Machine Learning Engineer Nanodegree
Capstone Proposal
Aaron Marquez
August 27, 2017

Domain Background

Personalized medicine is a medical procedure that divides patients into different groups with medical decisions, medicines, and products being tailored specifically to a patient based on their predicted response or risk of disease [1]. Also known as genome-based medicine, it is based on studying the human genome and looking at a person's genetic predisposition to certain medical conditions. It is a young but rapidly advancing field of healthcare and has a lot of potential to change how people are prescribed medicines. Instead of a single pill being designed for millions of people to consume as treatment for a disease, each person could have their own unique pill based on their genetic makeup.

Cancer research has learned a lot about genetic variety in cancer types and has created a new field called "Oncogenomics" that is the application of genomics and personalized medicine to cancer research and treatment. It has found success in targeted cancer treatments such as Gleevec, Herceptin, and Avastin. I think it is very important to continue to research personalized medicine, especially when it comes to cancer research, and using machine learning algorithms can accelerate this process by combing through large amounts of data faster than a human could. A paper titled "Applications of Machine Learning in Cancer Prediction and Prognosis" talks about personalized medicine and the various ways machine learning can aid in cancer research [2].

Problem Statement

Kaggle is currently hosting a competition titled "Personalized Medicine: Redefining Cancer Treatment" sponsored by Memorial Sloan Kettering Cancer Center (MSKCC). The following description is taken form the competition's homepage:

"Once sequenced, a cancer tumor can have thousands of genetic mutations. But the challenge is distinguishing the mutations that contribute to tumor growth (drivers) from the neutral mutations (passengers).
Currently this interpretation of genetic mutations is being done manually. This is a very time-consuming task where a clinical pathologist has to manually review and classify every single genetic mutation based on evidence from text-based clinical literature.
For this competition MSKCC is making available an expert-annotated knowledge base where world-class researchers and oncologists have manually annotated thousands of mutations.

We need your help to develop a Machine Learning algorithm that, using this knowledge base as a baseline, automatically classifies genetic variations [3].”

Datasets and Inputs

The data is available on Kaggle’s website and is already split into training and testing sets. The files are text files labeled: “test_text.txt”, “training_text.txt”, “test_variants.txt”, and “training_variants.txt.” There are a total of 9 classes that the algorithm should predict. Each class represents the different classes recommended by the American College of Medical Genetics to classify genetic mutations on. MSKCC encoded those classes into numbers between 1 and 9. These classes are genetic mutations based on clinical evidence, which in this case is located in text files and the training data is text. There are 3321 training observations and 5668 testing observations. The following file descriptions are taken from Kaggle’s website [4].

- training_variants - a comma separated file containing the description of the genetic mutations used for training. Fields are ID (the id of the row used to link the mutation to the clinical evidence), Gene (the gene where this genetic mutation is located), Variation (the aminoacid change for this mutations), Class (1-9 the class this genetic mutation has been classified on)
- training_text - a double pipe (||) delimited file that contains the clinical evidence (text) used to classify genetic mutations. Fields are ID (the id of the row used to link the clinical evidence to the genetic mutation), Text (the clinical evidence used to classify the genetic mutation)
- test_variants - a comma separated file containing the description of the genetic mutations used for training. Fields are ID (the id of the row used to link the mutation to the clinical evidence), Gene (the gene where this genetic mutation is located), Variation (the aminoacid change for this mutations)
- test_text - a double pipe (||) delimited file that contains the clinical evidence (text) used to classify genetic mutations. Fields are ID (the id of the row used to link the clinical evidence to the genetic mutation), Text (the clinical evidence used to classify the genetic mutation)

Solution Statement

Given that the training data is text then the problem is a natural language processing one and a supervised classification one as well. To start, the first thing to do would be to use a bag of words preprocessing method to transform the text into numerical data. One possible way to accomplish this would be to use scikit-learn’s built in feature extraction functions (CountVectorizer, TfidVectorizer, etc.) to process the text data. Another way would be to use the spaCy [5], which is a python library for advanced natural language processing. SpaCy seems like a good option given that it offers fast performance, pre-trained word vectors, and easy deep learning integration.

SpaCy requires loading a language model (in this case it would be English) and feeding the model text to get numerical data. Beyond the basic model there are additional steps that could be taken to improve the performance of the model. These include

removing stopwords (common words such as a, the, very, etc.), lemmatizing words (grouping different inflections of the same word: eat, eaten, eats), using n-grams (counting sequences of words instead of individual words: “dirty clothes” vs. “dirty” “clothes”), among others.

The next step would be to build the model. Naïve bayes models are known to work well on natural language processing problems [6], but I think it would be a good idea to explore other models (random forest classifier, neural networks) and in particular to try a deep learning approach with the keras python library [7]. The evaluation metric for this problem is a multi class log loss equation (described below). To make this project replicable the best thing to do would be to set a random seed so the results stay the same every time the program is run.

Benchmark Model

As of August 28th, 2017 there are 878 teams competing in the competition. There is also a leaderboard showing the rankings of teams based on their log loss score. I think good benchmark may be to compare my model’s performance against other competitor’s models. The midmost team in the leaderboard has a score of 0.74399. If I can get a score better than that then I will have done better than half of the current competitors.

Evaluation Metrics

The evaluation metric used by Kaggle is a multi class log loss equation. It has the following form:

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j}\log(p_{i,j})$$

Where N is the number of observations, M is the number of class labels, log is the natural logarithm, $y_{i,j}$ is the 1 if the observation i is in class j and 0 otherwise, and $p_{i,j}$ is the predicted probability that observation i is in class j. A lower log loss means better performance. It has been used in a few other Kaggle competitions as well.

Project Design

I want to first try to get a model up and running quickly using sci-kit learn just to see how well it performs. The first step is to preprocess the text into numerical data using a bag of words representation [8]. With sci-kit learn I would use the CountVectorizer method to convert the text to a matrix of token counts. Using an English stop words list will also help rid the text of common words such as “a,” “and,” “the,” etc. Then using TfidfTransformer I can transform the newly created matrix into tf-idf representation. This is done to scale down the impact of words that appear frequently and are less informative than words that appear less often. Finally I can fit the data to a RandomForestClassifier and a GuassianNB classifier and test the accuracy on a validation set (before I start I would need to split the data again to create a validation set since Kaggle provides only

training and testing sets). I think it would also be a good idea to use grid search to search for optimal parameters quickly. Some of the parameters I will test out will be number of estimators for random forest, criterion, max depth, n gram range for count vectorizer, and whether to enable inverse-document-frequency reweighting (use_idf) for the tfid transformer. I'll also set the random state parameter (where applicable) to an integer to make the results reproducible.

After testing how well that first approach goes I will move onto working with spaCy and keras for a different approach to the problem. I think it would be interesting to see how these two different approaches compare. My workflow would follow these steps:

1. Visualize some example text
2. Load English model from spaCy
    a. nlp = spacy.load('en')
3. Process training text using spaCy
    a. Tokenize text/extract keywords
4. Train deep learning model using keras
    a. Build Sequential model and add densely connected layers with relu activation
5. Predict and assess accuracy on validation data
6. Refine parameters and test on validation until optimal accuracy is reached

The spaCy library is capable of integrating keras and I plan to make use of this by creating a multilayer perceptron architecture with relu activation function and a softmax activation for output layer. I am going to experiment with the number of hidden layers and number of neurons when I get to that point and see which combination works best. After fitting this model to the training data I can make predictions on the test data (after preprocessing it using spaCy) and calculate the accuracy. I also plan to make use of some of spaCy's methods to visualize which parts of speech and maybe even plotting a word map of the training dataset.

References

[1] https://en.wikipedia.org/wiki/Personalized_medicine
[2] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2675494
[3] https://www.kaggle.com/c/msk-redefining-cancer-treatment
[4] https://www.kaggle.com/c/msk-redefining-cancer-treatment/data
[5] https://spacy.io
[6] https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/
[7] https://keras.io
[8] https://en.wikipedia.org/wiki/Bag-of-words_model