Portfolio Projects, Reports, and Presentations

The following is the poster I presented on my research, “Deep learning for Blood Pressure Estimation” at National Collegiate Research Conference, Harvard 2024 and Undergraduate Research Symposium, Boston University, 2023. The link for this poster is – https://github.com/jiveshj/UROP_Poster.git I have attached the sample image of the poster below:

Boston University Office of the Provost
Undergraduate Research Opportunities Program

Deep Learning for Blood Pressure Estimation

Jivesh Jain[1], Ariane Garett[2], Darren Roblyer[2]

1. Department of Electrical and Computer Engineering , Boston University, Boston, MA
2. Department of Biomedical Engineering , Boston University, Boston, MA

BOSTON UNIVERSITY

Abstract

Nearly half the population in the United States is suffering from hypertension. For continuous blood pressure estimation, various studies have constructed machine learning algorithms which use photoplethysmography [PPG] (which is an optical technique for detecting blood volume changes) and electrocardiograph (ECG) signals as inputs. In our laboratory, we are trying to show that the blood flow index [BFi] (which is an optical technique that gives information about vascular blood flow) is more correlated with blood pressure than PPG signals.Therefore, in this study, we used different deep learning techniques to estimate blood pressure for 20 subjects using the combination of BFi and PPG waveforms. Also, since we used BFi and PPG waveforms, only one optical set up was needed for taking measurements and hence this made the blood pressure estimation process more practical.

Introduction

This study has the following characteristics:

1) We use a dataset that contains highly variable blood pressure data to train our model.

2) We show that BFi and PPG combination is more helpful in predicting blood pressure than PPG alone.

- For this purpose, we trained a model on features extracted from PPG data and then trained the same model on features extracted from a combination of BFi and PPG data of the subjects.

- The deep learning model that we build consists of convolutional layers and LSTM layers to help predict time series data more accurately. [1]

At the end of the study, we also show that it is easier to predict blood pressure with greater accuracy if the dataset values are not fluctuating enough but instead the error starts to increase if the variance of the data increases.

Results

Figure 4. Correlation between predicted SBP vs True SBP as determined from only PPG features

Figure 5. Correlation between predicted SBP vs True SBP as determined from BFi + PPG features

Figure 6. Bland Altman plot for predicted SBP from PPG features

Figure 7. Bland Altman plot for predicted SBP from PPG + BFi features

Figure 8. Comparison of Root mean Square error for different subjects

	PPG	PPG+BFi
ME (mmHg)	15.89	5.62
SD (mmHg)	12.22	11.64
RMSE (mmHg)	20.76	13.86

33% improvement in RMSE

Table 1. Comparison between the PPG and BFi + PPG model errors

Figure 9. Correlation between Predicted SBP(mmHg) vs True SBP(mmHg) as determined from PPG + BFi features (less datapoints per subject)

	More Datapoints (Baseline + Exercise)	Less Datapoints (Baseline)
ME (mmHg)	5.62	-1.71
SD (mmHg)	11.46	9.64
RMSE (mmHg)	13.86	11.26

18% improvement in RMSE

Table 2. Comparison between less datapoints vs more datapoints per subject model.

Methods

Figure 1. Measurement set up. SCOS measurements are taken on the finger and wrist.
PCX: plano-convex [2]

Figure 2. Measurement Setup [2]

Figure 3. Blood Pressure Predicting Pipeline

Conclusions

1) Through the graphs at the top, it is easy to conclude that BFi + PPG is a better combination for blood pressure estimation (than PPG alone).

2) The graph and table shown at the bottom of the result's section prove that it is indeed easier to predict stable blood pressure values as compared to blood pressure values which are fluctuating often (real life BP values).

All these conclusions help us form considerations for what dataset to use, what model to build and which parameters to use to achieve portable, continuous, noninvasive blood pressure estimation.

Future Directions

-Build more complex models that can help build a machine that predicts blood pressure within the error range of FDA.

-Train the model on more data to increase its performance .

- Build CNN models which use PPG and BFi waveform images as input

Acknowledgements

This research was funded by Undergraduate Research Opportunities Program, Boston University

References

1. Mou, Hanlin, and Junsheng Yu. "CNN-LSTM Prediction Method for Blood Pressure Based on Pulse Wave." Electronics, vol. 10, no. 14, July 2021. https://doi.org/10.3390/electronics10141664
2. Garrett, Ariane et al. "Simultaneous photoplethysmography and blood flow measurements towards the estimation of blood pressure using speckle contrast optical spectroscopy". Biomedical Optics Express. 2023

Machine Learning:

The following is the project report on our project, “Classification for Sailor Award Prediction”. This project is going to be published on the Sailrank.com (https://sailrank.com/) soon.

Classification for Sailor Award Prediction

Jivesh Jain
Dept. of Electrical & Computer Engineering Boston University
Boston, MA 02215
jiveshj@bu.edu

Noah Robitshek
Dept. of Electrical & Computer Engineering Boston University
Boston, MA 02215
noahro@bu.edu

1. Abstract

While predicting the win/loss of sports has risen over the past few years, predicting awards such as Heisman (football), player of the year (soccer), and All-Stars (baseball) have not become as prominent. While they are often less lucrative for betters, they still offer an interesting problem for academics. In this study we created a dataset from data consisting of 264 Sailors from the Mid-Atlantic, and North-Atlantic conferences who competed in at least their Fall Championship, Spring Championship, or National Championship Regattas. Given these scores and additional features such as year in school, division, and finish place, we built a classifier to distinguish an individual as “All-American” or “Not All American”. We got an average accuracy (CCR) of around 0.8 across all the algorithms we implemented. We also drew several interesting conclusions from our data regarding All American status given to the sailors.

2. Introduction

Throughout the lifetime of sports, awards have been an integral part of competitions. Finish place awards such as 1st, 2nd, and 3rd place awards are easy to classify. If a competitor gets 1st place, they get the 1st place award, if the competitor gets 2nd place they get the 2nd place award. While these awards are straightforward, awards such as Top Athlete, All-State, All-Conference, and All American awards are more ambiguous. Some of these awards have clear criteria, such as top 5 fastest runners at states, while others are more clouded. In sailing one of the most precious awards for an open skipper, the driver male or female of the sailboat, is the All-American award. This award is chosen by a collection of college coaches. The selection committee has a few requirements that help clarify the selection process. Some of these include restrictions include “the total number of All-American awards which is capped at 18”. Other criteria the committee looks at is “How the record compares to those of fellow sailors, the quality of competition in which the record was achieved, how the record compares to those of All-Americans selected in previous years”. Additionally, information is released stating what committee does not consider such as: “Sailors' records in previous years, Sailors' records in non-college events, and the breadth of teams and conferences represented”. While considering the rough list of factors published by the committee, we set out to investigate which features of a sailor contribute the greatest to the probability of receiving an All American award. Additionally, we set out to see if given these factors, we could predict whether a sailor received the award or not (a binary classification problem).

3. Significance

Our project has a few key contributions to society. Firstly, is the creation of the dataset which will allow interested coworkers the opportunity to develop more comprehensive models. The current data for college sailing is found on the website Techscore. This website contains a record of a vast amount of information about sailors including year, regattas sailed, finish place at regattas and much more. While there is a vast amount of data, there is currently no data format that allows researchers to easily analyze the data. With the creation of our data sets, we are creating resources for more complex data analysis in the future. Additionally, our study is significant because it will give insights to college sailors about features that contribute the most to receiving an All American award. With this information, sailors will be more informed to compare themselves against their competitors as well as have the information to predict whether they will be given the All-American award or not. Finally, our project will give College Sailing coaches the opportunity to optimize the chance their sailors receive an All-American award. With the information from our project, we hope to allow college coaches to allocate team resources more effectively to maximize the probability their sailor receives the coveted award.

4. Literature Review

While there are currently no widely known works on machine learning for sailing, during the initial stages we reviewed a few works related to sports and classification.

The first related work is the website SailRank [1]. This website started as a project to rank sailors and teams and has evolved to include win/loss prediction and much more. The website utilizes a ranking system called "Power Ranking", a ranking commonly used in other sports such as football, golf, and basketball. While this website does not involve awards like the All-American it gave us inspiration to start by analyzing the finish place of individual participants in regattas.

The first paper we referenced was "Predicting Students' Performance using ID3 and C4.5 Classification Algorithms" [2]. This study addresses aims to predict students' future academic performance based on past students' performance. To achieve this, the authors utilized data mining classification technique to analyze a dataset comprising student information such as gender, board exam scores, entrance exam ranks, and previous batch's first-year results. The ID3 and C4.5 classification algorithms were applied to predict both general and individual performance of newly admitted students in future examinations. The choice of classification over clustering was justified to meet the goal of predicting students' performance into predefined classes - "Pass" and "Fail". During our search for related works, we compared the ability to classify "Pass" and "Fail" with the ability to classify a sailor as "All-American" vs "Not All-American".

5. Datasets

Throughout the project, a key component was the creation of the datasets for which our models would be trained. Most of the data that was used for the project was scraped from the website called Techscore. Techscore is the home of College Sailing results and provided us with historical data ranging from today until 2008. The website not only contains historical data on the finishing place of each individual in the regatta but also provides us information about the sailors such as graduation year, conference, regattas sailed, and races sailed in a regatta.

During the beginning stages of data collection, we considered writing a web scraping program to scrape the data off the Techscore website but opted for manual data collection instead because of the limited time frame for the project. From the start we were considering two parameters: scope of the dataset and number of features. While considering scope, we were looking at both the number of individuals per year as well as the number of years. Additionally, we had to consider the number of features. From the start, we began by looking at place finish at regattas but later expanded to include graduation year and information about the regatta sailed such as division (A or B) and information about whether they were swapped in or swapped out. Throughout this process we had to consider the event of missing data points in cases where an individual did not sail a regatta. In this case we decided to finish them last place plus 1, C division, and "not sailed" in the feature about getting swapped in/out.

In our initial dataset, Dataset 1, we included around 24 sailors per year over the course of 4 years. Additionally in Dataset 1, we only had 3 features. These features were the place finishes from the Fall Championship, Spring Championship, and the National Championship. After the creation of this dataset, we realized the need for both more individuals as well as more features. Initially we worked to include more individuals and created Dataset 2 which included sailors from 4 difference conferences, MAISA, NEISA, PCCSA, and SAISA. While this dataset included more sailors, it was still restricted to only division A sailors, a small subset of the regatta. Following this dataset, and constrained by time, we narrowed the next datasets down to only the MAISA and NEISA conferences but included sailors in the A and B divisions. The final three datasets include data taken from 2022, 2023, and both years respectively. The final data set, dataset 5, was developed with the following 11 features. While scraping the data, we utilized google sheets formulas to translate data such as divisions (A, B) into numerical values such as 1 and 2. Additionally, we transformed school year data from graduation year to year in school which ranged from 1 to 4.

Feature #	Name of Feature	Type of Feature
1	Confrence Number	Categorical
2	Year in School	Categorical
3	Fall Champs Changed	Categorical
4	Fall Champs Division	Categorical
5	Fall Champs Place	Categorical
6	Spring Champs Changed	Categorical
7	Spring Champs Division	Categorical
8	Spring Champs Place	Categorical
9	Nationals Changed	Categorical
10	Nationals Division	Categorical
11	Nationals Place	Categorical

Figure 1: Feature Descriptions of Final Dataset

With this final dataset comprised of 2022 and 2023 data, we performed basic explanatory analysis to understand the dataset before training our models. Firstly, we analyzed the range of the finish place data as well as the grade distribution to understand how our data was distributed.

	Mean	Min	Max	Standard Deviation
Year in School	2.7235	1	4	1.08711
Fall Champs Place	12.9318	1	19	5.93782
Spring Champs Place	12.1970	1	19	5.91182
Nationals Place	16.1894	1	19	4.98993

Figure 2: Statistical analysis of final Dataset

Additionally, we computed a basic correlation coefficient matrix to understand the initial linear relationships between the features (0-10) and the label (row 11). One of the most interesting observations we concluded from the matrix was the strong influence that the features of Nationals (whether they were swapped in/out, which division they sailed, and the place they received at nationals) had the strongest correlation with the label. Additionally, we saw an interesting observation that the features from the Fall Championships (such as position obtained, whether they were swapped in or out etc.) were linearly correlated as the features form the Spring Championships suggesting that the player's performance is correlated in the fall and spring regattas. Within each championship, whether a sailor participated in a regatta had much more influence (very strongly correlated with the label) than the other two features (which division they sailed, and what their final position/rank in the regatta was).

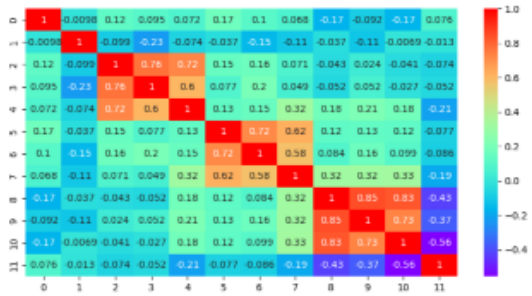


Figure 3: Correlation Coefficient Matrix of final dataset

6. Problem Formulation and Solution Approaches

With the creation of the final dataset, we began to work to map the 11 features into the label space. Our task at hand was to perform binary classification. We had to classify whether a sailor with a given feature vector could qualify as an All-American at the end of the sailing year. For the task of classifying the sailors, we employed multiple machine learning algorithms, specifically, we worked with Support Vector Machines (SVMs), Logistic Regression, Decision Trees, Random Forests, Linear Discriminant Analysis (LDA), Feed forward Neural network (Multilayer perceptron (MLP)), and Adaboost.

The following was the learning problem that we were faced with:

Given/Known:
Feature Space: $X \in R^{11}$
Label space: $y = \{0,1\}$
Labeled Training set: $\mathcal{D} = \{(\mathbf{x}_j, y_j) \in X \times y, j = 1, \ldots, 264\}$

Goal:
Build and test classifiers that could accurately map a sailor's feature vector in the feature space (a sailor from the labeled training data) to its appropriate label in the label space. In addition, help in determining which features are more important than others in deciding the label. For fulfilling our goal, we built all the algorithms listed above. The main objective behind using these algorithms was to leverage their ability in performing classification tasks. All of the algorithms listed above are standard algorithms that have been used extensively in binary as well as multiclass classification. Algorithms such as SVMs, Logistic regression, and LDA help in linearly separating the data points and classifying them accordingly. The general form of algorithms such as logistic regression and SVMs include a cost function which is to be minimized in the course that the algorithm runs so as to find the optimal parameters which could help in correctly classifying the data. The general cost functions for both of these algorithms are given below:

$$\frac{1}{2}\|\mathbf{w}\|^2 + \sum_{j=1}^{n} C_j \,\text{hinge}\left(\, y_j\left(\mathbf{w}^{\mathrm{T}}\,\mathbf{x}_j \;+\; b\right)\right)$$

Figure 4: The cost function for SVM algorithm

where **w** denotes the weights parameter (associated with each feature in a feature vector), C_j is the penalty term associated with each data vector in the training set, and b is the offset parameter.

$$g(\boldsymbol{\Theta}) = \lambda\left(\sum_{\ell=1}^{m}\|\boldsymbol{\theta}_\ell\|^2\right) + \text{NLL}(\boldsymbol{\Theta})$$

Figure 5: The loss function of logistic regression

where λ is the regularization parameter, θ_l is vector containing weights and offset parameters (for a given class) and is given by:

$$\boldsymbol{\theta}_\ell = \begin{pmatrix} \mathbf{w}_\ell \\ b_\ell \end{pmatrix}$$

where **w** and b are the weight and offset parameters associated with a given label and NLL is the negative log likelihood which is given as

$$\text{NLL} = -\sum_{j=1}^{n} \ln\left(p(y_j|\mathbf{x}_j, \boldsymbol{\Theta})\right)$$

Now in both of these cost functions, there is a penalty term and regularization term which helps in making sure that the parameters to be learnt do not become unconstrained and keep on increasing without bound. Thus, this was one of the main reasons to select SVM and logistic regression for our classification tasks. Also, the regularization terms help in limiting the overfitting of the algorithm on the training data and in addition, there are also weights terms (in C and N.I. terms) where you can give classes different class weights in order to make the algorithm give more priority to certain minority classes (suitable for class imbalance problems). SVMs and logistic regression could also be used to fit nonlinear boundaries with the help of kernel methods.

Random forest and decision trees were mainly used in this project for their ability to show their "decision making" process. Decision trees classify the data using true and false responses to certain questions. Random forest is an ensemble of decision trees, where different decision trees are trained on different subsets of features of the input data and at the end, the predictions from each of the trees are averaged together to form a unified prediction which is the output of the random forest (bagging). Random forest and decision tree help in visualizing which features are important in determining the label. Thus, random forest and decision trees were included in this project for determining feature importance. We also used adaboost as a gradient boosting algorithm in our study (as a way to improve the accuracy of weak predictors). We realized that adaboost has a high runtime complexity while experimenting with the different algorithms and hence, due to time constraints we were only able to apply adaboost with one other algorithm (SVM) as a base estimator. We also used MLP in our project in order to capture any non-linearity in the training data set (all the other methods could only apply linear or fixed nonlinear decision boundaries to the dataset) in an attempt to learn the actual nonlinearity of the data.

7. Implementation Methodology and Experiments

For implementing the algorithms mentioned before, we employed the following workflow:

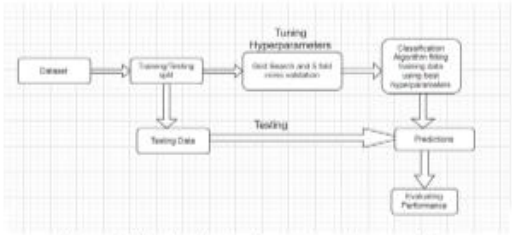


Figure 6: Showing the implementation of our workflow.

Our methodology included first using the entire dataset and then splitting it into training and testing batches. We performed an 80/20 split of the dataset with 80% of the data belonging to the training set and the rest of the data belonging to the testing set. We then normalized the training set by subtracting each feature column with its mean and dividing by its variance and then we applied the same normalization (with the mean and variance vector of the training dataset) to the testing dataset. Now, we used the training data to perform hyperparameter tuning. We did this by using grid search and 5-fold cross validation. We defined a grid space of hyperparameters (unique to an algorithm) for a model to try and then performed 5-fold cross validation with the training data (in which the training data is split into different batches: one for validation and the other for training and then the appropriate hyperparameters along with the weights and offset factors learnt during the training are applied on the validation set. This is done for a total of 5 times with the accuracy scores (CCR) obtained from each of the five splits then being averaged for a given set of hyperparameters). At the end, whichever pair of hyperparameters got the maximum accuracy were selected as the optimal hyperparameters for our implementation. After performing the hyperparameter tuning, we fitted the model with the optimal hyperparameters on the training set and then finally tested the model on the testing set. We evaluated the performance of our model based on performance metrics such as accuracy (CCR), F1 score, recall, and macro average. We used metrics such as F1 score and recall because we realized that our dataset had a class imbalance problem. Majority of our labels were of class 0 and class 1 was in minority (This is because very few sailors get the

All-American status at the end of the sailing year). The histogram below displays the fact:

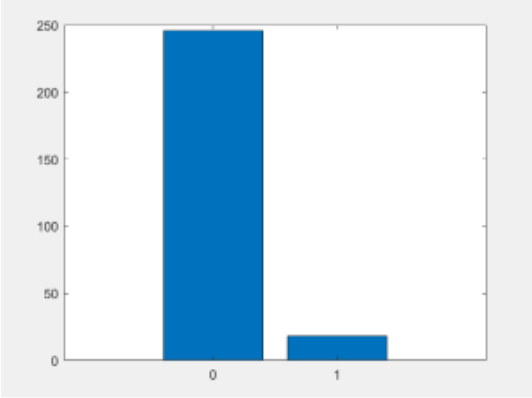


Figure 7: The histogram of class labels for the entire dataset.

We realized we had a class imbalance problem when we performed leave one out cross validation on the training set (before we implemented our ML workflow) that is we trained on 263 training vectors and tested on 1 and repeated this 264 times (once for each vector in the dataset). We realized that the algorithm only predicted class 0 as its final label and never class 1. Therefore, we tried to vary the class weights factor in the algorithms as well as used performance metrics such as F1 score and recall in order to better evaluate the performance of our model. The final hyperparameters as well as the final class weights for different algorithms is as follows:

SVM: {Class weights: (class 0 -0.1, class 1 - 1), C (regularization penalty) = 0.1, kernel = linear}

Logistic Regression: {Class weights: (class 0 -0.1, class 1 - 1), penalty = l2, kernel = linear, solver = liblinear}

LDA: {n_components = none, shrinkage = 0.1, solver = lsqr}

Random Forest: {max_depth = 3, n_estimators = 103}

Decision Tree: {criterion: entropy, max_depth = 3}

We didn't have enough time to find the hyperparameters for the MLP and adaboost. Therefore, we did some manual hyperparameter tuning (found the range of hyperparameters by manually varying them and implementing them on the training dataset). We found the following hyperparameters for:

MLP: {optimizer = adam, loss = binary_crossentropy, layers = 3, neurons = (32,8,1), activation_function = (Leaky_relu, Leaky_relu, sigmoid), learning_rate = 0.0002}

Adaboost: {n_estimators = 50, learning_rate = 1}.

We then proceeded to perform the testing by testing the model on the testing dataset (20% of the dataset).

8. Experimental Results

Algorithm	Accuracy(CCR)	F1 score (Weighted Average)	Recall (Weighted Average)	Precision (Weighted Average)
SVM	0.85	0.89	0.87	0.91
Logistic Regression	0.8	0.84	0.81	0.91
LDA	0.92	0.94	0.92	0.96
Adaboost (with SVM as base estimator)	0.96	0.94	0.96	0.93

Table 1: The results of our different algorithms in predicting the labels

Algorithm	Precision (for class 1)	F1 score (for class 1)	Recall (for class 1)
SVM	0.17	0.22	0.33
Logistic Regression	0.11	0.17	0.33
LDA	0	0	0
Adaboost (with SVM as base estimator)	0	0	0

Table 2: Results for the different algorithms for predicting class 1

In table 1, we used the weighted average of the F1 score, precision, recall (weighted average implies that the number of instances of the class were considered when the precision, f1 and recall scores for the different classes were calculated)
In Table 2, we used the precision, recall, and F1 score of only class 1(that is we used the precision, recall and F1scores only for class 1). All of this information was obtained using the function classification_report and is reported in appendix as figures 11-14.

From the above tables, we can see that the LDA and adaboost with SVM as base estimators were not good estimators for class 1. They predicted every testing feature vector as class 0. These algorithms did not really learn the mapping from the feature space to the label space correctly. The adaboost with SVM as its base estimator was not tuned for hyperparameters (because of time constraint) and hence, that might be one of the reasons as to why the adaboost algorithm might not have been able to predict class 1. Since adaboost and LDA were only predicting class 0, their accuracy, and weighted

average scores of F1 score, precision and recall were more than those of Logistic regression and SVM. However, Logistic regression and SVM were able to predict class 1 to a certain degree because of the class_weights hyperparameter.

Since, one of our goals was to also perform feature selection and see which features are the most important in determining the label, we used random forest and decision trees in that regard.

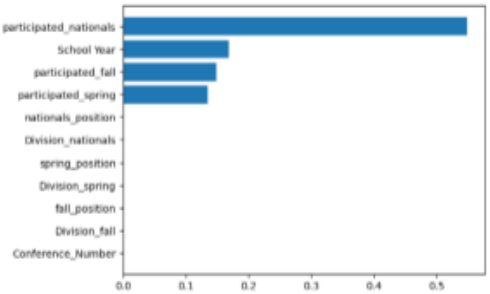


Figure 8: Different feature weights obtained from random forest algorithm.

We can see that some of the top features which helped in the decision-making process were whether the sailor participated in the national's competition, what was the school year of the sailor, and whether the sailor participated in the fall and spring regattas. The other features didn't contribute as much to the decision process. This makes sense because most of the All-American awards are given to sailors who secured a high rank in the national regatta. Therefore, if a sailor didn't participate in nationals (or was swapped out or swapped in that is he/she didn't sail for the entire regatta) then their chance of getting an All American is heavily changed or influenced/reduced. We also saw a similar weight distribution in logistic regression and SVM (since we used linear kernels in these algorithms, we could actually see which weights associated with what features were the highest). The weight distribution in logistic regression and SVM matched with that seen in random forest classifier and we have attached these weight figures (for only SVM) in appendix as figure 15.
We also wanted to see out of all the positions that a sailor got in a regatta (fall, spring, nationals), which one among these would influence the chance of getting an All American the most. We used decision trees for this and used only three features out of the 11 (position obtained in nationals, position obtained in spring, and position obtained in fall). We have pasted a figure regarding that in the appendix (Figure 10). We saw that the position obtained in nationals regatta was the most important factor and then came the fall championship (this matches what we know from domain knowledge).

At last, we also wanted to see if we just used our top feature (participated in nationals) as our only feature, what will the accuracy look like. We used the SVM algorithm (since we got the highest CCR from there) and tried only one feature from our dataset. We found out that we got an average accuracy of about 0.8 (CCR) by using just one feature. Thus, participating in nationals was truly the single most important feature in our dataset according to the algorithms we implemented.

9. Conclusion

A large part of this project was held back by the creation of the dataset. Although the algorithms could be created and developed in parallel with the creation of the dataset, a large portion of the project relied on the completion of the dataset. This was quite a bottleneck in the project and delays in the creation of the dataset had crippling effects on the efficiency of the project. Additionally, we did not expect multiple iterations of the dataset. Throughout the entire process, we developed more than 5 different datasets, each taking between 5-20 hours a piece. During this process, training of the algorithms was halted as we worked to finish the datasets. In the future we hope to create a more efficient process for creating the datasets allowing large datasets to be created in much less time. With the addition of basic web scraping tools, a large dataset encapsulating more than the current two years will be possible which will allow multiple years to be compared against each other to illuminate trends and patters. As a result, the next steps in the project will not be updating the ML algorithms but rather the data collection process.

10. Appendix

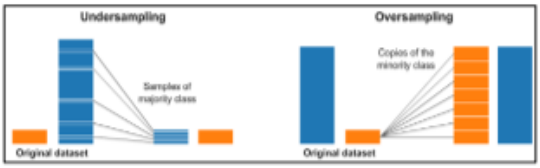


Figure 9: Actions to counteract the class imbalance problem.

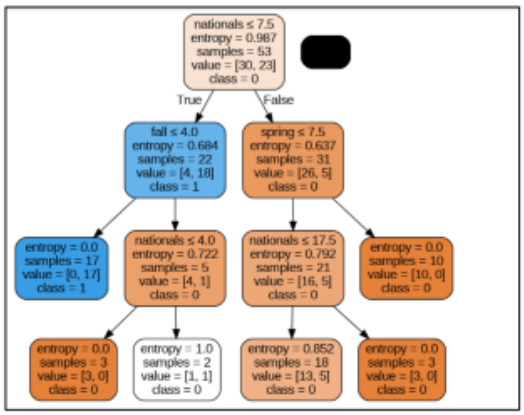


Figure 10: Decision Trees results from dataset 3 with 3 features: Fall Championship, Spring Championship, and National Championship.

	precision	recall	f1-score	support
0	0.96	0.90	0.93	50
1	0.17	0.33	0.22	3
accuracy			0.87	53
macro avg	0.56	0.62	0.58	53
weighted avg	0.91	0.87	0.89	53

Figure 11: Results for SVM

	precision	recall	f1-score	support
0	0.96	1.00	0.98	51
1	0.00	0.00	0.00	2
accuracy			0.96	53
macro avg	0.48	0.50	0.49	53
weighted avg	0.93	0.96	0.94	53

Figure 12: Results for Adaboost with SVM as base estimator

	precision	recall	f1-score	support
0	0.95	0.84	0.89	50
1	0.11	0.33	0.17	3
accuracy			0.81	53
macro avg	0.53	0.59	0.53	53
weighted avg	0.91	0.81	0.85	53

Figure 13: Results for Logistic Regression

	precision	recall	f1-score	support
0	0.98	0.94	0.96	52
1	0.00	0.00	0.00	1
accuracy			0.92	53
macro avg	0.49	0.47	0.48	53
weighted avg	0.96	0.92	0.94	53

Figure 14: Results for LDA

```
[[ 0.15199772  0.43069007  0.0447889  -0.14500768 -0.03085002  0.00569388
   0.0877784  -0.03397692 -0.46772537  0.07888739 -0.17724663]]
```

Figure 15: Weights for SVM.

11. Acknowledgements

We would like to thank Nicholas Lorentzen for sitting down with us at the beginning of the project. We would also like to thank Professor Prakash Ishwar for his suggestions relating to project implementation and dataset creation.

References

[1] "ICSA Hall of Fame." *Inter-Collegiate Sailing Association*, collegesailing.org/hall-of-fame/all-american/all-american-trophy-overview. Accessed 3 May 2024.

[2] Adhatrao, K., Gaykar, A., Dhawan, A., Jha, R. &Honrao, V. Predicting students' performance using ID3 and C4.5 classification algorithms. International Journal of Data Mining & Knowledge Management Process, 3(5), 2013, 39-52.

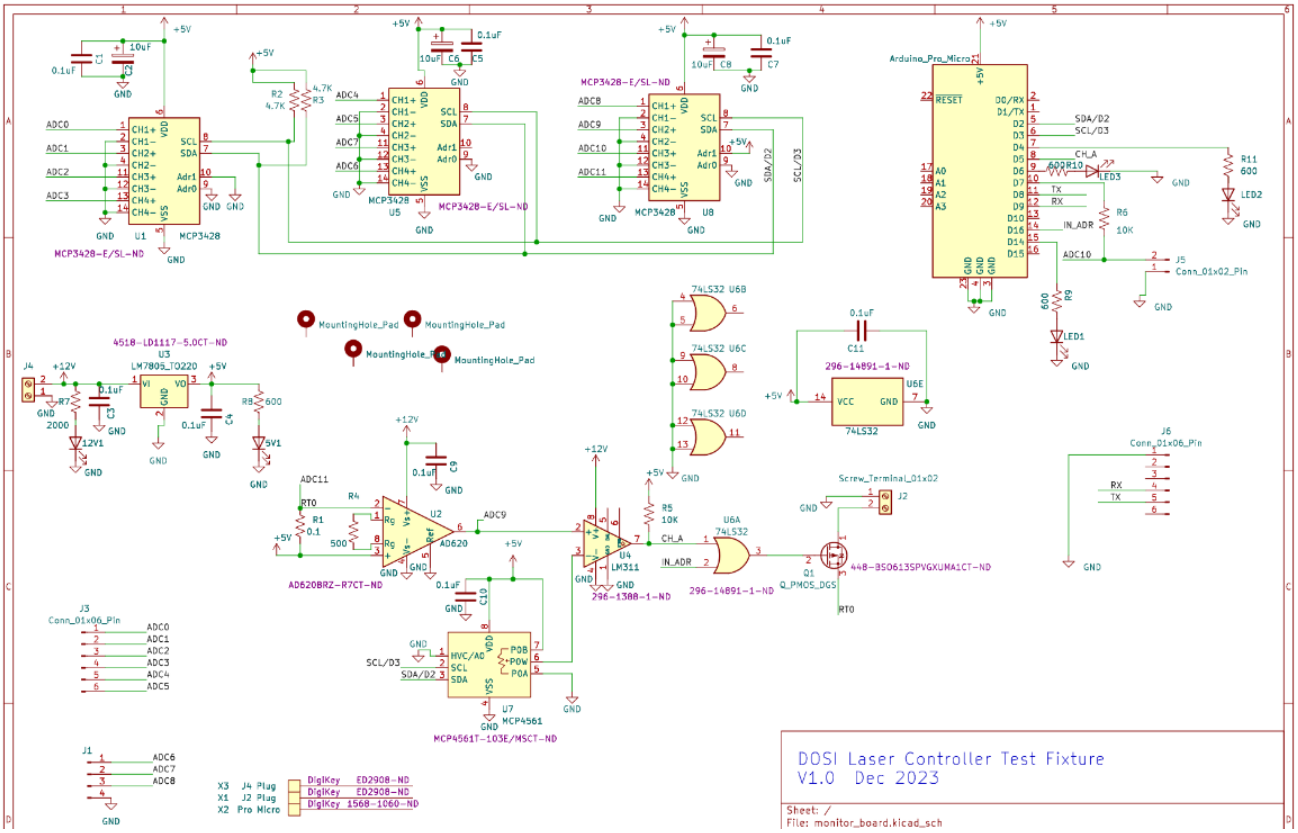
[3] "10 Techniques to Solve Imbalanced Classes in Machine Learning (Updated 2024)." *Analytics Vidhya*, 17 Jan. 2024, www.analyticsvidhya.com/blog/2020/07/10-techniques-to-deal-with-class-imbalance-in-machine-learning/#:~:text=One%20of%20the%20widely%20adopted,class%20(over%2Dsampling).

[4] SailRank. "College Sailing Rankings and Statistics." *SailRank*, sailrank.com/. Accessed 1 May 2024.

Electronics

This is a PCB schematic for an electronic board I designed to test laser controller boards. Unverified laser controller boards may have faults such as short circuits, which can damage expensive laser diodes when connected. To safeguard the laser diodes, I developed this board to monitor the current supplied to the laser controller.

Using MOSFETs and logic gates, the board can disconnect the laser controller when the current exceeds a predefined threshold. This feature helps identify faults in the laser controller board and ensures the safety of the laser diodes. This testing and debugging tool was utilized in the Biomedical Optical Technologies Lab at Boston University to support the optical setups of PhD students. This is the schematic:



Undergraduate Honors Thesis

The following report will summarize the need as well as the impact of my undergraduate honors thesis:

Project Summary

Large Language Models (LLMs) such as GPT-2, are crucial in AI due to their advanced text generation abilities. However, these models generate tokens based on prior words, which can lead to errors. If a predicted token is incorrect, all subsequent predictions may become inaccurate, compromising the context. To address this limitation, this project proposes using the Viterbi algorithm to improve sequence predictions. Instead of selecting tokens one-by-one, the Viterbi algorithm maximizes the probability of the entire sequence, considering future tokens to help predict even the current one. This approach enhances the overall coherence and accuracy of generated text. The project will involve testing the GPT-2 model and a modified version of GPT-2 integrated with the Viterbi algorithm, comparing their ability to optimize word sequences based on an initial context. The goal is to assess whether the modified architecture improves language model performance.

1 Need for this Project

Large Language Models (LLMs) are transformer-based models that have revolutionized natural language processing (NLP) by enabling a wide range of text-related operations, such as text recognition, generation, translation, and more. Among the most prominent of these models is Generatively Pretrained Transformer-2 (GPT-2), which has gained widespread adoption across various sectors due to its superior ability to generate and manipulate text. Industries such as marketing, customer service, gaming, education, healthcare, and content creation have integrated GPT-2 into their workflows to enhance efficiency and improve the quality of automated interactions and content delivery.GPT-2 has a vocabulary set (the words which it has been trained to use and understand) is about 50,000 tokens. The model predicts the next token in a sequence based on the prior context of words. It does this by calculating the probability of each potential token given the preceding string and selecting the one with the highest probability that is it implements the following equation [1]:

$$\mu_k(x_k) \in \arg\max_{x_{k+1}} p(x_{k+1} \mid x_k).$$

Equation 1.1 x_k is the previous context or sequence of states and x_{k+1} is the next potential token.

This mechanism has one inherent limitation which is that the model predicts one token at a time, and an incorrect prediction (or the one which is not appropriate to the context) early in the sequence can lead to a cascading effect of errors. This means that if a token is incorrectly selected, all subsequent tokens will be based on this flawed token, resulting in an output that can deviate significantly from the original context or intended meaning. This flaw can manifest in various ways, such as producing text that is off-topic, illogical, or even factually incorrect, particularly in cases where maintaining precise language or factual accuracy is essential, like in legal documents or medical reports.

To mitigate this issue, there is a need to refine the token prediction process to ensure more reliable and contextually accurate text generation. One promising approach is to use algorithms like the Viterbi algorithm, which evaluates entire sequences rather than individual tokens. The Viterbi algorithm seeks to maximize the probability of the entire sequence by considering both past and future tokens during the prediction process, thus reducing the likelihood of incorrect token generation and its subsequent cascading effect. By considering future tokens, the model can ensure that the current token fits within the broader context, leading to more coherent and accurate output.

Refining models like GPT-2 to reduce errors and enhance accuracy is crucial not only for addressing current limitations but also for advancing the development of future language models. The research and innovations we implement today lay the groundwork for more powerful and reliable LLMs in the future. As language models evolve, they will require even more sophisticated algorithms and techniques to handle increasingly complex tasks and larger datasets, making today's improvements foundational for future advancements.

In summary, this project seeks to address the limitations of current token-by-token prediction methods in GPT-2 by exploring the use of algorithms like the Viterbi algorithm to enhance sequence prediction. This could potentially result in better contextual responses, benefiting industries and future developments in AI language models.

2 Problem Statement and Deliverables

2.1 Problem Statement

The goal of our project is to mitigate a fundamental flaw in the token-by-token generation process, where an incorrect token early in the sequence can cause subsequent tokens to deviate from the intended meaning, resulting in incoherent or irrelevant outputs. The Viterbi algorithm, traditionally used in Hidden Markov Models (HMMs), identifies the most likely sequence of states that lead to observed outputs.

Our design based on implementing Viterbi algorithm with a transformer architecture like GPT-2 can be particularly useful for token generation. Instead of selecting the most probable token at each step, the algorithm will help maximize the probability of the entire sequence. In our design, we will create a trellis of possible token paths with the tokens representing the hidden states in the diagram. Instead of choosing the top token at each iteration, the model will traverse this trellis and select the path that maximizes the overall probability of the entire sequence. Therefore, the key idea is that even the future tokens (along with the past tokens) are used to inform and adjust the current prediction. This backward- and forward-looking feature ensures that the model can adjust current predictions based on likely future outcomes and the given past context, resulting in text that could potentially flow more logically.

For example, at the end of the first iteration, the second-most probable token might lead to a path with a higher overall probability than the top token. By following the entire trellis, the Viterbi algorithm can identify this situation and return a string corresponding to the path states of the second token which fits better within the overall context, whereas the GPT-2 architecture will just pick the most probable token and inherently lead to a path where the entire probability of the sequence is compromised.

2.2 Deliverables

A computational algorithm which is capable of:

1. **Creating Computational Trellis**: Constructing a Viterbi trellis using the top k tokens generated at each iteration of GPT-2. The states in the trellis represent potential tokens, and the transition probabilities between states will be derived from the transformer's softmax layer.
2. **Token Sequence Optimization**: This model architecture will optimize token generation by considering future token possibilities and refining predictions as the sequence unfolds. It will ensure that the final output string is globally optimized in terms of coherence, accuracy, and contextual appropriateness, unlike traditional methods that rely solely on local token-by-token predictions.
3. **Efficient Computational Performance**: Another deliverable is to create the model that can efficiently compute the optimal token path without excessive computational costs.
4. **Scalable and Adaptable Framework**: The algorithm and model will be developed in a way that allows scalability for future models or additional architectures. This means the project could be expanded beyond GPT-2 to improve other LLM architectures or be adapted for specific industry needs, enabling broader applicability.
5. **Transparency**: This project will also focus on making the decisions of the Viterbi algorithm more transparent and interpretable to a human user (that is we are trying to make the architecture of the model more than a mere black box).

3 Visualization

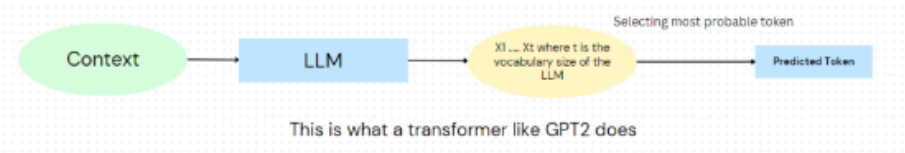


Figure 1.1 Here is how the current transformer architecture of GPT-2 looks like. You first feed it with some context (initial input) which goes to the pretrained model. The model then outputs a layer of vocabulary words/tokens as potential candidates for the next word. The transformer then selects the most probable word from this list and returns it as the next word of the given prompt.

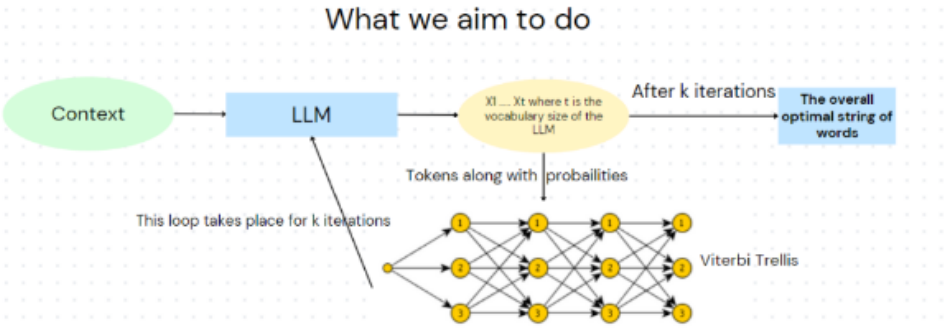


Figure 1.2 This visualization represents what we aim to do. In this picture we can see that we first feed in the context to the LLM and the LLM produces the vocabulary words. After this is done, these vocabulary words (or top n of these words where n is some experimentally determined memory constraints conscious number) are filled into the first column of the Viterbi Trellis (here the trellis shows n = 3). The top 3 tokens are determined on the likelihood of seeing those given the initial context (so we essentially pick the top 3 most probable tokens). After this we go back to the LLM and add to the context the new token which is filled in the Viterbi trellis and again repeat the same process of generating the candidate tokens and filling the Viterbi trellis new layer. We repeat this process a k (in this case 4) number of times and populate the Viterbi trellis (Note – In this example we are assuming that the new tokens produced by all the previous tokens have the same top 3 candidate tokens which is not always the case!). Once we completely fill the Viterbi trellis, we implement the Viterbi algorithm which finds the most probable path and returns the sequence of tokens which are encountered during that path. These sequence of tokens is then returned as the final output of the transformer. Thus, we are post-processing the transformer outputs.

4 Competing Technologies

The following are some of the existing competing technologies/standard algorithms which are implemented in the transformer token generation and selection processes. These algorithms are implemented widely in the Natural Language Processing area. Some of the key insights derived from these algorithms for my own project are as follows:

Beam Search
Features and Requirements:

- This algorithm is a widely used search algorithm in the sequence generation tasks to explore multiple candidate sequences simultaneously.
- It works by computing the most likely sequence by keeping the top k sequences (where k is the beam width) at each step offering a more comprehensive exploration of possible sequences than greedy token-by-token generation.
- It employs heuristic rules in pruning and finding the top k sequences (so that it doesn't evaluate the entire token space) thus, focusing the scope of its search.

Incorporation to my project: Our project also requires efficient memory management and narrowing the search to a fixed number of tokens, rather than considering the entire token vocabulary. This approach helps avoid an exponentially growing algorithm. As a result, some of the efficient data structures implementation in Beam Search inspires our model's implementation too. However, a key difference lies in the scope: Beam Search operates primarily during the pre-token generation phase, while our algorithm is applied in the post-token generation process to refine the selected tokens. Despite this distinction, the underlying principles of Beam Search still guide the workings of our algorithm.

Top-k Sampling and Nucleus Sampling
Features and Requirements:

- These are algorithms used to in limiting the randomness of generated text by selecting a subset of tokens based on their probabilities and then selecting a token from only that subset of tokens randomly.
- A mechanism to filter and rank tokens based on their probabilities.
- Configurable parameters to determine the sampling threshold (k for Top-k or cumulative probability for Nucleus Sampling).
- Implementing some balance between the diversity of the randomly picked token and the contextual sense of that token.

Incorporation in my Project: These methods rely heavily on ranking tokens, which also aligns with my project's need to evaluate token probabilities appropriately in order to build the Viterbi trellis. However, while they focus on controlling randomness, our method seeks to optimize the entire sequence's path by not working in a somewhat "stochastic" setting rather finding the path of maximum probability deterministically. The focus on doing effective token ranking and pruning (the k parameter) is a shared requirement for my project as well in order to narrow the search space and still get meaningful, derivable results.

5 Engineering Requirements

These requirements can be categorized into the following categories: Functions, Objectives, constraints, and the means.

Functions:

- **Viterbi Trellis Construction:** The Viterbi trellis is appropriately populated with tokens and transition probabilities at each iteration.
- **Influence Token Generation**: The main function of this project is to post process the transformer outputs in order to inform the process of token generation of a transformer architecture.
- **Returning Optimized String**: The predictive valid sequence of words that come out of our modified architecture align with the most probable sequence of words as decoded from the Viterbi trellis.
- **Scalability Across Different Transformers**: The algorithm is easily adaptable to other transformer architectures such as GPT-3 etc.

Objectives:

- **Prediction Accuracy/Contextual Appropriateness**: The Viterbi algorithm in decoding the maximum likelihood path should give word sequences which are on average 5% to 10% more accurate than what the transformer architecture of GPT-2 will give. The contextual accuracy could be either evaluated by humans or metrics such as BERT score.
- **Fast Inference Time**: The Implementation of Viterbi algorithm with the transformer should result in a fast inference time of the path and should ideally not grow exponentially with increasing iterations. Ideally, the strings should be computed within the following range: 1.5 to 2 times the time it will take the transformer to give results.
- **Handle Memory Issues**: The algorithm should be able to handle large token spaces and vocabulary of the Transformers which is about 50,000 words.
- **Maximize Overall Likelihood:** Maximize the overall likelihood of the sequence rather than just picking the most probable token one by one (as is done in a transformer). Ideally, the transformed architecture should be able to give a more or same probable path sequence as the transformer 90% of the time.

Constraints/Challenges:

- **Multiple Paths**: Presence of multiple equally probable non intersecting paths from the initial to the final state in the Viterbi trellis, resulting in multiple non overlapping valid sequences as the output of the Viterbi algorithm given a prompt.
- **Compute Power:** Limited compute power available to test the transformers and our algorithm on large scale inputs.

Means:

- Pretrained models of GPT-2 available on Hugging Face.

The above report summarizes the need of my project. Throughout this ongoing semester, I was working on constructing my Viterbi Modified Transformer architecture. Here is the algorithm that **I developed**:

Algorithm 1 Viterbi Modified GPT-2 algorithm

Input: The Viterbi Trellis depth, T and the pruning constant, c. An input text string.

PHASE 1

Set the input string into the GPT-2 model and find the probability distribution, $P_X(x)\ \forall\ x \in$ Token, where Token $= x_1 \dots x_K$ contains all the tokens in the vocabulary of the transformer.

Select the top c candidate tokens in an array *Unique* along with their transition probability.

Set $s = c$.

for $t = 2 \dots T$ **do**

  for $m = 1 \dots s$ **do**

    Repeat step 1 in PHASE 1 with an input sentence that is modified by adding context up till *Unique*$[m]$ and passing the sentence to the GPT-2 model.

  end for

  Select all unique tokens in the given iteratoin.

  Set $s =$ number of unique tokens.

end for

return Constructed Viterbi Trellis.

PHASE 2

Run the Viterbi Algorithm implementation on the Constructed Trellis.

return Best Path nodes, Best Path Probability

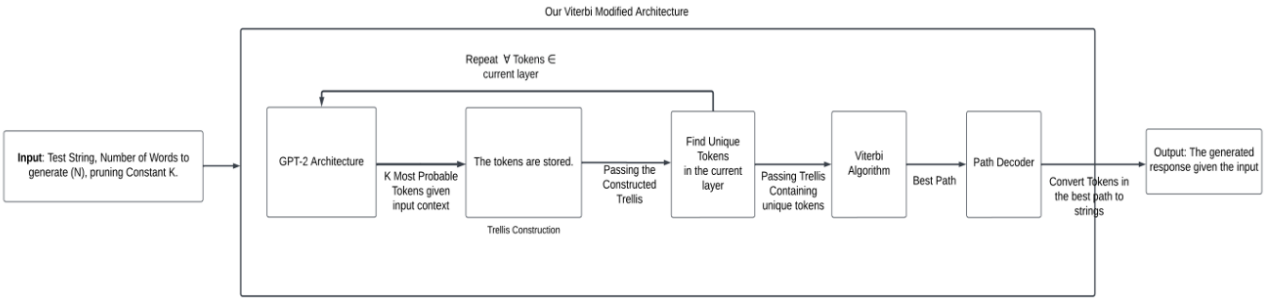
PHASE 3

Call the Decoding String function on the Best path and get the corresponding string associated with that part.

return The decoded string, Best Path Probability

This figure shows the high-level working of my modified Viterbi and Transformer decoding algorithm. PHASE-1 represents the trellis generation which is required before starting PHASE-2. PHASE-2 represents running the Viterbi algorithm on the trellis. PHASE-3 represents using the decoder to interpret the selected tokens back to their string representations.

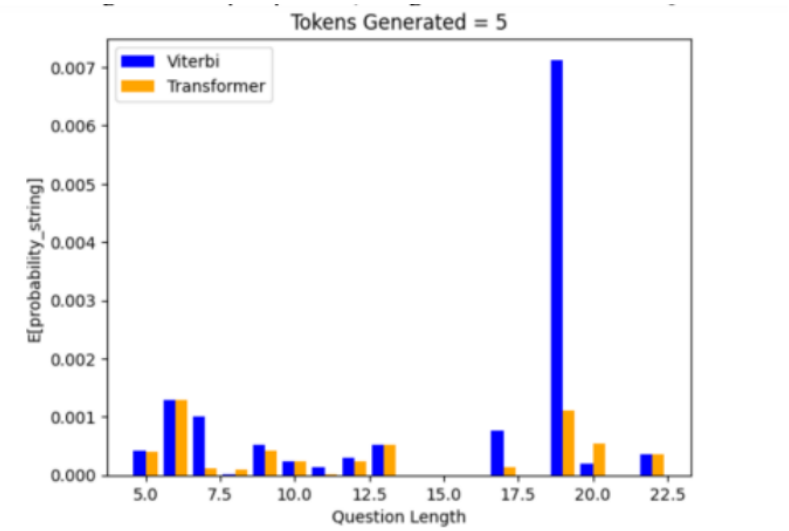
This is the high-level block diagram of my architecture:



The process begins with an input string fed into the transformer model (e.g., GPT-2). At each iteration, the transformer generates a probability distribution over the vocabulary for the next token. From this distribution, the Top K most probable tokens are selected, where K is a user-defined pruning constant, and stored as nodes in the trellis for the current layer. A partial sequence is constructed by appending the token to the string generated so far, and these partial sequences are

then fed back into the transformer to generate new tokens and their distributions for subsequent layers, updating the trellis iteratively. This process continues for N iterations until the trellis is fully constructed, containing all potential paths formed by the selected Top K tokens at each layer. Once the trellis is complete, it is passed into the Viterbi algorithm, which evaluates all possible paths and identifies the sequence with the highest likelihood. The optimal path of tokens is then passed to the Path Decoder, which converts the selected token sequence into corresponding output string. The resulting string is returned as the final output of the modified transformer model.

These are some of the preliminary results:



In this figure, Y axis represents the expected value of the probability of the output string and the X axis represents the length (number of words) in the input prompt (question) given to my modified architecture and the traditional greedy Transformer architecture. In this particular case I asked both the models to produce 5 output words or tokens. We can see that on average, my algorithm produces higher probability strings than the traditional transformer architecture showing that my model has the capability to replace the existing transformer decoding algorithms.

To read the full report (20 pages), you can go to the following link:
https://github.com/jiveshj/SeniorThesis.git and click on the file SemesterReport.pdf.

To check my presentation on this topic, you can go to the same link and click on the SemesterPresentation.pptx.

You can also check the video demonstration of my prototype that I developed this semester by going on the following link:

https://drive.google.com/file/d/1c29_eaOyFmJc4uWQL2ROQ0tRLNtjSxH5/view?usp=sharing

Other Interests - Mathematics

I also enjoy working on math related problems and using probability theory to prove the convergence rate of many optimization algorithms to the true minimizer of the loss function.

1. Show that

$$-\mathbb{E}\left[\sum_{t=1}^{T}\alpha_t\mathcal{L}(\mathbf{w}_\star)\right] \leq \mathbb{E}\left[\sum_{t=1}^{T}\left(\sum_{i=1}^{t-1}\alpha_i\right)\mathcal{L}(\mathbf{w}_t) - \left(\sum_{i=1}^{t}\alpha_i\right)\mathcal{L}(\mathbf{x}_t)\right] + \frac{\|\mathbf{w}_\star - \mathbf{y}_1\|^2}{2\eta} + \frac{\sigma^2\eta\sum_{t=1}^{T}\alpha_t^2}{2} + \frac{\eta}{2}\mathbb{E}\left[\sum_{t=1}^{T}\alpha_t^2\|\nabla\mathcal{L}(\mathbf{x}_t)\|^2\right]$$

Solution:
For this question, we can first start by noting through convexity that:

$$l(x_t, z_t) - l(w_\star, z_t) \leq \langle \nabla l(x_t, z_t), x_t - w_\star\rangle$$

which implies:

$$\sum_{t=1}^{T}\alpha_t(l(x_t, z_t) - l(w_\star, z_t)) \leq \sum_{t=1}^{T}\alpha_t\langle\nabla l(x_t), x_t - w_\star\rangle$$
$$= \sum_{t=1}^{T}\alpha_t\langle\nabla l(x_t, z_t), x_t - y_t\rangle + \sum_{t=1}^{T}\alpha_t\langle\nabla l(x_t, z_t), y_t - w_\star\rangle$$

where $\alpha_t\langle\nabla l(x_t, z_t), y_t - w_\star\rangle = \langle g_t, y_t - w_\star\rangle$ and we know from theorem 15.1 that

$$\sum_{t=1}^{T}\langle g_t, y_t - w_\star\rangle \leq \frac{||w_\star - y_1||^2}{2\eta} + \frac{\eta\sum_{t=1}^{T}||g_t||^2}{2}$$

so therefore, we get:

$$\sum_{t=1}^{T}\alpha_t(l(x_t, z_t) - l(w_\star, z_t)) \leq \sum_{t=1}^{T}\alpha_t\langle\nabla l(x_t, z_t), x_t - y_t\rangle + \frac{||w_\star - y_1||^2}{2\eta} + \frac{\eta\sum_{t=1}^{T}||g_t||^2}{2}$$

(Equation 1)

Now, we know from notes that

$$\alpha_t(x_t - y_t) = \textstyle\sum_{i=1}^{t-1}\alpha_i(w_t - x_t)$$

$$\implies \langle\nabla l(x_t, z_t), \alpha_t(x_t - y_t)\rangle = \sum_{i=1}^{t-1}\alpha_i\langle\nabla l(x_t, z_t), (w_t - x_t)\rangle$$

Now, we know from convexity that $l(w_t, z_t) \geq l(x_t, z_t) + \langle\nabla l(x_t, z_t), w_t - x_t\rangle$ so putting all of this together in equation 1 we will get the following:

$$\sum_{t=1}^{T}\alpha_t(l(x_t, z_t) - l(w_\star, z_t)) \leq \sum_{t=1}^{T}\sum_{i=1}^{t-1}\alpha_i(l(w_t, z_t) - l(x_t, z_t)) + \frac{||w_\star - y_1||^2}{2\eta} + \frac{\eta\sum_{t=1}^{T}||g_t||^2}{2}$$

Now, if we subtract $\sum_{t=1}^{T}\alpha_t l(x_t, z_t)$ from both sides, we will get:

$$\sum_{t=1}^{T} - l(w_\star, z_t)) \leq \sum_{t=1}^{T}(\sum_{i=1}^{t-1}\alpha_i)l(w_t, z_t) - (\sum_{i=1}^{t}\alpha_i)l(x_t, z_t) + \frac{||w_\star - y_1||^2}{2\eta} + \frac{\eta\sum_{t=1}^{T}||g_t||^2}{2}$$

Now, we can take expectation on both sides and we know that $\mathbb{E}[l(w_t, z_t)] = L(w_t, z_t)$, $\mathbb{E}[l(w_\star, z_t)] = L(w_\star)$ and $\mathbb{E}[||g_t||^2] \leq \alpha_t^2\,\mathbb{E}[||\nabla L(x_t)||^2 + \sigma^2]$ so we get the following equation:

$$\sum_{t=1}^{T} - \mathbb{E}[L(w_\star))] \leq \mathbb{E}[\sum_{t=1}^{T}(\sum_{i=1}^{t-1}\alpha_i)L(w_t) - (\sum_{i=1}^{t}\alpha_i)L(x_t)] + \frac{||w_\star - y_1||^2}{2\eta} + \frac{\eta\sum_{t=1}^{T}\alpha_t^2\,\mathbb{E}[||\nabla L(x_t)||^2]}{2} + \frac{\sigma^2\eta\sum_{t=1}^{T}\alpha_t^2}{2}$$

which can be further written as:

$$-\mathbb{E}\left[\sum_{t=1}^{T}\alpha_t\mathcal{L}(\mathbf{w}_\star)\right] \leq \mathbb{E}\left[\sum_{t=1}^{T}\left(\sum_{i=1}^{t-1}\alpha_i\right)\mathcal{L}(\mathbf{w}_t) - \left(\sum_{i=1}^{t}\alpha_i\right)\mathcal{L}(\mathbf{x}_t)\right] + \frac{\|\mathbf{w}_\star - \mathbf{y}_1\|^2}{2\eta} + \frac{\sigma^2\eta\sum_{t=1}^{T}\alpha_t^2}{2} + \frac{\eta}{2}\mathbb{E}\left[\sum_{t=1}^{T}\alpha_t^2\|\nabla\mathcal{L}(\mathbf{x}_t)\|^2\right]$$

Hence, proved.

My solution to this problem uses various concepts from algebra, probability theory etc. to prove the given inequality. Using this inequality, we can later prove the convergence bound of the Accelerated gradient descent algorithm. This is shown below:

3. Show that for any $\eta \leq \frac{1}{H}$:

$$\sum_{t=1}^{T} \alpha_t \, \mathbb{E}\left[\mathcal{L}(\mathbf{w}_{T+1}) - \mathcal{L}(\mathbf{w}_\star)\right] \leq \frac{\|\mathbf{w}_\star - \mathbf{y}_1\|^2}{2\eta} + \sigma^2 \eta \sum_{t=1}^{T} \alpha_t^2$$

Solution:
Now, from question 2, we know that

$$\mathbb{E}\left[-\mathcal{L}(\mathbf{x}_t)\right] \leq \mathbb{E}\left[-\mathcal{L}(\mathbf{w}_{t+1}) - \frac{\eta}{2}\|\nabla \mathcal{L}(\mathbf{x}_t)\|^2 + \frac{\eta\sigma^2}{2}\right]$$

so putting this into the final equation of question 1, we will get the following expression:

$$-\mathbb{E}\left[\sum_{t=1}^{T} \alpha_t \mathcal{L}(\mathbf{w}_\star)\right] \leq \mathbb{E}\left[\sum_{t=1}^{T}\left(\sum_{i=1}^{t-1}\alpha_i\right)\mathcal{L}(\mathbf{w}_t) + \left(\sum_{i=1}^{t}\alpha_i\right)(-L(\mathbf{w}_{t+1}) - \frac{\eta||\nabla L(\mathbf{x}_t)||^2}{2} + \frac{\eta\sigma^2}{2})\right] + \frac{\|\mathbf{w}_\star - \mathbf{y}_1\|^2}{2\eta} + \frac{\sigma^2\eta\sum_{t=1}^{T}\alpha_t^2}{2} + \frac{\eta}{2}\mathbb{E}\left[\sum_{t=1}^{T}\alpha_t^2\|\nabla\mathcal{L}(\mathbf{x}_t)\|^2\right]$$

This can be written as:

$$-\mathbb{E}\left[\sum_{t=1}^{T} \alpha_t \mathcal{L}(\mathbf{w}_\star)\right] \leq \mathbb{E}\left[\sum_{t=1}^{T}\left(\sum_{i=1}^{t-1}\alpha_i\right)\mathcal{L}(\mathbf{w}_t) + \left(\sum_{i=1}^{t}\alpha_i\right)(-L(\mathbf{w}_{t+1}) - (\sum_{i=1}^{t}\alpha_i)\frac{\eta||\nabla L(\mathbf{x}_t)||^2}{2} + (\sum_{i=1}^{t}\alpha_i)\frac{\eta\sigma^2}{2})\right] + \frac{\|\mathbf{w}_\star - \mathbf{y}_1\|^2}{2\eta} + \frac{\sigma^2\eta\sum_{t=1}^{T}\alpha_t^2}{2} + \frac{\eta}{2}\mathbb{E}\left[\sum_{t=1}^{T}\alpha_t^2\|\nabla\mathcal{L}(\mathbf{x}_t)\|^2\right]$$

Now, we know that the sum $\sum_{t=1}^{T}(\sum_{i=1}^{t-1}\alpha_i)L(w_t) - \sum_{t=1}^{T}(\sum_{i=1}^{t}\alpha_i)L(w_{t+1})$ telescopes and therefore, we will get the following:

$$-\mathbb{E}\left[\sum_{t=1}^{T} \alpha_t \mathcal{L}(\mathbf{w}_\star)\right] \leq \mathbb{E}\left[-\sum_{t=1}^{T}L(\mathbf{w}_{t+1}) - \sum_{t=1}^{T}(\sum_{i=1}^{t}\alpha_i)\frac{\eta||\nabla L(\mathbf{x}_t)||^2}{2} + \sum_{t=1}^{T}(\sum_{i=1}^{t}\alpha_i)\frac{\eta\sigma^2}{2})\right] + \frac{\|\mathbf{w}_\star - \mathbf{y}_1\|^2}{2\eta} + \frac{\sigma^2\eta\sum_{t=1}^{T}\alpha_t^2}{2} + \frac{\eta}{2}\mathbb{E}\left[\sum_{t=1}^{T}\alpha_t^2\|\nabla\mathcal{L}(\mathbf{x}_t)\|^2\right]$$

which after rearranging gives us:

$$\sum_{t=1}^{T} \alpha_t \, \mathbb{E}[L(\mathbf{w}_{t+1}) - \mathcal{L}(\mathbf{w}_\star)] \leq \mathbb{E}\left[-\sum_{t=1}^{T}(\sum_{i=1}^{t}\alpha_i)\frac{\eta||\nabla L(\mathbf{x}_t)||^2}{2} + \sum_{t=1}^{T}(\sum_{i=1}^{t}\alpha_i)\frac{\eta\sigma^2}{2})\right] + \frac{\|\mathbf{w}_\star - \mathbf{y}_1\|^2}{2\eta} + \frac{\sigma^2\eta\sum_{t=1}^{T}\alpha_t^2}{2} + \frac{\eta}{2}\mathbb{E}\left[\sum_{t=1}^{T}\alpha_t^2\|\nabla\mathcal{L}(\mathbf{x}_t)\|^2\right]$$

Now, noting that $\sum_{i=1}^{t}\alpha_i = \alpha_t^2$ from the definition of α_t. Thus, we get the following:

$$\sum_{t=1}^{T} \alpha_t \, \mathbb{E}[L(\mathbf{w}_{t+1}) - \mathcal{L}(\mathbf{w}_\star)] \leq \mathbb{E}\left[-\sum_{t=1}^{T}\alpha_t^2\frac{\eta||\nabla L(\mathbf{x}_t)||^2}{2}\right] + \mathbb{E}\left[\sum_{t=1}^{T}\alpha_t^2\frac{\eta\sigma^2}{2})\right] + \frac{\|\mathbf{w}_\star - \mathbf{y}_1\|^2}{2\eta} + \frac{\sigma^2\eta\sum_{t=1}^{T}\alpha_t^2}{2} + \frac{\eta}{2}\mathbb{E}\left[\sum_{t=1}^{T}\alpha_t^2\|\nabla\mathcal{L}(\mathbf{x}_t)\|^2\right]$$

Now, after cancelling the terms $\sum_{t=1}^{T}\alpha_t^2\frac{\eta||\nabla L(\mathbf{x}_t)||^2}{2}$ and adding the terms $\frac{\sum_{t=1}^{T}\alpha_t^2\eta\sigma^2}{2}$ we get the following:

$$\sum_{t=1}^{T} \alpha_t \, \mathbb{E}\left[\mathcal{L}(\mathbf{w}_{T+1}) - \mathcal{L}(\mathbf{w}_\star)\right] \leq \frac{\|\mathbf{w}_\star - \mathbf{y}_1\|^2}{2\eta} + \sigma^2 \eta \sum_{t=1}^{T} \alpha_t^2$$

Hence, proved.

4. Choose a value for η such that:

$$\mathbb{E}[\mathcal{L}(\mathbf{w}_{T+1}) - \mathcal{L}(\mathbf{w}_\star)] \leq O\left(\frac{H\|\mathbf{w}_\star - \mathbf{y}_1\|^2}{T^2} + \frac{\sigma\|\mathbf{w}_\star - \mathbf{y}_1\|}{\sqrt{T}}\right)$$

Your choice for η may depend on values unknown in practice, such as $\|\mathbf{w}_\star - \mathbf{y}_1\|$. You would normally have to tune the learning rate to obtain this result without this knowledge.

Solution:

Now, through course notes, we know that $\sum_{t=1}^{T} \alpha_t$ is bounded by $\frac{t^2}{9} \leq \sum_{t=1}^{T} \alpha_t \leq t^2$. Now, from question 3, we know that

$$\sum_{t=1}^{T} \alpha_t \, \mathbb{E}\left[\mathcal{L}(w_{T+1}) - \mathcal{L}(w_\star)\right] \leq \frac{\|w_\star - y_1\|^2}{2\eta} + \sigma^2 \eta \sum_{t=1}^{T} \alpha_t^2$$

$$\implies \mathbb{E}\left[\mathcal{L}(w_{T+1}) - \mathcal{L}(w_\star)\right] \leq \frac{\|w_\star - y_1\|^2}{2\eta \sum_{t=1}^{T} \alpha_t} + \frac{\sigma^2 \eta \sum_{t=1}^{T} \alpha_t^2}{\sum_{t=1}^{T} \alpha_t}$$

Now, using the inequality mentioned at the starting we can see the following:

$$\mathbb{E}\left[\mathcal{L}(w_{T+1}) - \mathcal{L}(w_\star)\right] \leq \frac{\|w_\star - y_1\|^2 (9)}{2\eta T^2} + \frac{(9)\sigma^2 \eta \sum_{t=1}^{T} \alpha_t^2}{T^2}$$

(Equation 1)

Now, again from the above inequality, we can see the following results:

$$\frac{t^2}{9} \leq \sum_{t=1}^{T} \alpha_t = \alpha_t^2 \leq t^2$$

$$\implies \sum_{t=1}^{T} \frac{t^2}{9} \leq \sum_{t=1}^{T} \alpha_t^2 \leq \sum_{t=1}^{T} t^2$$

Now, we know the sum of squares of n natural numbers, so we get the following:

$$\frac{T(T+1)(2T+1)}{54} \leq \sum_{t=1}^{T} \alpha_t^2 \leq \frac{T(T+1)(2T+1)}{6}$$

Therefore, $\sum_{t=1}^{T} \alpha_t^2 \leq O(T^3)$. Therefore, if we substitute that into the equation 1 we will get the following:

$$\mathbb{E}\left[\mathcal{L}(w_{T+1}) - \mathcal{L}(w_\star)\right] \leq O\left(\frac{\|w_\star - y_1\|^2}{\eta T^2} + \frac{\sigma^2 \eta T^3}{T^2}\right)$$

$$\implies \mathbb{E}\left[\mathcal{L}(w_{T+1}) - \mathcal{L}(w_\star)\right] \leq O\left(\frac{\|w_\star - y_1\|^2}{\eta T^2} + \sigma^2 \eta T\right)$$

Now, if we set $\eta = \frac{\|w_\star - y_1\|}{\sigma T^{\frac{3}{2}}}$ and substitute that into the variance term, we will get the following:

$$\sigma^2 \eta T = \frac{\sigma \|w_\star - y_1\|}{\sqrt{T}}$$

and therefore, we will get the following:

$$\mathbb{E}\left[\mathcal{L}(w_{T+1}) - \mathcal{L}(w_\star)\right] \leq O\left(\frac{\|w_\star - y_1\|^2}{\eta T^2} + \frac{\sigma \|w_\star - y_1\|}{\sqrt{T}}\right)$$

Now, recalling our assumption that as long as $\eta \leq \frac{1}{H}$, we can get a upper bound for our T^{-2} term which is :

$$\mathbb{E}\left[\mathcal{L}(w_{T+1}) - \mathcal{L}(w_\star)\right] \leq O\left(\frac{\|w_\star - y_1\|^2 (H)}{T^2} + \frac{\sigma \|w_\star - y_1\|}{\sqrt{T}}\right)$$

Therefore, if we set $\eta = min(\frac{1}{H}, \frac{\|w_\star - y_1\|}{\sigma T^{\frac{3}{2}}})$ then we will get the convergence bound we want:

$$\mathbb{E}[\mathcal{L}(\mathbf{w}_{T+1}) - \mathcal{L}(\mathbf{w}_\star)] \leq O\left(\frac{H\|\mathbf{w}_\star - \mathbf{y}_1\|^2}{T^2} + \frac{\sigma \|\mathbf{w}_\star - \mathbf{y}_1\|}{\sqrt{T}}\right)$$

Hence, proved.

My proof proves the convergence rate of T^{-2} of the final output by the accelerated gradient descent with the optimal minimizer of the loss function which is represented by the following notation: $\mathcal{L}(\mathbf{w}_\star)$. I really enjoy proving mathematical proofs and I believe this will help me in my research at graduate school where I will have to understand and produce complex mathematical proofs to show the validity and usability of my interpretable neural network architectures.

Certification

I have got this certification for clearing the Azure Data Scientist exam. I believe that this certification has equipped me with skills such as using cloud for running machine learning pipelines, performing experiments on the available data and logging the models etc. I believe that all these skills will come in handy in my research to make interpretable neural networks.