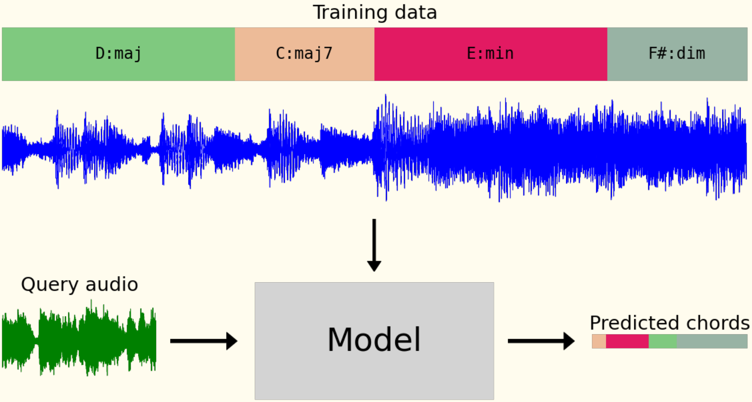
Large-Scale Content-Based Matching of Audio and MIDI Data

Colin Raffel and Dan Ellis
with help from Kitty Shi and Hilary Mogul

CCRMA DSP Seminar, January 13, 2015

Music Information Retrieval Pipeline



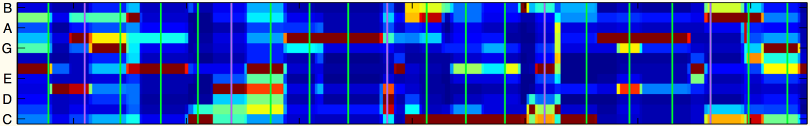
The Million Song Dataset

```
         artist: 'Tori Amos'
        release: 'LIVE AT MONTREUX'
          title: 'Smells Like Teen Spirit'
             id: 'TRKUYPW128F92E1FC0'
            key: 5
           mode: 0
       loudness: -16.6780
          tempo: 87.2330
 time_signature: 4
       duration: 216.4502
    sample_rate: 22050
      audio_md5: '8'
     7digitalid: 5764727
    familiarity: 0.8500
           year: 1992
```

100.0 — cover	5.0 — cover songs
57.0 — covers	4.0 — soft rock
43.0 — female vocalists	4.0 — nirvana cover
42.0 — piano	4.0 — Mellow
34.0 — alternative	4.0 — alternative rock
14.0 — singer-songwriter	3.0 — chick rock
11.0 — acoustic	3.0 — Ballad
8.0 — tori amos	3.0 — Awesome Covers
7.0 — beautiful	2.0 — melancholic
6.0 — rock	2.0 — k00l ch1x
6.0 — pop	2.0 — indie
6.0 — Nirvana	2.0 — female vocalistist
6.0 — female vocalist	2.0 — female
6.0 — 90s	2.0 — cover song
5.0 — out of genre covers	2.0 — american

```
%5489,4468, Smells Like Teen Spirit
TRTUOVJ128E078EE10 Nirvana
TRFZJOZ128F4263BE3 Weird Al Yankovic
TRJHCKN12903CDD274 Pleasure Beach
TRELTOJ128F42748B7 The Flying Pickets
TRJKBXL128F92F994D Rhythms Del Mundo feat. Shanade
TRIHLAW128F429BBF8 The Bad Plus
TRKUYPW128F92E1FC0 Tori Amos
```

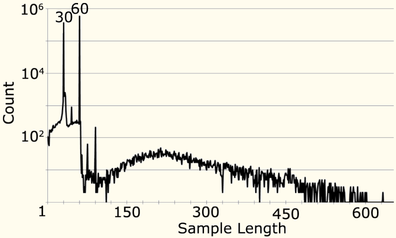
12	hello	6	here	3	is
11	i	6	us	3	with
10	a	6	entertain	3	oh
9	and	4	the	3	out
7	it	4	feel	3	an
6	are	4	yeah	3	light
6	we	3	to	3	less
6	now	3	my	3	danger



Thierry Bertin-Mahieux et al. *"The million song dataset"*

Audio? One solution:

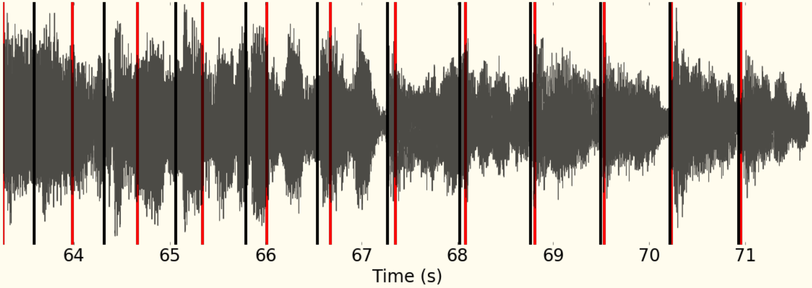
↓



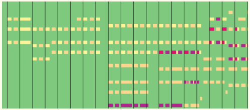
Samplerate		
22	768,710	77,26%
44	226,169	22,73%
other	81	0,01%
Bitrate		
128	646,120	64,94%
64	343,344	34,51%
other (VBR)	5,494	0,55%
Channels		
Mono	6,342	0.64%
Stereo	150,779	15.15%
Joint stereo / dual channel	837,839	84.21%

Schindler et al. *"Facilitating Comprehensive Benchmarking Experiments on the Million Song Dataset"*

Ground Truth?



Ground Truth from MIDI



←

→

↓

110 bpm

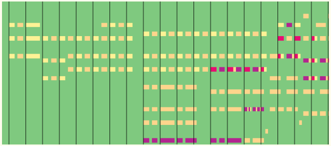
Extracting with `pretty_midi`

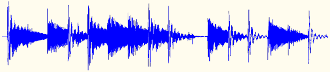
```python
import pretty_midi
# Load MIDI file into PrettyMIDI object
midi_data = pretty_midi.PrettyMIDI('midi_file.mid')
# Get a beat-synchronous piano roll
piano_roll = midi_data.get_piano_roll(times=midi_data.get_beats())
# Compute the relative amount of each semitone across the entire song, a proxy for key
print [sum(semitone)/sum(sum(midi_data.get_chroma())) for semitone in midi_data.get_chroma()]
# Shift all notes up by 5 semitones
for instrument in midi_data.instruments:
    # Don't want to shift drum notes
    if not instrument.is_drum:
        for note in instrument.notes:
            note.pitch += 5
# Synthesize the resulting MIDI data using sine waves
audio_data = midi_data.synthesize()
```

http://github.com/craffel/pretty-midi

MIDI + Audio + MSD

```
     artist: 'Tori Amos'
    release: 'LIVE AT MONTREUX'
      title: 'Smells Like Teen Spirit'
         id: 'TRKUYPW128F92E1FC0'
   duration: 216.4502
sample_rate: 22050
  audio_md5: '8'
 7digitalid: 5764727
       year: 1992
```

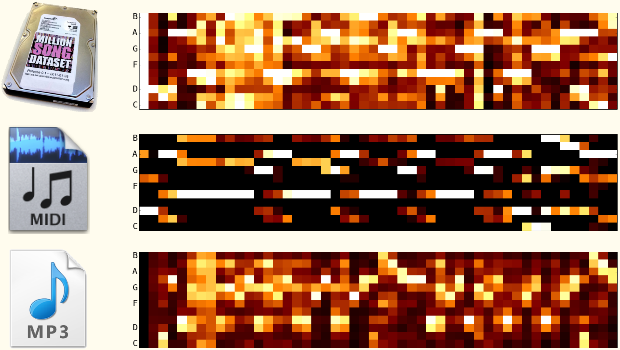




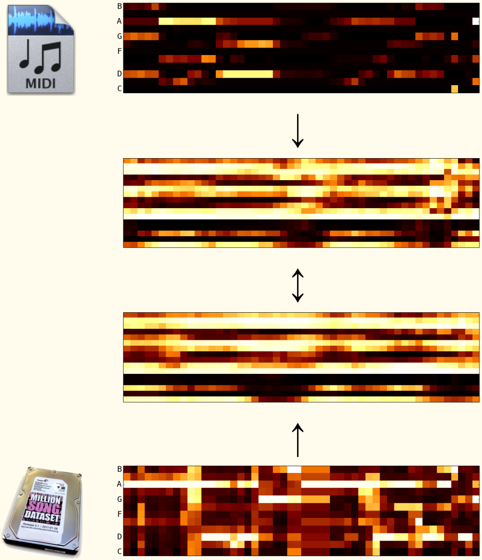
Matching by Text

```
J/Jerseygi.mid
V/VARIA18O.MID
Carpenters/WeveOnly.mid
2009 MIDI/handy_man1-D105.mid
G/Garotos Modernos - Bailanta De Fronteira.mid
Various Artists/REWINDNAS.MID
GoldenEarring/Twilight_Zone.mid
Sure.Polyphone.Midi/Poly 2268.mid
d/danza3.mid
100%sure.polyphone.midi/Fresh.mid
rogers_kenny/medley.mid
2009 MIDI/looking_out_my_backdoor3-Bb192.mid
```

Matching by Content



Idea: Map to a Common Space



The Plan

1. Obtain a large collection of MIDI files

The Plan

1. Obtain a large collection of MIDI files
2. Manually find a subset with good metadata

The Plan

1. Obtain a large collection of MIDI files
2. Manually find a subset with good metadata
3. Match them against known MP3 collections

The Plan

1. Obtain a large collection of MIDI files
2. Manually find a subset with good metadata
3. Match them against known MP3 collections
4. Perform MIDI to audio alignment

The Plan

1. Obtain a large collection of MIDI files
2. Manually find a subset with good metadata
3. Match them against known MP3 collections
4. Perform MIDI to audio alignment
5. Learn a mapping between feature spaces

The Plan

1. Obtain a large collection of MIDI files
2. Manually find a subset with good metadata
3. Match them against known MP3 collections
4. Perform MIDI to audio alignment
5. Learn a mapping between feature spaces
6. Use the mapping to **efficiently** match MIDI files without metadata to MSD entries

Unique MIDIs

500,000 $\longrightarrow$ 250,000

Finding Good Metadata

```
J/Jerseygi.mid
V/VARIA18O.MID
Carpenters/WeveOnly.mid
2009 MIDI/handy_man1-D105.mid
G/Garotos Modernos - Bailanta De Fronteira.mid
Various Artists/REWINDNAS.MID
GoldenEarring/Twilight_Zone.mid
Sure.Polyphone.Midi/Poly 2268.mid
```

↓

```
Mc Broom, Amanda/The Rose.mid
Men At Work/Down Under.mid
Beach Boys, The/Barbara Ann.mid
Star Wars/Cantina.mid
T L C/CREEP.MID
Beatles/help.mid
Idol, Billy/White Wedding.mid
```

Cleaning Metadata

`Mc Broom, Amanda/The Rose.mid`	`Amanda McBroom/The Rose.mid`
`Men At Work/Down Under.mid`	`Men At Work/Down Under.mid`
`Beach Boys, The/Barbara Ann.mid`	`The Beach Boys/Barbara Ann.mid`
`Star Wars/Cantina.mid`	
`T L C/CREEP.MID`	`TLC/Creep.mid`
`Beatles/help.mid`	`The Beatles/Help!.mid`
`Idol, Billy/White Wedding.mid`	`Billy Idol/White Wedding.mid`

25,000 → 17,000 (9,000)

Matching to Existing Collections

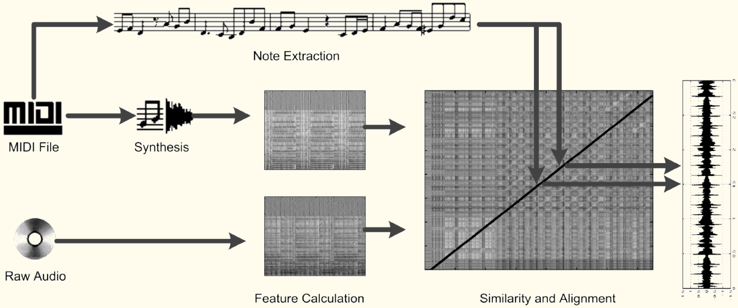
Amanda McBroom/The Rose.mid
Men At Work/Down Under.mid
The Beach Boys/Barbara Ann.mid
TLC/Creep.mid
The Beatles/Help!.mid
Billy Idol/White Wedding.mid

men_at_work/Brazil/07-Down_Under.mp3

tlc/Crazy_Sexy_Cool/02-Creep.mp3
The Beatles - Help!.mp3

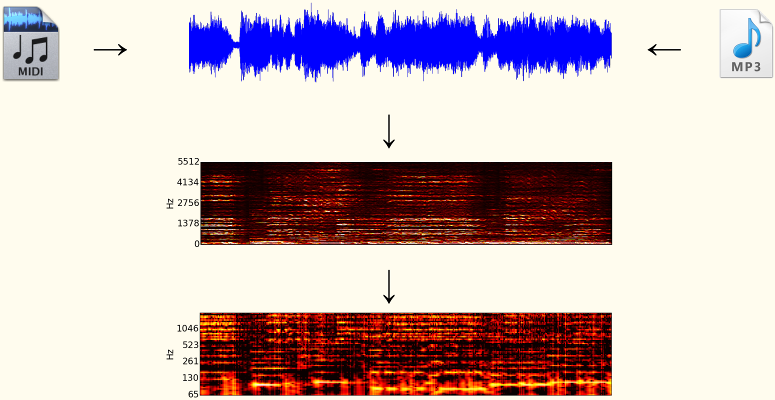
17,000 (9,000) → 5,000 (2,000)

Alignment



Turetsky and Ellis, *"Ground-Truth Transcriptions of Real Music from Force-Aligned MIDI Syntheses"*

Feature Extraction for Alignment

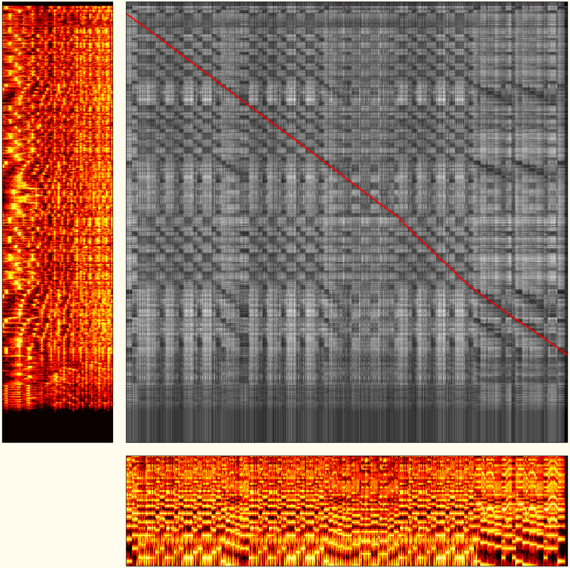


Feature Extraction with librosa

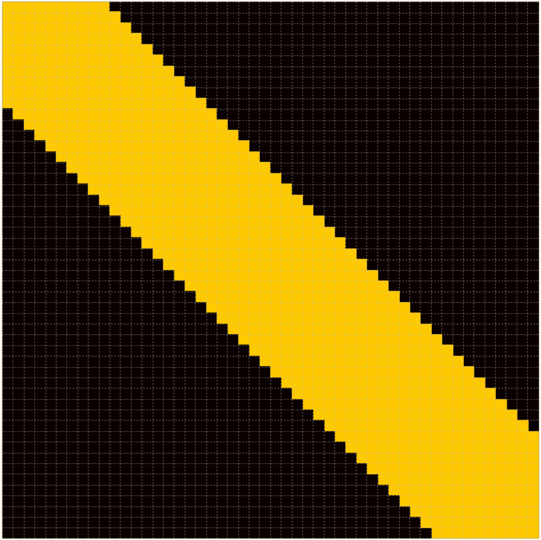
```
import librosa
# We could also obtain audio data from pretty_midi's fluidsynth method
audio, fs = librosa.load('audio_file.mp3')
# Separate harmonic and percussive components
audio_stft = librosa.stft(audio)
H, P = librosa.decompose.hpss(audio_stft)
audio_harmonic = librosa.istft(H)
# Compute log-frequency spectrogram of original audio
audio_gram = np.abs(librosa.cqt(y=audio_harmonic, sr=fs, hop_length=hop,
                                fmin=librosa.midi_to_hz(36), n_bins=60))
# Convert to decibels
log_gram = librosa.logamplitude(audio_gram, ref_power=audio_gram.max())
# Normalize the columns (each frame)
normed_gram = librosa.util.normalize(log_gram, axis=0)
```

http://www.github.com/bmcfee/librosa

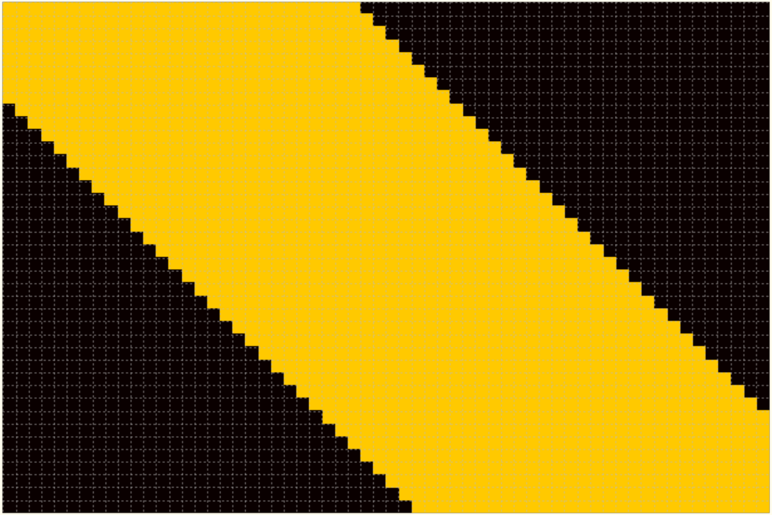
Dynamic Time Warping



Traditional DTW Constraint



Sequences of Different Length



Reporting a Confidence Score

1. Compute the total distance between aligned frames

Reporting a Confidence Score

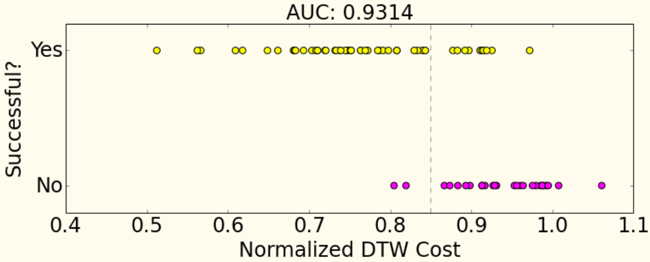
1. Compute the total distance between aligned frames
2. Normalize by the path length

Reporting a Confidence Score

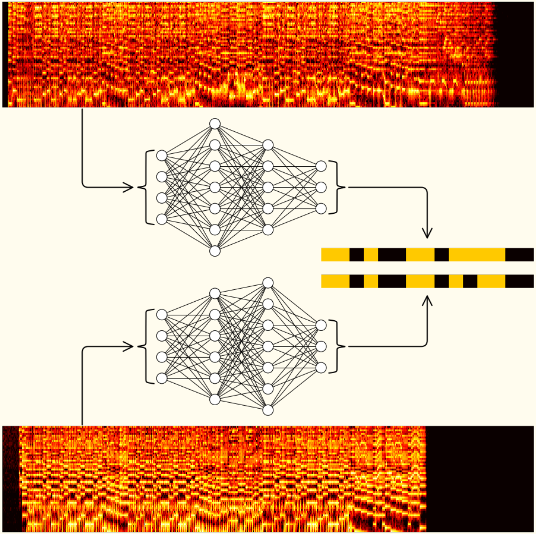
1. Compute the total distance between aligned frames
2. Normalize by the path length
3. Normalize by the mean distance between all frames

Reporting a Confidence Score

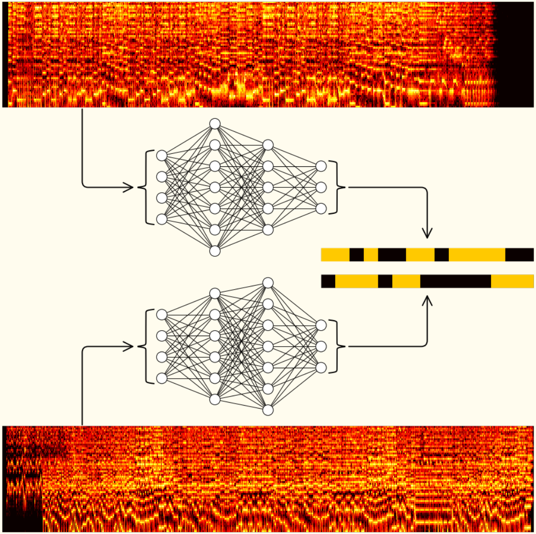
1. Compute the total distance between aligned frames
2. Normalize by the path length
3. Normalize by the mean distance between all frames



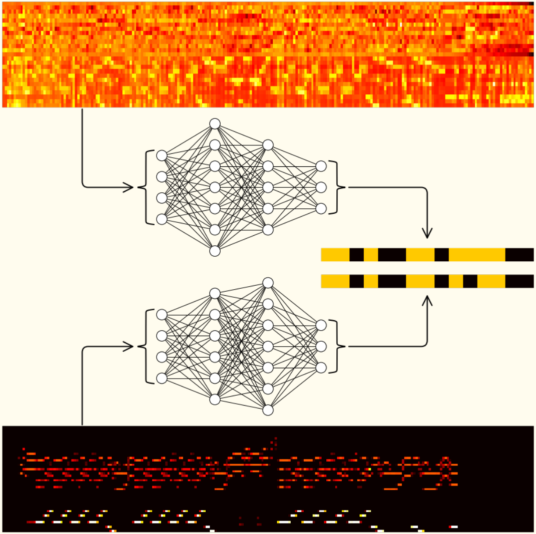
Similarity-Preserving Hashing



Similarity-Preserving Hashing

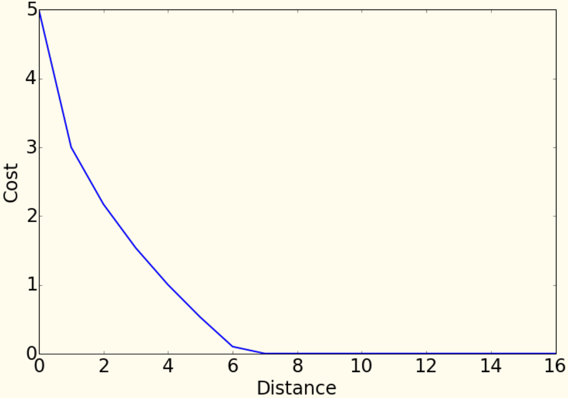


Cross-Modality Hashing



Cost Thresholding for Negatives

$$\max(0, m - \|x - y\|_2)^2$$



Neural Network Details

- $\approx$ 1.4M examples, 10% used as validation set

Neural Network Details

- $\approx$ 1.4M examples, 10% used as validation set
- Negative examples chosen at random

Neural Network Details

- $\approx$ 1.4M examples, 10% used as validation set
- Negative examples chosen at random
- Inputs shingled and Z-scored

Neural Network Details

- $\approx$ 1.4M examples, 10% used as validation set
- Negative examples chosen at random
- Inputs shingled and Z-scored
- SGD with Nesterov's Accelerated Gradient

Neural Network Details

- $\approx$ 1.4M examples, 10% used as validation set
- Negative examples chosen at random
- Inputs shingled and Z-scored
- SGD with Nesterov's Accelerated Gradient
- tanh units in every layer

Neural Network Details

- $\approx$ 1.4M examples, 10% used as validation set
- Negative examples chosen at random
- Inputs shingled and Z-scored
- SGD with Nesterov's Accelerated Gradient
- tanh units in every layer
- Early-stopping using validation set cost

Neural Network Details

- $\approx$ 1.4M examples, 10% used as validation set
- Negative examples chosen at random
- Inputs shingled and Z-scored
- SGD with Nesterov's Accelerated Gradient
- tanh units in every layer
- Early-stopping using validation set cost
- No other regularization needed

Neural Network Details

- $\approx$ 1.4M examples, 10% used as validation set
- Negative examples chosen at random
- Inputs shingled and Z-scored
- SGD with Nesterov's Accelerated Gradient
- tanh units in every layer
- Early-stopping using validation set cost
- No other regularization needed
- Hyperparameters chosen using `hyperopt`

Neural Network Details

- $\approx$ 1.4M examples, 10% used as validation set
- Negative examples chosen at random
- Inputs shingled and Z-scored
- SGD with Nesterov's Accelerated Gradient
- tanh units in every layer
- Early-stopping using validation set cost
- No other regularization needed
- Hyperparameters chosen using `hyperopt`
- Model objective: Ratio of mean in-class and mean out-of-class distances

Neural Network Details

- $\approx$ 1.4M examples, 10% used as validation set
- Negative examples chosen at random
- Inputs shingled and Z-scored
- SGD with Nesterov's Accelerated Gradient
- tanh units in every layer
- Early-stopping using validation set cost
- No other regularization needed
- Hyperparameters chosen using `hyperopt`
- Model objective: Ratio of mean in-class and mean out-of-class distances
- 16-bit hashes created by thresholding output

Neural Nets with `lasagne`

```python
import lasagne
layers = []
# Input layer signals end of network computations
layers.append(lasagne.layers.InputLayer(shape=(batch_size, n_features)))
# Add each hidden layer recursively
for num_units in hidden_layer_sizes:
    # A dense layer implements \sigma(Wx + b)
    layers.append(lasagne.layers.DenseLayer(layers_X[-1], num_units=num_units))
    # Dropout is implemented as a layer
    layers.append(lasagne.layers.DropoutLayer(layers[-1]))
# Add output layer
layers.append(lasagne.layers.DenseLayer(layers[-1], num_units=n_output))
# Get a list of all network parameters
params = lasagne.layers.get_all_params(layers[-1])
# Define a cost function using layers[-1].get_output(input)
# Compute updates for Nesterov's Accelerated Gradient
updates = lasagne.updates.nesterov_momentum(cost, params, learning_rate, momentum)
```

http://www.github.com/benanne/Lasagne

Why Hash?

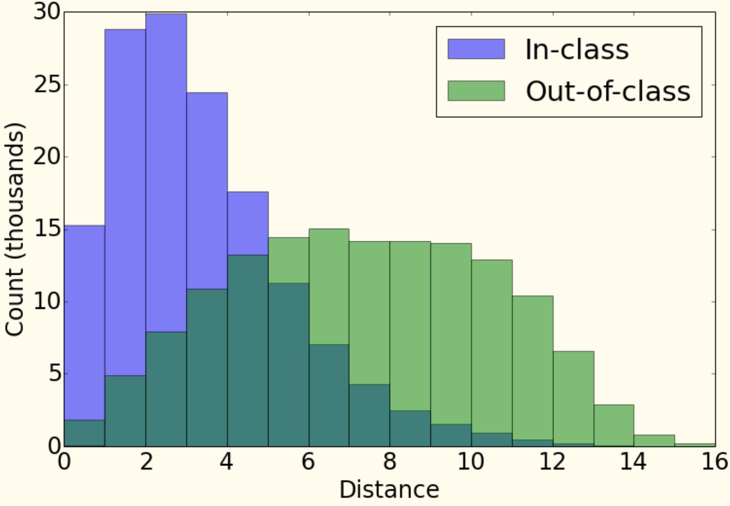
$$x \in \mathbb{R}^{M \times I}, y \in \mathbb{R}^{N \times I}$$

$$distance[m, n] = \sum_i (x[m, i] - y[n, i])^2$$

$$x \in \mathbb{R}^{M}, y \in \mathbb{R}^{N}$$

$$distance[m, n] = bits\_set[x[m]^\wedge y[n]]$$

Validation Set Distances



Content-Based Matching Pipeline

1. Pre-compute hash sequences for all MSD entries

Content-Based Matching Pipeline

1. Pre-compute hash sequences for all MSD entries
2. Store sorted list of MSD entry durations

Content-Based Matching Pipeline

1. Pre-compute hash sequences for all MSD entries
2. Store sorted list of MSD entry durations
3. Compute hash sequence for query MIDI file

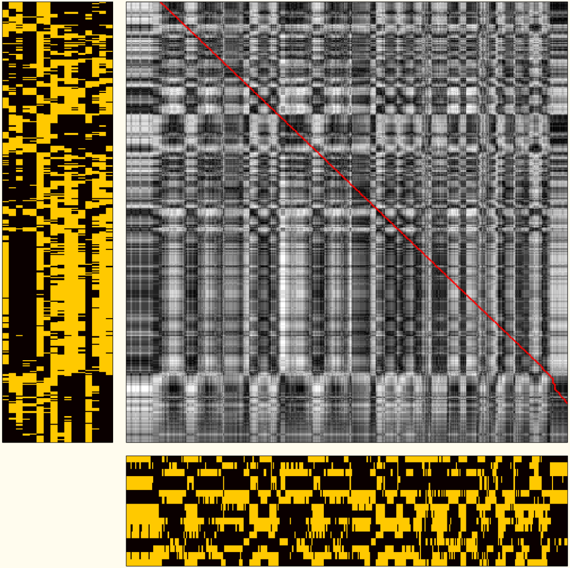
Content-Based Matching Pipeline

1. Pre-compute hash sequences for all MSD entries
2. Store sorted list of MSD entry durations
3. Compute hash sequence for query MIDI file
4. Select MSD hash sequences within a tolerance of MIDI file duration

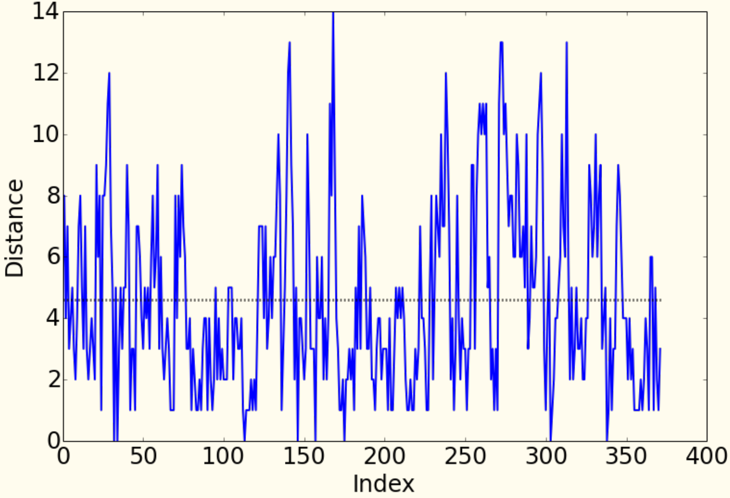
Content-Based Matching Pipeline

1. Pre-compute hash sequences for all MSD entries
2. Store sorted list of MSD entry durations
3. Compute hash sequence for query MIDI file
4. Select MSD hash sequences within a tolerance of MIDI file duration
5. Compute DTW distances to these sequences

Example: Hash Sequence DTW



Example: Distance Along Path



Confounding Factors

- MIDI and MSD durations aren't within chosen tolerance

Confounding Factors

- MIDI and MSD durations aren't within chosen tolerance
- Beat tracking varies drastically

Confounding Factors

- MIDI and MSD durations aren't within chosen tolerance
- Beat tracking varies drastically
- MIDI is a poor transcription

Confounding Factors

- MIDI and MSD durations aren't within chosen tolerance
- Beat tracking varies drastically
- MIDI is a poor transcription
- Hashing fails

Future Work

- Better hashing (recurrence)

Future Work

- Better hashing (recurrence)
- Faster DTW

Future Work

- Better hashing (recurrence)
- Faster DTW
- Better text-based matching

Future Work

- Better hashing (recurrence)
- Faster DTW
- Better text-based matching
- Regular alignment after matching

Future Work

- Better hashing (recurrence)
- Faster DTW
- Better text-based matching
- Regular alignment after matching
- Quantitative evaluation!

Future Work

- Better hashing (recurrence)
- Faster DTW
- Better text-based matching
- Regular alignment after matching
- Quantitative evaluation!
- Dataset release

Related Work

Thanks!

http://github.com/craffel/midi-dataset
http://github.com/craffel/pretty-midi
http://github.com/bmcfee/librosa
http://github.com/benanne/Lasagne

craffel@gmail.com