milestone03_yujiaochen_brianho_jonjay_part00_intro

April 19, 2017

1 AC209b / CS109b Final Project - Milestone 3

Yujiao Chen, Brian Ho, Jonathan Jay // 04/19/2017

Traditional statistical and machine learning methods, due Wednesday, April 19, 2017

Think about how you would address the genre prediction problem with traditional statistical or machine learning methods. This includes everything you learned about modeling in this course before the deep learning part. Implement your ideas and compare different classifiers. Report your results and discuss what challenges you faced and how you overcame them. What works and what does not? If there are parts that do not work as expected, make sure to discuss briefly what you think is the cause and how you would address this if you would have more time and resources.

You do not necessarily need to use the movie posters for this step, but even without a background in computer vision, there are very simple features you can extract from the posters to help guide a traditional machine learning model. Think about the PCA lecture for example, or how to use clustering to extract color information. In addition to considering the movie posters it would be worthwhile to have a look at the metadata that IMDb provides.

You could use Spark and the ML library to build your model features from the data. This may be especially beneficial if you use additional data, e.g., in text form.

You also need to think about how you are going to evaluate your classifier. Which metrics or scores will you report to show how good the performance is?

The notebook to submit this week should at least include:

- Detailed description and implementation of two different models
- Description of your performance metrics
- Careful performance evaluations for both models
- Visualizations of the metrics for performance evaluation
- Discussion of the differences between the models, their strengths, weaknesses, etc.
- Discussion of the performances you achieved, and how you might be able to improve them in the future

1.0.1 Overview

This week, we applied traditional statistical and machine learning methods to the problem of genre prediction within our dataset. Using a database of 990 movies, with a third each classified as horror, romance or science ficiton, we trained and tested two genre classifiers based on differently engineered and extracted features. As a whole, the work comprises:

A model based on bag-of-word features from movie description

In the first model (seen below) we vectorize the overview feature (i.e. the movie summary) in a bag-of-words approach in conjunction with basic metadata from the TMDB API results. We began with a gradient-boosted tree approach utilizing XGBoost, which was approximately 73% accurate. As this performance was unsatisfactory, we then took a more sophisticated approach involving PCA dimensionality reduction.

In“milestone03_yujiaochen_brianho_jonjay_part01_word” and “milestone03_yujiaochen_brianho_jonjay_part02_PCA_SVM.pdf” we perform both EDA of the word features themselves and PCA to features that account for 90% of the variance. We then employ SVM with radial basis function to classify the horror, romance and scifi movies based on this reduced feature set. The final predicting accuracy on the test set using this model is around 80%, which is a satisfactory result.

A model based on title features

In “milestone03_yujiaochen_brianho_jonjay_part03_titles” we build upon our analysis using word count features, by attempting to add the features of movie titles. We find that it's possible to train a model to predict between two distant classes (romance and horror) using only simple features engineered from the title itself, such as character/word length, presence of grammatical symbols, and sentiment. We also attempt to extend the model to predict among three classes–first using title alone – as well as integrating our established model for predicting on description word counts, although resulting improvement is not satisfactory.

Exploratory data analysis and vizualization of movie posters

In “milestone03_yujiaochen_brianho_jonjay_part04_color.pdf” we perform initial analysis and exploration of our poster data set on a few sample posters, with a particular emphasis on color decomposition and analysis of frequencies.

In “milestone03_yujiaochen_brianho_jonjay_part05_poster.pdf” we perform exploratory analysis and visualization of the complete poster dataset, to understand properties within the data other than the posters themselves.

In [1]:
```
## Import libraries
import pandas as pd
import numpy as np

import imdb
import requests
from ast import literal_eval
from xgboost import XGBClassifier
from sklearn.cross_validation import StratifiedKFold
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import CountVectorizer
```

In [74]:
```
## Read in the data
movies = pd.read_csv("Movie subset for poster analysis_990 movies.csv")
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 990 entries, 0 to 989
Data columns (total 21 columns):
Unnamed: 0            990 non-null int64
```

```
X                    990 non-null int64
adult                990 non-null bool
backdrop_path        981 non-null object
genre_ids            990 non-null object
id                   990 non-null int64
original_language    990 non-null object
original_title       990 non-null object
overview             986 non-null object
popularity           990 non-null float64
poster_path          990 non-null object
release_date         990 non-null object
title                990 non-null object
video                990 non-null bool
vote_average         990 non-null float64
vote_count           990 non-null int64
genre_names          985 non-null object
date                 990 non-null object
year                 990 non-null int64
genres               990 non-null object
decade               990 non-null int64
dtypes: bool(2), float64(2), int64(6), object(11)
memory usage: 149.0+ KB
```

In [75]:
```
## Cleanup column names
movies.rename(columns={"Unnamed: 0":"result_id"}, inplace = True)
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 990 entries, 0 to 989
Data columns (total 21 columns):
result_id            990 non-null int64
X                    990 non-null int64
adult                990 non-null bool
backdrop_path        981 non-null object
genre_ids            990 non-null object
id                   990 non-null int64
original_language    990 non-null object
original_title       990 non-null object
overview             986 non-null object
popularity           990 non-null float64
poster_path          990 non-null object
release_date         990 non-null object
title                990 non-null object
video                990 non-null bool
vote_average         990 non-null float64
vote_count           990 non-null int64
genre_names          985 non-null object
```

```
date                    990 non-null object
year                    990 non-null int64
genres                  990 non-null object
decade                  990 non-null int64
dtypes: bool(2), float64(2), int64(6), object(11)
memory usage: 149.0+ KB
```

In [76]:
```
## Filter out movies with invalid information
valid_overview = [type(i) is str for i in movies["overview"]]
movies = movies[valid_overview]
# valid_title_filter = [type(i) is str for i in movies["title"]]
# movies = movies[valid_title_filter]

movies = movies.reset_index()

movies.tail()
```

Out[76]:
```
     index  result_id   X  adult                     backdrop_path  \
981    985       8478  15  False  /4liSXBZZdURI0c1Id1zLJo6Z3Gu.jpg
982    986       8165   2  False  /cfVoH243KjWXV6JoLzwxqWNb23i.jpg
983    987       7964   1  False  /jxdSxqAFrdioKgXwgTs5Qfbazjq.jpg
984    988       8167   4  False  /cZkPJ0noQvcR3oCCZ4pwYZeWUYi.jpg
985    989       8380  17  False  /oZY3DOlEZbEZvRxWynWkFTe4UgE.jpg

                genre_ids      id original_language     original_title  \
981    [878, 14, 28, 12]   76757                en  Jupiter Ascending
982      [878, 12, 9648]   70981                en         Prometheus
983        [12, 28, 878]   10138                en         Iron Man 2
984        [28, 53, 878]   59967                en             Looper
985  [53, 878, 18, 9648]  157353                en      Transcendence

                                              overview   ...    release_da
981  In a universe where human genetic material is ...   ...      2015-02-
982  A team of explorers discover a clue to the ori...   ...      2012-05-
983  With the world now aware of his dual life as t...   ...      2010-04-
984  In the futuristic action thriller Looper, time...   ...      2012-09-
985  Two leading computer scientists work toward th...   ...      2014-04-

                 title  video vote_average vote_count  \
981  Jupiter Ascending  False          5.2       2206
982         Prometheus  False          6.2       4135
983         Iron Man 2  False          6.6       5601
984             Looper  False          6.6       4053
985      Transcendence  False          5.9       1861

                                      genre_names        date  year  \
981  [Science Fiction, Fantasy, Action, Adventure]  2015-02-04  2015
```

```
982          [Science Fiction, Adventure, Mystery]  2012-05-30  2012
983           [Adventure, Action, Science Fiction]  2010-04-28  2010
984            [Action, Thriller, Science Fiction]  2012-09-26  2012
985    [Thriller, Science Fiction, Drama, Mystery]  2014-04-16  2014

                  genres  decade
981    878, 14, 28, 12    2010
982      878, 12, 9648    2010
983        12, 28, 878    2010
984        28, 53, 878    2010
985  53, 878, 18, 9648    2010

[5 rows x 22 columns]
```

In [77]:
```
## Create bag-of-words feature representation from movie summaries
corpus = movies["overview"].tolist()
vectorizer = CountVectorizer(min_df=20, stop_words="english")
words = vectorizer.fit_transform(corpus)
print words.toarray().shape
print vectorizer.get_feature_names()

## Convert to data frame
words = pd.DataFrame(words.A, columns=vectorizer.get_feature_names())
```

```
(986, 149)
[u'alien', u'american', u'away', u'based', u'battle', u'beautiful', u'begin', u'beg
```

In [21]:
```
## Create predictors from metadata and
X = movies[[u'adult', u'id', u'popularity',
            u'year',
            u'vote_average', u'vote_count']]

X = X.join(words, how="left", rsuffix="_word")
```

In [22]:
```
## A function to add a label for the response variable (which genre withi
def classify(ids):
    if "10749" in ids:   # romance
        return 0
    elif "27" in ids:    # horror
        return 1
    elif "878" in ids:   # science fiction
        return 2

## Create response from complete genre labels
movies["label"] = movies.apply(lambda x: classify(x["genre_ids"]), axis=1)
Y = movies["label"].values
```

In [72]:
```
X.head()
```

Out[72]:
```
   adult     id  popularity  year  vote_average  vote_count  alien  americ
0  False    164    3.978771  1961           7.5         769      0
1  False    907    3.292234  1965           7.3         193      0
2  False    284    3.270765  1960           8.0         383      0
3  False  37247    3.257459  1967           7.5         637      0
4  False  11113    3.092740  1964           7.4         269      0

   away  based  ...    way  wife  woman  women  work  world  year_word  ye
0     0      0  ...      0     0      1      0     0      0          0
1     0      0  ...      1     0      0      0     0      0          0
2     0      0  ...      1     0      0      0     0      0          0
3     0      0  ...      0     1      0      0     0      0          0
4     0      0  ...      0     0      0      0     0      0          0

   york  young
0     0      0
1     0      0
2     1      0
3     0      0
4     0      0

[5 rows x 155 columns]
```

In [31]:
```
from sklearn.grid_search import GridSearchCV
```

In [68]:
```
## Set up K-fold cross validation
kf = StratifiedKFold(Y, n_folds=5, shuffle=True)

model = XGBClassifier()

n_estimators = [100, 150, 200, 300, 400]
max_depth = [1, 2, 4, 6]
learning_rate = [0.1, 0.2, 0.3, .4, .5]
param_grid = dict(max_depth=max_depth, n_estimators=n_estimators, learning

grid_search = GridSearchCV(model, param_grid, n_jobs=-1, cv=kf)

grid_result = grid_search.fit(X, Y)
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_par
```

```
Best: 0.743408 using {'n_estimators': 200, 'learning_rate': 0.3, 'max_depth': 2}
```

In [79]:
```
from sklearn.cross_validation import ShuffleSplit
from sklearn.metrics import confusion_matrix

rs = ShuffleSplit(len(X), n_iter=3, test_size=.25, random_state=0)
```

```python
for train, test in rs:
    train_x, test_x = X.iloc[train,:], X.iloc[test,:]
    train_y, test_y = Y[train], Y[test]

model = XGBClassifier(n_estimators=300, max_depth=2, learning_rate=0.3)
model.fit(train_x, train_y)

y_pred = model.predict(test_x)
print confusion_matrix(test_y, y_pred)
print accuracy_score(test_y, y_pred)
```

```
[[56 12 11]
 [15 48  8]
 [12 15 70]]
0.704453441296
```

The gradient-boosted tree model had an accuracy of 74% after 5-fold cross validation and parameter tuning. Based on the confusion matrix,it appears there were roughly equal rates of misclassification between labels. The poor rate of accuracy might be related to the relatively small size of the dataset and expansive age range — having only 990 movie across several decades could make classification more difficult.