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

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0.64</th>
<th>0.64</th>
<th>0</th>
<th>...</th>
<th>3.756</th>
<th>61</th>
<th>278</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.21</td>
<td>0.28</td>
<td>0.5</td>
<td>0</td>
<td>...</td>
<td>5.114</td>
<td>101</td>
<td>1028</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.96</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>1.147</td>
<td>5</td>
<td>78</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0.65</td>
<td>0</td>
<td>...</td>
<td>1.25</td>
<td>5</td>
<td>40</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Methods

- K-means & PCA

![Elbow Plot for K-means dataset (transformed)](image)
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

train accuracy: 0.634496250408
test accuracy: 0.631029986962

LINEAR REGRESSION:
accuracy: 0.832464146023
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

<table>
<thead>
<tr>
<th></th>
<th>48</th>
<th>49</th>
<th>45</th>
<th>...</th>
<th>56</th>
<th>56</th>
<th>54</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>52</td>
<td>39</td>
<td>...</td>
<td>52</td>
<td>50</td>
<td>51</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>50</td>
<td>42</td>
<td>...</td>
<td>48</td>
<td>53</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>48</td>
<td>50</td>
<td>...</td>
<td>45</td>
<td>46</td>
<td>46</td>
</tr>
<tr>
<td>45</td>
<td>54</td>
<td>49</td>
<td>...</td>
<td>46</td>
<td>47</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>51</td>
<td>51</td>
<td>...</td>
<td>47</td>
<td>46</td>
<td>46</td>
<td></td>
</tr>
</tbody>
</table>

**Matrix to image:**
- import matplotlib.pyplot as plt
- PIL
Methods for Image Segmentation

Spectral clustering:
- With Otsu’s method
- With K-means
- With discretization

```python
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, th1, th2, blur, 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

```python
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 - 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

\[
\begin{align*}
total \text{ invest} & = 9000 \\
profit & = 158.629876201 \\
percentage \text{ profit} & = 1.76255418001 \%
\end{align*}
\]

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?
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

Chord

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)
Bad quality MIDI files do not consistently provide good data for note duration.

Lack of significant number of MIDI files 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 results.

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

Excellent documentation
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
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 0.2610966057412535 0.0 0.0 0.04267161410018553
0.0 0.5476190476190477 0.1058201682015801 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 0.061224489795918366
0.0 0.3225 0.0 0.0 0.08970588235294118
0.0 0.518324607329843 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
0.0069444444444444 0.5243055555555556 0.06597222222222222 0.0 0.1
0.0 0.548460661345496 0.06385404789053592 0.0 0.06447534766118837
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.
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
Motivation

- translating signs on the fly
- deciphering bad handwriting
- digitizing notes
- quick lookup
Chars74K Dataset

6  D  K  R  b
f  m  0  q  t

62 classes (a-z A-Z 0-9)
55 samples per class
3410 total samples
Pre-processing

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

1. Unsupervised
   - PCA
   - K-means

2. Supervised
   - CNN using TensorFlow
K-means
Results

K-means with $k = 4$

K-means with $k = 6$
Comparing Accuracy

Visualizing Digit Classification
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)
----- 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

```python
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


Thank you!

Questions? Suggestions?