Machine Learning Engineer Nanodegree

Capstone Proposal

Natalya Rapstine
February 3rd, 2017

Proposal

Domain Background

Applying to graduate school is a long and tedious process that requires a lot of preparation and research. One of the most important aspects of applying to graduate school is to find a potential advisor or research group that fits an individual interests and background. Priceton Review recommends starting to research grad schools in May and finalizing the list of prospective graduate schools in September. My goal is to shorten 5 months long research process to a few hours.

Anyone who has gone through the long process of applying to graduate school most likely found the task of finding a research group to work in or a suitable professor tedious. Moreover, because the research of potential school is so time-consuming and superficial with the best efforts, it is nearly impossible to get a grasp of how research groups collaborate and what institutions do research in the same field. Machine learning to the rescue!

This project was partly inspired by this paper from Google on how YouTube recommender system that uses deep neural networks and partly by personal frustrating experience of applying to graduate school.

Problem Statement

The problem is to find top matches for professors for each student interested in applying to graduate school. By best, I mean the most relevant in terms of publications that the professor produces to the student's background and interests. The problem can be thought of as multiclass classification with each professor being a distinct class. My goal is to accurately predict what abstracts are classified to which author. Then given new input of student's interests, I can predict which classes (professors) are the most likely thus outputting the most suitable matches.

Datasets and Inputs

I tend to build a fairly large database of academic publications and count word frequency in each abstract by scraping the web for author's abstracts. The inputs are each abstract that will be converted into a word vector with normalized word frequencies and true labels are the author of the abstract. I will split my dataset into the training and testing datasets to evaluate the model's performace on unseen data.

For example, arXiv.org provides open access to over 1 million research publications in Physics, Mathematics, Computer Science, Quantitative Biology, Quantitative Finance and Statistics which are suitable for my database of publications in STEM fields.

Solution Statement

I aim to aid students and working professionals thinking about applying to graduate school (in the United States) in a STEM field by building a recommender system for students who are unsure of where to apply based on their current knowledge and interests. This process will automate long and tedious Google search to first find schools, narrow down professors and read through their recent publications. My recommender system would output ordered suitable professors or institutions given student's interests in a form of key words so that the student can be confident that there is a match between what research group/professor is publishing and what student wants to pursue and she does not have to spend half a year researching different institutions and reading professors' publications.

As a solution, I propose building a Convolutional Neural Network (CNN) and optimizing its performance by using a sophisticated optimizer such as ADAM instead of the gradient descent optimizer. After the model is trained, the new input in the form of key words from a student will produce a probability distribution over all professors. My recommender system will output the top matches (with the highest probabilities) as its top ranked professors for the student.

Benchmark Model

For a benchmark model, I will build a simple logistic regression model that predicts class (professor) label given the input of word vectors for thousands of abstract samples.

Evaluation Metrics

To evaluate the performance of my recommender system, I will use the accuracy score metric on the test data that the model has not seen while training for the benchmark and solution models.

The accuracy score computes the fraction of correctly predicted samples. The lowest possible accuracy score is 0 and the highest is 1. Accuracy is computed as follows:

$$accuracy(y, \hat{y}) = \frac{1}{N} \sum_{i} 1(\hat{y}_i = y_i)$$

where *i* is the sample index, *y* is the true label, *y hat* is the predicted class, *N* is the number of samples, and *1* is the indicator function.

Project Design

The first step is to gather the data off the web. In the raw form, I scrape the webpages and filter out abstracts and authors to make two lists - one for publication abstracts and one for authors which will be my class labels. The raw data needs to be preprocessed to be suitable for classification. I follow a typical Natural Language Processing workflow of cleaning the data by removing capitalization, stop words (common words in English language), punctuation, and stemming the words. Then I count the frequency of each word stem occurrence in the abstract so that I end up with the word stem frequency as the word vector for each sample. The author of the abstract is its label.

When the dataset is ready for analysis, I split it into the training and testing sets.

I build a benchmark model of logistic regression to predict class labels given the training dataset and evaluate it on the test data using the accuracy score metric.

Then, I build a convolutional neural network model to predict labels given the input training dataset and also, evaluate it on the test dataset to compare against the benchmark model. I intend to use TensorFlow library for Python to build my CNN model.