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: (v_i, v_j) + "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

(Not OK to use video IDs as features.)

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 51.8% of the time. P-value: nil
- judgments often discordant: '-iuk0PbfaHY wDx28Y2RcCI' left and right each "funnier" 119 times
- accuracy of ideal classifier for training data: 73.4% (includes videos with no comments)
- if order is ignored, accuracy of ideal classifier drops to 65%

PageRank

- for individual videos, directed graph of "funnier than"
- can summarize that graph by PageRank
- PageRank algorithm:
  Assign each video ID (graph node) a 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.
$PR(p_i; t+1) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j;t)}{L(p_j)}$
Stop when $\lvert PR(t+1) - PR(t) \rvert \leq \epsilon$
d is a damping factor, default 0.85 in scikit-learn.

- Amounts to using the power method to estimate the top eigenvector of incidence matrix

Prediction using metadata: Feature selection

- Acquired metadata: queried YouTube with video IDs using Google APIs w/i Python
- 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 using log(average rating), # views, # votes had negligible "lift"
- Comments more promising

Comments: complications and cleaning

- Data Encoding: re-encoded comments in ASCII; omit garbled words
- Inconsistent Spellings
  - Spelling: tried Norvig's spelling corrector, ported to AWS.
    (e.g.: "speling" $\rightarrow$ "spelling")
    drawback: not accurate on this corpus
    (e.g.: "youtube" $\rightarrow$ "couture", "lol" $\rightarrow$ "ll", "haha" $\rightarrow$ "hata")
  - Text-speak: used RegExp to standardize words

Pattern	standard
lol, lolll, llol, lollololl	lol
ha, hahahh, ahaha, jhajha	ha

  - Emoticons: used RegExps to replace happy faces (e.g., ":-]") with "happyface" before stripping other punctuation
- Stemmed with nltk PorterStemmer; kept only meaningful words
- Most frequent words: like, love, lol, funni, video

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 in any comment for video ID
  - (relative) frequency of comments that contain a given word
  - logOdds of frequencies and relative frequencies
- derived feature: PageRank predicted by linear regression
  using (among other things) arctan(difference in LOL counts)/($\pi/2$)

Logistic Regression: log Bayes factor as feature

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 in comment of "significant" winner/loser, 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)

3. derive new feature:
   $logOdds(v) = \sum_w m_w \log(\frac{\mathbb{P}_1(w)}{\mathbb{P}_0(w)}) + (n - m_w) \log(\frac{1-\mathbb{P}_1(w)}{1-\mathbb{P}_0(w)})$

   n: #comments on video v
   m_w: #comments on video v containing w.

Linear regression

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

$$Logit\ Y = \beta_0 + \beta_1 x_1 + ... + \beta_m x_m = \beta^T X$$

- $Logit\ Y$: log odds of each ID pair, i.e $log(\frac{\#\ left\ \ funnier}{\#\ right\ \ funnier})$
- x_i: difference in #comments containing w_i for v_{left} and v_{right}
- Build X in B blocks of $20,000$ rows; save all the blocks to disk.
- Dimension of X: 359,874 $\times$ 3392

$$X^T X = \sum_{b=1}^{B} X_b^T X_b$$

$$X^T Y = \sum_{b=1}^{B} X_b^T Y_b$$

PageRank Linear regression classifier

- Predict continuous PageRank of video IDs from word frequencies by linear regression
- Classify pair as "right" if predicted PageRank of right video is higher than left
- Classify pair as "left" if predicted PageRank of left video is higher than right

CART

- Features: high frequency words
- Model

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

- R_k: partition of feature space.
- $c_k \in \{0, 1\}$.
- How to find region R_k?

CART–FIX ME!

- 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

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

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: *mlpy*.

Test data

- 225,593 records
- 75,447 distinct ordered pairs of videos
- right video won 51.6% of the time

Performance on test data

$$\text{Accuracy: } \frac{\text{correct predictions}}{\text{pairs with comments}}$$

- Accuracy of ideal (Bayes) classifier: 69.5%
- Accuracy of Bayes classifier w/o order info: 65%
- logistic regression: 52.3%
- linear regression: 52.17%
- PageRank linear regression classifier: 51.3%
- CART: 52.79% [FIX ME!]

Conclusions

- Quantitative metadata: not informative
- Comments:
  - heavy cleanup—must look at data by hand to understand problems
  - treat comments as bags of bags of words
  - derive features from the bags
  - logistic regression
  - linear regression
  - PageRank linear regression classifier
  - CART
- Results not encouraging: problem is HARD
- Humor very personal
- Even the order of presentation matters

Improvements

- Make better use of GitHub
- Ability to group/model individual raters might help, a la Netflix

Acknowledgement

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