

Predicting YouTube Comedy Slam Winners

STAT 222 Class 2013

Department of Statistics
University of California, Berkeley

7 May 2013

YouTube Comedy Slam Data

- from UCI ML Repository; donated by Google employee
- a trial is an ordered pair of video IDs (v_i, v_j)
- data: ordered pair + “left” or “right” found funnier by viewer
- order of the pair in each trial was random
- repository has training data and test data
- YouTube has metadata about videos
- no data about viewers

Goal

Predict which video in a pair will be funnier, from metadata

(Is it OK to use the video IDs in the prediction?)

Tools

- Github
- IPython notebook
- numpy, scipy, matplotlib, nltk, scikit-learn

Descriptive statistics of training data

- 912,969 records
- 18,474 distinct video IDs
- 267,211 distinct video ID pairs
- 359,874 distinct ordered pairs of video IDs
- right video won 51.77% of the time. P -value: nil
- judgments often discordant:
e.g., '-iuk0PbfaHY wDx28Y2RcCl' left and right each “funnier”
119 times
- accuracy of ideal classifier for training data: 73.449% (includes videos with no comments)
- for individual videos, directed graph of “funnier than”
- can summarize that graph by PageRank

PageRank

- Assign each video ID (node) in the graph a numerical value between 0 to 1, known as its *PageRank*.

| |
|--|
| At $t = 0$, $PR(p_i; 0) = \frac{1}{N}$, N is the total number of nodes. |
| $PR(p_i; t + 1) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j; t)}{L(p_j)}$ |
| Algorithm ends when $ PR(t + 1) - PR(t) \leq \epsilon$ |
| d is a damping factor, default 0.85 in scikit-learn. |

Prediction using metadata: Feature selection

- Acquired metadata: queried YouTube with video IDs using Google APIs w/i Python
- Compute mutual information:

| Variable | Mutual Info (bits) |
|-----------------|--------------------|
| average rating | 0.00227 |
| number of views | 0.00431 |
| number of votes | 0.00468 |
| views per day | 0.00382 |

- Logistic regression of log score of average rating, number of views and number of votes
- Comments more promising

Issues with comments

- Data Encoding: encoded the words to ASCII format and omit the garbled words
- Inconsistent Spellings
 - Correct Spelling: tried Norvig's spelling corrector, ported to AWS.
(e.g.: "spelng" → "spelling")
drawback: not accurate on this corpus
(e.g.: "youtube" → "couture", "lol" → "ll", "haha" → "hata")
 - Text-speak: used RegExp to standardize words

| Patterns | As |
|----------------------------|-----|
| lol, lolll, llol, lollloll | lol |
| ha, hahahh, ahaha, jhajha | ha |

- Stemming: used Porter Stemmer from NLTK package
- Addressing Emoticons: used RegExps to replace happy faces (e.g., ":-]") with "happyface" before stripping other punctuation

Bags of Bags of Words

- ordered pair of videos reduced to two “bags of bags of words”
each comment is a bag, each video has a bag of bags
- features derived from bags of bags:
 - presence of a word among video ids
 - frequency of comments with a given word
 - relative frequency of comments with a given word
 - logOdds of frequencies and relative frequencies
- another derived feature: PageRank predicted by linear regression
using (among other things) $\arctan(\text{difference in LOL counts})/(\pi/2)$

Models

Logistic regression with Log Bayes Factor

Binary Output (Left v.s. Right) as response variable
log bayes factor of the two ordered ids and a constant term as features

Logistic regression Wenchang

Explain

CART

Explain

Page Rank

Binary Output and "LOL" Count as link's weight indicators
Each node represents a video, if a video A is funnier than B, then a directed link with weight = $\# \text{ of A funnier than B} + \arctan(\# \text{ of lol(A-B)})/(\pi/2)$
(Implemented arctan to shrink an integer to (0, 1), small modification)

Features for Logistic Regression 1

Log Bayes Factors

- 1 determine whether video v_i won more than it lost, or vice versa.
if v_i won more than lost, it's a "winner"
if v_i lost more than won, it's a "loser"

- 2 for each word w , find:

$\mathbb{P}_1(w)$ = percentage of winner comments that contain w

$\mathbb{P}_0(w)$ = percentage of loser comments that contain w

(pad to avoid zeros)

(Only used comments of "significant" winners/losers)

- 3 derive new feature:

$$B(v) = \logOdds(v) = \sum_w m_w \log\left(\frac{\mathbb{P}_1(w)}{\mathbb{P}_0(w)}\right) + (n - m_w) \log\left(\frac{1 - \mathbb{P}_1(w)}{1 - \mathbb{P}_0(w)}\right)$$

n : number of comments on video v

m_w : number of comments on video v containing w .

Logistic Regression 1 with Log Bayes Factors

$$\text{logit}(\mathbb{P}) = X\beta$$

- \mathbb{P} = Probability that the right video is funnier



$$X = \begin{pmatrix} 1 & B(v_{11}) & B(v_{12}) \\ \vdots & \ddots & \vdots \\ 1 & B(v_{n1}) & B(v_{n1}) \end{pmatrix}$$

Logistic regression 2 Data preprocessing

- Clean comments: only keep meaningful words.
`nltk.tokenize`, `nltk.corpus`
- Build Dictionary: order words by frequency. (Top frequent words: like, love, lol,funni, video)
Structure of dictionary, `dict()` and Set, `set()`
- Building X matrix blockwise: divide 20000 rows as a block and save all the blocks to disk.

$$X = \begin{pmatrix} X_1 \\ X_2 \\ \dots \\ X_B \end{pmatrix}$$

logistic regression 2

- Model: treating high frequency (≥ 1500) words as features

$$\text{Logit } Y = \beta_0 + \beta_1 x_1 + \dots + \beta_m x_m = \beta^T X$$

- *Logit* Y : log odds of each ID pair, i.e. $\log\left(\frac{\text{No. left funnier}}{\text{No. right funnier}}\right)$
- x_i : difference of appearance of word i in each ID pair.
- Size of matrix X : 359874×3392

$$X^T X = \sum_{b=1}^{\text{Block No.}} X_b^T X_b$$

$$X^T Y = \sum_{b=1}^{\text{Block No.}} X_b^T Y_b$$

Regression Tree

- Model

$$f(x) = \sum_{k=1}^K c_k I(x \in R_k)$$

R_k : partition of feature space. (still treat high frequency words as features)

- By minimizing MSE, we get

$$\hat{c}_k = \text{ave}(y_i | x_i \in R_k)$$

- How to find region R_k ?

Regression Tree

- Greedy algorithm to find partitions

$$R_1(j, s) = \{X | x_j \leq s\} \text{ and } R_2(j, s) = \{X | x_j > s\}$$

Seek the splitting variable j and split point s by solving

$$\min_{j, s} [\min_{c_1} \sum_{x_i \in R_1(j, s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j, s)} (y_i - c_2)^2]$$

Inner minimization is solved by

$$\hat{c}_1 = \text{ave}(y_i | x_i \in R_1(j, s)), \quad \hat{c}_2 = \text{ave}(y_i | x_i \in R_2(j, s))$$

Scan through all of the inputs to determinate the best pair (j, s) .
Repeat the splitting process on each of the two regions.

- Python package: *mipy*.

Performance on the training set

Common criterion to measure performances

$$\text{score} = \frac{\text{Number of Correct Prediction}}{\text{Number of Predictable Pairs}}$$

- Logistic with bayes odds: 53.3%

The Test data

- 225,593 records
- 75,447 distinct ordered pairs of videos
- Accuracy of the ideal classifier: 0.6946 (excluding pairs with no comments)
- right video won: 51.62% of the time

Performance on test data

- Bayes classifier from training data applied to test data: 52.3%
- logistic regression: 52.17%
- CART: 52.79%

Conclusions

- Quantitative metadata: not informative
- Comments:
 - Logistic Regression with Bayes Classifier
 - Logistic Regression with LogOdds
 - CART
 - PageRank
- Results
- Goal achieved?

Improvements

- Make better use of GitHub
- Having data that are of higher quality
- Incorporate personal preference in the dataset

Acknowledgement

Thanks to Philip, David and Aaron
for the valuable help!