Instrument Classification

Nick Meyer, Jack Killian, Tim Aris

Abstract

Automatic instrument recognition can be accomplished by transforming an amplitude signal from a song into many features related to frequencies, pitch and power and feeding these features into a learning machine. We investigated the IRMAS dataset containing ~6,000 three second recordings in which more than one instrument may be playing, but only one is the “lead”. To classify clips by lead instruments, we implemented three styles of learning machines that have shown promising results on this task: K-nearest-neighbors, Support Vector Machines, and Neural Networks (deep and shallow). We found that shallow networks performed the best at ~57% accuracy over 11 classes, though the performance of all the machines was heavily dependent on the input feature set. More work guided by audio engineering techniques is needed to further improve accuracy.

Intro

Identify the main instrument present in an audio file. Our data set (IRMAS) focuses on melodic instruments [1].

Implications in the Field of Music Information Retrieval.

- **Indexing** songs by instrument (Spotify, Pandora, or iTunes)
- **Genre classification** (lead instrument can correlate with genre)

Background

- Built off of 2012 thesis paper and dataset [2]
- 6705 training samples / 2874 test samples
  - 11 instruments
  - 3 seconds each
- Thesis describes using SVM, K-NN, and Neural Nets
  - Best results with **SVM (~63%)**
  - **K-NN (57.4%)**, Neural Net **(57.9%)**

← **We want to match/beat these**

- Other work from literature suggests these as best methods for sound classification (with basic learning machines)
- **Most important step is Feature Extraction/Selection**

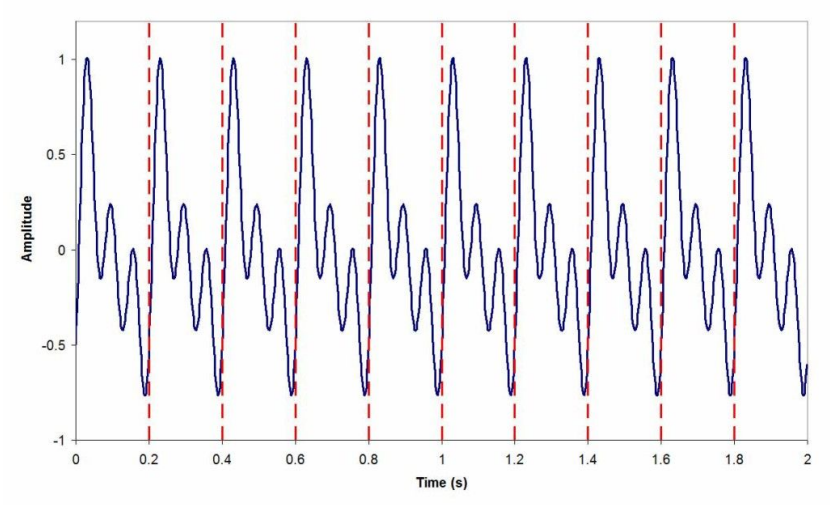
Feature Extraction

- Our data (wav files) starts as an array of amplitudes over time
- This in **raw form is difficult** to use to discern between instruments
- Feature extraction is extremely important
- Most features derived by **converting to frequency space**
    - Mel Frequency Cepstral Coefficients (MFCC)
    - Spectral features (centroid, spread, energies, valleys, rolloff)
    - Zero Crossing Rate

Mel Frequency Cepstral Coefficients (MFCC)

1. Frame the signal into short frames.

- Window Lengths of 10-100 ms are common
- Will calculate coefficients for each window

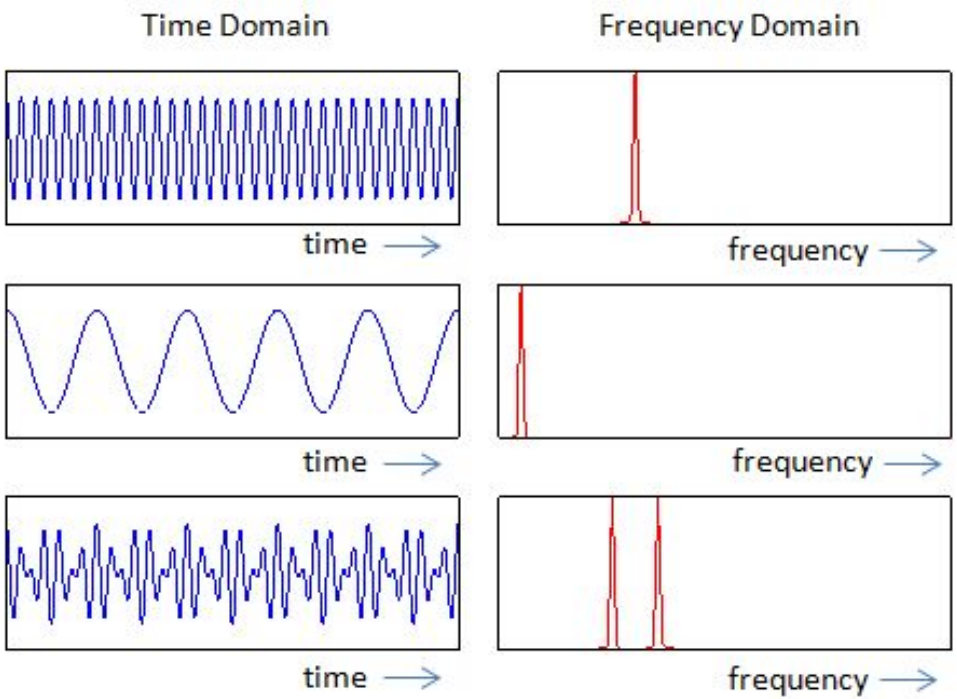


[3]

MFCC

2. Convert each window from time domain to frequency domain.

Accomplished by taking a Fourier Transform.

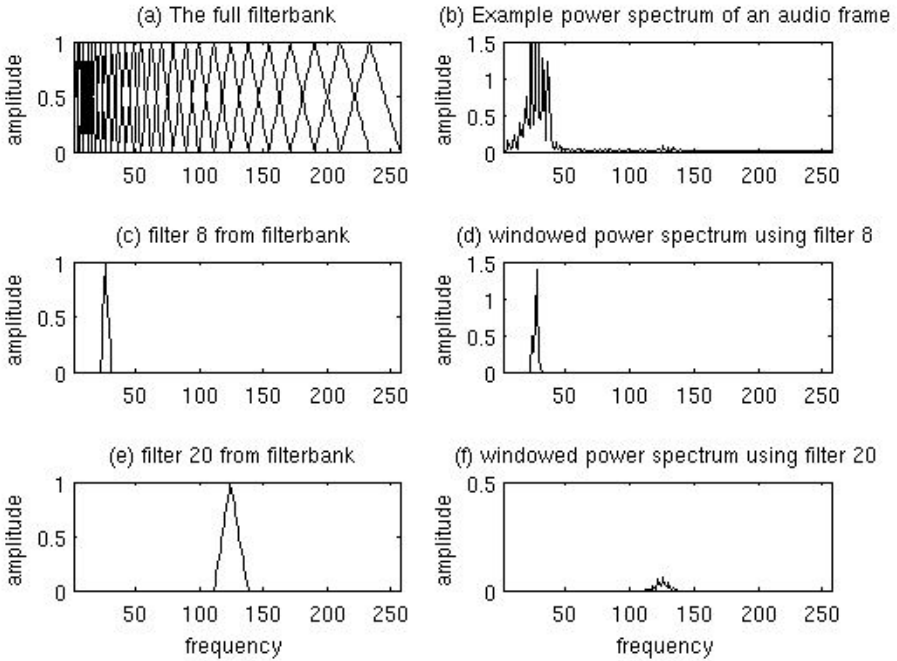


[3]

MFCC

3. Determine a set of windows (or filter banks - the triangular regions) and sum the energies in each window.

Typical number of filters is 26



[3]

MFCC

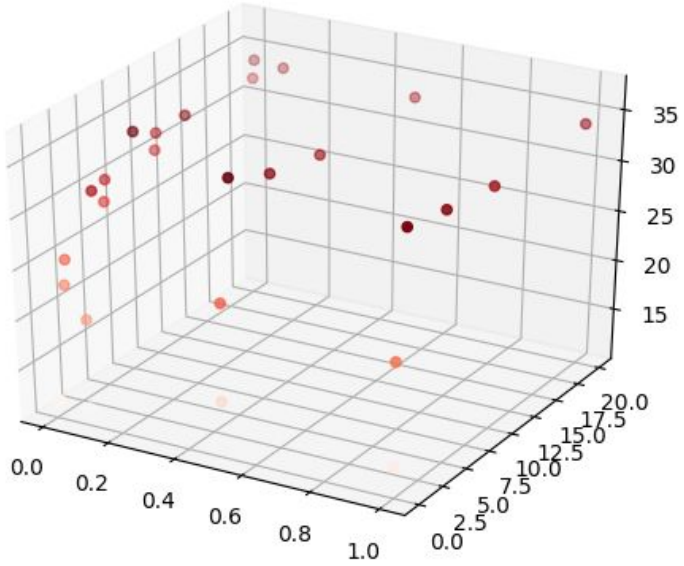
4. Take the logarithm of all filterbank energies.

5. Take the Discrete Cosine Transform (DCT) of the log filterbank energies.

6. Keep DCT coefficients 2-13 (make sure to discard 0 as this contains volume and zero-frequency information)

[3]

SVM Implementation



Accuracy over many hyperparameters.

X - Gamma
Y - C
Z - Accuracy

The SVM was implemented using libsvm and sklearn.

An RBF kernel was used.

The confidence and learning rate hyperparameters were found using a naive grid search and checked with cross validation.

To deal with multiple classes, many two class (One vs One or One vs All) SVMs are created and the most likely result is chosen.

SVM - Feature Selection and Results

Feature selection/elimination using sklearn's SelectKBest function.

- Varied the number of features, never getting an increase in accuracy

Aggregated best features over pairwise combinations of labels to estimate the best features for each label

Created an SVM using each label's best feature set and used either max or average over the results to pick a label

- Also did not help, with accuracy in the low 30%s.

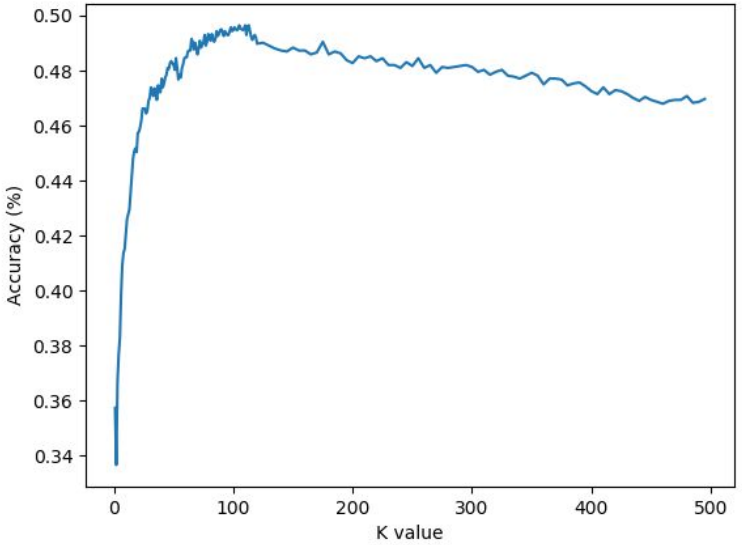
Best result so far uses all of the mentioned features + barkbands, achieving 37% accuracy.

KNN

K-nearest-neighbors was implemented with sklearn in python as well as manually.

All methods of calculating distance resulted in basically the same accuracy. (Euclidean, Manhattan, Minkowski)

The final accuracy was around 49%, with a k value of 105.



Recursive Feature Elimination

This removes the worst feature for predicting the data, then removes the next worst until the number of features matches the target number.

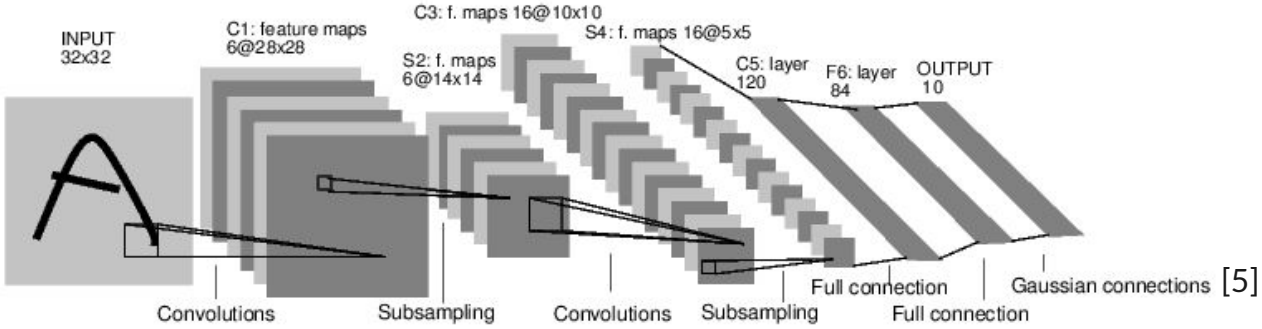
RFE was tried with KNN.

The thought was that because KNN can't attach weights to features, so some features might have been damaging the results.

Apparently all features were useful, however, as just reducing them from 13 to 12 resulted in a ~3% loss in accuracy.

Deep Neural Net - Implementation

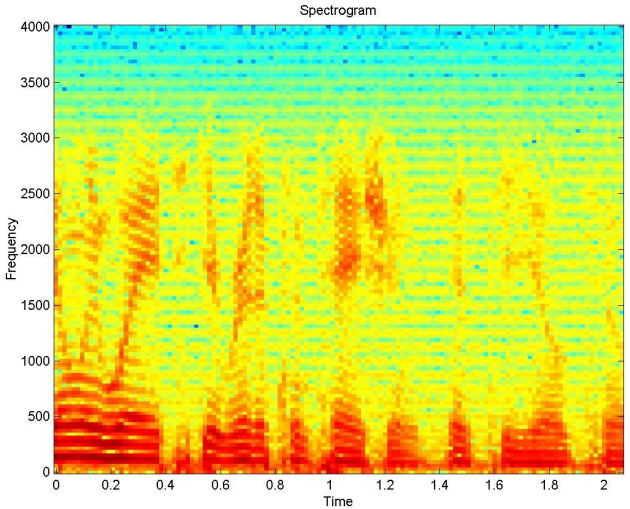
- Implemented with TensorFlow
  - Structure [4]: (32x32x1) -> CONV (32x32x32) -> POOL (16x16x32) -> CONV (16x16x64) -> POOL (8x8x64) -> FC (4096x1024) -> FC (1024x13) -> OUT
  - Softmax, Gradient Descent



[5]

Deep Neural Net - Implementation

- Input a spectrogram
  - Matrix where each row is FFT of a window of sound
- Hyperparameter: Window Size
  - Smaller windows → larger spectrograms
  - Laptop could only handle training for spectrograms of ~5 rows
    - (since sampling rate is so high at 44kHz = #columns)

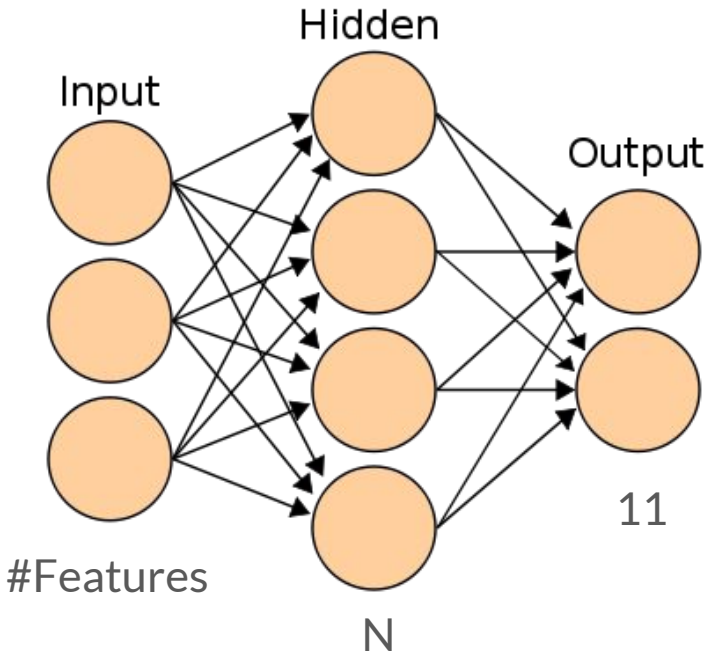


Deep Neural Net - Results

- Maximum accuracy achieved was **~38%.**
- Likely could have achieved higher with smaller windows
  - need more memory!
- MFCC space spectrogram?

Shallow Neural Net - Implementation

- Implemented with TensorFlow
- Structure:
  - One hidden layer of size N
  - Output layer of size 11 (softmax)
- Cross Entropy Loss
  - appropriate for softmax
  - faster training than MSE
- Hyperparameters (learning rate, N) selected via cross validation for each attempted feature set



Shallow Neural Net - Results

- Average MFCC (plus 1st order differences)
  - Accuracy: **32.13%**
- Average MFCC (plus 2nd order differences)
  - Accuracy: **30.82%**
- MFCC - Flattened Covariance matrix
  - Accuracy: **45.82%**
- “Full Set” - Average MFCC plus Spectral features
  - Accuracy: **48.63%**

Shallow Neural Net - Feature Selection

- Calculated Feature Importance using SciKit-Learn's ExtraTreesClassifier
  - Builds a forest of trees from an input training set
  - Calculates the importance of each feature in making classifications

Shallow Neural Net - Feature Selection

- Top 35% most important features from “Full Set”
  - Accuracy: **50.25%**
- Top 50% most important features from “Full Set” + MFCC covariance
  - Best Accuracy: **57.75%**
  - Closest match to thesis results
  - **Within <1% of thesis results**

Discussion

- Our dataset is less forgiving than most.
  - Variable instruments, loudness, recording quality, what makes a “lead” instrument
- Not all methods reached the accuracy achieved from the thesis, but comparing our results to Fuhrmann's still gives insights.
  - He got the best accuracy from an SVM, but our best accuracy came from a NN, and SVM gave our worst.
- There's more to be researched about what features most strongly affect prediction for different learning algorithms.

Limitations

- Computational Power
  - Feature: Linear Prediction Coefficients
  - Important in thesis results
  - Too long for us to feasibly compute for 6K training samples
- Memory
  - Deep Nets work with raw/ semi-processed input
  - Insufficient memory to support this
  - Truncated input likely caused poor results for Deep Net

References

- [1] https://www.upf.edu/web/mtg/irmas
- [2] http://www.dtic.upf.edu/~ffuhrmann/PhD/ffuhrmann_PhDthesis.pdf
- [3] http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/
- [4] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324.
- [5] https://medium.com/@awjuliani/recognizing-sounds-a-deep-learning-case-study-1bc37444d44d