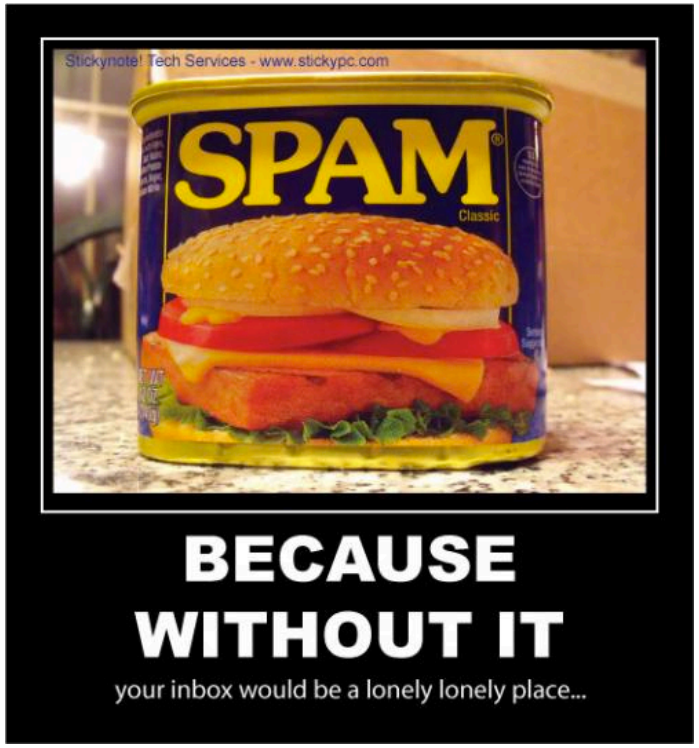
Email Spam Filtering

Jenny Huang

What is spam?

Spam is irrelevant or inappropriate messages sent on the Internet to a large number of recipients, so unsolicited bulk email.



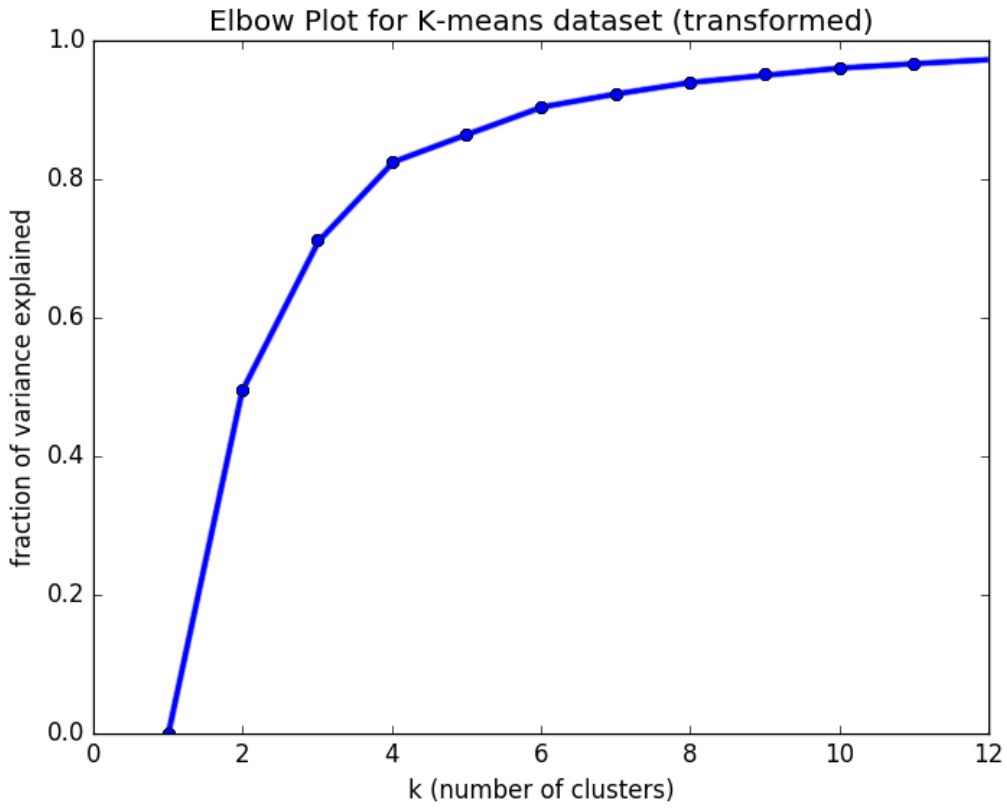
Data

- Number of instances (m): 4601
- Number of attributes (p): 58

0	0.64	0.64	0	...	3.756	61	278	1
0.21	0.28	0.5	0	...	5.114	101	1028	1
				.				
				.				
				.				
0.96	0	0	0	...	1.147	5	78	0
0	0	0.65	0	...	1.25	5	40	0

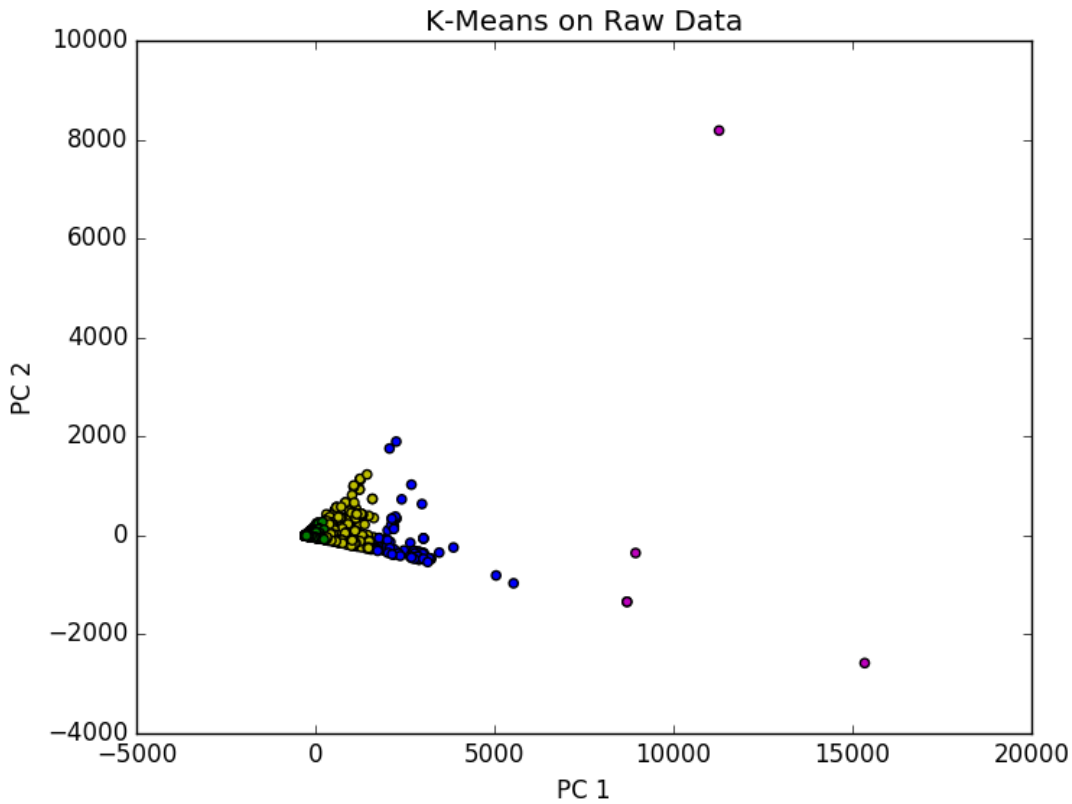
Methods

- K-means & PCA



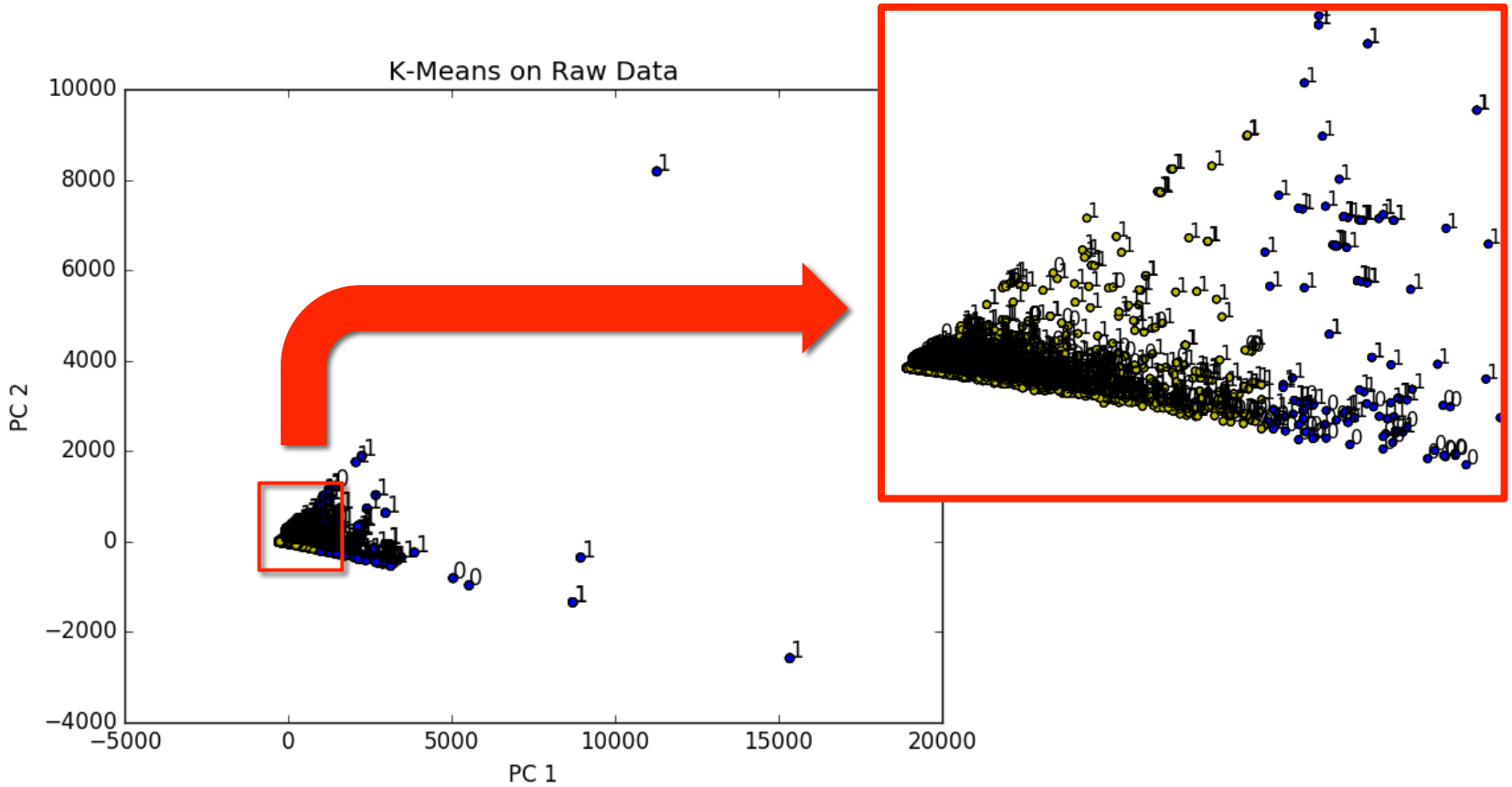
Methods

- K-means & PCA



Methods

- K-means & PCA



Methods

- Nearest Neighbors & Linear Regression
- K-means Accuracy

```
NEAREST NEIGHBORS:
k=1, 0.704041720991
k=2, 0.810951760104
k=5, 0.698826597132
k=10, 0.726205997392
k=15, 0.680573663625
k=20, 0.699478487614

LINEAR REGRESSION:
accuracy: 0.832464146023
```

```
train accuracy: 0.634496250408
test accuracy: 0.631029986962
```

Challenges & Future Work

- Unable to fully interpret the data
- What do the clusters mean with k=4?
- Run same methods on subsets of data
- 3 component PCA plot

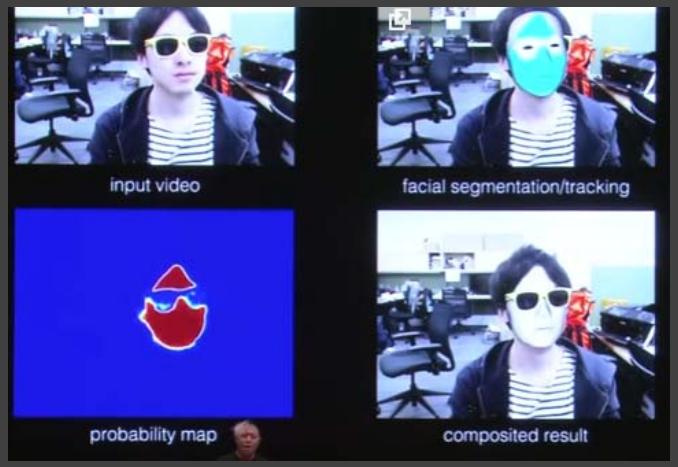
Thank you!

Image Segmentation

JESSICA TRAN, CSC390

Motivation

Human Digitization – Hao Li



Data

Database of Faces from Cambridge University

- 92x112 pixels, with 256 grey levels per pixel

Labeled Faces in the Wild Home

- More than color 13,000 images of faces collected from the web

Any image!

- Using OpenCV (cv2.imread) we can get the image data for any image in color(1), grayscale(0) or unchanged(-1)

Database of Faces from Cambridge University

Database of Faces from Cambridge University

Image to matrix:

- Each grayscale image → 92 columns, 112 rows

Matrix to image: ???

- import matplotlib.pyplot as plt
- PIL

```
[[48 49 45 ..., 56 56 54]
 [45 52 39 ..., 52 50 51]
 [45 50 42 ..., 48 53 50]
 ...,
 [50 48 50 ..., 45 46 46]
 [45 54 49 ..., 46 47 47]
 [51 51 51 ..., 47 46 46]]
```

Methods for Image Segmentation

Spectral clustering:

- With Otsu's method
- With K-means
- With discretization

```
from sklearn.cluster import spectral_clustering
```

```
# global thresholding
ret1,th1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)

# Otsu's thresholding
ret2,th2 = cv2.threshold(img,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

# Otsu's thresholding after Gaussian filtering
blur = cv2.GaussianBlur(img,(5,5),0)
ret3,th3 = cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

# plot all the images and their histograms
images = [img, 0, th1,
          img, 0, th2,
          blur, 0, th3]
titles = ['Original Noisy Image','Histogram','Global Thresholding (v=127)',
          'Original Noisy Image','Histogram',"Otsu's Thresholding",
          'Gaussian filtered Image','Histogram',"Otsu's Thresholding"]
```

K-means clustering

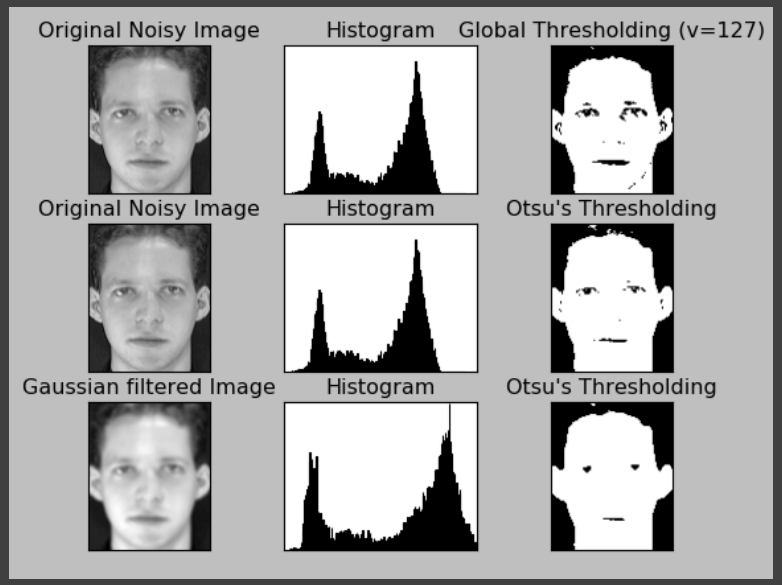
```
import numpy as np
```

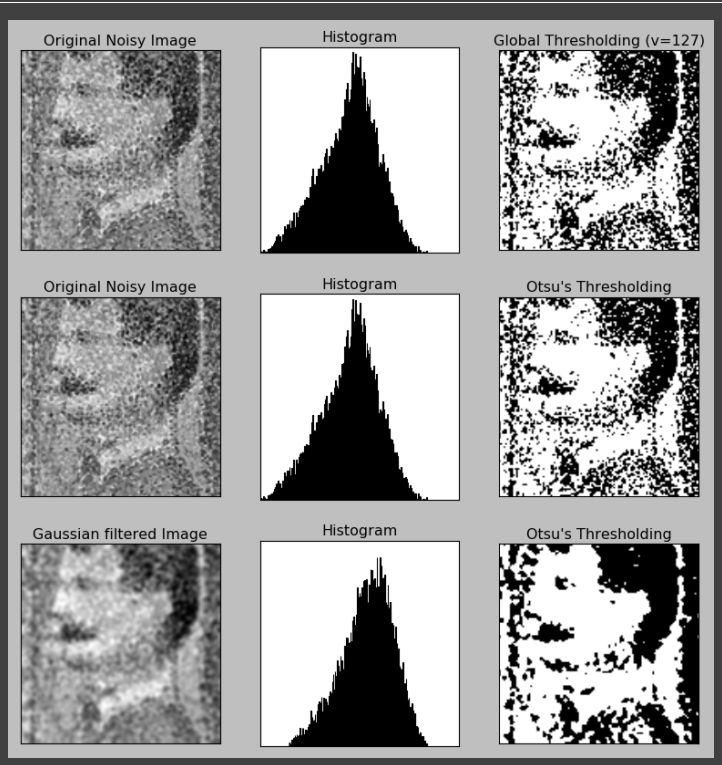
Otsu’s Method

Clustering-based image thresholding

Transforms image to a binary image

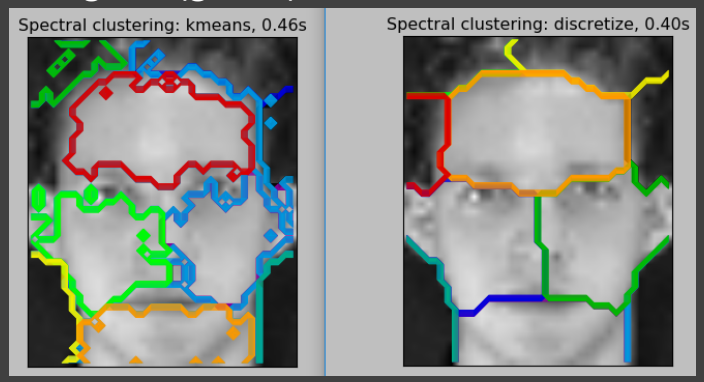
- Foreground and background pixels



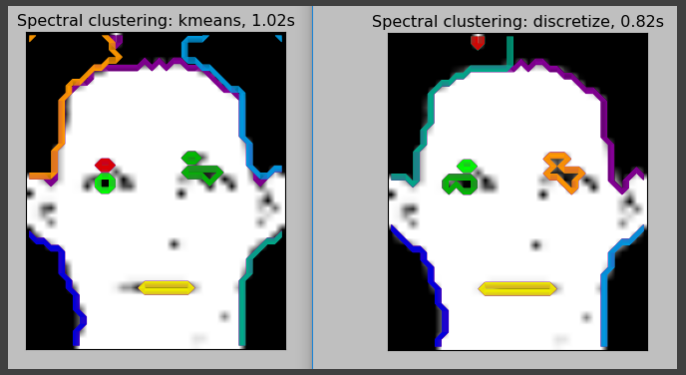


Spectral clustering

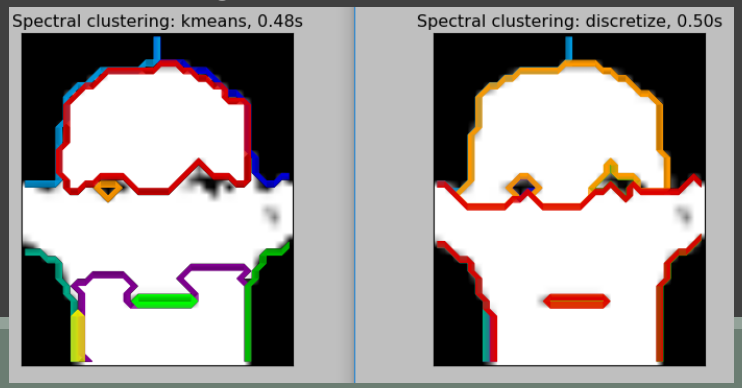
10 regions (global)



10 regions with Otsu's Method

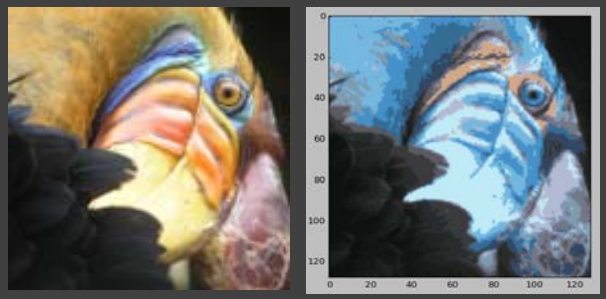


10 regions with Gaussian Blur

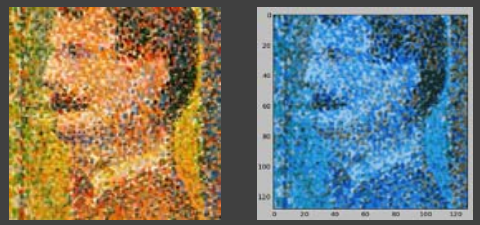


K-Means Clustering

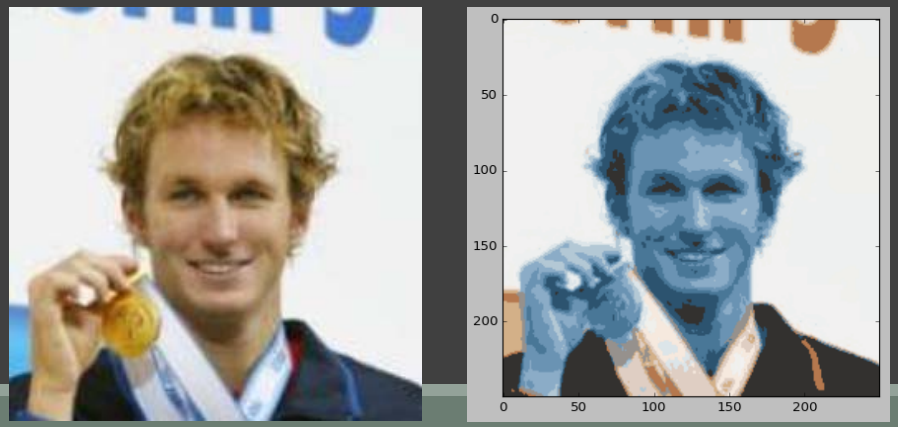
K = 5



K = 16



K = 16



K = 16

Interpretation and Future Work

K-means clustering > Spectral clustering

Otsu's method might be helpful in real time

Further work in training, identifying, and focusing on features (crop image)

Questions?

Stock Market Analysis: How To Be a Billionaire?

Motivation

- apply machine learning approaches in stock market analysis

- test common opinions
  - Efficient market hypothesis?
  - Stocks in the same sectors behave similarly?
  - Is it hard to outperform the market?

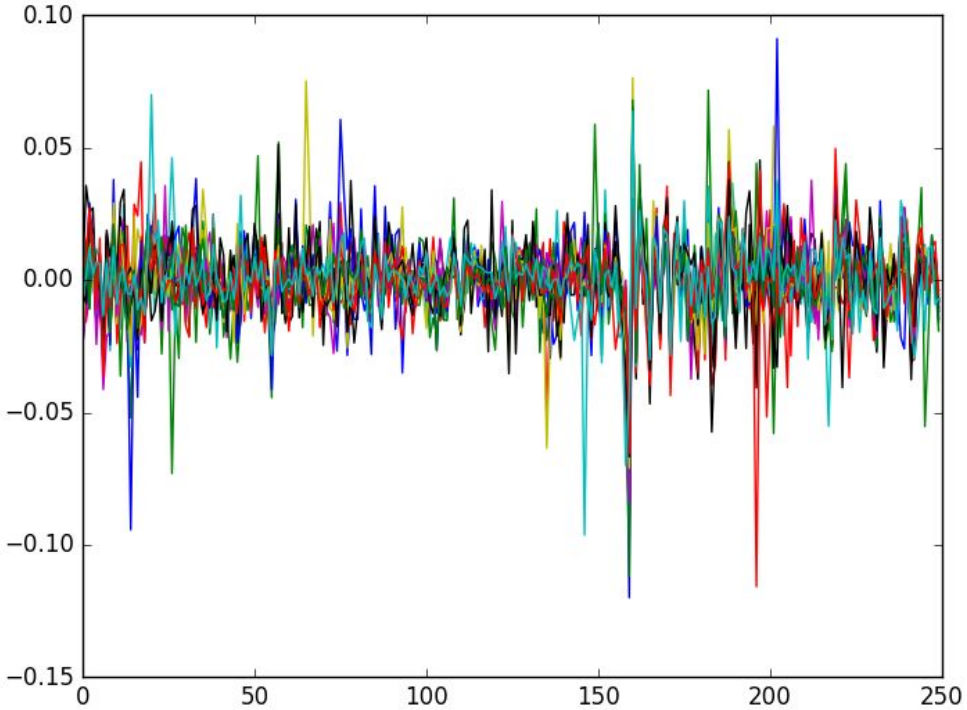
Methods & Data

- Clustering
  - PCA + K-means
  - k = 6

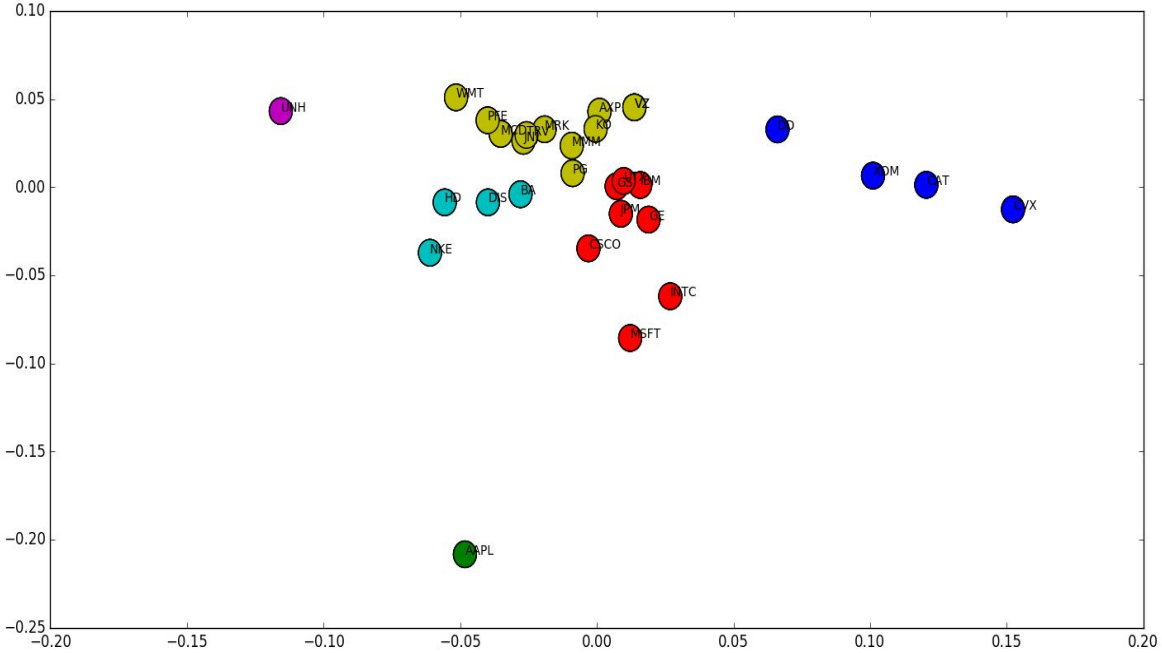
    consumer, energy & utilities, industrials, financials, IT, healthcare
- GMM
  - n = 2: rise or fall
  - Predict hidden state for each stock
- Data
  - Dow Jones Components (30 companies)
  - Time Period: 2015 (250 days)

    leave the last 7 days as testing
  - Pre-processed to obtain %return

Results - Plotting Returns



Results - Clustering (k=6)



-Apple

-United Healthcare

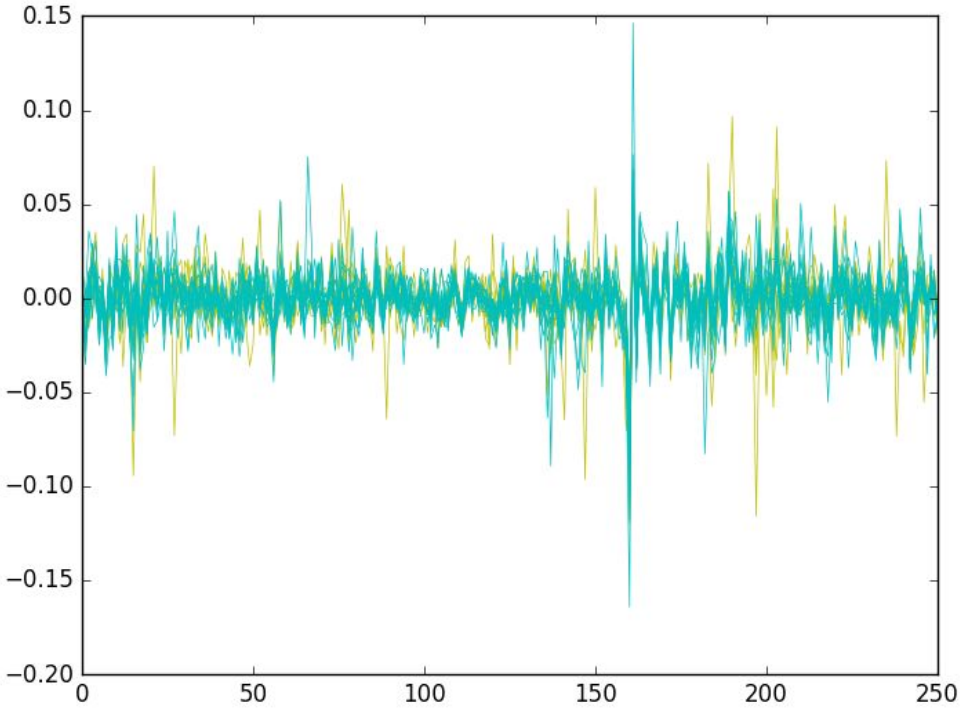
-Boeing
-Home Depot
-NIKE
-Walt Disney

-Cisco
-GE
-Goldman Sachs
-Intel
-IBM
-JPMorgan Chase
-Microsoft
-United Technologies

-Caterpillar
-Chevron
-DuPont
-Exxon Mobil

-3M
-American Express
-Coca-Cola
-Johnson & Johnson
-McDonald's
-Merck
-Pfizer
-P&G
-The Travelers Companies
-Verizon
-Visa
-Walmart

Results - GMM (n=2)



Results - GMM (Continued)

Simple simulation on test data:

- Find the state that has a higher %return on average
- Invest evenly on the companies (i.e. $1000 each)
- calculate profit

```
total invest =  9000
profit =  158.629876201
percentage profit =  1.76255418001 %
```

As a benchmark, Dow Jones %change = 0.314%

We outperform the market!

...Not Really

+=

with uncertainty.

Future Work

- Compare true industries with predicted ones
  - Rand Index
- Moving Average
  - GMM: history doesn’t matter
- Include company fundamentals

Questions?

IDENTIFYING COMPOSER FROM MIDI FILES

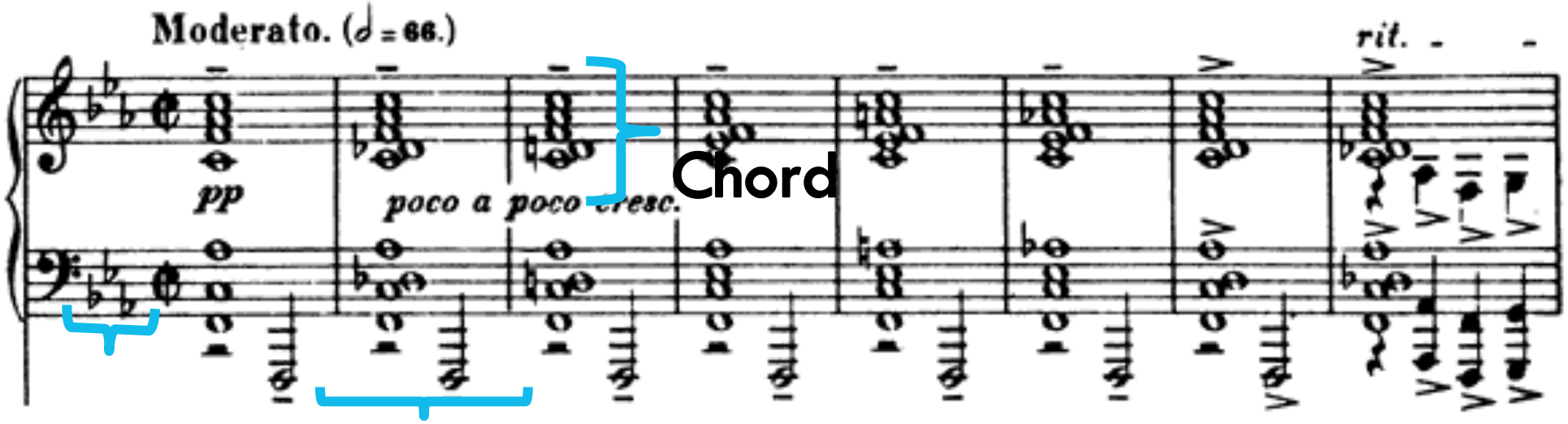
Sharon Vizcaíno

MOTIVATION

Music analysis is tedious and time-consuming to do by hand, especially when composers have hundreds of works

Often hear classical music and wonder who the composer is

BASIC MUSIC THEORY



Key signature

Measure

MORE BASIC MUSIC THEORY

Interval: Distance between two notes, counted by step (each key on a piano)

Seventh Chord: Chord made from stacking 4 notes, with an interval of 3 between each note

Dissonance: Combination of notes that creates a clash/tension

Chord quality: Type of chord (major: happy, minor: sad, etc)

PROBLEMS

Bad quality MIDIs do not consistently provide good data for note duration

Lack of significant number of MIDIs for several major composers

Limited knowledge of music theory leads to a less-than-ideal data set, in which factors considered may not be the best ones to focus on

Key changes may affect result

Analyzing the music takes too long, varies depending on length of song (and seemingly MIDI quality)

MUSIC21

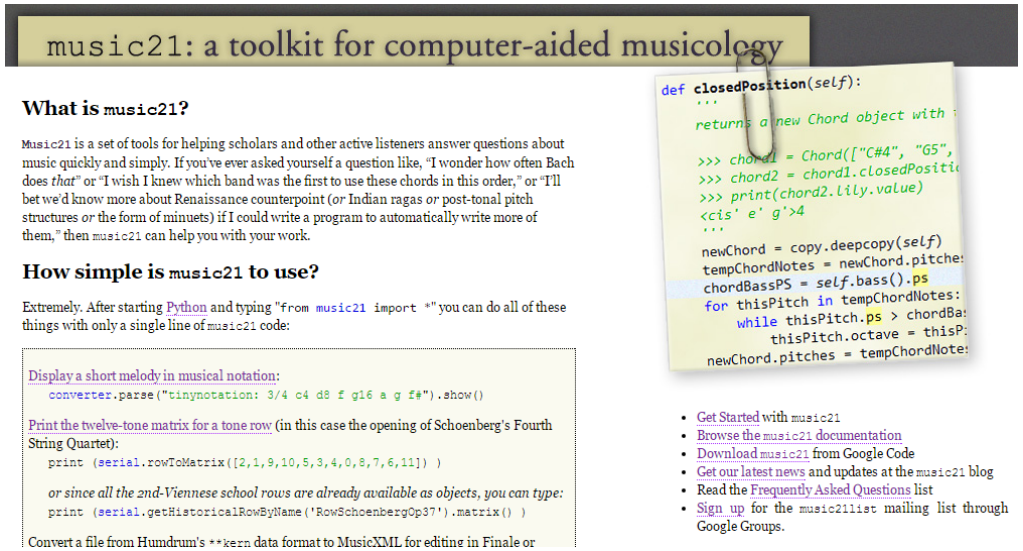
Python library for musical analysis

Capabilities include “chordifying”, identification of chord quality, intervals, grace notes, etc

Simple to use

Support for multiple formats

<u>Excellent documentation</u>



music21: a toolkit for computer-aided musicology

What is music21?

Music21 is a set of tools for helping scholars and other active listeners answer questions about music quickly and simply. If you've ever asked yourself a question like, “I wonder how often Bach does *that*” or “I wish I knew which band was the first to use these chords in this order,” or “I'll bet we'd know more about Renaissance counterpoint (*or* Indian ragas *or* post-tonal pitch structures *or* the form of minuets) if I could write a program to automatically write more of them,” then music21 can help you with your work.

How simple is music21 to use?

Extremely. After starting Python and typing "from music21 import *" you can do all of these things with only a single line of music21 code:

Display a short melody in musical notation:

```
converter.parse("tinynotation: 3/4 c4 d8 f g16 a g f#").show()
```

Print the twelve-tone matrix for a tone row (in this case the opening of Schoenberg's Fourth String Quartet):

```
print (serial.rowToMatrix([2,1,9,10,5,3,4,0,8,7,6,11]) )
```

or since all the 2nd-Viennese school rows are already available as objects, you can type:

```
print (serial.getHistoricalRowByName('RowSchoenbergOp37').matrix() )
```

Convert a file from Humdrum's **kern data format to MusicXML for editing in Finale or

```
def closedPosition(self):
    '''
    returns a new Chord object with
    >>> chord1 = Chord(["C#4", "G5",
    >>> chord2 = chord1.closedPositi
    >>> print(chord2.lily.value)
    <cis' e' g'>4
    '''
    newChord = copy.deepcopy(self)
    tempChordNotes = newChord.pitche
    chordBassPS = self.bass().ps
    for thisPitch in tempChordNotes:
        while thisPitch.ps > chordBa
            thisPitch.octave = thisP
    newChord.pitches = tempChordNote
```

- Get Started with music21
- Browse the music21 documentation
- Download music21 from Google Code
- Get our latest news and updates at the music21 blog
- Read the Frequently Asked Questions list
- Sign up for the music21list mailing list through Google Groups.

ANALYZING THE MUSIC

5 features:

- Percentage of notes that don't belong to the key
- Percentage of chords that are Augmented Sixth chords
- Percentage of chords that are inversions
- Percentage of chords that are seventh chords
- Percentage of chords that are dissonant

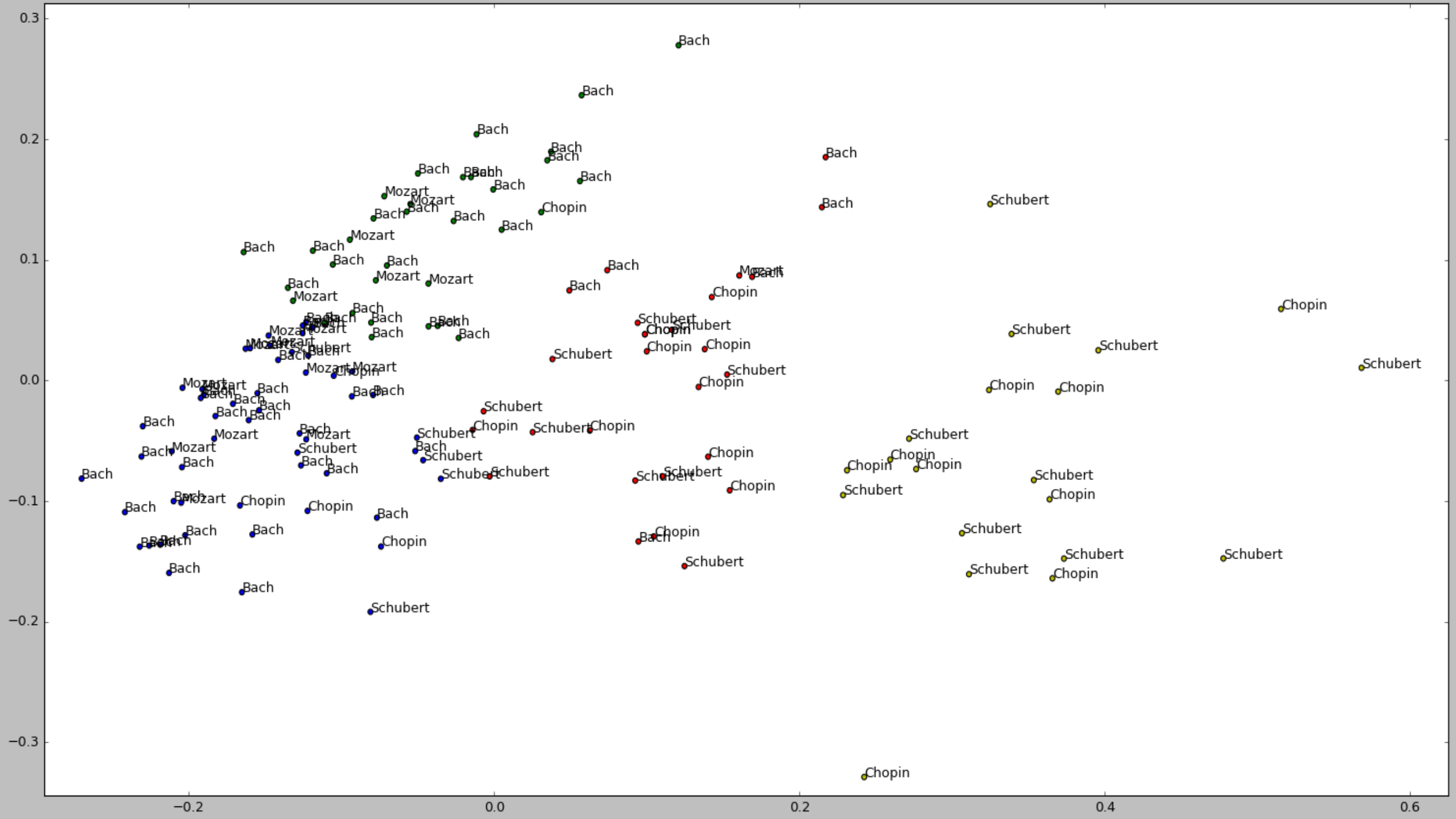
ANALYSIS USING MUSIC21

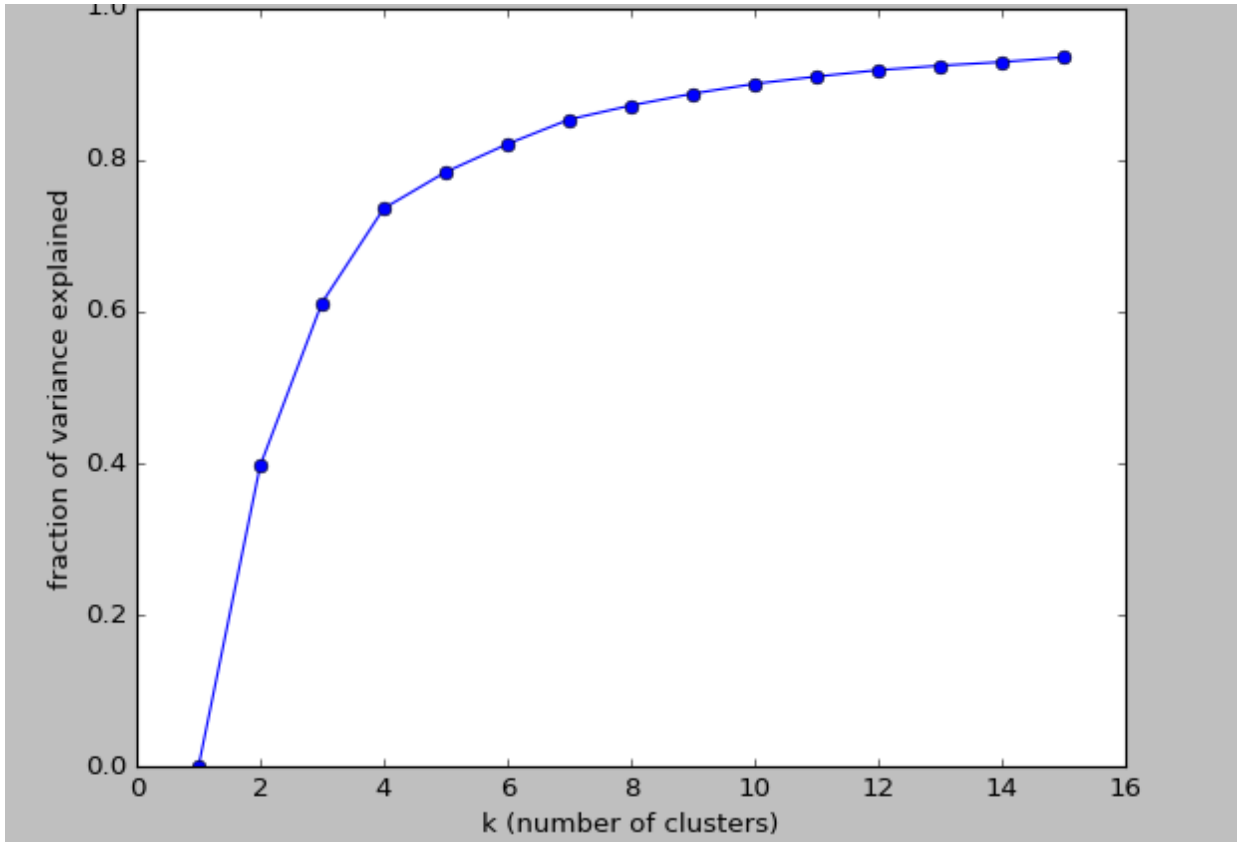
```
for chord in chord_stream.recurse().getElementsByClass('Chord'):
    chord_count += 1
    if chord.isAugmentedSixth():
        chord_dict['aug6'] += 1
    elif not chord.isConsonant():
        chord_dict['diss'] += 1
    elif chord.isPrimeFormInversion:
        chord_dict['inv'] += 1
    elif chord.containsSeventh():
        chord_dict['sev'] += 1
```

```
for note in stream.pitches:
    if note.name not in keyPitches:
        notes_not_in_key += 1
```

DATA

```
0.0 0.5280898876404494 0.08146067415730338 0.0 0.04340567612687813
0.0 0.47280334728033474 0.1297071129707113 0.0 0.05555555555555555
0.0 0.26109660574412535 0.0 0.0 0.04267161410018553
0.0 0.5476190476190477 0.10582010582010581 0.0 0.09292035398230089
0.0 0.3977272727272727 0.0 0.0 0.06818181818181818
0.0 0.1994750656167979 0.0 0.0 0.08346213292117466
0.0 0.5490196078431373 0.08333333333333333 0.0 0.07142857142857142
0.0 0.4980544747081712 0.08949416342412451 0.0 0.06122448979591836 6
0.0 0.3225 0.0 0.0 0.08970588235294118
0.0 0.51832460732984 3 0.0549738219895288 0.0 0.08372093023255814
0.0 0.603082851637765 0.06551059730250482 0.0 0.09265944645006016
0.0 0.29955947136563876 0.0 0.0 0.07194244604316546
0.006944444444444444 0.5243055555555556 0.06597222222222222 0.0 0.18
0.0 0.548460661345496 0.06385404789053592 0.0 0.06447534766118837
```


0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.2
0.0
0.2
0.4
0.6
Bach
Mozart
Chopin
Schubert


1.0
0.8
0.6
0.4
0.2
0.0
fraction of variance explained
0
2
4
6
8
10
12
14
16
k (number of clusters)

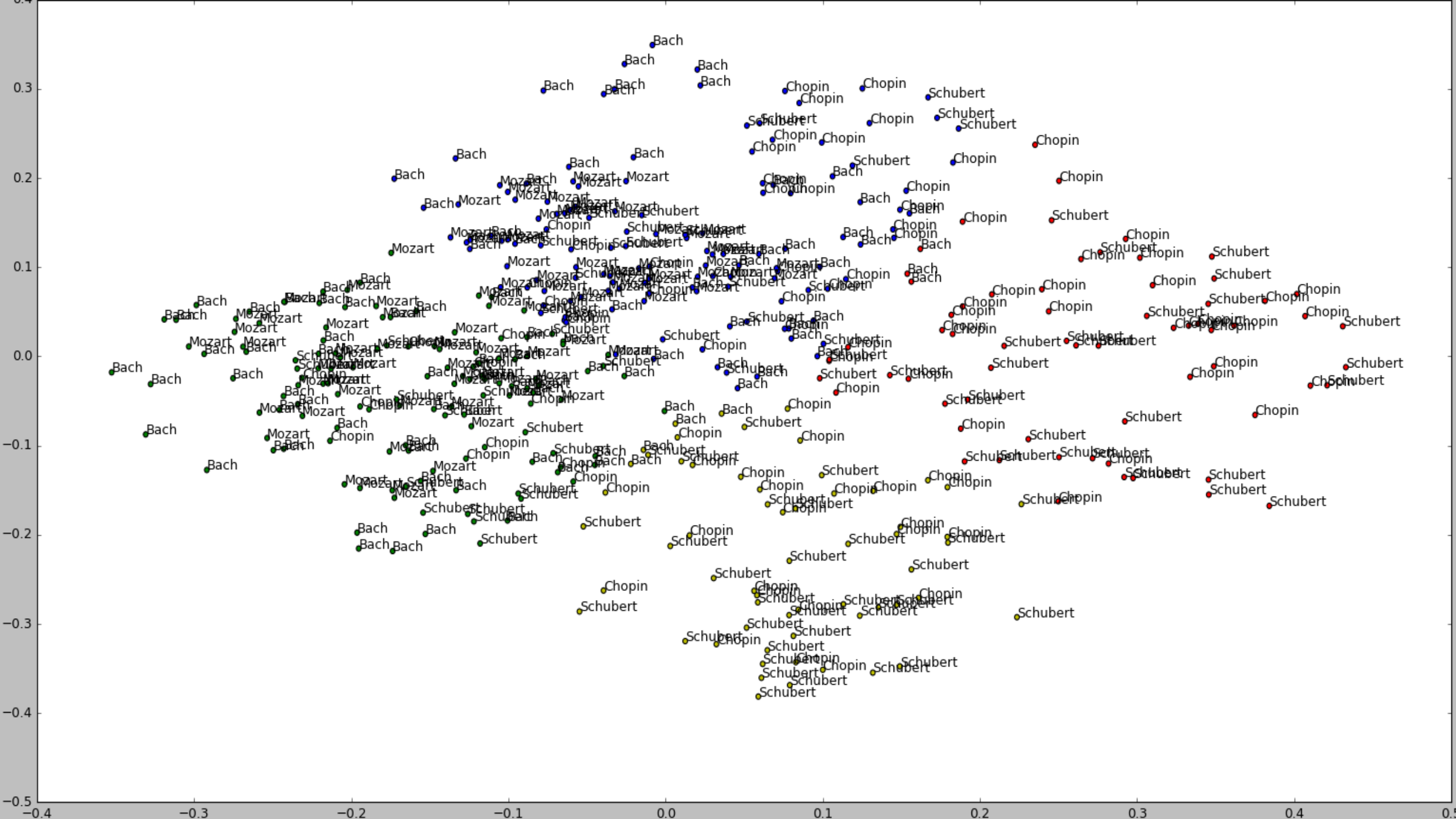
…FAKE DATA

Analysis time varies wildly per piece, taking anywhere from 10 sec to 2 hours

Not feasible at the moment

Generate fake data using ranges obtained from analysis of 20-30 pieces per composer

Weak, as it doesn’t take into account correlation between features


0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.4
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
0.5

CONCLUSIONS & FUTURE STEPS

Works better than expected

Need better information on which features are useful

Real data, and at least 150-200 works per composer

More composers

Test with other genres of music

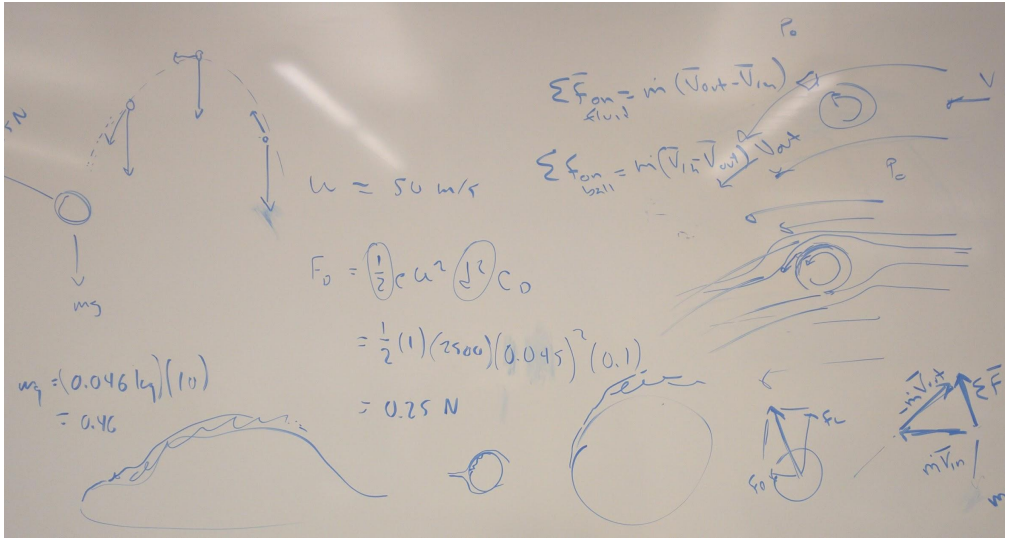
Use of better features, ask music professors

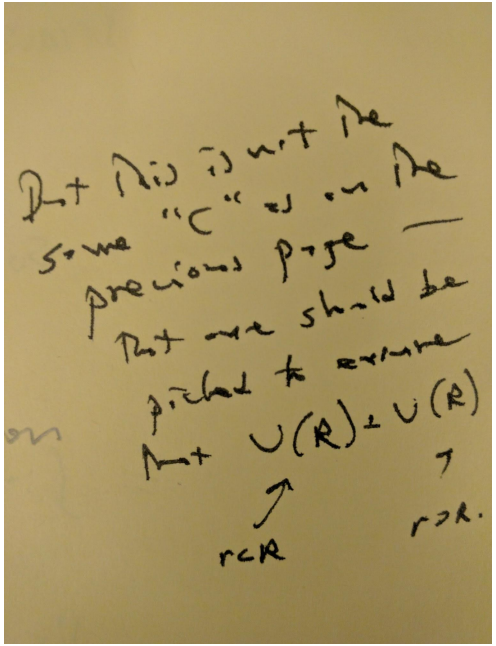
Test with more methods

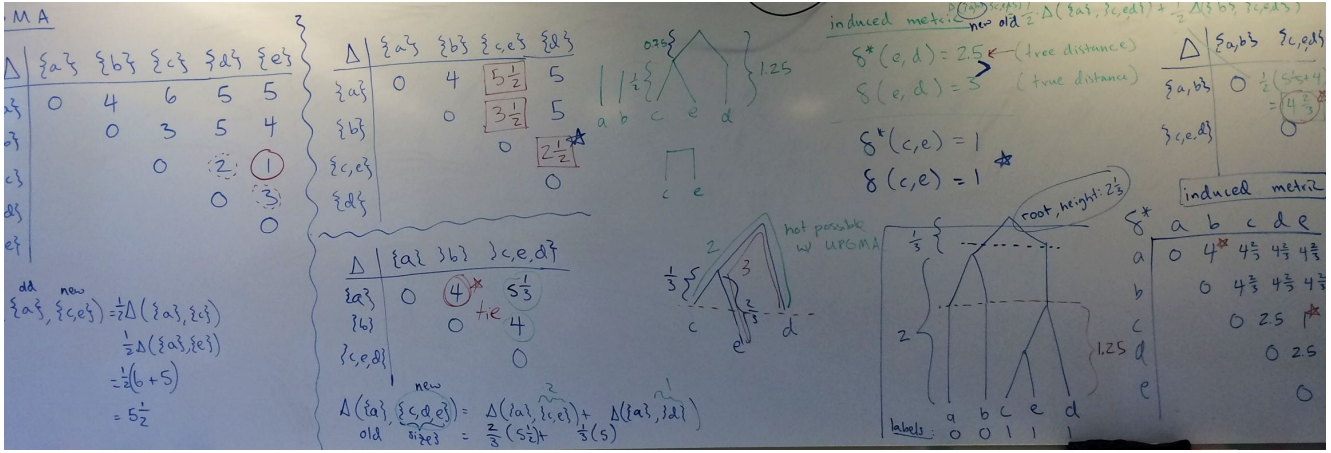
Character Recognition In the Wild

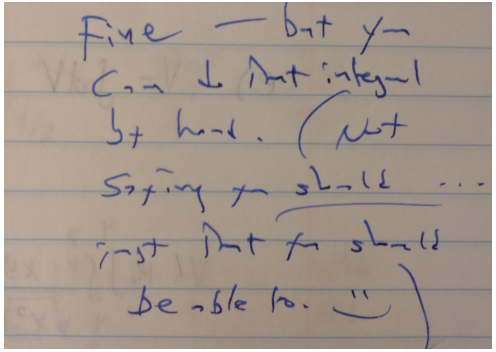
CSC 390 | Farida Sabry


ARRÊT


u ≈ 50 m/s
mg
mg = (0.046 kg)(10)
= 0.46
F_D = (1/2) ρ u² A C_D
= 1/2 (1)(2500)(0.045)²(0.1)
= 0.25 N


But this is not the same "C" as on the previous page —
But we should be picked to assume
that U(R) = U(R)
r<R
r>R.


UPGMA
induced metric
δ*(c,e) = 1
δ(c,e) = 1
not possible w/ UPGMA
root, height 2 1/3
labels


Fine — but you can do that integral by hand. (Not saying you should ... just that you should be able to. :)

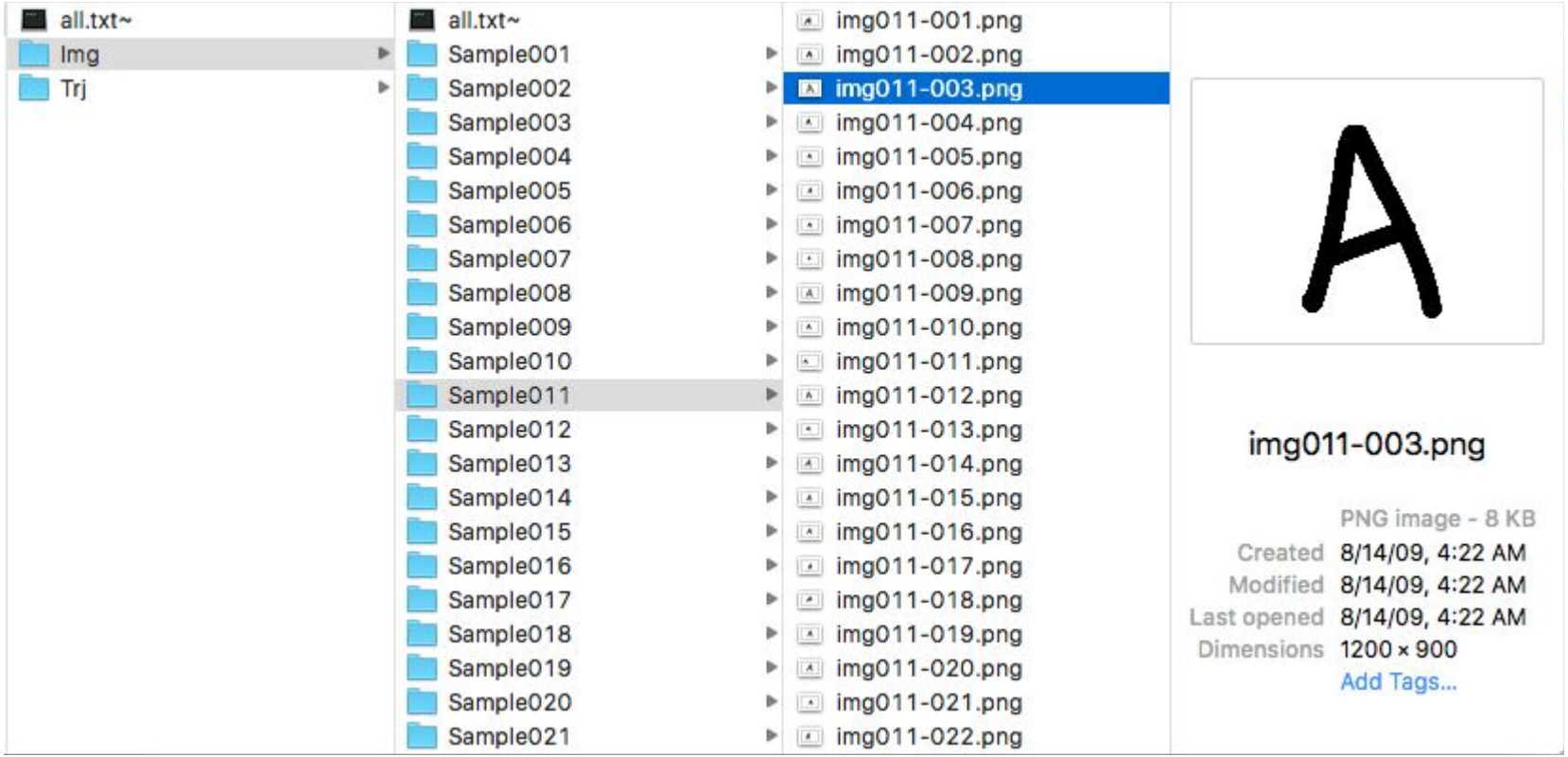
Motivation

- translating signs on the fly
- deciphering bad handwriting
- digitizing notes
- quick lookup

Chars**74K** Dataset

62 classes (a-z A-Z 0-9)
55 samples per class

3410 total samples


all.txt~
Img
Trj
all.txt~
Sample001
Sample002
Sample003
Sample004
Sample005
Sample006
Sample007
Sample008
Sample009
Sample010
Sample011
Sample012
Sample013
Sample014
Sample015
Sample016
Sample017
Sample018
Sample019
Sample020
Sample021
img011-001.png
img011-002.png
img011-003.png
img011-004.png
img011-005.png
img011-006.png
img011-007.png
img011-008.png
img011-009.png
img011-010.png
img011-011.png
img011-012.png
img011-013.png
img011-014.png
img011-015.png
img011-016.png
img011-017.png
img011-018.png
img011-019.png
img011-020.png
img011-021.png
img011-022.png
A
img011-003.png
PNG image - 8 KB
Created 8/14/09, 4:22 AM
Modified 8/14/09, 4:22 AM
Last opened 8/14/09, 4:22 AM
Dimensions 1200 × 900
Add Tags...

Pre-processing

Images were **1200 x 900**
compressed them to **28 x 28**
& converted each character
to a normalized **pixel array**
using **Pillow**

0

Classification

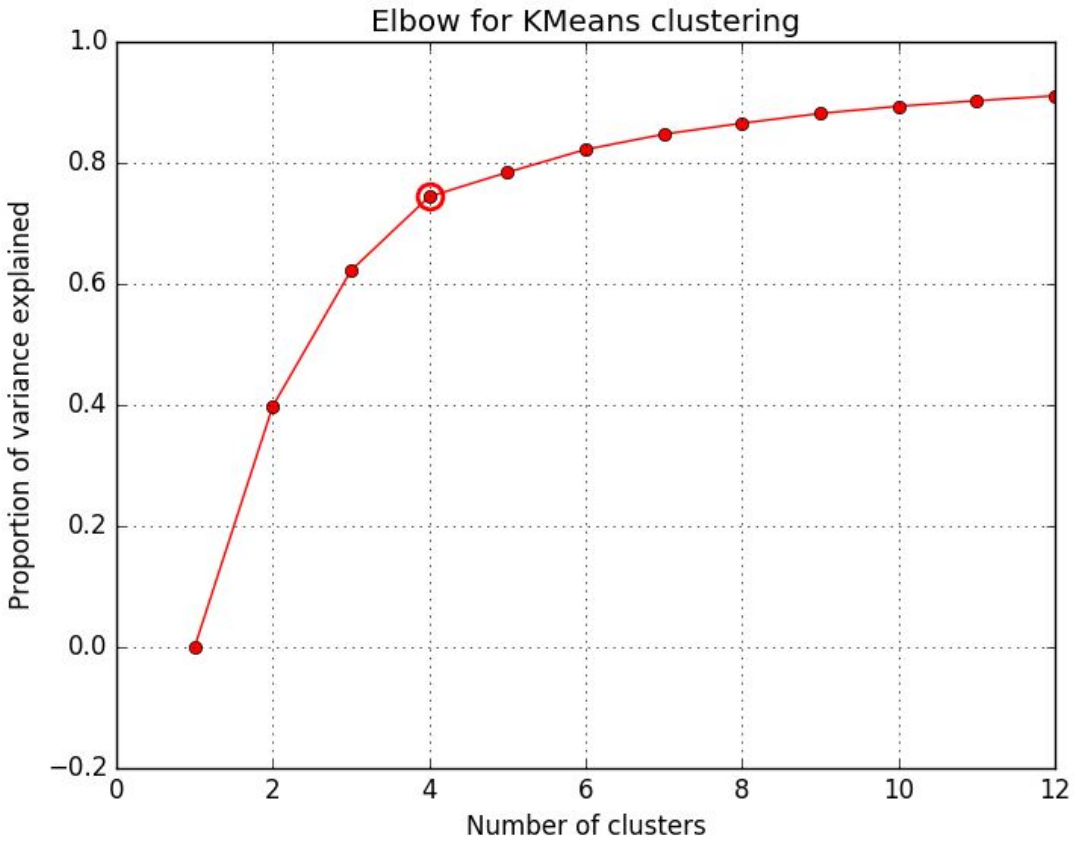
1. Unsupervised

    - PCA
    - K-means

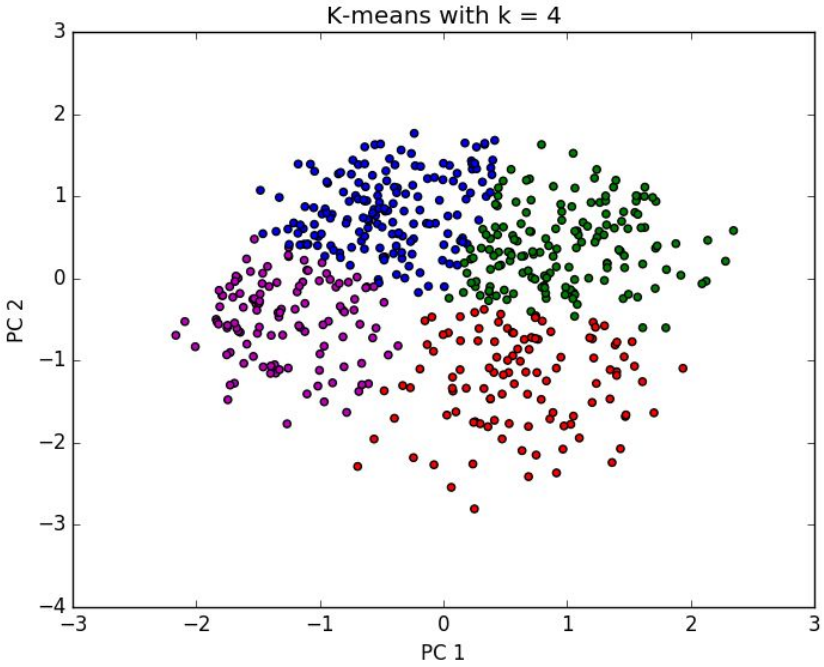
2. Supervised

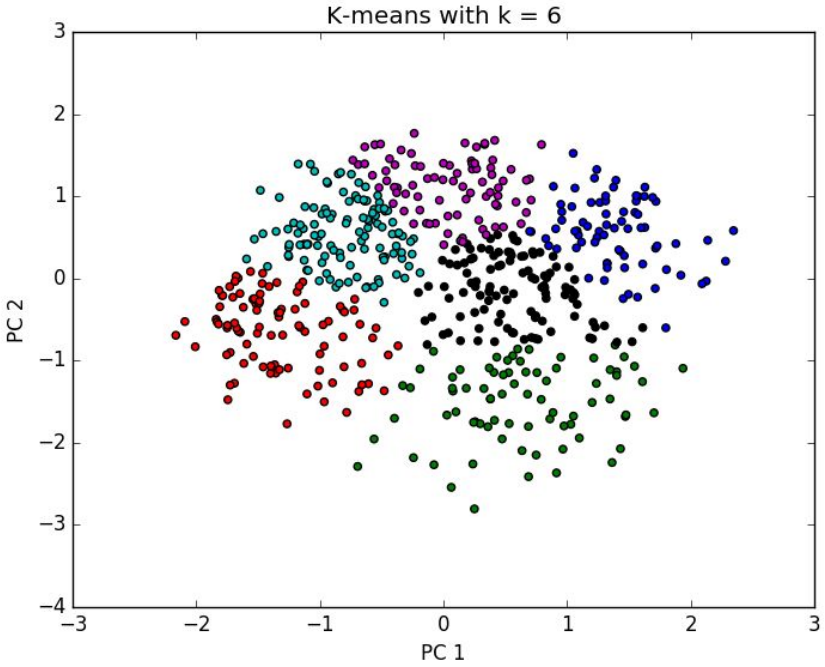
    - CNN using TensorFlow

K-means

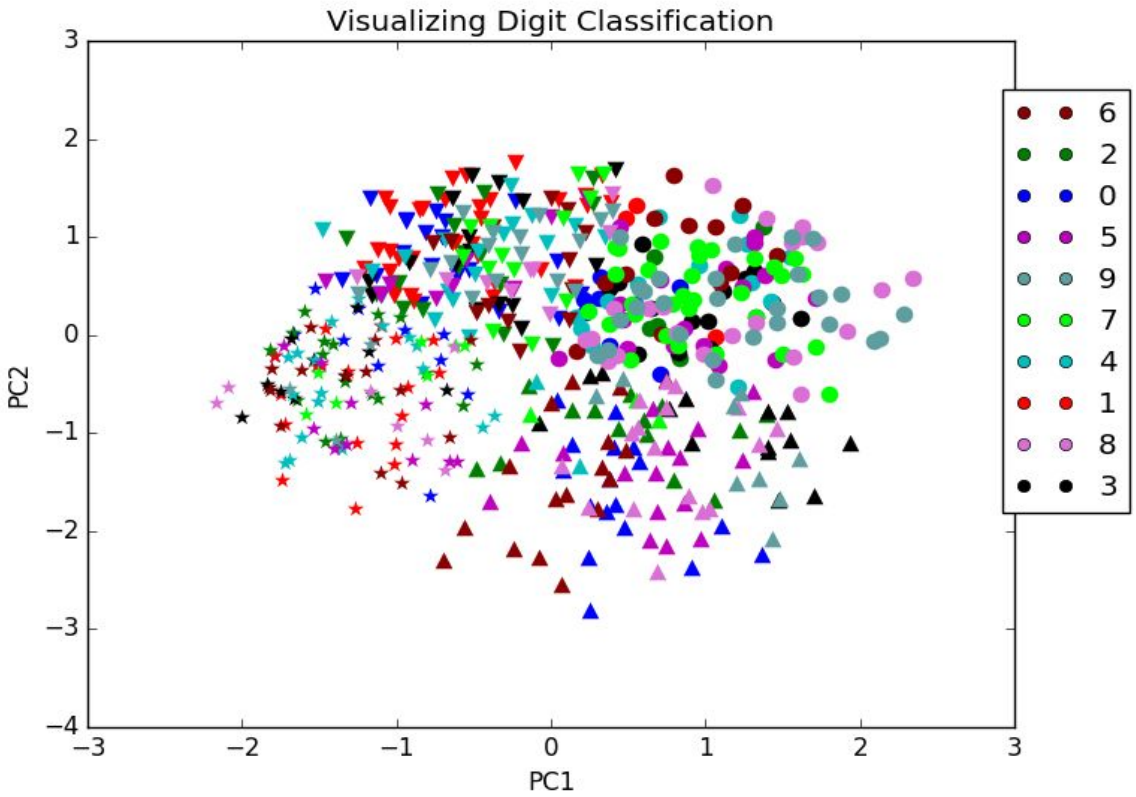

Elbow for KMeans clustering
Proportion of variance explained
Number of clusters
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
2
4
6
8
10
12

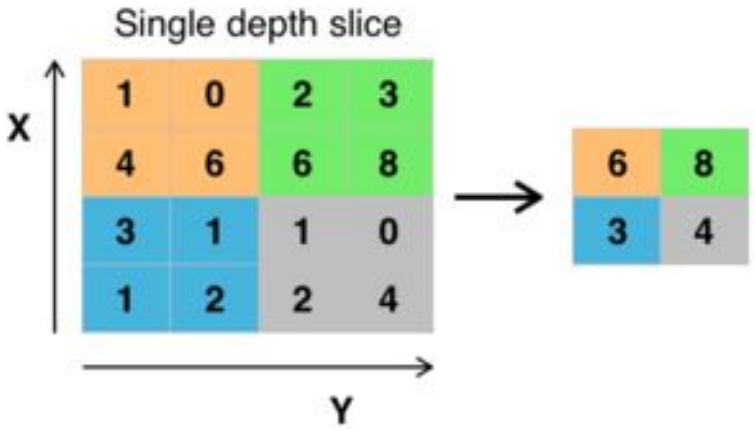
Results

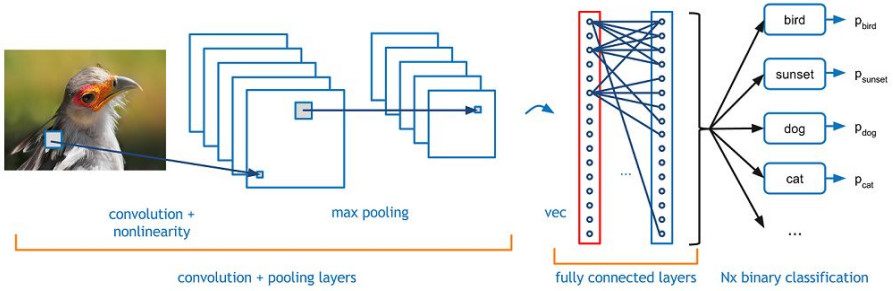

K-means with k = 4
PC 2
PC 1
3
2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3


K-means with k = 6
PC 2
PC 1
3
2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3

Comparing Accuracy







Convolutional Neural Networks

Modern CNNs are traditionally made of alternating convolution & max-pooling layers followed by a small number of connected layers

** used the MNIST data to train

Building the TensorFlow Model

```
def main():
    # initialize interactive session
    sess = tf.InteractiveSession()

    # placeholders
    x  = tf.placeholder(tf.float32, shape=[None, 784])
    y_ = tf.placeholder(tf.float32, shape=[None, 10])

    # variables
    W = tf.Variable(tf.zeros([784,10]))
    b = tf.Variable(tf.zeros([10]))

    # prediction
    y = tf.nn.softmax(tf.matmul(x,W) + b)
```

```
step 17500, training accuracy 0.96
step 17600, training accuracy 0.98
step 17700, training accuracy 1
step 17800, training accuracy 0.98
step 17900, training accuracy 1
step 18000, training accuracy 0.98
step 18100, training accuracy 1
step 18200, training accuracy 1
step 18300, training accuracy 1
step 18400, training accuracy 1
step 18500, training accuracy 0.98
step 18600, training accuracy 0.98
step 18700, training accuracy 1
step 18800, training accuracy 0.98
step 18900, training accuracy 1
step 19000, training accuracy 1
step 19100, training accuracy 1
step 19200, training accuracy 1
step 19300, training accuracy 1
step 19400, training accuracy 1
step 19500, training accuracy 1
step 19600, training accuracy 0.98
step 19700, training accuracy 1
step 19800, training accuracy 1
step 19900, training accuracy 1
Model saved in file:  model.ckpt

[Restored Dec 15, 2016, 2:09:46 AM]
Last login: Thu Dec 15 02:09:36 on console
```

```python
#---- CONVOLUTIONAL NEURAL NETWORK ----#

# First convolutional Layer
W_conv1 = weight_variable([5, 5, 1, 32])
b_conv1 = bias_variable([32])

# re-shape x to a 4d tensor
x_image = tf.reshape(x, [-1, 28, 28, 1])
h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
h_pool1 = max_pool_2x2(h_conv1)

# Second Convolutional Layer
W_conv2 = weight_variable([5, 5, 32, 64])
b_conv2 = bias_variable([64])

h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)

# Densely Connected Layer
W_fc1 = weight_variable([7 * 7 * 64, 1024])
b_fc1 = bias_variable([1024])
```

Saver Class

```
saver = tf.train.Saver()
sess.run(tf.global_variables_initializer())
```

```
save_path = saver.save(sess, "model.ckpt")
print("Model saved in file: ", save_path)
```

```
with tf.Session() as sess:
    sess.run(init_op)
    saver.restore(sess, "model.ckpt.data")
    prediction = tf.argmax(y_conv,1)
    return prediction.eval(feed_dict={x: [imvalue],keep_prob: 1.0},
        session=sess)
```

Challenges

- I spent a lot of time trying to get my images into a **usable format**
- My laptop has **zero computing power** and basically no RAM so I encountered a lot of 'freezing'
- TensorFlow is a little **confusing** and I'm still trying to figure it out

Future Work

Now and Later...

1. Figure out how to retrieve my model and re-use it
2. Have my model predict numbers on the fly
3. **Ambitious:** image segmentation & patches

References

1. T. E. de Campos, B. R. Babu and M. Varma. **Character recognition in natural images.** In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP), Lisbon, Portugal*, February 2009.

2. LeCunn, Yann, Corinna Cortes, and Christopher J.C. Burges. "THE MNIST DATABASE." MNIST Handwritten Digit Database*, Yann LeCun, Corinna Cortes and Chris Burges*. N.p., n.d. Web. 15 Dec. 2016.

3. "TensorFlow." *TensorFlow*. N.p., n.d. Web. 15 Dec. 2016.

Thank you!

Questions? Suggestions?