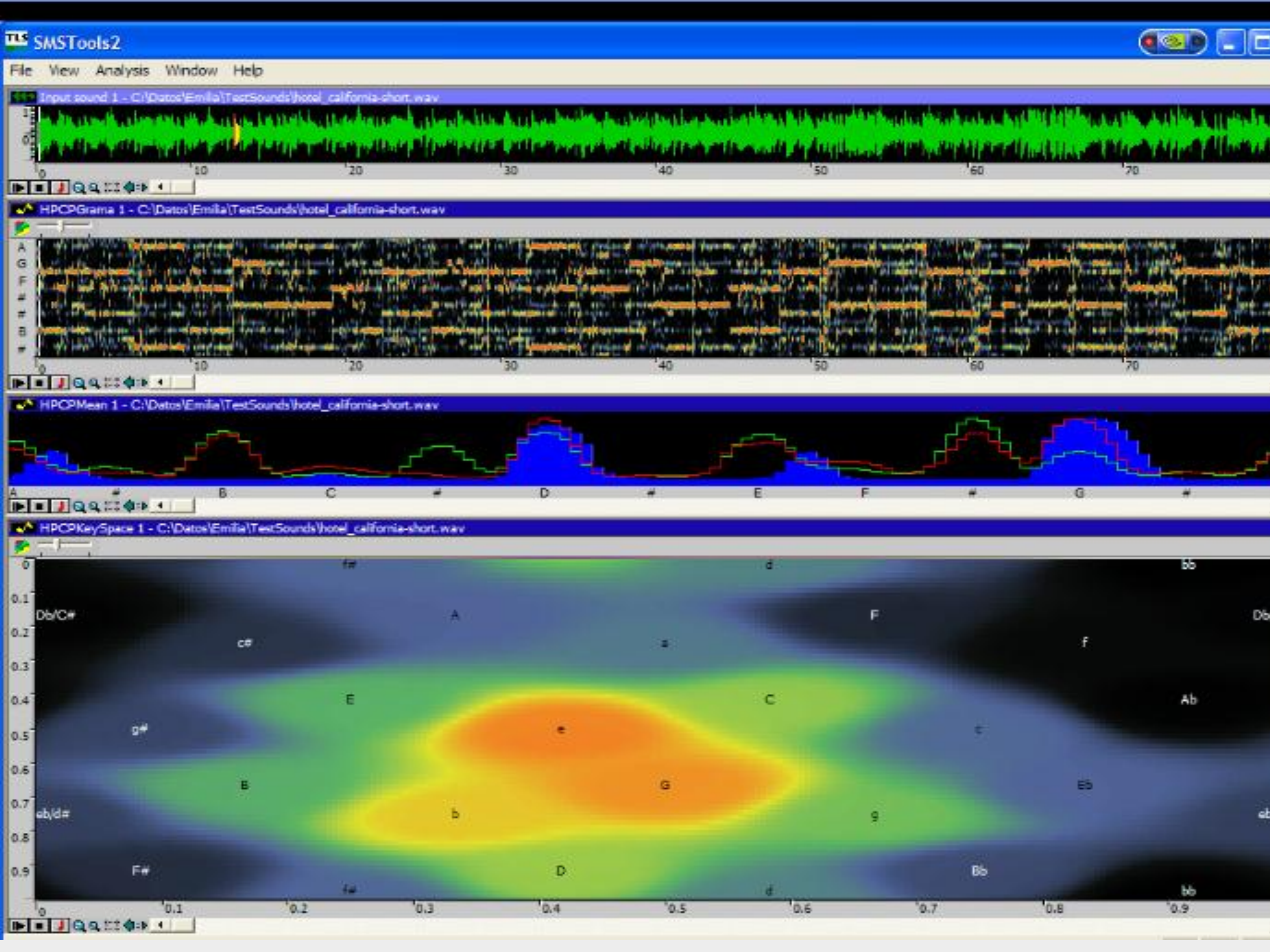
Picture courtesy: Olivier Lartillot

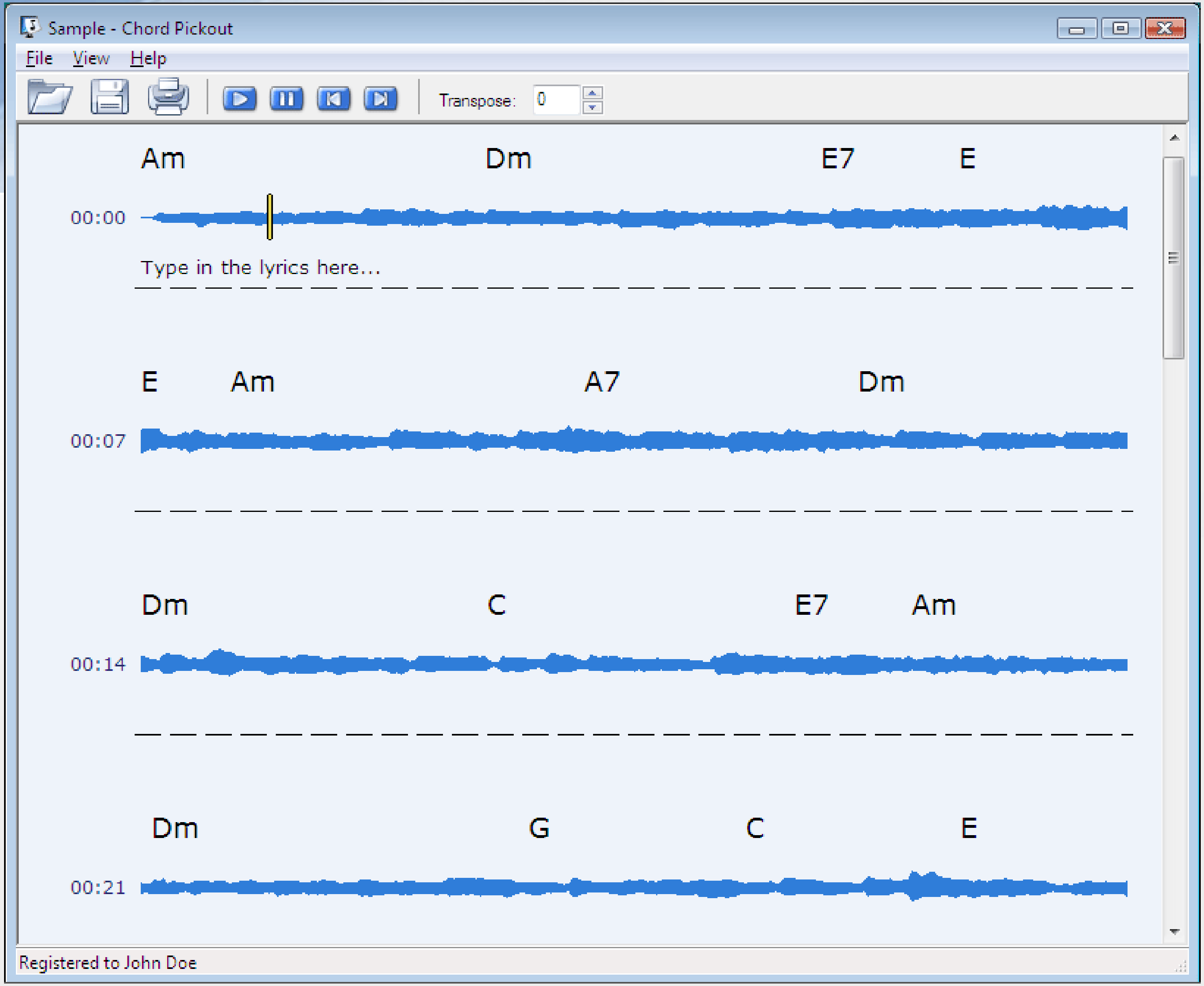


The resulting graph indicate the cross-correlation score for each different tonality candidate.



Picture courtesy: Olivier Lartillot

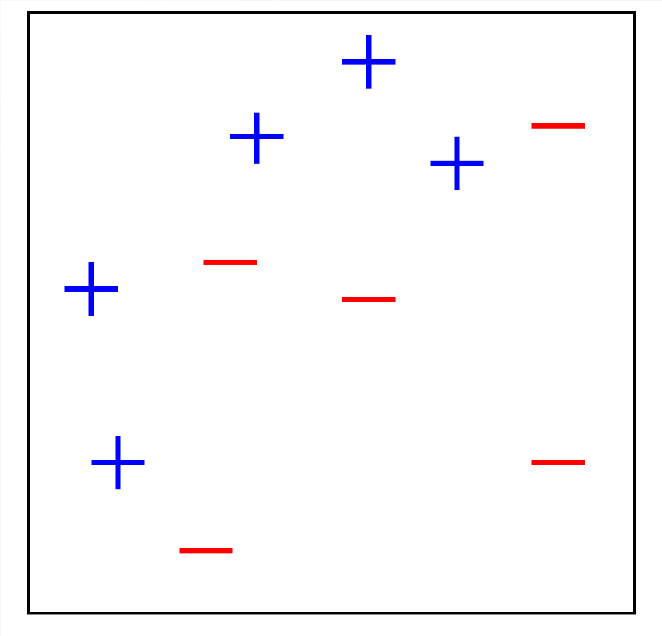

SMSTools2
File View Analysis Window Help
Input sound 1 - C:\Datos\Emilia\TestSounds\hotel_california-short.wav
HPCPGrama 1 - C:\Datos\Emilia\TestSounds\hotel_california-short.wav
HPCPMean 1 - C:\Datos\Emilia\TestSounds\hotel_california-short.wav
HPCPKeySpace 1 - C:\Datos\Emilia\TestSounds\hotel_california-short.wav



- http://www.chordpickout.com/index.html

Decision stumps

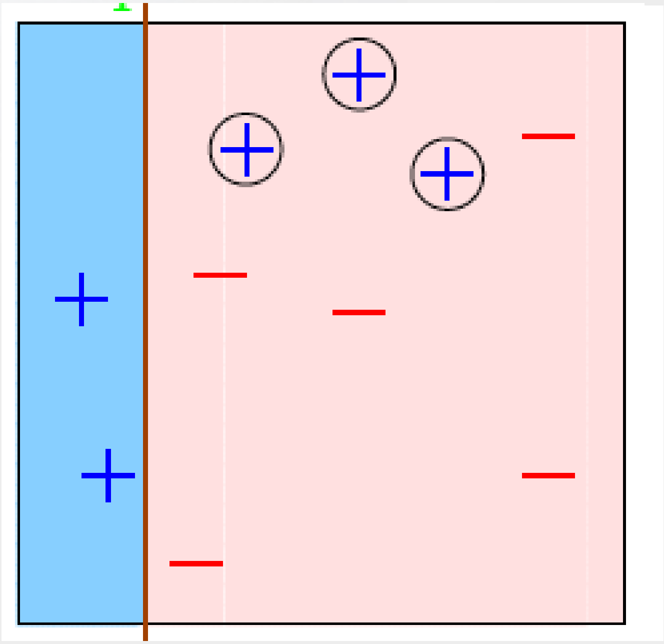
- An example dataset:



This section contains slides adapted from Rob Schapire @ Princeton.

A decision threshold

- Single threshold: e.g., “output ‘+’ iff x < .2”



- Decision stump: 1 threshold decision

Many thresholds: Decision trees

- Consists of many decisions in succession (like a flowchart)
- General approach:
  - Recursively split training data into subsets based on simple thresholds
  - Optionally prune to avoid overfitting
- Common algorithms: CART, ID3 => C4.5 (J48)

Decision Trees

- Advantages:
  - Easy to interpret
  - Decision boundary is explicit and straightforward
- Disadvantages:
  - Can take a long time to learn
    - Finding optimal tree can be NP-complete
  - Prone to overfitting
  - Inherently heuristic
  - Slight perturbations of data can lead to very different trees

Boosting

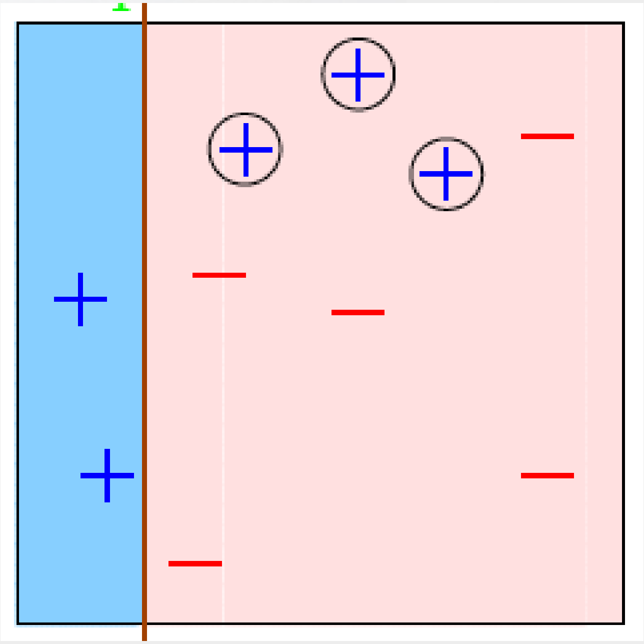
- A “meta-algorithm” for creating a “strong” learner from many “weak” learners
- Iteratively train weak learners on variations of the dataset and combine in a principled way to produce classification outputs.

AdaBoost

- A popular boosting algorithm from Freund and Schapire
- Robust to overfitting: emphasis on **maximizing the margin**

Back to stumps

- Single threshold: e.g., “output ‘+’ iff x < .2”



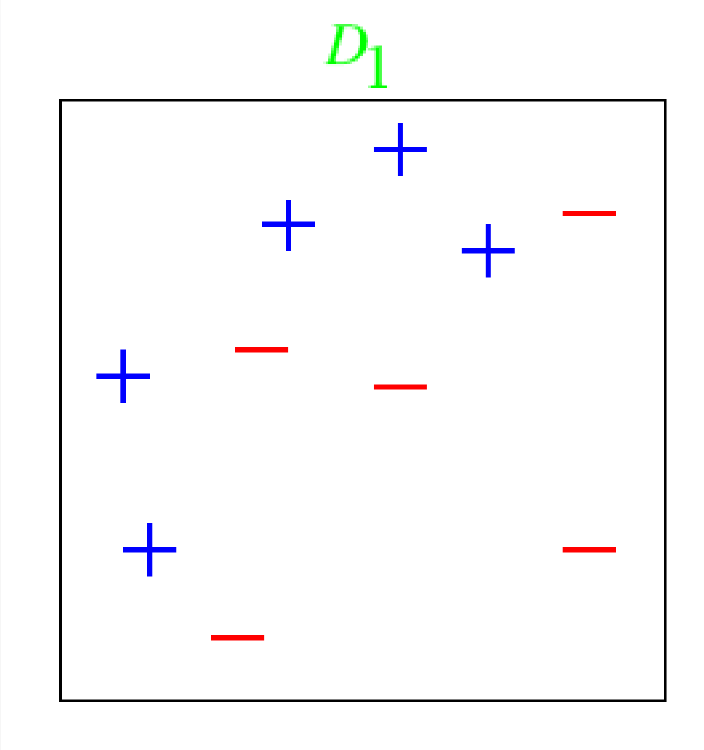
- Makes a nice weak learner!

The AdaBoost algorithm

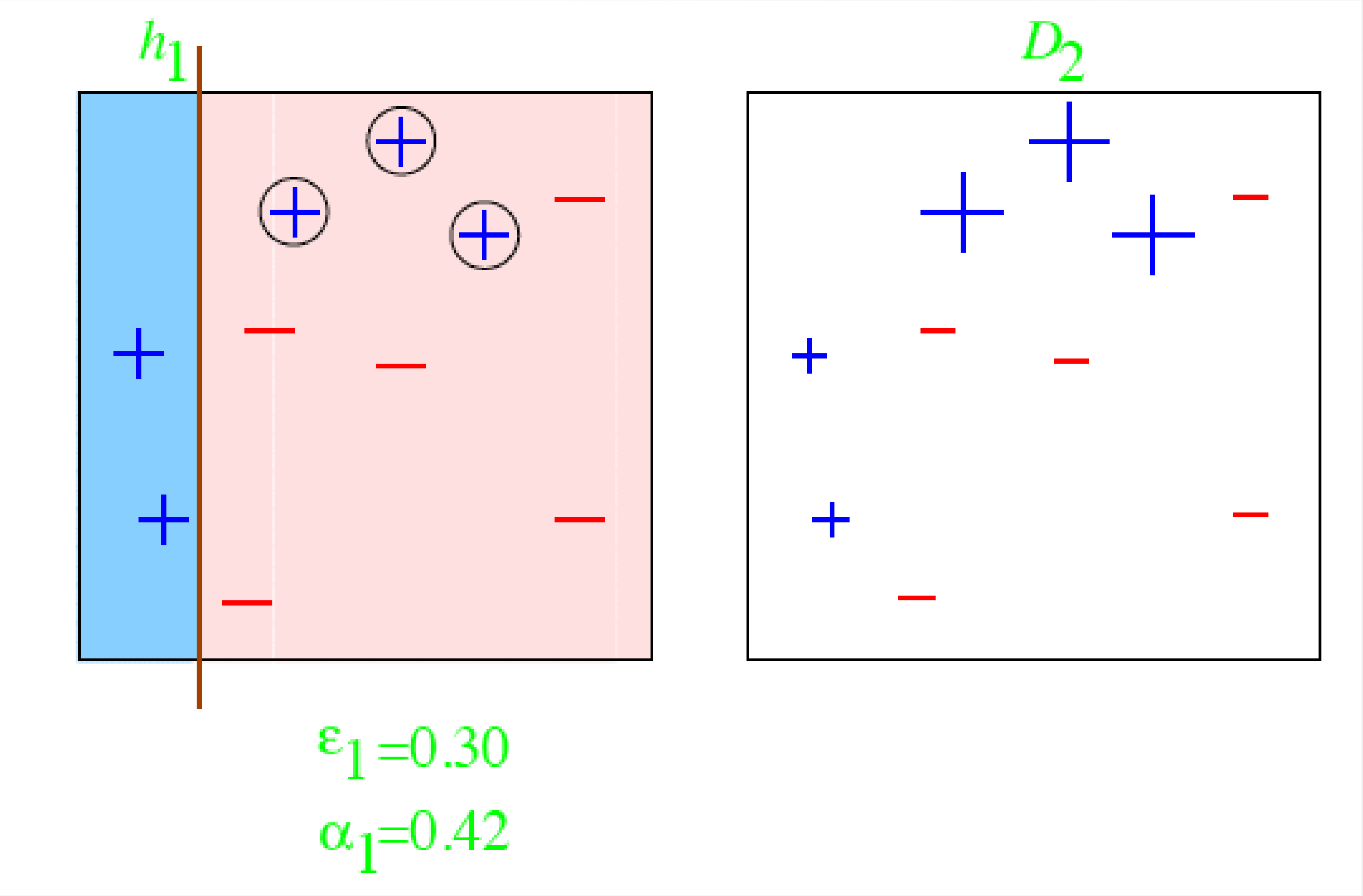
- Initialize D_1 to be the dataset with each example equally weighted.
- for round t in 1 to T:
  - Train a weak learner, h_t, on the dataset D_t
  - If h_t can't achieve 50% accuracy, stop.
  - Choose $alpha_t$ according to error rate of ht on D_t (better ht => higher $alpha_t$)
  - Update data weights D_{t+1} to **increase** weight of examples ht got wrong, and **decrease** weight of examples h_t got right.
- To classify new data, take a weighted majority vote of all weak learners, each h_t weighted by its $alpha_t$.

AdaBoost illustrated

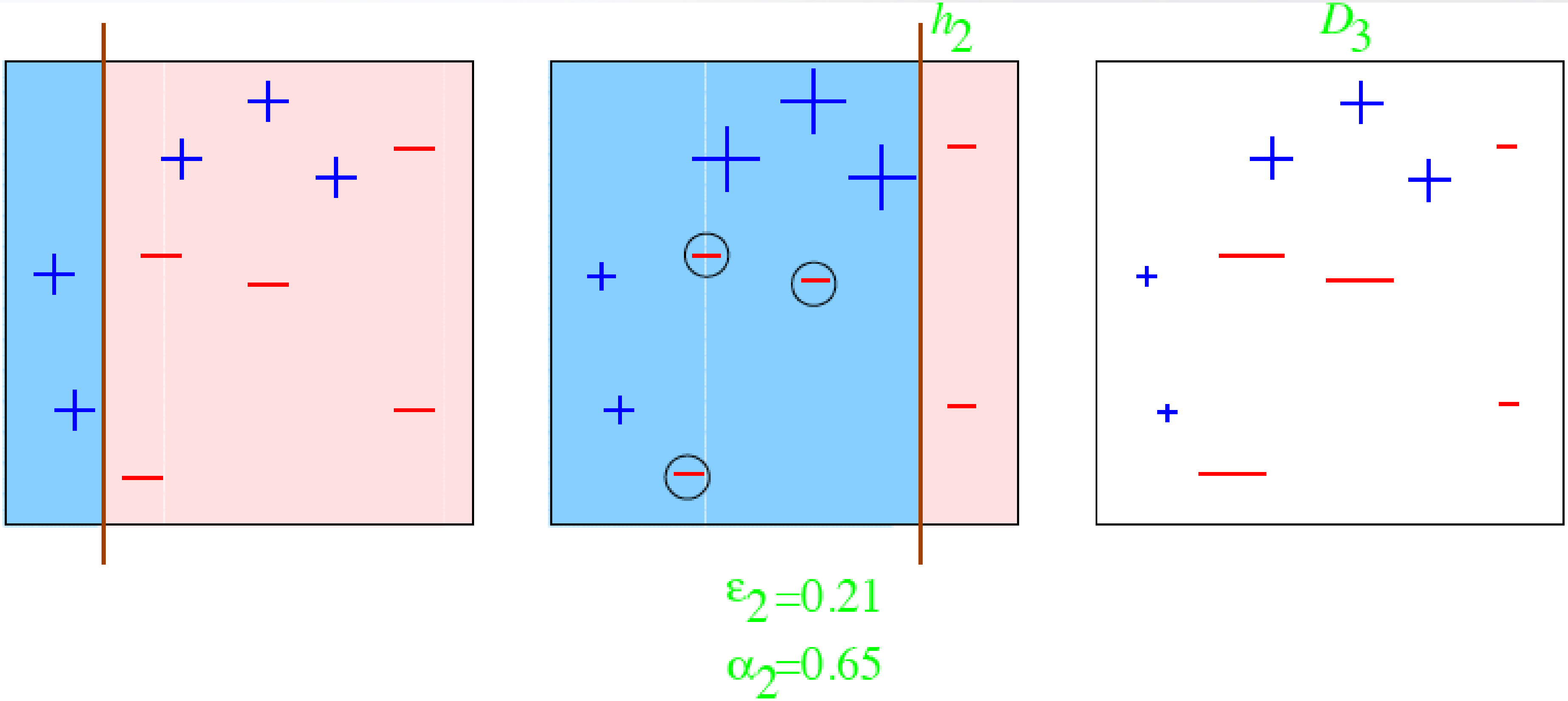
- Initial data:



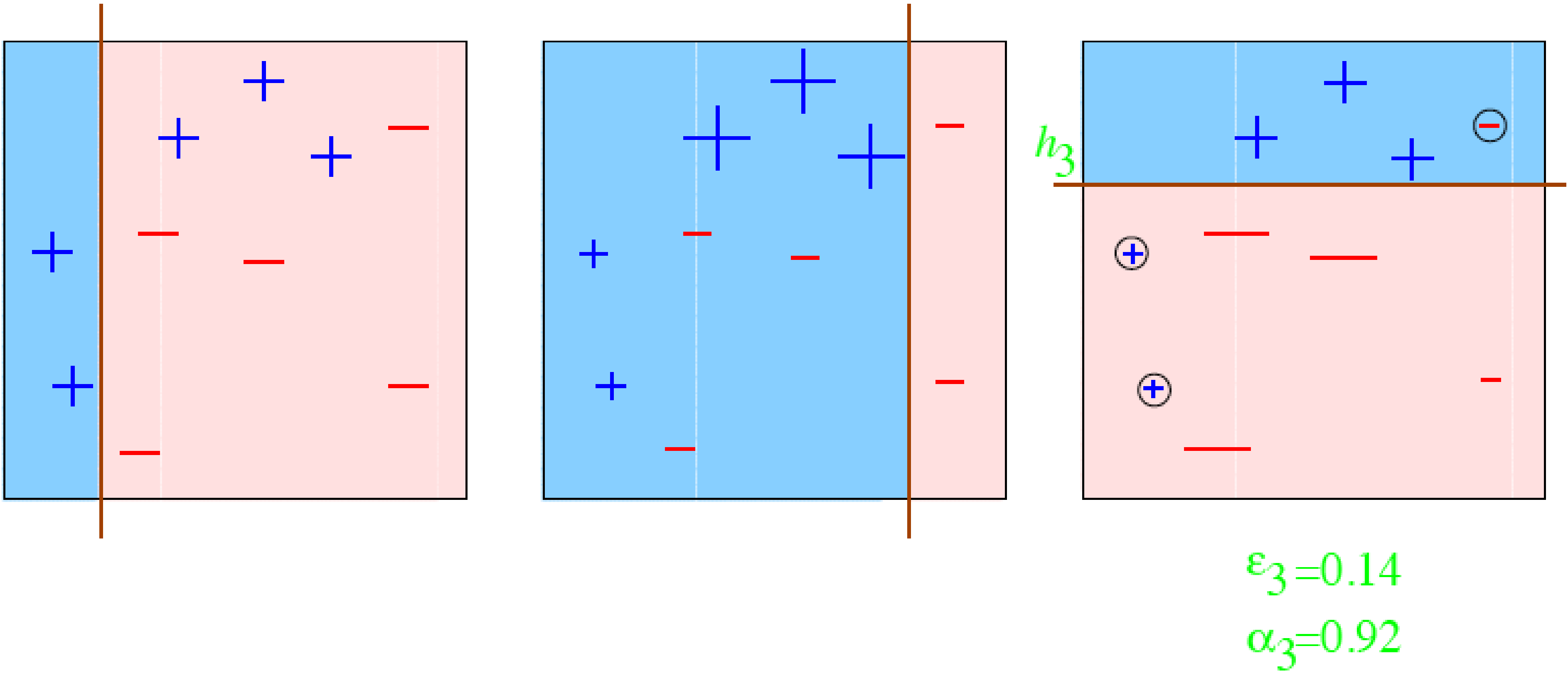
Round 1



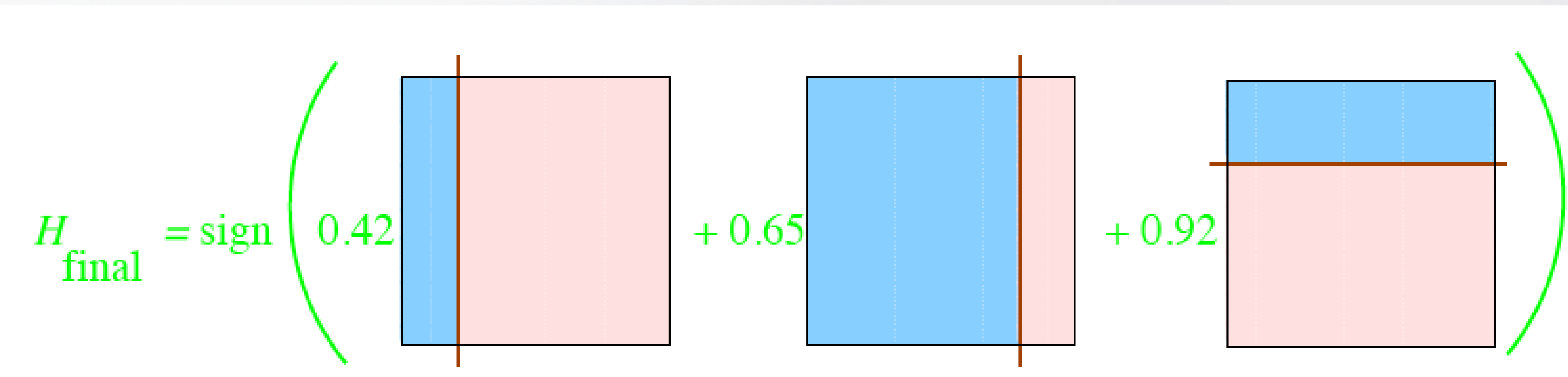
Round 2



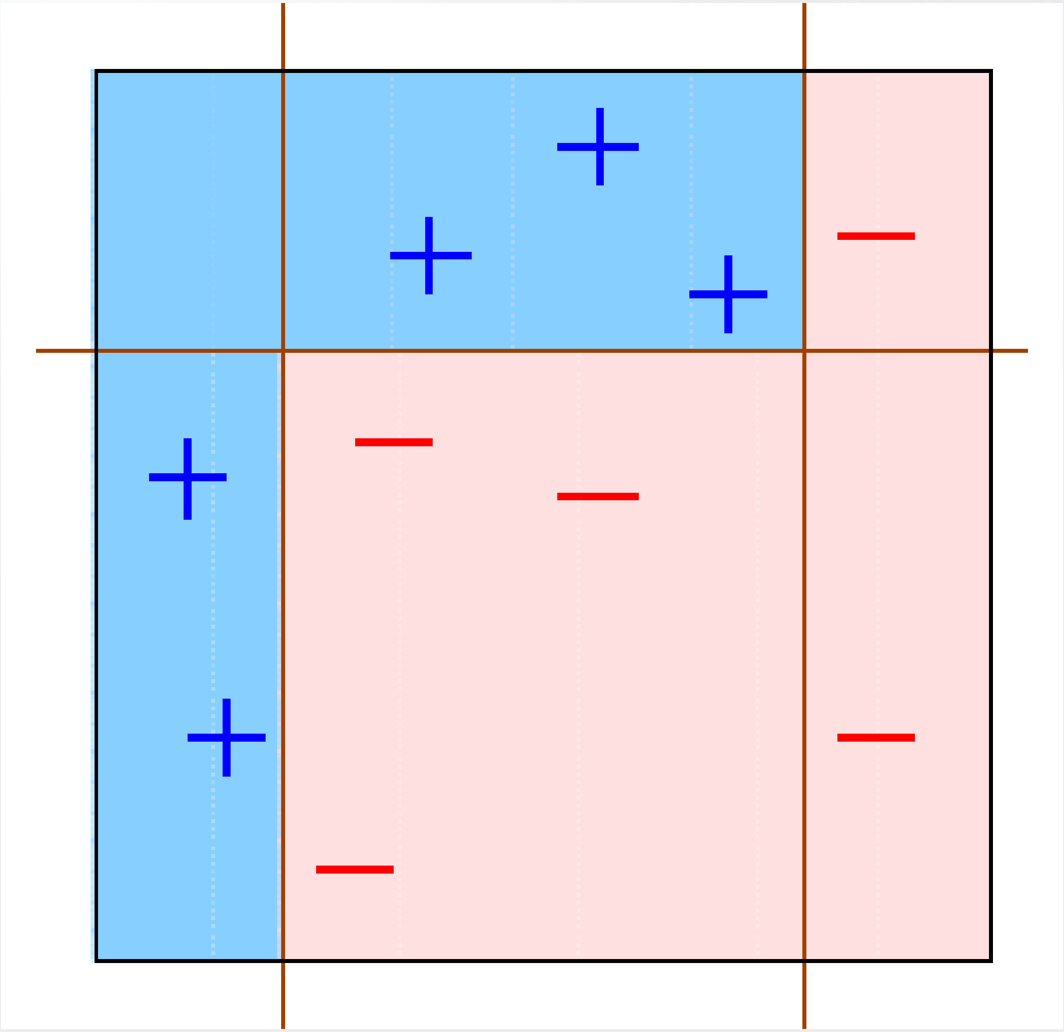
Round 3



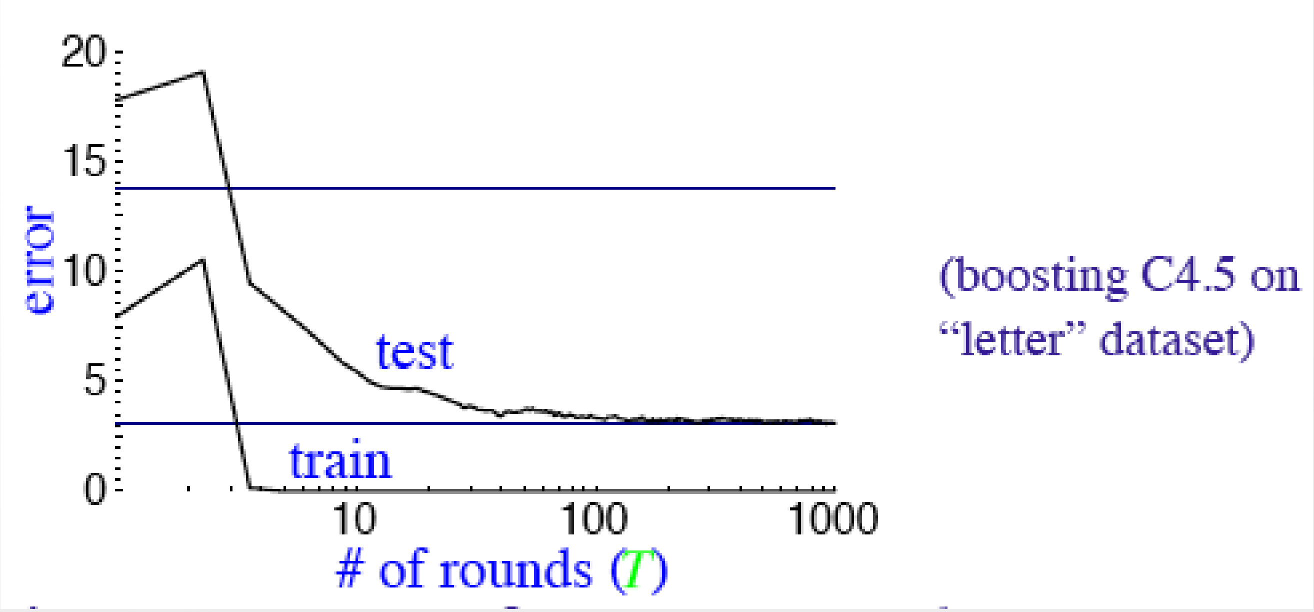
Final classifier



Final classifier: decision boundary



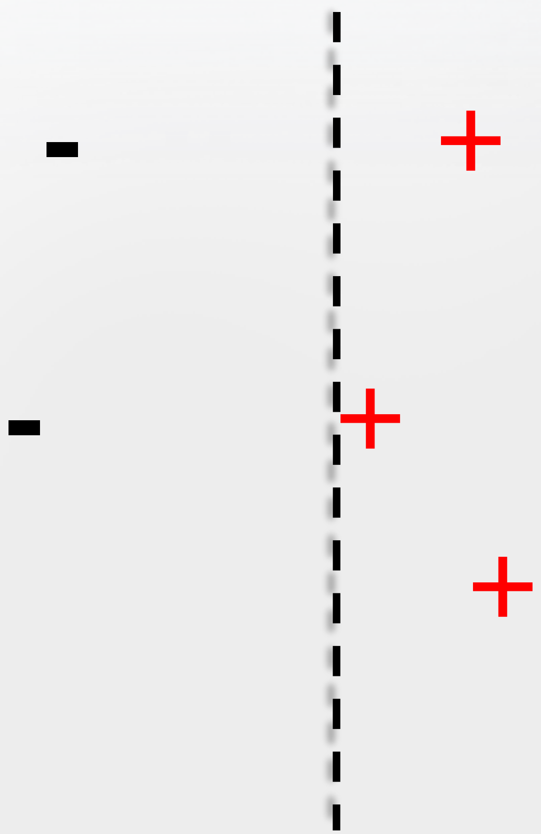
A typical AdaBoost run



- Test error does not increase, even after 1000 rounds
- Test error continues to drop, even after training error = 0.

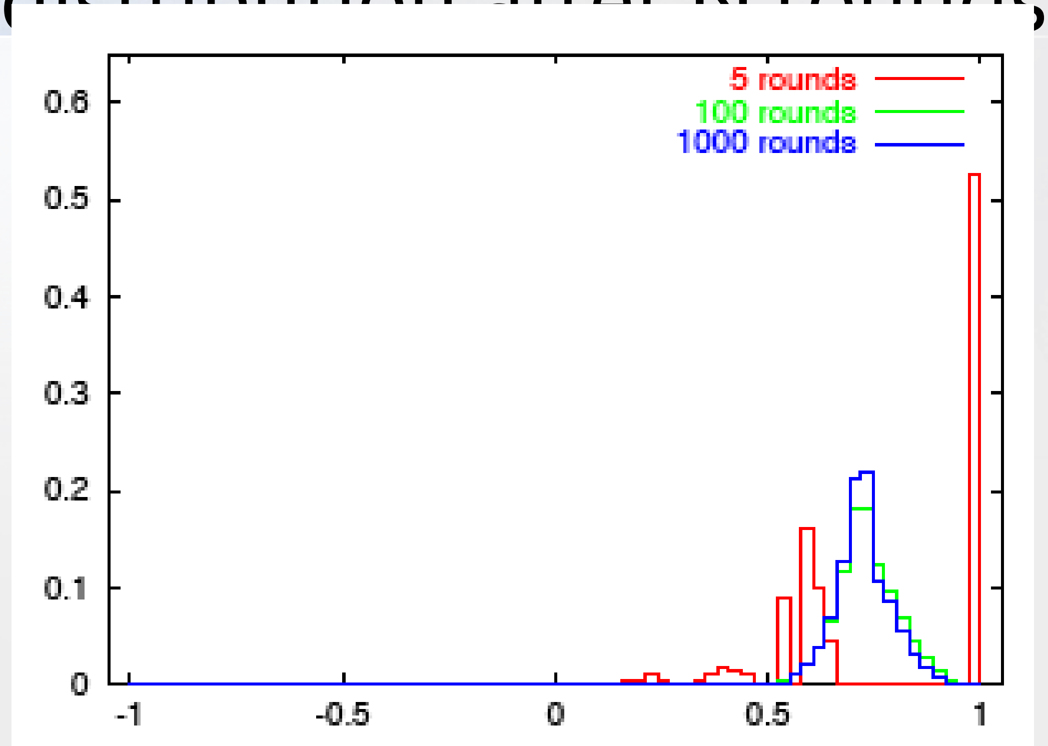
The margin

- Narrow margin



- Wide margin

Margin distribution after N rounds



	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1
% margins ≤ 0.5	7.7	0.0	0.0
minimum margin	0.14	0.52	0.55

AdaBoost pro & con

- Advantages:
  - Robust to overfitting
  - Conceptually simple
  - Statistically very nice: maximizing the margin, game-theoretic understanding
  - Can work with any base learner
  - No parameters to tune
- Disadvantages:
  - Weak learner must achieve >50% or failure
  - Original formulation binary only
    - AdaBoost.M1 handles multi-class, but more required of weak learner

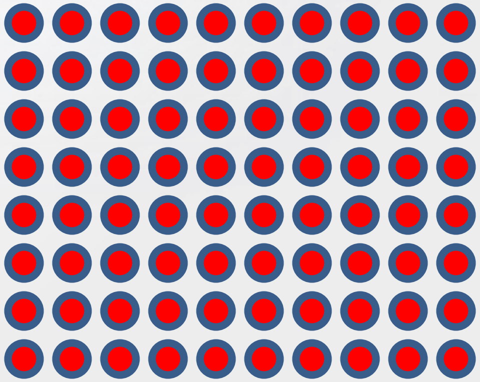
EVALUATION

Our classifier accuracy is 83.4%

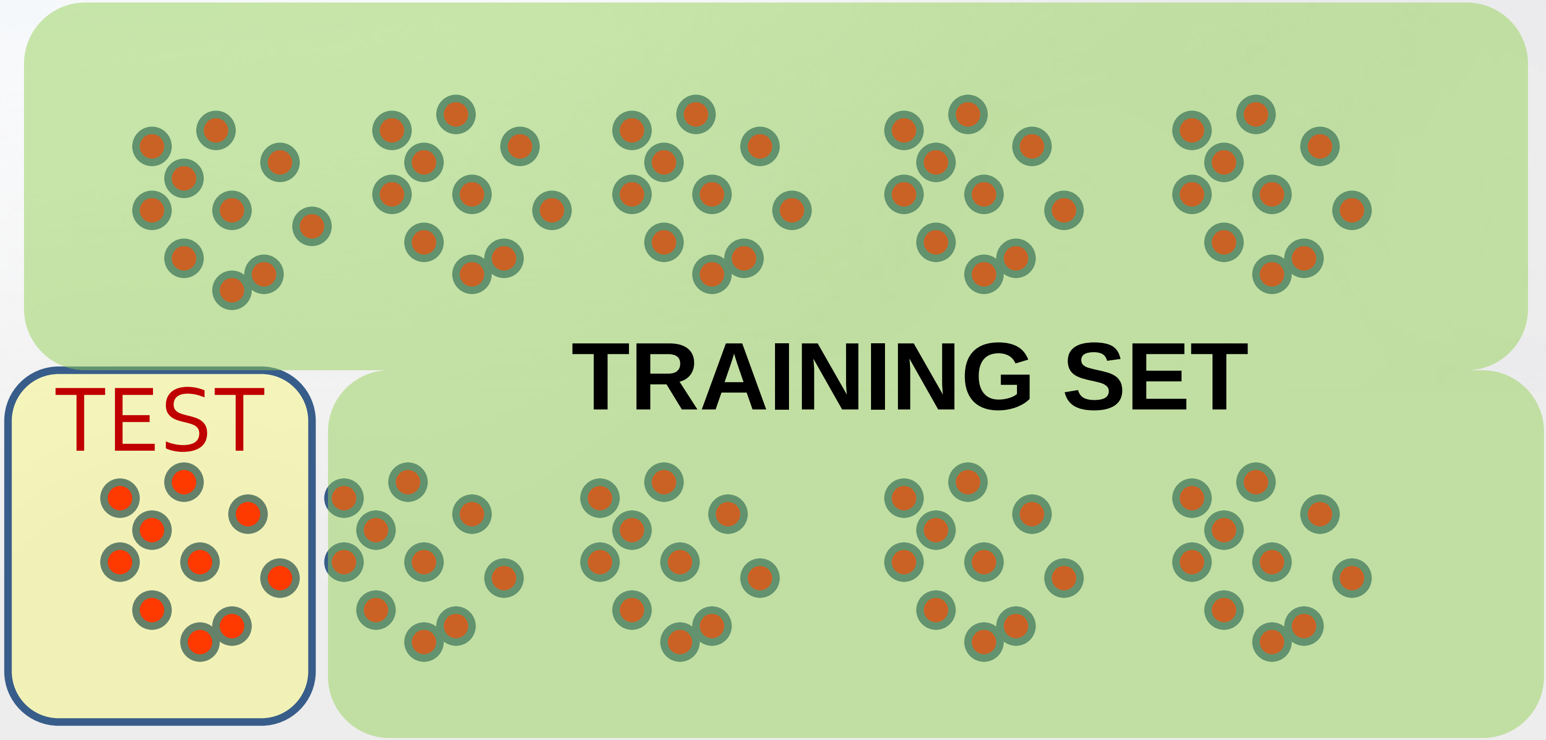
Cross-validation

- Say, 10-fold cross validation
- Divide test set into 10 random subsets.
- 1 test set is tested using the classifier trained on the remaining 9.
- We then do test/train on all of the other sets and average the percentages. Helps prevent over fitting.
- Do not optimize too much on cross validation – you can severely overfit. Sanity check with a test set.

Cross-validation



Cross-validation



Fold 1: 70%

Cross-validation

Fold 1: 70%
Fold 2: 80%

Cross-validation

Fold 1: 76%
Fold 2: 80%
Fold 3: 77%
Fold 4: 83%
Fold 5: 72%
Fold 6: 82%
Fold 7: 81%
Fold 8: 71%
Fold 9: 90%
Fold 10: 82%
Mean = 79.4%

Stratified Cross-Validation

- Same as cross-validation, except that the folds are chosen so that they contain equal proportions of labels.

> End Day 2