Predicting Popularity of YouTube Video

Loc Do
Heinz College, Carnegie Mellon University
Pittsburgh PA, 15213
halocd@andrew.cmu.edu

Joseph Richardson
School of Computer Science, Carnegie Mellon University
Pittsburgh PA, 15213
jmrichar@andrew.cmu.edu

Abstract

In this work we address the problem of ranking two objects given their features. The problem is applied to a context of guessing the more popular one out of two given YouTube videos. We formulate this problem into two different approaches, namely, binary classification and linear regression. Experiments are conducted in our data set crawled from YouTube for one month. Results show that both approaches yields a slightly above 50% accuracy.

1 Introduction

YouTube is one of the most popular video-sharing online platform in the Internet. It attracts billion of unique user visit monthly and has diverse topics in their video content. YouTube users can earn money from the number of views of their uploaded videos. Hence, understanding the secrets of making a popular YouTube video is essential for people who want to make benefits from the site. There are many factors to determine popularity of a video, but in general we can pin down to have the following two: having good content and good marketing plan. One of the techniques to attract more viewerships is to make the video's "visual appearance" look appealing such as catchy keywords, informative cover picture, etc. Our approach is to utilise a video's metadata to predict their popularity.

The ranking problem (aka. Learning to rank[1]) is a typical supervised learning problem of predicting the rank of a set of items regarding to a set of criteria. It has a numerous applications in a broad domains such as web search, multimedia retrieval, recommender systems, etc. Results from these such problems can bring benefits to Internet users such as saving their time by introducing the most relevant products/articles/web pages to their interests. In YouTube context, ranking problems can be raised to recommend most relevant videos to a given video. Another application is to predict the ranking of videos regarding to their popularity. This can reveal the most important visual factors in determine popularity of a video.

Problem statement. Given a set of videos, each is associated with a set of bag-of-word and numeric features. A pair of videos is said to have an order on their popularity by comparing their number of views. Video with more viewership is considered to be more popular than the other. Given two videos with their features, the video ranking problem is to construct a model to accurately predict which one has more number of views.

[1] http://en.wikipedia.org/wiki/Learning_to_rank

2 Ranking by Classification

Our goal is to construct a prediction rule f that can rank the videos w.r.t. their popularity using their meta-data as feature inputs. Since there are only two outcomes in ranking two videos, we choose to model the output of f as a binary class label. Hence, we can formulate the ranking problem as a binary classification problem as in Section 2.1. This approach poses two problems. First, it does not take into account the magnitude in the ranking, meaning there is no difference between pairs of videos which are significantly different in their viewerships and pairs which are just slightly different. We address this problem by borrowing ideas of re-weighted training set from the Boosting technique in Section 2.2. Second, video viewership also depends on the video's "age", i.e. number of days passed since it was uploaded to YouTube to the day it was crawled. It is comparable to rank popularity of a video uploaded years ago with recent ones. Therefore, we bin videos according to their "age", and learn a distinct classifier f for each group. We also construct an ensemble of these time-specific classifiers in order to enhance the overall predictive performance. All are described in Section 2.3.

2.1 Problem Formulation

We can reformulate the ranking prediction problem between two videos as a binary classification problem. To be specific, let $X_i \in \mathbb{R}^D$ and $X_j \in \mathbb{R}^D$ be feature vectors of two videos i and j correspondingly. Each video pair (i, j) is associated with a binary label Y_{ij} defined as follows

$$Y_{ij} = \begin{cases} 1, & \text{if \#\_of\_views\_i} \geq \text{\#\_of\_views\_j} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

We can form a representative vector of the pair (i, j) as follows

$$\mathcal{X}_{ij} = k(X_i, X_j), \tag{2}$$

where $k : (\mathcal{R}^D, \mathcal{R}^D) \rightarrow \mathcal{R}^{D'}$ is a feature transformation function. There are several options for k.

- Difference between two feature vectors: $\mathcal{X}_{ij} = X_i - X_j$
- Concatenation of two feature vectors: $\mathcal{X}_{ij} = [X_i, X_j]$ (Matlab notation)
- Kernel functions, e.g. $\mathcal{X}_{ij} = ||X_i - X_j||^2$

At the moment, we cannot find any theories/signals to indicate which form of k is the most appropriate. For the scope of the project, we choose to represent X_{ij} as difference between X_i and X_j, meaning $D = D'$. The kernelized version is left in future work.

The function form of classifier $P(Y_{ij}|\mathcal{X}_{ij}, \mathbf{w})$ as follows

$$P(Y_{ij} = 1|\mathcal{X}_{ij}, \mathbf{w}) = \frac{1}{1 + \exp(w_0 + \sum_d w_d \mathcal{X}_{ij}^d)} = \frac{1}{1 + \exp(\mathbf{w}^T \mathcal{X}_{ij})} \tag{3}$$

The model parameters $\mathbf{w} \in \mathbb{R}^D$ can be learnt using MAP.

$$\begin{aligned} \hat{\mathbf{w}}_{MAP} &= \arg\max_{\mathbf{w}} \prod_{(i,j)} P(Y_{ij}|\mathcal{X}_{ij}, \mathbf{w}) P(\mathbf{w}) \qquad (P(\mathbf{w}) \sim \mathcal{N}(0, \tau^2 I)) \\ &= \arg\max_{\mathbf{w}} \prod_{\{(i,j)|Y_{ij}=1\}} P(Y_{ij}|\mathcal{X}_{ij}, \mathbf{w}) P(\mathbf{w}) \qquad (Y_{ij} + Y_{ji} = 1, \mathcal{X}_{ij} = -\mathcal{X}_{ji}) \\ &= \arg\max_{\mathbf{w}} \sum_{\{(i,j)|Y_{ij}=1\}} lnP(Y_{ij}|\mathcal{X}_{ij}, \mathbf{w}) + lnP(\mathbf{w}) \\ &= \arg\max_{\mathbf{w}} \sum_{\{(i,j)|Y_{ij}=1\}} ((1 - Y_{ij})\mathbf{w}^T \mathcal{X}_{ij} - ln(1 + \exp(\mathbf{w}^T \mathcal{X}_{ij}))) - \lambda_w \|\mathbf{w}\|_2^2 = l(\mathbf{w}) \end{aligned} \tag{4}$$

We can optimise Equation 4 (a concave function) using Gradient Ascent algorithm with the following update rule

$$w_d^{t+1} \leftarrow w_d^t + \eta \frac{\partial l(\mathbf{w})}{\partial w_d} = w_d^t + \eta \Big(\sum_{\{(i,j)|Y_{ij}=1\}} \mathcal{X}_{ij}^d ((1 - Y_{ij}) - \frac{\exp(\mathbf{w}^T \mathcal{X}_{ij})}{1 + \exp(\mathbf{w}^T \mathcal{X}_{ij})}) - \lambda_w w_d \Big) \tag{5}$$

Since the size of training set is less than the number of features, we opt to using Stochastic Gradient Ascent algorithm with L2-regularization.

$$w_d^{t+1} = w_d^t + \eta(\mathcal{X}_{ij}^d((1 - Y_{ij}) - \frac{\exp(\mathbf{w}^T \mathcal{X}_{ij})}{1 + \exp(\mathbf{w}^T \mathcal{X}_{ij})}) - \lambda_w w_d) \tag{6}$$

Algorithm 1: Stochastic Gradient Ascent Algorithm

Initialize **w** as zero-valued vector.
Initialize $epoch = 1$.
Initialize $\lambda = 0.01$.
Initialize $\eta = 1$.
while *epoch* $<$ *maxIter* **do**
  for $\forall i, j$ **do**
    Update **w** using Equation 7
  epoch++;
Return **w**.

2.2 Extension 1: Re-weighting the features

By definition in Equation 1, Y_{ij} can only capture which video has more viewerships, but not how much their viewerships differ. Hence, the classifier f cannot weight correctly important features in determining video's popularity. Assume that we have three YouTube videos v_1, v_2 and v_3, each attracted 1000, 10 and 1 views after one day since they were uploaded to the web. Also assume that we only use bag-of-word features and there are five words in our dictionary $\{w_1, w_2, w_3, w_4, w_5\}$.The following table contains all necessary information for this example.

Video	Title	Number of views	Bag of word features
v_1	$\{w_1, w_2, w_3\}$	1000	$\{1, 1, 1, 0, 0\}$
v_2	$\{w_2, w_3, w_4\}$	10	$\{0, 1, 1, 1, 0\}$
v_3	$\{w_3, w_4, w_5\}$	1	$\{0, 0, 1, 1, 1\}$

As the above scheme in Section 2.1, we can represent the pair (v_1, v_2) and (v_2, v_3) with feature vectors $\mathcal{X}_{12} = \{1, 0, 0, -1, 0\}$ and $\mathcal{X}_{23} = \{0, 1, 0, 0, -1\}$. When training a classifier f on these two vectors with both Y_{12} and Y_{23} equal to 1, we can see no differences between the weights (i.e. model parameters) on feature w_1 and w_2, w_4 and w_5. Hence, f cannot correctly rank a pair of videos with titles of $\{w_1, w_4\}$ and $\{w_2, w_5\}$ correspondingly.

A general idea to solve this problem is to incorporate the magnitude in difference between two videos' number of views into f. Basically, we want to have more weights on features that are frequently attached with popular videos. In this work we propose two ad-hoc solutions: 1) Augmenting the representative feature vectors $\mathcal{X}_{ij}$ and 2) Re-weighting the gradient.

2.2.1 Augmenting the representative feature vectors $\mathcal{X}_{ij}$

In this approach, we scale the representative feature vectors $\mathcal{X}_{ij}$ for pairs of videos (i, j) in the training set, and train a classifier f on these augmented features. The scaling factor is determined based on the ratio of numbers of views of the corresponding video pair i and j. To avoid the overflow problem caused by high variance in number of views (2 billions versus 10), we transform the number of views into log space before computing the ratio. Let consider the pair $\mathcal{X}_{12}$ in the above example. The scaling factor is computed as follow: $\alpha = \frac{\log 1000}{\log 10} = 3$. Hence the augmented representative features $\mathcal{X}_{12} = \{3, 0, 0, -3, 0\}$.

There are other various options to compute the scaling factor α such as using a different log base, or normalise all the view counts, etc. However, all these approaches are akin to the proposed one, and hence do not have a theoretical guarantee on the overall performance. We demonstrate one way of using ratio of log base 10 in this work.

2.2.2 Re-weighting the gradient

Another way to tackle the above problem is to re-weighting the gradient. Similarly as the first approach, we also compute a scaling factor α from the two videos' number of views. The only difference is instead of augmenting the representative feature vectors $\mathcal{X}_{ij}$, we re-weight the gradient in Equation 6 as follows

$$w_d^{t+1} = w_d^t + \eta\alpha(\mathcal{X}_{ij}^d((1 - Y_{ij}) - \frac{\exp(\mathbf{w}^T\mathcal{X}_{ij})}{1 + \exp(\mathbf{w}^T\mathcal{X}_{ij})}) - \lambda_w w_d) \tag{7}$$

The idea is borrowed from the AdaBoost technique by re-weighting the training data points. However, the difference is that AdaBoost keeps updating weights for every iteration, meanwhile we fix the weight. Our goal is to stress the importance of pairs having high variance in their viewerships by forcing the corresponding parameters to update more gradients of these pairs.

2.3 Extension 2: Ensemble of time-specific classifiers

Number of views of a video also depends on time passed since it was first uploaded to YouTube. It is more comparable to measure the popularity between videos sharing the same passed days. Such videos are clustered into a bin and a distinct classifier f is trained on them. For the above example, we have 3 different bins, $B_{1000} = \{v_1\}$, $B_{10} = \{v_2\}$ and $B_1 = \{v_3\}$, with corresponding classifiers f_{1000}, f_{10} and f_1. Given a new video pair v_4 and v_5, we first find the bin that both videos belong to according to their age, and use the corresponding classifier to compare their popularity.

This approach has a drawback which makes the learning model suffers from data sparseness. Since our data is a one-month sample from YouTube repository, it is possible to have keywords do not exist in the observed video pairs of a bin. To overcome this issue, we suggest to construct an ensemble of all bins' classifier to enhance the predictive performance.

According to [4], ensemble methods are learning algorithms that blend results from different hypotheses to perform some prediction tasks on new data points. There are two main reasons behind these methods. First is statistical. Basically, we often do not have sufficient data to identify the best, but equally accurate, hypotheses. By taking the majority votes or average results of these hypothesis, we can reduce the risk of choosing the wrong hypothesis. Secondly, different hypotheses may have different starting points and explore different local optima. Hence an ensemble of these hypotheses can give a better approximation than any of individual hypothesis. Bagging and Boosting are common ensemble algorithms.

The above two reasons motive us to construct an ensemble of bin-specific classifiers with selection of the majority weighted voting scheme. First, it is possible to have a keyword w does not exist in an arbitrary bin's training data, but can occur in other bins. Hence we can borrow information from those bins to weight w. Second, to incorporate the fact that the video's popularity will be diminishing over time, the weights from bins further away from the selected bin must have smaller impact to its neighbouring bins. We therefore introduce a majority weighted voting scheme as follow. Let f_t be the classifier trained on data of bin B_t, containing videos of t-days old. Assume that we have T such bins. For a pair of videos i, j in the bin B_t's testing set, we compute the probability of ranking as follows

$$\begin{aligned} P(Y_{ij}^t = 1|\mathcal{X}_{ij}^t, f_1, \ldots, f_T) &= \sum_{t'} \frac{1}{1 + |t - t'|} P(Y_{ij}^t = 1|\mathcal{X}_{ij}^t, f_{t'}) \\ &= \sum_{t'} \frac{1}{1 + |t - t'|} P(Y_{ij}^t = 1|\mathcal{X}_{ij}^t, \mathbf{w}^{t'}) \end{aligned} \tag{8}$$

We can compute $P(Y_{ij}^t = 0|\mathcal{X}_{ij}^t, f_1, \ldots, f_T)$ similarly. The class with higher probability is selected as final prediction.

3 Ranking by Regression

Another approach to the ranking problem is to predict the number of views for each video and then compare that, rather than comparing two videos directly. Finding the number of views can be treated

as a simple linear regression problem. Our linear function assumes that are labels come from our input X_u plus some noise ϵ:

$$Y_u = X_u\beta + \epsilon \tag{9}$$

We therefore seek a function of the form

$$f(X) = X\beta \tag{10}$$

and attempt to minimize the mean squared error loss function, giving

$$\beta = argmin_\beta 1/n(A\beta - Y)^T(A\beta - Y) \tag{11}$$

where

$$A = [X_1...X_n]^T, Y = [Y_1...Y_n]^T \tag{12}$$

We can use either the closed form or Gradient Descent to learn the β parameters. However, since our feature space may be quite large, we opt for the latter. We therefore initialize β^0 to 0, and thereafter use the update step

$$\beta^{t+1} = \beta^t - \eta A^T(A\beta^t - Y) \tag{13}$$

Since our context is a bad conditioning problem, we opt Stochastic Gradient Descent with Regularization. The new update function

$$\beta^{t+1} = \beta^t - \eta(x_i(x_i\beta^t - Y_i) + \beta^t) \tag{14}$$

After the learning stage is complete, the predicted ranking can be done as follows

$$\hat{Y}_{uv} = \mathbb{I}(\beta X_u > \beta X_v), \tag{15}$$

where $\mathbb{I}$ is the indicator function, return 1 if the expression as argument is true, and 0 otherwise.

Directly predicting the actual popularity is not strictly necessary for our ranking problem, however, and we can instead predict anything with the same ranking properties. In order to help deal with enormous variance in the number of views, we also attempted replace Y with $log(Y)$ and perform least-squares to predict that instead. Since order of magnitude is more significant than the actual number of views, this is a reasonable substitution.

We anticipated that direct ranking with logistic regression would be more accurate than first performing regression on popularity. Comparing the two methods is one of the goals of this project.

4 Experimental study

4.1 Dataset

We implemented our own crawler in Java and started to collect information from YouTube since October, 1st, to November, 5th, 2014. Our crawling strategy is to initialize the crawler with several random "seed" videos, which mostly are in the *Movie* and *Music* category, , and recursively explores all other videos that YouTube suggests as being related to that videos. For each video, we extract all of its metadata such as title, uploader, description, upload date, number of views/likes/dislikes, video length, and a number of other attributes, as well as a list of around 30 videos YouTube recommends as being similar.

4.1.1 Preliminary Data Statistics

Although the crawler was suspended twice due to technical issues and upgrades, thus far we have gathered a decent amount of data:

- Number of videos crawled: 1,432,213
- Number of uploaders: 628,072
- Most viewed video: 2,104,518,656 views.
- Most "liked" video: 8,639,650 likes.
- Most "disliked" video: 4,184,769 dislikes.
- Size of "bag of words" dictionary produced for the titles and descriptions: 2,447,603 entries.

As the statistics show, we have a large magnitude in ranges of number of view, likes and dislikes. This motivates us to do some preprocessing on data, such as feature normalisation or data standardization, to ensure the numerical stability and good speed of convergence on the learning algorithm. We discuss more on this step in the Section 4.1.2.

We plot in Figure 1 the distribution of number of views in our current data. Interestingly, the distribution is in the shape of Gaussians, instead of following the Power Law distribution as frequently observed in social network. We guess this observation is due to the way we sample the data, by following the recommended links, and imply that popularity of a video is one of the highly-impacted factors in YouTube recommender systems. Whether we can uncover the underlying reasons of links suggested by YouTube, namely, based on similarity or popularity or both, is an interesting question and may be addressed in our future work.

4.1.2 Pre-processing data

To Joseph: can you help me to work on this part ? You can revise from your discussion of taking the log-form and put it here.

4.1.3 Feature extraction

The first step is to build a dictionary mapping the uploader to the number of videos they have uploaded and the total number of views there videos have. We also take care to prevent "cheating": In order to ensure that our predictor has only such information as would be available before the video's publishing is ever used, we temporarily reduce these number of video-views and the total number of video uploads for the uploader according to the publish date of the video under current consideration.

We train a linear regression model for each of our three outputs on the following features:

- Many features extracted via a bag-of-words model on the title, using TF-IDF.
- The # of videos uploaded by the uploader prior to the current video's upload date.
- The total # of views for an uploader due to videos released prior to the current video's upload date.

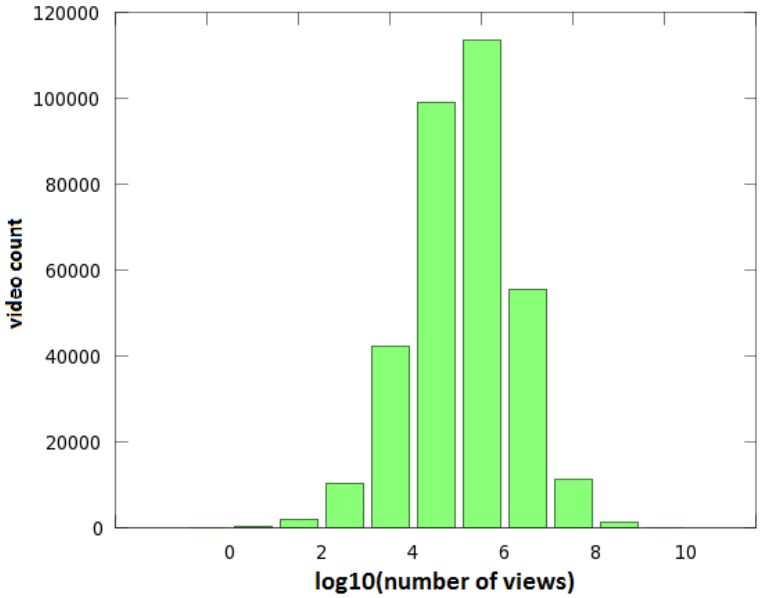

Figure 1: Histogram on the distribution of number of views (in log-scale).

- The fraction of the previous two features (the average number of views per video for videos uploaded by the same uploader prior to the current video's upload date).
- The runtime of the video, in seconds.
- The age of the video at the time of crawling, in days.

4.2 Evaluation Metrics

Three evaluation measures widely used to evaluate ranking approaches are 0-1 loss function, Area under Curve (AUC). **0-1 loss function** is the ratio of correctly ordered video pairs over total number of pairs in testing set.

$$0/1_{loss} = \sum_{(u,v)_{test}} \frac{1}{|(u,v)_{test}|} \mathbf{1}[\hat{Y}_{uv} - Y_{uv} == 0] \tag{16}$$

Since our label values in $\{0, 1\}$, we use **AUC Loss** as another ranking-based performance metric.

$$AUC_{loss} = 1 - AUC, \tag{17}$$

where AUC is the area under the ROC curve.

4.3 Preliminary Results

Here we show some of our preliminary results using a simple linear regression.

5 Related work

- Local Collaborative Ranking [1]
- Logits model for sets of ranked items [2]
- AdaRank [3]

Ordinal Regression Ranking learning or ordinal regression is a learning task of predicting class values possessing a natural order. For example, students' exam paper are often graded in scale of $F < D < C < B < A$. There are several approaches in Machine Learning have been proposed to tackle this problem.

- Frank and Hall (2001) transformed the K-class ordinal problem into K-1 binary classification problems.

6 Conclusion

6.1 To-do list

There is one additional piece of information that we decided our crawler should collect, which still needs to be added: the number of subscribers for each user. Since this was not needed for our original proposal, we will need to go back and gather this information, which may take some time considering the vast quantity of videos crawled.

6.2 Stretch Goals

We certainly hope to attempt increasingly sophisticated learning techniques to reduce our loss as much as possible. Time allowing, there are also other interesting results we can persue. Chief among these in our mind is to make more time-dependent predictions. It may be, for example, that video A is very popular at first, but that video B maintains it's popularity better over time, eventually overtaking video A in terms of the numbers of views and of likes.

Acknowledgments

Our sincere thanks to Anthony for his patience and value inputs to improve our project.

References

[1] Joonseok Lee, Samy Bengio, Seungyeon Kim, Guy Lebanon and Yoram Singer. Local Collaborative Ranking. Proceedings of the 23rd International World Wide Web Conference (WWW) 2014

[2] Allison, Paul D., and Nicholas A. Christakis. "Logit models for sets of ranked items." Sociological methodology 24.1994 (1994): 199-228.

[3] Xu, Jun, and Hang Li. "Adarank: a boosting algorithm for information retrieval." Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2007.

[4] Dietterich, Thomas G. "Ensemble methods in machine learning." Multiple classifier systems. Springer Berlin Heidelberg, 2000. 1-15.