The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)

Proceedings of the Undergraduate Consortium Mentoring Program

New York, New York, USA

February 7-12, 2020

Quantifying the Effect of Unrepresentative Training Data on Fairness Intervention Performance

Jessica Dai[1], **Sarah M. Brown**[2]
Brown University
[1]jessica.dai@brown.edu
[2]sarah_m_brown@brown.edu

Abstract

While many fairness interventions exist to improve outcomes of machine learning algorithms, their performance is typically evaluated with the assumption that training and testing data are similarly representative of all relevant groups in the model. In this work, we conduct an empirical investigation of fairness intervention performance in situations where data from particular subgroups is systemically under- or over- represented in training data when compared to testing data. We find post intervention fairness scores vary with representation in often-unpredictable and dataset-specific ways.

Introduction

As machine learning algorithms are applied to ever more domains, the data used to train these models is also increasingly messy; fairness interventions aim to prevent algorithms from reproducing societal biases encoded in data. These interventions are generally evaluated under the assumption that the training data is well-representative of the data on which the model will be deployed. However, systemic disparities in group representation in training data is uniquely likely in domains where historical bias is prevalent.

Our main question is: how does the oversampling or undersampling of a particular group affect the performance of the post-intervention algorithm, in terms of overall accuracy and in terms of various measures of fairness? We resample existing datasets to simulate different proportions of demographic groups for training and testing, extending the previous work of Friedler et al. 2019 to evaluate the performance of those interventions on our artificially unrepresentative test-train splits of the data; this serves as our proxy for real-world unrepresentative data.

We find that changing the representation of a protected class in the training data affects the ultimate performance of fairness interventions in somewhat unpredictable ways. For the rest of this paper, we will use *representation effects* to describe the way in which changing representation affects fairness performance. In particular, our results are:

1. **Fairness-accuracy tradeoff.** Representation effects with regards to the fairness-accuracy tradeoff are inconsistent even within a specific dataset; in each of the datasets we analyzed, they differ depending on the algorithm and intervention being analyzed. The only generalization to be made is that representation effects for an *intervention* on a baseline algorithm follow the same pattern as representation effects on the baseline itself.

2. **Calibration-error rate tradeoff.** Representation effects with respect to the calibration-error rate tradeoff are also inconsistent across datasets, but representation effects of different algorithms *are* consistent within a single dataset.

Related work

Existing fairness interventions and metrics. A wide variety of intervention strategies exist. These include modifications or reweighting of the training data (Feldman et al. 2015) and modifications of the objective function (Kamishima et al. 2012; Zafar et al. 2017), among other approaches. At the same time, there are a variety of ways to quantify "fairness," including base rates and group-conditioned classification statistics (i.e. accuracy and true/false positive/negative rates for each group).

Class imbalance and distribution shift. While these are known problems in machine learning broadly, they are largely unconsidered in the context of fairness interventions; they typically assume variation in the distribution of the target variable, while we are interested in the distribution of the protected class. Scholarship in this area often suggests some method for oversampling (e.g. Fernández, García, and Herrera), albeit more nuanced than the experiments run here.

Existing empirical survey. Friedler et al. published an empirical comparison of several fairness interventions across multiple datasets with an open source framework for replication. They found that intervention performance is context-dependent—that is, varies across datasets—and empirically verified that many fairness metrics directly compete with one another. This survey also investigated the relationship between fairness and accuracy (which has often been characterized as a tradeoff), noting that stability in the context of fairness is much lower than accuracy.

Experimental setup

We preserve the experimental pipeline of Friedler et al.: For each dataset, we run standard algorithms (SVM, Naive Bayes, Logistic Regression) and several fairness intervention algorithms introduced by Feldman et al.; Kamishima et al.; Calders and Verwer, and Zafar et al.. For each run of each algorithm, we compute overall accuracy and a variety of fairness metrics. However, in each experiment, we replace the train-test splits—which were random in Friedler et al.'s original work—to simulate unrepresentative data.

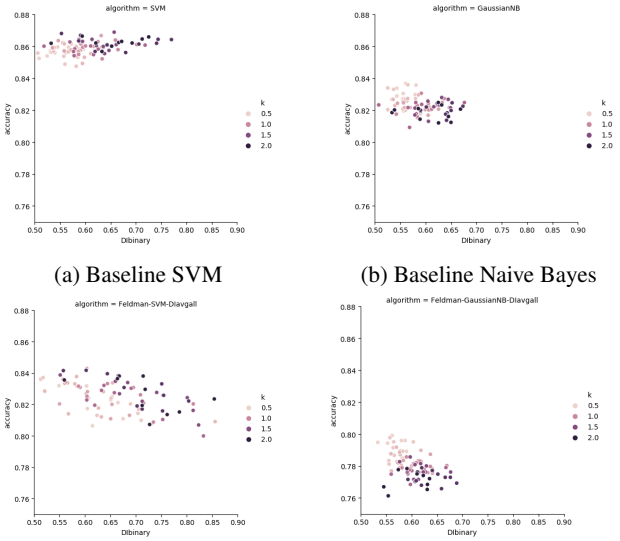
To simulate unrepresentativeness, we create train-test splits for each dataset that represent a variety of distribution shift or oversampling possibilities. We introduce a parameter $k = \frac{q}{r}$, where q is the proportion of the protected class in the training set and r is the proportion of the protected class in the testing set, so that for $k = 0.5$ the disadvantaged group is half as prevalent in the training set as it is in the testing set (underrepresented), and for $k = 2.0$ the protected class is twice as prevalent (overrepresented). We run a series of experiments each for a value of k in $\frac{1}{2}, \frac{4}{7}, \frac{2}{3}, \frac{4}{5}, 1, \frac{5}{4}, \frac{3}{2}, \frac{7}{4}, 2$. We use 80-20 test train splits in all experiments. Most new work on this project is my own; Dr. Sarah Brown advises this project, providing guidance on framing research questions and formulating new approaches.

Results & evaluation

For the following figures, each dot represents the statistic calculated for one run of the algorithm. A darker dot indicates a higher *k*.

Fairness-accuracy tradeoff. Representation effects in the context of the fairness-accuracy tradeoff are inconsistent not only across datasets but for algorithms within the same dataset as well. The Adult dataset (figure 1) is one example of this phenomenon: increasing training representation appears to increase both fairness and accuracy in SVM algorithms, but reduces accuracy with little impact on fairness in Naive Bayes algorithms. Interestingly, when fairness interventions are applied to the same baseline algorithms, the representation effects on the interventions follow the same general pattern as representation effects on the baseline algorithms. Here, fairness is measured through disparate impact (formally, $\frac{P(\hat{Y}=1,S\neq 1)}{P(\hat{Y}=1,S=1)}$ where $\hat{Y}$ is the predicted label and $S = 1$ indicates the privileged demographic).

Calibration-error rate tradeoff. It is impossible to achieve equal true positive and negative calibration rates across groups, given unequal base rates (Kleinberg, Mullainathan, and Raghavan 2016). More formally, we compare the true positive rate (TPR) of the unprivileged demographic ($P(\hat{Y} = 1|Y = 1, S = 0)$ to the negative calibration of the same demographic ($P(Y = 1|\hat{Y} = 1, S = 0)$). Here, representation effects also differ across datasets, though different algorithms within the same dataset respond similarly to changes in representation, as illustrated in figure 2. While the general shape of the tradeoff follows the expected downward slope in each dataset and each algorithm, note that in the Adult dataset, representation appears to have little effect on TPR, while it tends to increase TPR in the Ricci dataset.



(a) Baseline SVM (b) Baseline Naive Bayes

(c) Feldman SVM (d) Feldman Naive Bayes

Figure 1: Fairness-accuracy tradeoff of a subset of algorithms run on the Adult dataset. Disparate impact is on the horizontal axis, while accuracy is on the vertical axis.

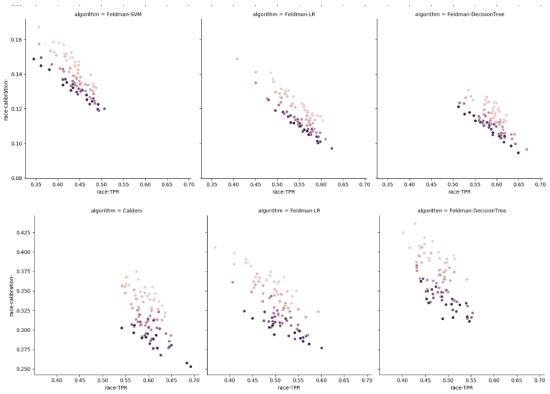


Figure 2: Calibration-error rate tradeoff in Adult (top) and ProPublica (bottom) datasets. TPR is on the horizontal axis.

Discussion

These empirical results further illustrate the importance of context and domain awareness when considering the "fairness" of an algorithm. In particular, the somewhat unpredictable representation effects across datasets and algorithms suggest a need for a rethinking of approaches to fairness interventions; while (over)representation may sometimes be helpful, it is clear that datasets contain some intrinsic properties that affect observed fairness.

In future work, we hope to develop a model which provides a theoretical explanation for our results; this also aids us in commenting on the interpretation of "fairness results," as well as arriving at a framework for understanding *a priori* when overrepresentation in training data may be helpful.

References

Calders, T., and Verwer, S. 2010. Three naive bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery* 21(2):277–292.

Feldman, M.; Friedler, S. A.; Moeller, J.; Scheidegger, C.; and Venkatasubramanian, S. 2015. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 259–268. ACM.

Fernández, A.; García, S.; and Herrera, F. 2011. Addressing the classification with imbalanced data: open problems and new challenges on class distribution. In *International Conference on Hybrid Artificial Intelligence Systems*, 1–10. Springer.

Friedler, S. A.; Scheidegger, C.; Venkatasubramanian, S.; Choudhary, S.; Hamilton, E. P.; and Roth, D. 2019. A comparative study of fairness-enhancing interventions in machine learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 329–338. ACM.

Kamishima, T.; Akaho, S.; Asoh, H.; and Sakuma, J. 2012. Fairness-aware classifier with prejudice remover regularizer. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 35–50. Springer.

Kleinberg, J.; Mullainathan, S.; and Raghavan, M. 2016. Inherent trade-offs in the fair determination of risk scores. *arXiv preprint arXiv:1609.05807*.

Zafar, M. B.; Valera, I.; Rogriguez, M. G.; and Gummadi, K. P. 2017. Fairness Constraints: Mechanisms for Fair Classification. In Singh, A., and Zhu, J., eds., *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, 962–970. PMLR.

Activity Recognition Using Deep Convolutional Networks for Classification in the SHL Recognition Challenge

Michael Sloma
University of Toledo
msloma@rockets.utoledo.edu

Abstract

This paper provides an overview of my contribution to the participation in the Sussex-Huawei Locomotion-Transportation (SHL) challenge. The SHL recognition challenge considers the problem of human activity recognition using sensor data collected from an Android smartphone. My main contributions included cleaning and preprocessing the data, and in the development of a neural network based model for detecting the mode of locomotion. The applications of this project include smartphone-based fitness trackers, productivity trackers and improved general activity recognition.

Project Goals

The primary goal of the Sussex-Huawei Locomotion-Transportation (SHL) challenge is to recognize different modes of locomotion and transportation using the sensor data of the smartphone. The SHL dataset (Gjoreski, Ciliberto, Wang, Morales, Mekki, Valentin and Roggen 2018) included 366 hours of recorded smartphone data, 271 hours for training (including activity labels) and 95 hours for testing (no labels available). This data was collected from a single person using Huawei Mate 9 smartphone, worn in the front right pants pocket. Potential applications for a model like this are in the activity recognition field such as fitness trackers and in the productivity field, allowing users to granularly track their actions day to day.

Previous Work

Prior work such as that by Bao et. al. (Bao and Intille 2004) and Ravi et. al. (Ravi, Dandekar, Mysore, and Littman. 2005) suggest that multiple accelerometers can effectively discriminate many activities. In addition, work from Lara et. al. (Lara and Labrador 2013) provides multiple methods and proposes a framework for activity recognition problems using wearable sensors. These papers also provide some preprocessing methods traditionally used in activity recognition that we wanted to challenge and see if they were still necessary with the prevalence of deep learning methods. Hammerla et. al. (Hammerla, Halloran, and Plötz 2016) describe several potential deep learning approaches for activity recognition, including tradition deep feed forward networks, convolutional networks and recurrent networks. This paper, in addition to the prevalence of deep learning in all machine learning applications inspired us to attempt to utilize these methods for the challenge.

Personal Contributions

My main contributions to the project were two-fold, in cleaning and preprocessing the data, and in the development of a neural-network based model for detecting the mode of locomotion. Since the data came in a raw format, there were missing data points that needed to be filled and values that were nonsensical that needed to be corrected for. After the data was cleaned, the values for each feature needed to be scaled to within a reasonable range which allows for a) faster learning and b) allows the model to extract information better due to not needing to overcome the scaling issues. I scaled all the values for each feature independently to the range of [-1, 1], so the max value for each feature was 1 and the minimum value was -1.

The data was then sliced windows to allow for the creation of a supervised classification problem, where each window's labels were the most frequent value of all the labels in that time span. The length of the window was varied to determine what the optimal window length was for the algorithms we used.

Once the data was prepared, we then decided on using two approaches for classification; a deep learning approach done by myself and a random forest approach done by a master's student working in our lab, both using the data

that I prepared. For the deep learning approach, I selected to use 1-D (temporal) convolutional neural network (CNN) layers to capture the time related aspect of our data. These layers were interleaved with max-pooling layers to provide translational invariance to the model. After the block of convolutional and max-pooling layers, traditional fully connected layers were used to learn non-linear combinations of the features extracted via the convolutional layers. The final layer was fed into a softmax function across the 8 possible activity classes, which allows for the output to be interpreted directly as a probability of being in each class.

Since each model was costly to train it was important to be as efficient as possible in our searches, so I used a mixture of manual and random searches to find optimal hyperparameters for the CNN model. Once I had a suitable CNN model, we compared the CNN model that I had created against the random forest model that the master's student had created. Once we learned what each model exceled on, we both went back and refined our models further to incorporate the new information we had found out.

Results & Learning Opportunities

The initial results of our paper on our validation and testing sets were quite promising, resulting in a mean F1 score over all activities of 0.973 (max score possible of 1.0). When the results for the held-out test set data were provided from the challenge creators, our team scored a 0.532. Naturally, this was a surprise to us as we thought we were doing quite well however, we had made several mistakes in the splitting of our data as we neglected that time series data needs to be treated differently. Initially, we treated the data as if each window was independent of each other and used a random stratified split to maintain the distribution of classes in our train set and test set, without regard to the temporal nature of the data. This, however, resulted in the possibility of sequential time windows being in the train set and test set, making it substantially easier for the model to predict test set results as the model had practically seen almost all the data before. As a team we learned a great amount from this challenge and went on to write a book chapter about our process so others can learn from our mistakes and prevent this from happening to their studies.

Application of Contributions

Ideally the contributions provided by this work would have been providing more reliable methods for detecting activities while the user has a smartphone on them, however the main contribution of this works turned out to be helping others learn from our mistakes and prevent test-set leakage in time series applications. From going through this process and revising our methods, we provided a resource to other researchers on how to correctly prepare their data to prevent extraneous results in other similar research.

Prospective Next Steps

There are several potential directions that this project could be taken going forward. Due to the limited computational power of our lab, we were unable to effectively explore introducing recurrent networks as a potential solution. Since recurrent networks have been shown to be effective in modeling temporal data (Hammerla, Halloran, and Plötz 2016) this would be an excellent area for further exploration. Given the hardware needed to complete this, the timeline for this would be on the order of about a month.

Another potential direction would be improving the CNN architecture that we currently have using improved methods for hyperparameter searches such as Asynchronous HyperBand (Li, Jamieson, Rostamizadeh, Gonina, Hardt, Recht and Talwalkar 2018), which exploits both parallelism and early-stopping techniques to provide searches an order of magnitude faster than random search. These methods are readily available in packages such as Ray Tune (Liaw, Liang, Nishihara, Moritz, Gonzalez and Stoica 2018) allowing for a quick implementation of these methods, thus the timeline for implementing a better search algorithm could be on the order of days, with searching taking several weeks to find new optimal models.

References

Ling Bao and Stephen S. Intille. 2004. Activity recognition from user-annotated acceleration data. In *Proceedings of the 2nd International Conference on Pervasive Computing*. 1–17.

H. Gjoreski, M. Ciliberto, L. Wang, F. J. O. Morales, S. Mekki, S. Valentin, and D. Roggen. 2018. The University of Sussex-Huawei Locomotion and Transportation dataset for multimodal analytics with mobile devices. *IEEE Access* In Press (2018).

Nils Y. Hammerla, Shane Halloran, and Thomas Plötz. 2016. Deep, convolutional, and recurrent models for human activity recognition using wearables. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. 1533–1540.

Óscar D. Lara and Miguel A. Labrador. 2013. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys & Tutorials* 15 (2013), 1192–1209.

Lisha Li, Kevin Jamieson, Afshin Rostamizadeh, Katya Gonina, Moritz Hardt, Benjamin Recht and Ameet Talwalkar. 2018. Massively Parallel Hyperparameter Tuning. *https://openreview.net/forum?id=S1Y7OOlRZ.* (2019)

Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez and Ion Stoica. 2018. Tune: A Research Platform for Distributed Model Selection and Training. *https://ray.readthedocs.io/en/latest/tune.html.* (2019)

Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L. Littman. 2005. Activity recognition from accelerometer data. In *Proceedings of the 17th Conference on Innovative Applications of Artificial Intelligence.* 1541–1546.

Michael Sloma, Makan Arastuie, Kevin S. Xu. 2018. Activity Recognition by Classification with Time Stabilization for the SHL Recognition Challenge. In *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*. 1616–1625.

Michael Sloma, Makan Arastuie, Kevin S. Xu. 2019. Effects of Activity Recognition Window Size and Time Stabilization in the SHL Recognition Challenge. *Human Activity Sensing* (2019). 213-231.

Harmful Feedback Loops in Unequitable Criminal Justice AI Models

Pazia Luz Bermudez-Silverman
Brown University, Department of Computer Science, Data Science Initiative
78 Arnold St. Fl 2
Providence, RI 02906
pazia_bermudez-silverman@brown.edu

Abstract

My research focuses on the harm that AI models currently cause and the larger-scale potential harm that they could cause if nothing is done to stop them now. In particular, I am focusing on AI systems used in criminal justice, including predictive policing and recidivism algorithms. My work synthesizes previous analyses of this topic and steps to make change in this area, including auditing these systems, spreading awareness and putting pressure on those using them.

Introduction

As many researchers have recently shown, AI systems used by law enforcement and the public to make decisions that directly impact people's lives perpetuate human biases surrounding race, gender, language, skin color, and a variety of intersections of these identities (Albright 2019; Buolamwini and Gebru 2018; Lum and Isaac 2016). While these biases already existed in our society long before AI and modern technology, AI algorithms and models reinforce them at an unprecedented scale. In addition, these models' feedback loops strengthen such biases by perpetuating harm to communities already at risk (O'Neil 2016). We see these algorithms and their harmful feedback loops in areas such as education, criminal justice and housing, but this paper will focus on criminal justice algorithmic models and their effects on lower-income communities of color.

Background

Feedback loops and proxy attributes are essential for understanding the scale of harm and influence AI models have on this society, especially in the criminal justice system.

Feedback Loops

Feedback is essential to the accuracy of AI algorithms and models. Without it, a model will never know how well or how poorly it is performing and thus, it will never get better. However, depending on which feedback is given to a model, that model will change and behave in particular ways according to that feedback. This is a feedback loop.

Cathy O'Neil characterizes one of the main components in her definition of a "Weapon of Math Destruction" (WMD) as having a "pernicious feedback loop" (O'Neil 2016) that contributes to the power and harm of the WMD. Such feedback loops occur when an algorithm either does not receive feedback on its output or receives feedback that is not accurate in some way. O'Neil cites that organizations using AI algorithms allow for the continuation and growth of these pernicious feedback loops because they look at the short term satisfaction of their consumers rather than the long term accuracy of their models (O'Neil 2016). As long as companies are making money or organizations are meeting their goals, it is much easier for them ignore the harm that these models are causing. Additionally, Virginia Eubanks cites this "feedback loop of injustice" (Eubanks 2018) as harming specifically our country's poor and working-class people, mostly people of color, through examples of automated algorithms used in welfare, child and family and homeless services.

Proxy Attributes

While most predictive policing and other AI models used in law enforcement such as recidivism algorithms used to determine sentence length and parole opportunities do not directly take into account sensitive attributes such as race, other attributes such as zip code, friends and family criminal history, and income act as proxies for such sensitive attributes (Adler et al. 2018). In this way, predictive policing and recidivism prediction algorithms do directly make decisions having to do with race and other sensitive attributes.

These proxy attributes can actually directly lead to pernicious feedback loops not only because they are easier to "game" or otherwise manipulate, but also because the use of such proxies might make the model calculate something other than what the designers/users think. This can lead to false data (false positives or negatives) that is then fed back into the model, making each new iteration of the model based on false data, further corrupting the feedback loop (O'Neil 2016). Eubanks exemplifies this in her description of how using proxies for child maltreatment cause higher racial biases in automated welfare services (Eubanks 2018).

Predictive Policing

The most common model used for predictive policing in the U.S. is PredPol. We will focus on how this software perpetuates racial and class-based stereotypes and harms lower-income communities of color particularly through its pernicious feedback loops.

One reason for skewed results that are biased toward lower-income neighborhoods populated mostly by people of color is the choice of including two different types of crimes. Either only "Part 1 crimes" which are more violent like homicide and assault or also "Part 2 crimes" which are less violent crimes/misdemeanors such as consumption and sale of small quantities of drugs (O'Neil 2016). "Part 2 crimes" are often associated with these types of neighborhoods in our society. By following these results, law enforcement will send more officers into those areas, who will then "catch more crime," feed that data back into the model, perpetuating this pernicious feedback loop and continuing to send officers to these communities instead of other, more affluent and white areas.

Feedback Loops in PredPol Software

Crime is observed in two ways: law enforcement directly sees "discovered" crime and the public alerts them to "reported" crime. "Discovered" crime is a part of the harmful feedback loop: the algorithm sends officers somewhere and then when they observe crime the predictions are confirmed. Predpol is trained on observed crime which is only a proxy for true crime rates. PredPol lacks feedback about areas with "lower" crime-rates according to the model by not sending officers there, contributing further to the model's belief that one region's crime rate is much higher than the other. Given two areas with very similar crime rates, the PredPol algorithm will *always* choose the area with the slightly higher crime rate because of the feedback loop (Ensign et al. 2017).

Auditing Practices and Further Interventions

Auditing has been used by researchers such as Raji and Buolamwini, Adler et al. and Sandvig et al. to evaluate the accuracy and abilities of AI models as well as potential harm they could cause by inaccuracy such as through pernicious feedback loops. Corporations are not likely to change the way they use these models if change does not contribute to one of the following areas: "economic benefits, employee satisfaction, competitive advantage, social pressure and recent legal developments" (Raji and Buolamwini 2019). This means that external pressure, namely public awareness and organizing is necessary for change.

Ensign et al. propose an "Improvement Policy" for the PredPol software which suggests a filtering of feedback given to the model. They recommend that the more police are sent to a given district, the less weight discovered incidents in that area should count in feedback data (Ensign et al. 2017). They conclude that most "discovered" crime should not be counted in feedback data. This may still miss some crimes, but it is a better proxy, especially for use in algorithms, because it is not directly influenced by the model's previous choices. This should create a more equitable outcome that does not continue to target impoverished neighborhoods and communities of color.

A Socio-Technical Analysis of Feedback loops

One key question I am exploring involves how these analyses fit into the traps defined by Selbst et al., particularly the Ripple Effect Trap, which essentially speaks to the concept of feedback loops.

To solve this problem, we propose an investigation into whether such feedback loops contribute to the amplification of existing human biases or to the creation of new biases unique to such technology. Additionally, we hope to analyze the best ways to spread public awareness and influence companies and organizations to make changes in the way they use AI technologies. First analyses of these methods are discussed below and will be continued throughout this research.

Public Awareness

So, how do we combat these pernicious feedback loops and actually change the structure of the models and the way that organizations use them? The first step to making positive change in AI is spreading public awareness of the harm that AI systems currently cause and the misuse of them by law enforcement in these cases particularly. This can and should be through not only academic papers and articles, but through political activism on and off the web. As Safiya Noble has shown, the more people that understand what is currently happening and what we can possibly do to change it, the more pressure that is put on the companies, organizations and institutions that use these harmful models, which will encourage them to change the way they use the models and the way the models work in general (Noble 2018).

Next Steps and Timeline

I am currently working on a survey paper evaluating feedback loops and bias amplification through a socio-technical lens. Specifically, I will focus on the unique roles of AI researchers, practitioners, and activists in combating the harm caused by feedback loops. To investigate the question of how feedback loops amplify existing societal biases and/or create new unique biases, I am analyzing texts more relevant to my background in Africana Studies. This helps to provide societal context and background to this AI research.

Algorithms currently important in my research include PredPol (predictive policing software (Ensign et al. 2017)), COMPAS (the recidivism algorithm (Larson et al. 2016; Broussard 2018)), VI-SPDAT (used in automated services for the unhoused (Eubanks 2018)), as well as facial recognition software used by law enforcement (Garvie et al. 2016).

Following this academic version, I will translate and convey these findings into actionable insights accessible to all. Making my research available to many people will contribute to the public awareness I believe is necessary to combat the negative impacts of AI.

References

Adler, P.; Falk, C.; Friedler, S. A.; Nix, T.; Rybeck, G.; Scheidegger, C.; Smith, B.; and Venkatasubramanian, S. 2018. Auditing black-box models for indirect influence. *Knowledge and Information Systems* 54(1):95–122.

Albright, A. 2019. If You Give a Judge a Risk Score: Evidence from Kentucky Bail Decisions.

Broussard, M. 2018. *Artificial unintelligence: how computers misunderstand the world*. MIT Press.

Buolamwini, J., and Gebru, T. 2018. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, 77–91.

Ensign, D.; Friedler, S. A.; Neville, S.; Scheidegger, C.; and Venkatasubramanian, S. 2017. Runaway feedback loops in predictive policing. *arXiv preprint arXiv:1706.09847*.

Eubanks, V. 2018. *Automating inequality: How high-tech tools profile, police, and punish the poor*. St. Martin's Press.

Garvie, C.; Privacy, G. U. C. o.; Technology; and Technology, G. U. L. C. C. o. P. . 2016. *The Perpetual Line-up: Unregulated Police Face Recognition in America*. Georgetown Law, Center on Privacy & Technology.

Larson, J.; Mattu, S.; Kirchner, L.; and Angwin, J. 2016. How we analyzed the COMPAS recidivism algorithm. *ProPublica (5 2016)* 9.

Lum, K., and Isaac, W. 2016. To predict and serve? *Significance* 13(5):14–19.

Noble, S. U. 2018. *Algorithms of oppression: How search engines reinforce racism*. nyu Press.

O'Neil, C. 2016. *Weapons of math destruction: How big data increases inequality and threatens democracy*. Broadway Books.

Raji, I. D., and Buolamwini, J. 2019. Actionable auditing: Investigating the impact of publicly naming biased performance results of commercial ai products. In *AAAI/ACM Conf. on AI Ethics and Society*, volume 1.

Sandvig, C.; Hamilton, K.; Karahalios, K.; and Langbort, C. 2014. Auditing algorithms: Research methods for detecting discrimination on internet platforms. *Data and discrimination: converting critical concerns into productive inquiry* 22.

Selbst, A. D.; Boyd, D.; Friedler, S. A.; Venkatasubramanian, S.; and Vertesi, J. 2019. Fairness and abstraction in sociotechnical systems. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 59–68. ACM.

Search Tree Pruning for Progressive Neural Architecture Research Summary

Deanna Flynn
deannaflynn@gci.net
University of Alaska Anchorage
401 Winfield Circle
Anchorage, Alaska 99515

Abstract

A basic summary of the research in search tree pruning for progressive neural architecture search. An indiciation of the work which I contributed, as well as the advantages and possible ways the research can continue.

Summary of Research

We develop a neural architecture search algorithm that explores a search tree of neural networks. This work contrasts with cell-based networks ((Liu et al. 2017), (Liu, Simonyan, and Yang 2018)) and uses Levin search, progressively searching a tree of candidate network architectures (Schmidhuber 1997). The algorithm constructs the search tree using Depth First Search (DFS). Each node in the tree builds upon its parent's architecture with the addition of a single layer and a hyperparameter optimization search. Hyperparameters are trained greedily, inheriting values from parent nodes, as in the compositional kernel search of the Automated Statistician (Duvenaud et al. 2013).

We use two techniques to constrain the architecture search space. First, we constructed a transition graph to specify which layers can be inserted into the network, given preceding layers. The input and output layers are defined by the problem specification. Second, we prune children from the tree based on their performance relative to their parents' performance or we reach a maximum depth. The tree search is halted when no children out-perform their parents or we have reached the maximum depth.

The algorithm was tested on the CIFAR-10 and Fashion-MNIST image datasets ((Xiao, Rasul, and Vollgraf 2017), (Krizhevsky, Hinton, and others 2009)). After running our algorithm on a single Intel i7 8th generation CPU on the Fashion-MNIST dataset for four days, we generated 61 networks with one model achieving a benchmark performance of 91.9% accuracy. It is estimated our algorithm only traversed between a fifth and a third of the search tree. This result was acquired in less time than it took other benchmark models to train and test on the same dataset. Table 1 shows a comparison of other benchmark models on the Fashion-MNIST dataset.

When testing the CIFAR-10 dataset, our processing time was limited to 24 hours. We placed a depth limit on the search tree and trained on ten percent of the original dataset. In thirteen hours, on an Intel Xeon Gold 6154 CPU and NVIDIA Tesla V100-SXM2-32GB, the algorithm generated a network with an accuracy of 55.9%.

This algorithm is limited to finding feed-forward networks. Although contingent on the transition graph, the algorithm is simple to implement. However, by dealing with layers directly, it incorporates the macro-architecture search required in cell-based neural architecture search. The progressive nature of Levin search makes the algorithm relevant to resource-constrained individuals who need to find the simplest network that accomplishes their task.

What I Contributed

The research presented was intended for my summer internship at NASA working on a project called Deep Earth Learning, Training, and Analysis (DELTA). DELTA analyzes satellite images of rivers to predict flooding and uses the results to create flood maps for the United States Geological Survey (USGS). Also, work was beginning to examine images for identifying buildings which were damaged in natural disasters. For both aspects, a manually created neural network was used for the identification and learning process of each image.

The suggestions of the outcome for the research were predefined by my mentors who wanted to investigate having a neural network be automatically generated for a specific task. First, some form of search tree was to be created with every node representing a neural network. Next, each child node in the search tree would contain an additional network layer somewhere within the previous neural architecture. Finally, the tree was to have the capability of performing basic pruning.

The first problem which needed to be addressed was the creation of the neural networks. This included both what layers were used within the architecture and how to methodically insert the layers within pre-existing architectures. For simplicity purposes, only five layers were considered in the initial design: *Convolution, Flatten, Max Pooling, Dropout,* and *Dense*. As for determing what layers can be inserted, I constructed a transition graph through studying and testing multiple neural networks and carefully watching what

Model	Accuracy (%)	# of Parameters	Time to Train	Computer Specification
GoogleNet	93.7	4M		CPU
VGG16	93.5	∼26M	Weeks	Nvidia Titan Black GPUs
AlexNet	89.9	∼60M	6 days	2 Nvidia GeForce 580 GPUs
Our Model	91.9	98K	4 days	Intel i7 8th Generation CPU

Table 1: Size and execution time of some Fashion-MNIST networks compared to the generated network.

sequence of layers worked and what did not.

The transition graph is represented within the algorithm as a dictionary with each layer represented as strings. The actual insertion occurs between every layer pair in the parent architecture. The algorithm then uses the dictionary to determine the the new layer to add within the neural network. Each new insertion results in a new child node within the search tree.

The second problem needing to be solved was hyperparameter optimization. Hyperparamers are the basic attributes of each neural network layer. Certain libraries were originally used to handle testing hyperparameter possibilities, but they did not deliver the results we wanted. So, I created my own version of hyperparameter optimization. Just like the transition graph, hyperparameters are stored in a dictionary based on the layer type. Each hyperparameter combination for the specific layer is then tested by incrementally training against a subset of images from either CIFAR-10 or Fashion-MNIST. The best hyperparameter combination resulting in the highest accuracy is added to the neural architecture and, potentially, to the search tree.

The final problem I needed to address was the aspect of pruning. How pruning is approached is already mentioned within the summary of the research. As a child network is generated, the accuracy is compared to the accuracy of its parent. If the child performs worse, the child is removed entirely and the branch is not considered. If all new child nodes are created but no new network is added to the search tree, then the expansion of the search tree is done. However, keeping track of the children and the parent to perform the comparison of accuracy without having to store the entire tree needed to be determined. As a result, the algorithm became recursive DFS and passed the parent's accuracy down to the child.

Advantages of the Research

There are several advantages to this research and algorithm. To begin with, the only human interaction required is in the initial setup of the algorithm. This includes modifications of the transition graph and or addition of new hyperparameters. Next, the algorithm's simplicity allows for individuals not as knowledgable in the area of neural networks to use and create simple neural networks that can solve their problem. Only the initial setup requries more knowledge about neural networks. Once the algorithm is running, a network will be generated. Finally, the algorithm takes considerably less time to run and trains more neural networks than other architectures and has been shown to generate similar results. Looking back to Table 1, our algorithm generated a result in 4 days. The next algorithm was 6 days. However, our algorithm created and trained *61* models compared to only one.

Future Work

There are several areas which can be pursued in the future of our research. First, continuing tests on both CIFAR-10 and Fashion-MNIST but with the full datasets and larger trees. Second, using datasets that are text-based and not images. This can illustrate the versatility of our algorithm by solving a *specific task*. Third, modifying the current structure of our algorithm to use the Breadth-First search, Best-First search, and varying pruning algorithms. The results can be used to determine varying runtimes and how much accuracy and loss are affected by modifying the algorithm.

Besides changing the algorithm's basic structure, using Bayesian optimization to test the possibilities of hyperparameters is being considered to speed up our algorithm. Optimizing hyperparameters is the most time-consuming aspect of our algorithm. Anything which can speed up our process further will be very beneficial. Finally, adding autoencoders to distinguish between more minute details within images or textual patterns.

References

Duvenaud, D.; Lloyd, J. R.; Grosse, R.; Tenenbaum, J. B.; and Ghahramani, Z. 2013. Structure discovery in nonparametric regression through compositional kernel search. *arXiv preprint arXiv:1302.4922*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

Liu, C.; Zoph, B.; Shlens, J.; Hua, W.; Li, L.; Fei-Fei, L.; Yuille, A. L.; Huang, J.; and Murphy, K. 2017. Progressive neural architecture search. *CoRR* abs/1712.00559.

Liu, H.; Simonyan, K.; and Yang, Y. 2018. DARTS: differentiable architecture search. *CoRR* abs/1806.09055.

Schmidhuber, J. 1997. Discovering neural nets with low kolmogorov complexity and high generalization capability. *Neural Networks* 10(5):857–873.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Reconstructing Training Sets by Observing Sequential Updates

William Jang
Amherst College
220 South Pleasant Street
Amherst, Massachusetts 01002
wjang20@amherst.edu

Abstract

Machine learning methods are being used in an increasing number of settings where learners operate on confidential data. This emphasizes the need to investigate machine learning methods for vulnerabilities that may reveal information about the training set to a persistent attacker. We consider the task of reverse engineering aspects of a training set by watching how a learner responds to additional training data. Specifically, an adversary Alice observes a model learned by Bob on some original training set. Bob then collects more data, and retrains on the union of the original training set and the new data. Alice observes the new data and Bob's sequence of learned models with the aim of capturing information about the original training set. Previous work has addressed issues of data privacy, specifically in terms of theoretical limits on the amount of information leaked by publishing a model. Our contribution concerns the novel setting of when Alice observes a sequence of learned models (and the additional training data that induces this sequence), allowing her to perform a differencing attack. The successful completion of this line of work will yield a better understanding of the privacy guarantees of learners in real world settings where attacker and learner act in time.

Introduction

Using machine learning methods in practice introduces security vulnerabilities. An attacker may manipulate data so as to trick a learned model or a learner in process of training. Such is the study of adversarial learning (Lowd and Meek 2006; Vorobeychik and Kantarcioglu 2018; Joseph et al. 2019; Biggio and Roli 2018). In addition, in deploying a learned model, one may inadvertently reveal information about the training data used. The aim of privacy-preserving learning (Dwork et al. 2010) is to create learning methods with guaranteed limits on the amount of information revealed about the underlying data. Often in practice a learned model is deployed and then later (after additional training data has been gathered), a new model trained on the union of the old and new data is deployed. In this work we seek to quantify how much information about a training set can be gained by an attacker which observes not only the deployed model, but how that model evolves over time as new training data is introduced.

We consider the setting where a learner Bob uses a training set $D = (X_0, Y_0)$, where $X_0 \in \mathbb{R}^{n \times d}$, $Y_0 \in \mathbb{R}^n$, to learn a model θ and an attacker Alice attempts to reverse engineer aspects of D. There is a rich collection of prior work in data privacy, in particular differential privacy (Dwork et al. 2006) which addresses this problem. In contrast to prior work, we model Alice as observing not only θ, but also a sequence of new points and subsequently learned models. Formally, Bob learns θ_0 from D with learning algorithm L: $\theta_0 = L(D)$. He then gathers new data D' and learns a new model θ_1 from $D \cup D'$: $\theta_1 = L(D \cup D')$. Alice observes θ_0, D', and θ_1. This continues with Alice observing additional data sets, Bob training a model on the growing set of points, and Alice observing his model. She attempts to reverse engineer some aspect of the original D (e.g., the mean of a particular feature, whether or not some specific instance is present in the training set, etc.). Our preliminary results show that this sequential observation process results in Alice having substantially more capability to reverse engineer the training set than if she had only observed the first model.

Methods

As an illustrative example, suppose Bob trains a linear regression model using ordinary least squares and Alice aims to reverse engineer aspects of the original training set. That is, Bob learns a model θ_0 which satisfies the normal equations $A_0\theta_0 = B_0$ where $A_0 = X_0^\top X_0$ and $B_0 = X_0^\top Y_0$. For simplicity, we further assume that Alice simply observes, and has no control over the additional points added sequentially to the training process. Bob performs a series of updates. For the ith update, he receives a new data point (x_i, y_i) and learns a model on $D \cup_{j \le i} (x_j, y_j)$. Let x_1, y_1 be the first point in the new data and θ_1 be the resulting model. Note that after Alice observes x_1, y_1, θ_1, she knows that $A_1\theta_1 = B_1$ which we write as

$$(A_0 + x_1 x_1^\top)\theta_1 = B_0 + y_1 x_1 \tag{1}$$

After k updates, Alice knows:

$$\left(A_0 + \sum_{i=1}^{k} x_i x_i^\top\right)\theta_k = B_0 + \sum_{i=1}^{k} y_i x_i \tag{2}$$

$$\mathcal{X} = \left\{ \begin{bmatrix} 3 & 5 \\ -2 & 0 \end{bmatrix}, \begin{bmatrix} -7/6 & 0 \\ 0 & 7/11 \end{bmatrix}, \begin{bmatrix} -56/\sqrt{613} & -21/\sqrt{613} \\ 5 & 610/121 \end{bmatrix}, \begin{bmatrix} 2 & 3 \\ 5 & 4 \end{bmatrix}, \begin{bmatrix} \sqrt{29} & 26/\sqrt{29} \\ 0 & 7/\sqrt{29} \end{bmatrix} \right\}$$
$$\mathcal{Y} = \left\{ \begin{bmatrix} 8 \\ 8 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 15/\sqrt{613} \\ 40/11 \end{bmatrix}, \begin{bmatrix} 3 \\ 2 \end{bmatrix}, \begin{bmatrix} 16/\sqrt{29} \\ 11/\sqrt{29} \end{bmatrix} \right\}, \theta = \left\{ \begin{bmatrix} -6/7 \\ 11/7 \end{bmatrix}, \begin{bmatrix} 152/331 \\ 88/331 \end{bmatrix}, \begin{bmatrix} 1148/1639 \\ -1498/8195 \end{bmatrix} \right\}$$
$$x = \left\{ \begin{bmatrix} 8 & 3 \end{bmatrix}, \begin{bmatrix} 0 & 9 \end{bmatrix} \right\}, y = \left\{ \begin{bmatrix} 5 \end{bmatrix}, \begin{bmatrix} -2 \end{bmatrix} \right\}$$

Figure 1: Each pair $\mathcal{X}_i, \mathcal{Y}_i$ represents a potential initial training set. When Alice observes the initial model θ_0, she has 6 unknowns and 3 equations. By solving this underdetermined system of equations, Alice knows that $\mathcal{X}_0, \mathcal{Y}_0$ could not be the original training set. Similarly, when Alice observes the first update, x_1, y_1, θ_1, she now has 5 equations and can rule out $\mathcal{X}_1, \mathcal{Y}_1$ as a possible initial training set. After she observes x_2, y_2, θ_2, she has 7 equations and 6 unknowns, meaning she can solve the fully determined system of equations to find values for A_0, B_0 that satisfy equation (2). At this point, $\mathcal{X}_3, \mathcal{Y}_3$ and $\mathcal{X}_4, \mathcal{Y}_4$ could be the initial training set. Note however, Alice cannot distinguish between the two datasets as both yield the same A and B, meaning that for all future updates they will satisfy equation (2).

Let $u_0 = 0$ and $u_k = \sum_{i=1}^{k} [y_i x_i] - \sum_{i=1}^{k} [x_i x_i^\top] \theta_k$. Then

$$A_0 \theta_k - B_0 = u_k \tag{3}$$

for $k = 0, ..., K$. Notice that this system of equations is linear in the unknowns, A_0 and B_0.

There are $d^2 + d$ unknowns. Alice starts with d equations when there are zero updates, and $(d^2 - d)/2$ additional equations of the form $A_{ij} = A_{ji}$ as A is symmetric. Each update yields d more equations. Thus Alice needs to observe K updates to have a fully determined system, where K is the smallest integer such that $Kd + (d^2 - d)/2 + d \geq d^2 + d$ (i.e., $K = \lceil (d+1)/2 \rceil$). In order to solve for A_0 and B_0, we let

$$M = \left[\begin{array}{c|c} \theta_0^\top \otimes I & -I \\ \theta_1^\top \otimes I & -I \\ \vdots & \vdots \\ \theta_K^\top \otimes I & -I \end{array} \right] \tag{4}$$

where $\otimes$ denotes the Kronecker product and I is the $d \times d$ identity matrix. We then have:

$$M \begin{bmatrix} \text{Vec}(A_0) \\ B_0 \end{bmatrix} = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_K \end{bmatrix}. \tag{5}$$

Solving this linear system yields A_0, B_0.

Impossibility Result

Two datasets X_0, Y_0 and $\tilde{X_0}, \tilde{Y_0}$ may be identical except for the ordering of the rows. This ordering gets lost during training, meaning that Alice can't distinguish between the $n!$ permutations of X_0, Y_0. This means Alice is never able to fully reconstruct Bob's training set (X_0, Y_0) by solely observing updates. In addition, transforming a dataset X_0 into the Gram matrix A_0 represents a further fundamental loss in information when Bob learns his model θ_0. There could be an alternative training set $\tilde{X_0}, \tilde{Y_0}$ that differs from X_0, Y_0 by more than permutation, such that $\tilde{X_0} \neq X_0$ and $\tilde{Y_0} \neq Y_0$ but $\tilde{X_0}^\top \tilde{X_0} = X_0^\top X_0$ and $\tilde{X_0}^\top \tilde{Y_0} = X_0^\top Y_0$. In such a case, these training sets cannot be distinguished.

Next Steps

Next steps include quantifying the information communicated with each additional training step. Namely, when Alice observes θ_1 there is an equivalence class of training sets that would have yielded that model. As Alice observes additional training points and the corresponding (updated) models, this equivalence class shrinks. In this way, the additional points and models are communicating information about the training set. A natural question we intend to explore is: how much information is communicated by each additional (set of) point(s) and model? Figure 1 demonstrates how each additional point and updated model provides information about the initial training set.

Furthermore, we intend to explore more sophisticated learners and attacker goals by using an Artificial Neural Network (ANN). For example, if a learned (ANN) used for image classification in a unmanned aerial vehicle is captured by enemy forces, they may seek to find out whether or not a particular collection of images was used to train that ANN. Our work specifically considers the scenario where the enemy observes multiple learned models as they are updated over time with additional training. Additionally, for ordinary least squares, we analytically reversed engineered aspects of X_0, Y_0, namely A_0 and B_0, and by knowing these, we also know how the model will update with each additional point. For more sophisticated learners, we will train an ANN to learn a function that will approximate how a model will update given a specific point.

Conclusion

We investigate the task of reverse engineering aspects of a training set by observing a series of models, each updated by the addition of training points. We approach this task along two trajectories: analytic computation for simple learners and automated learning from data. Along the first trajectory we find while one cannot fully reverse engineer the training set, one can reverse engineer aspects of the training set by solving a system of linear equations. After $\lceil (d+1)/2 \rceil$ single point updates, there is no new information about the original training set to infer. Along the second trajectory, we deploy ANNs to predict a general update step for a learner. Preliminary results show promise, but the architecture of the neural network has not yet been dialed in.

References

Biggio, B., and Roli, F. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition* 84:317 – 331.

Dwork, C.; McSherry, F.; Nissim, K.; and Smith, A. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, 265–284. Springer.

Dwork, C.; Naor, M.; Pitassi, T.; and Rothblum, G. N. 2010. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*, 715–724. ACM.

Joseph, A.; Nelson, B.; Rubinstein, B.; and Tygar, J. 2019. *Adversarial Machine Learning*. Cambridge University Press.

Lowd, D., and Meek, C. 2006. Adversarial learning. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 641–647. ACM.

Vorobeychik, Y., and Kantarcioglu, M. 2018. *Adversarial Machine Learning*. Morgan Claypool Publishers.

Ranking to Detect Users Involved in Blackmarket-based Collusive Retweeting Activities

Brihi Joshi*
Indraprastha Institute of Information Technology, Delhi
brihi16142@iiitd.ac.in

Abstract

Twitter's popularity has fostered the emergence of various illegal user activities – one such activity is to artificially bolster visibility of tweets by gaining large number of retweets within a short time span. The natural way to gain visibility is time-consuming. Therefore, users who want their tweets to get quick visibility try to explore shortcuts – one such shortcut is to approach **Blackmarket services** online, and gain retweets for their own tweets by retweeting other customers' tweets. Thus the users unintentionally become a part of a collusive ecosystem controlled by these services. Along with my co-authors, I designed `CoReRank`, an unsupervised algorithm to rank users and tweets based on their participation in the collusive ecosystem. Also, if some sparse labels are available, `CoReRank` can be extended to its semi-supervised version, `CoReRank+`. This work was accepted as a full paper (Chetan et al. 2019) at the 12th ACM International Conference on Web Search and Data Mining (WSDM), 2019. Being a first author, my contribution to this project was a year long effort - from data collection and curation, to defining the problem statement, designing the algorithm, implementing and evaluating baselines and paper writing.

Motivation and Related Work

Blackmarket services are categorized into two types based on the mode of service – *premium* and *freemium*. Premium blackmarkets provide services upon deposit of money. On the other hand, freemium services provide an additional option of unpaid services where customers themselves become a part of these services, participate in fake activities (following, retweeting other, etc.) and gain (virtual) credits. Hence, they become a part of the collusive ecosystem controlled by these services. Current state-of-the-art algorithms either focus on Bot Detection or Spam Detection. Detection of collusive users is challenging for two reasons:

- Unlike bots, they do not have a fixed activity and purpose. Unlike fake users, they are normal users and thus, not flagged by in-house algorithms deployed by Twitter. What makes them interesting to study is that they demonstrate an amalgamation of *inorganic and organic behavior* - they retweet content associated with the blackmarket services and they also promote content which appeals to their interest.
- Collecting large scale labeled data of collusive users is extremely challenging. This necessitates the design of an unsupervised approach to detect collusive users.

	(Davis et al. 2016)	(Wang 2010)	(ElAzab 2016)	(Giatsoglou et al. 2015)	(Dutta et al. 2018)	(Hooi et al. 2016)	Our
Address collusion phenomenon					✓		✓
Consider graph information		✓				✓	✓
Consider topic information							✓
Unsupervised approach	✓			✓		✓	✓
Return ranked list of users				✓			✓
Detect both collusive users and tweets							✓
Theoretical guarantees						✓	✓

Table 1: Comparison of `CoReRank` and other baseline methods w.r.t different dimensions of an algorithm.

Table 1 compares `CoReRank` with other related work. For our work, we address the following research questions:

- How can we design an efficient system to **simultaneously** detect users (based on their unusual retweeting pattern) and tweets (based on the credibility of the users who retweet them) involved in collusive blackmarket services?
- How can we develop an algorithm that detects collusive users, addressing the fact that there is a scarcity of labelled data? Can some labelled data be leveraged to enhance the algorithms?
- Is collusion detection really different from other fraudulent detection algorithms? How do other state-of-the-art algorithms perform in detecting collusion?

*Work done in collaboration with Aditya Chetan, Hridoy Sankar Dutta and Tanmoy Chakraborty, all from the same institute.

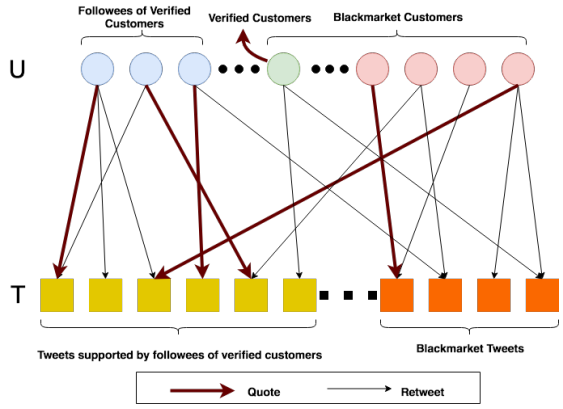


Figure 1: Graph Construction: $G(U, T, E)$ where $|U| = 10450$, $|T| = 2440320$, $|E| = 2962737$

Contribution

Dataset and Preliminaries

We used *Active Probing* strategy to mine tweets and users from Credit-based services (eg. YouLikeHits, etc.) and their 1-hop follower-followee network on Twitter. The data-collection and annotation took 4 months of rigorous participation in the Blackmarket ecosystem to draw inferences from the behaviour. [1] From the collected data, a directed bipartite graph called $G(U, T, E)$ (as demonstrated in Figure 1), is constructed, where U is the set of users, T is the set of tweets, and E is the set of edges. A quote edge has a higher weight than a retweet edge. We define **credibility of a user** u, $C(u)$: $C(u) \in (0, 1)$ as an indication of how likely they are to support a tweet based on their genuine agreement with the content of the tweet and **merit of a tweet** t , $M(t)$: $M(t) \in (0, 1)$ as an indication of the genuine organic support of users.

Methodology

CoReRank incorporates network dependencies from G, behavioral activities (using (Hooi et al. 2016)), topical similarities, cold start and label (for CoReRank+) to update the Credibility and Merit scores, motivated by (Kumar et al. 2018). (Highlighted text corresponds to respective contributions in the equations). This is similar to label propagation methods for semi-supervised learning.

$$M(t) = \frac{\gamma_{1t} \cdot \sum_{u \in \text{In(t)}} C(u) \cdot S(u,t) + \gamma_{2t} \cdot \pi_T(t) + \gamma_{3t} \cdot \mu_T + \alpha_T(t)}{\gamma_{1t} + \gamma_{2t} + \gamma_{3t} + |\text{In(t)}|} \quad (1)$$

$$C(u) = \frac{\gamma_{1u} \cdot \sum_{t \in \text{Out(u)}} M(t) \cdot S(u,t) + \gamma_{2u} \cdot \pi_U(u) + + \gamma_{3u} \cdot \tau_U(u) + \gamma_{4u} \cdot \mu_U + \alpha_U(u)}{\gamma_{1u} + \gamma_{2u} + \gamma_{3u} + \gamma_{4u} + |\text{Out(u)}|} \quad (2)$$

The γs listed here are parameters learnt by Parameter Sweeping. $C(u)$ and $M(t)$ are mutually updated at every iteration, until convergence.

[1] Code and dataset available at - https://github.com/LCS2-IIITD/CoReRank-WSDM-2019

Theoretical Guarantees

Apart from contributing to the development of the algorithm, I also led the derivation of theoretical guarantees for our algorithm - theorem of convergence, bound on iterations and complexity analysis ($O(|E|)$), along with scalability, robustness and ablative analysis, which cannot be listed here in detail due to space constraints.

Impact and Future Roadmap

Our algorithm returns a list of users and tweets, ranked by their collusive nature. This can be effectively used to identify such users and remove/limit their influence. Identifying collusive users and their acquired collusive retweets can be important for marketing/brand promotion, viral content detection and improving recommender systems by removing collusive outliers. For the broader AI Community, we present a new research direction to study and analyse, along with the code and dataset made available for the purpose of reproducibility. Further, I hope to continue this work as follows:

- Scaling this as a chrome-extension or a web-demo would help us make the algorithm more accessible.
- CoReRank studies collusion from the perspective of Retweets. A multi-view approach (Retweets, follower-followee network and likes) would help us gain a more holistic understanding of collusion.

References

Chetan, A.; Joshi, B.; Dutta, H. S.; and Chakraborty, T. 2019. Corerank: Ranking to detect users involved in blackmarket-based collusive retweeting activities. In *Proceedings of WSDM 2019*, 330–338.

Davis, C. A.; Varol, O.; Ferrara, E.; Flammini, A.; and Menczer, F. 2016. Botornot: A system to evaluate social bots. In *Proceedings of WWW 2016*, 273–274.

Dutta, H.; Chetan, A.; Joshi, B.; and Chakraborty, T. 2018. Retweet us, we will retweet you: Spotting collusive retweeters involved in blackmarket services. In *2018 IEEE/ACM ASONAM*, 242–249.

ElAzab, A. 2016. Fake accounts detection in twitter based on minimum weighted feature. *World*.

Giatsoglou, M.; Chatzakou, D.; Shah, N.; Beutel, A.; Faloutsos, C.; and Vakali, A. 2015. Nd-sync: Detecting synchronized fraud activities. In *PAKDD*, 201–214.

Hooi, B.; Shah, N.; Beutel, A.; Günnemann, S.; Akoglu, L.; Kumar, M.; Makhija, D.; and Faloutsos, C. 2016. Birdnest: Bayesian inference for ratings-fraud detection. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, 495–503. SIAM.

Kumar, S.; Hooi, B.; Makhija, D.; Kumar, M.; Faloutsos, C.; and Subrahmanian, V. 2018. Rev2: Fraudulent user prediction in rating platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, 333–341.

Wang, A. H. 2010. Detecting spam bots in online social networking sites: a machine learning approach. In *IFIP Annual Conference on Data and Applications Security and Privacy*, 335–342.

Incremental Open World Reference Resolution

Thomas Bennett
Colorado School of Mines
Golden, Colorado 80401
tbennett@mymail.mines.edu

Abstract

Humans commonly refer to people, places, and objects in natural language according to their characteristics and relationships. Identifying the targets of referring expressions, reference resolution, is a central part of language understanding. Recent work on reference resolution has sought either to make reference resolution algorithms operate incrementally, or to allow reference resolution to be performed in open worlds. I am working to combine these ideas to enable robots to incrementally identify intended referents in open worlds; an advancement that will both enable more efficient human-robot interaction and more plausible cognitive models of human reference resolution.

Introduction

Reference resolution is the process of identifying real world objects, locations, and people, that are referred to in natural language. In traditional reference resolution algorithms, this is achieved by first listening to an entire sentence, parsing that sentence into a set of semantic constraints, and then identifying the objects described in the sentence based on a knowledge base of potential candidates in the current environment and the semantic constraints that apply to them (Chai, Hong, and Zhou 2004). Critically, this approach is inconsistent with psycholinguistic accounts of reference resolution, which suggest that humans *incrementally* resolve references as an utterance unfolds word by word, rather than waiting to hear an entire sentence (Poesio and Rieser 2011).

Both incremental and open world reference resolution have been pursued independently and have seen good results. Approaches toward incremental reference resolution have been presented by Kennington and Schlangen (2015) as well as Poesio and Rieser (2011). Similarly, approaches towards open-world reference resolution have been presented by Williams and Scheutz (2015; 2016) and by Duvallet et al. (2016). In Williams and Scheutz's work on the Probabilistic, Open-World Entity Resolution (POWER) algorithm, for example, they show how new object representations can be hypothesized and asserted into a knowledge base during reference resolution, allowing robots to communicate about entities in their environment without needing to have previously observed those entities. However, to our knowledge, there had been no work on open-world reference resolution that is also incremental to date. I propose to fill this research gap by developing an incremental version of the POWER algorithm.

Algorithmic Approach

My approach to *incremental* probabilistic open-world entity resolution (IPOWER) takes the following parameters:

- S: A set of semantic constraints created by the parser from the most recently heard portion of a referring expression.
- P: An initially empty set of variables used to keep track of the semantic history of the utterance.
- M: A consultant[1] able to provide information about the entities within the environment, their properties, and their relationships to each other.
- H: An initially empty set of hypotheses for bindings between variables V appearing in S and entities in M.
- U: An initially empty set of set of semantic constraints that is used to store variables that need to be hypothesized at the end of the utterance.
- E: A flag indicating that the current semantic constraint is the last semantic constraint of the clause being analyzed.

Using these parameters, IPOWER seeks to incrementally find the set of all bindings from variables appearing in semantic constraints $(v \in S^V)$ to entities $m \in M$ such that the joint probability of all semantic constraints being satisfied under that binding is above some threshold defined in DIST-CoWER (an algorithm for reference resolution under a simplifying closed world assumption (Williams 2017b)). If no hypotheses meet the threshold, this signals that new entities must be hypothesized for one or more of the entities referred to thus far. This process is defined in Algorithm 1.

The algorithm first checks if the set of hypotheses is empty. If so, IPOWER creates a set of initial hypotheses, each of which binds the first variable of the first semantic

[1] Cp. the consultant framework presented by Williams (2017a).

Algorithm 1 $IPOWER(S, P, H, M, U, E)$

if $H = \emptyset$ **then**
  $v = S_0^{V_0}$
  for all $m \in M$ **do**
    $b = (v \rightarrow m)$
    $H = H \cup \{\{b\}, P, 1.0\}$
  end for
end if
$H' = DIST - CoWER(S^V, S, H, M)$
if $H = \emptyset$ **then**
  $S' = \{s \in (P \cup S) | s^V = s_0^{V_0}\}$
  return $IPOWER((P \cup S \setminus S'), \emptyset, \emptyset, M, (U \cup S'), E)$
else
  if E = true **then**
    return $(\emptyset, P \cup S, H', M.update(U), \emptyset, E)$
  else
    return $(\emptyset, P \cup S, H', M, U, E)$
  end if
end if

constraint ($S_0^{V_0}$) to a different entity $m \in M$. IPOWER then uses DIST-CoWER to reduce the current set of hypotheses to only those that sufficiently satisfy the set of semantic constraints S.

If DIST-CoWER yielded no satisficing hypotheses, IPOWER will assume that the first unbound variable appearing in constraints in S should be associated with a previously unknown entity, and set aside all constraints that involve that variable. A recursive call is then made to IPOWER using only the remaining semantic constraints, plus any semantic constraints IPOWER previously thought it had satisfactorally dealt with.

If instead DIST-CoWER returns a non-empty set of hypotheses, IPOWER will move S into the semantic history, and, if flag E was set to mark the end of the clause, update M with the contents of U. IPOWER then returns the final state of the semantic history, resolution hypotheses, knowledge base, and set of knowledge base updates to perform at clause end (if any).

Future Work

I am currently implementing this algorithm as part of the DIARC cognitive architecture (Scheutz et al. 2019). Once evaluation of this implementation is complete, many avenues will be open to build on this work. I hope, for example, to pursue the use of distributed knowledge during incremental reference resolution, as previously achieved by POWER (Williams and Scheutz 2016). This entails having multiple knowledge bases distributed throughout the virtual robot system in order to reduce bottleneck issues that come up when there is one centralized knowledge base.

When looking at real life scenarios, there is another application that becomes apparent for IPOWER. Currently reference resolution focuses on physical objects in the environment around the robot and how they are referred to. IPOWER could be applied to non-physical entities, like abstract concepts that are not in the environment, and how they are referenced. IPOWER could also be parallelized, leveraging the powerful GPU technology on the market today. This could also allow IPOWER to more easily leverage large knowledge bases expected to be used in real-life scenarios. This idea is supported in the work of Poesio and Rieser (2011) where they suggest that, unlike Frazier's Garden Path Theory (Frazier 1978; Coltheart 2016) which follows a serial approach, hypotheses can be generated and analyzed in parallel. Finally, while I expect IPOWER to open up many avenues of future research in reference resolution, its success ultimately depends upon every other piece of the natural language pipeline also being incremental, without which it may be difficult to achieve the positive effects I expect with respect to efficiency and perceived naturality, competence, and intelligence. As such, I am also interested in exploring incremental approaches to other aspects of computational natural language pragmatics.

References

Chai, J. Y.; Hong, P.; and Zhou, M. X. 2004. A probabilistic approach to reference resolution in multimodal user interfaces. In *Proceedings of the 9th international conference on Intelligent user interfaces*, 70–77. ACM.

Coltheart, M. 2016. *Attention and performance XII: The psychology of reading*. Routledge.

Duvallet, F.; Walter, M. R.; Howard, T.; Hemachandra, S.; Oh, J.; Teller, S.; Roy, N.; and Stentz, A. 2016. Inferring maps and behaviors from natural language instructions. In *Experimental Robotics*, 373–388. Springer.

Frazier, L. 1978. On comprehending sentences: Syntactic parsing strategies. *Doctoral dissertation, University of Connecticut*.

Kennington, C., and Schlangen, D. 2015. Simple learning and compositional application of perceptually grounded word meanings for incremental reference resolution. In *Proc. ACL*, 292–301.

Poesio, M., and Rieser, H. 2011. An incremental model of anaphora and reference resolution based on resource situations. *D&D* 2:235–277.

Scheutz, M.; Williams, T.; Krause, E.; Oosterveld, B.; Sarathy, V.; and Frasca, T. 2019. An overview of the distributed integrated cognition affect and reflection diarc architecture. In *Cognitive Architectures*. Springer. 165–193.

Williams, T., and Scheutz, M. 2015. Power: A domain-independent algorithm for probabilistic, open-world entity resolution. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1230–1235. IEEE.

Williams, T., and Scheutz, M. 2016. A framework for resolving open-world referential expressions in distributed heterogeneous knowledge bases. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Williams, T. 2017a. A consultant framework for natural language processing in integrated robot architectures. *IEEE Intelligent Informatics Bulletin*.

Williams, T. 2017b. Situated natural language interaction in uncertain and open worlds. *AI Matters* 3(2).

Human-in-the-Loop Gaussian Process-Based Learning and Coverage in Multi-Robot Settings

Andrew McDonald
Distributed Cyber Physical Human Systems Lab
Michigan State University
428 South Shaw Lane
East Lansing, MI 48824
mcdon499@msu.edu

1 Introduction

Suppose a team of environmentalists were to deploy a swarm of pollutant-removal robots in the area surrounding an industrial accident. Areas with high concentrations of pollutant would correspond to areas of high demand for robots, while unpolluted areas would correspond to areas of low demand; the *demand function* across the region would be proportional to pollutant levels. Surely, the environmentalists deploying the swarm would have a rough idea regarding the characteristics and shape of the distribution of pollutant, and could thereby provide valuable prior information to the swarm. Given this inherently uncertain human-input prior, how should the swarm divide its time *exploring* by measuring pollutant levels at different locations across the region, and *exploiting* by converging on areas of high pollutant concentration for cleanup?

More generally, given a team of robots, a planar region over which an unknown demand function is defined, and a human-input estimate of the demand function, the problem we consider is twofold: first, we wish to traverse the region and sample the demand function at various points to learn its mapping — second, we wish to position our robots in a manner which equally partitions the demand volume over the region into a collection of Voronoi cells, each centered at the position of a robot. Intuitively, we consider an *explore-exploit* problem in which *exploration* entails learning the demand function given a human-input prior and *exploitation* involves converging on a configuration which equally divides volume under the demand function over our region among robots in a locationally optimal manner.

2 Problem

We frame the problem of *coverage* as in (Diaz-Mercado, Lee, and Egerstedt 2015; Cortés and Egerstedt 2017) — let $D \subset \mathbb{R}^2$ be a convex region in the plane, and let $f : D \to \mathbb{R}^+$ be a demand function which assigns a non-negative $f(x)$ to each point $x \in D$. Let $A = \{a_1, a_2, ..., a_N\}$ be a collection of N robotic agents each capable of traversing D and measuring the value of the demand function $f(x)$ at any given $x \in D$. For a given configuration $X = \{x_1, x_2, ..., x_N\}$ where robot a_i is positioned at $x_i \in D$, define the Voronoi partition $\mathcal{V}_D(X)$ of D to be the set $\mathcal{V}_D(X) = \{v_1, v_2, ..., v_N\}$ of Voronoi cells $v_i \subset D$ which partition D by assigning points closest to robot a_i to the cell v_i

$$v_i = \{x \in D \mid \|x - x_i\| \leq \|x - x_j\| \ \forall j \neq i\} \tag{1}$$

and define the loss function L which assigns a non-negative loss score to a given Voronoi partition $\mathcal{V}_D$ by

$$L(\mathcal{V}_D(X)) = \sum_{i=1}^{N} \int_{v_i} \|x - x_i\|^2 f(x) dx \tag{2}$$

Moreover, define the mass m_i and center of mass c_i of the cell v_i to be

$$m_i = \int_{v_i} f(x) dx, \quad c_i = \frac{1}{m_i} \int_{v_i} x f(x) dx \tag{3}$$

Assuming an initial configuration of $X_0 = \{x_{1_0}, x_{2_0}, ..., x_{N_0}\}$ where robot a_i is positioned at $x_{i_0} \in D$ and where we have no prior information regarding the behavior of f, we ultimately wish to reach a final configuration $X_f = \{x_{1_f}, x_{2_f}, ..., x_{N_f}\}$ where robot a_i is positioned at $x_{i_f} \in D$ and the Voronoi partition $\mathcal{V}_D(X_f)$ associated with this configuration minimizes our loss function, i.e.

$$X_f = \underset{X}{\operatorname{argmin}}[L(\mathcal{V}_D(X))] \tag{4}$$

Defining the problem in this manner leads robots to cluster more densely in higher-demand areas and spread more sparsely in lower-demand areas as we intuitively seek.

Once the behavior of f is known, converging to X_f is trivial as shown in (Cortés and Egerstedt 2017): iteratively applying Lloyd's Algorithm at each timestep t to create a new Voronoi partition $\mathcal{V}_D(X_t)$ with centers of mass $c_i \in v_i$ for each $v_i \in \mathcal{V}_D(X_t)$ and updating robot positions to the configuration $X_{t+1} = \{c_1, c_2, ..., c_N\}$ where each robot a_i moves to the center of mass c_i of its Voronoi cell v_i will achieve near-optimal performance asymptotically.

Hence, the key challenge is to perform coverage of D while simultaneously learning the demand function f. To

learn f, however, a large number of samples may be needed, effectively prohibiting an approach that strictly learns f from scratch before performing coverage. Human input can provide a good prior such that initial coverage is sufficiently accurate — such an approach forms the basis for our work.

3 Approach

Previous works such as (Todescato et al. 2017; Low, Dolan, and Khosla 2008; Ouyang et al. 2014) leverage nonparametric Gaussian Process regression to learn f with impressive success — our primary contribution lies in extending this approach to incorporate human input, kickstarting and accelerating the learning process to minimize use of time and capital. Unlike other approaches which exclusively use samples collected from the robots to estimate hyper-parameters of the GP kernel and refine estimates of the demand function, we leverage human input to first construct a prior distribution of f and thereby reduce the set of sample points necessary to learn f to a given level of accuracy.

Formally, we model $f(x) \sim GP(\mu(x), k(x, x'))$ as a draw from a *Gaussian Process* with mean $\mu(x) = \mathbb{E}[f(x)]$ and covariance $k(x, x') = \mathbb{E}[(f(x)-\mu(x))(f(x')-\mu(x'))]$, defined to be a collection of random variables from which any finite subset are jointly Gaussian (Rasmussen and Williams 2006). Assuming we have sampled f and observed $y = f(x) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$ denotes Gaussian noise on a set of points $X \subset D$, we may predict the mean value $\bar{f}_*$ of $f_* \triangleq f(X_*)$ on a set of test points X_* by the equation

$$\bar{f}_* = K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1} y \tag{5}$$

where $K(X_*, X), K(X, X)$ are the covariance matrices of points in X_*, X with X defined by k, respectively (Rasmussen and Williams 2006). Likewise, we may predict the covariance matrix of f_* on a test set X_* by the equation

$$\begin{aligned} cov(f_*) =& K(X_*, X_*) - \\ & K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1} K(X, X_*) \end{aligned} \tag{6}$$

and hence may predict the variance of f_* at each individual point $x_* \in X_*$ by taking the diagonal entries of $cov(f_*)$, giving us a metric of the confidence we may stake in our prediction $\bar{f}_*$.

Within this framework, we propose two approaches to learning f and covering D, each incorporating human input. Both begin equivalently: a human with prior knowledge of f is presented with a graphical interface representing D, and clicks points $\{p_1, p_2, ...p_r\} \subset D$ between 1 and m times each to specify an estimate of the value of $f(p_i)$ on a scale of 1 to m at every $p_i \in D$. These sets $P = \{p_1, p_2, ...p_r\}$ and $f(P) = \{f(p_1), f(p_2), ...f(p_r)\}$, along with a global confidence estimate α (input by the user) which determines σ_n^2 in (5), (6) are used to initialize a Gaussian Process model GP, forming the basis for learning to build upon.

In our first approach, we fix a variance threshold δ and divide our algorithm into two phases: exploration and exploitation. Exploration proceeds iteratively: for each timestep t, we first compute the covariance matrix $cov(f_*)$ on a discrete mesh of test points $X_* \subset D$ using (6), and consider its diagonal entries which give the variance of f_* at each individual point $x_* \in X_*$. We find the point $x_{**} \in X_*$ for which $f(x_{**})$ has maximum variance; i.e.,

$$x_{**} = \underset{x_* \in X_*}{\operatorname{argmax}}[cov(f(x_*), f(x_*))] \tag{7}$$

and compute $\bar{f}(x_{**})$ using (5). We then condition our model GP on $(x_{**}, \bar{f}(x_{**}))$ as though we have observed it, reducing global variance, and proceed to timestep $t + 1$ until the point x_{**} with maximum variance satisfies $cov(f(x_{**}), f(x_{**})) \leq \delta$.

At this point, we have a set of variance-maximizing points $X_{**} = \{x_{**1}, x_{**2}, ..., x_{**m}\}$ at which to sample $f(X_{**})$ in order to reduce global variance below δ, and we switch to exploitation. Assuming N robots, the points of X_{**} are partitioned into N clusters by a k-means algorithm, and each robot is assigned a cluster. The robots perform a Traveling Salesman Tour of their cluster to collectively sample $f(X_{**})$, and the model GP is conditioned on the new set of observations $(X_{**}, f(X_{**}))$. Finally, a discrete mesh of test points $X_* \subset D$ are used to estimate $\bar{f}_*$ by (5), and coverage is performed over D to minimize (2).

In our second approach, we draw inspiration from (Todescato et al. 2017) and merge exploration with exploitation more subtly. We proceed iteratively, and at each timestep t decide to explore (0) or exploit (1) based on the value of a random variable $Y \sim Bernoulli(p)$ distributed with $p = e^{-c\lambda}$ where $c > 0$ is a real constant and λ is the maximum variance of a test point in the discrete grid $X_* \subset D$, i.e.

$$\lambda = \max_{x_* \in X_*} [cov(f(x_*), f(x_*))] \tag{8}$$

To execute an explore step, each robot $a_i \in A$ samples f at the variance-maximizing point x_{**i} from (7) within its Voronoi cell v_i from (1) with X_* restricted to $X_* \cap v_i$, and the model GP is conditioned on these samples $(x_{**i}, \bar{f}(x_{**i}))$ for $i = 1, 2, ...N$. To execute an exploit step, each robot $a_i \in A$ moves to the center of mass c_i from (3) of its current Voronoi cell v_i, effectively performing one iteration of Lloyd's Algorithm (Cortés and Egerstedt 2017).

Note that this strategy leads to a gradual shift in policy, favoring exploration while λ is large and exploitation as λ becomes small. As shown in (Todescato et al. 2017), we converge to a local minimum of (2) as $t \to \infty$.

4 Next Steps

Leveraging the open-source swarm robotics control framework OpenSwarm (McDonald 2019), we are in the process of implementing the two approaches discussed herein to achieve physical coverage of a testbed given human input, comparing the performance of each approach with its null-prior analog on the basis of time and distance traveled to achieve a threshold level of convergence. Such experiments will demonstrate the impact of human input in the dual challenge of learning and coverage, and may inspire applications to other explore-exploit problems in the process. We plan to submit a full review of our work to IEEE IROS 2020 (Las Vegas) by the submission deadline of March 1.

References

Cortés, J., and Egerstedt, M. 2017. Coordinated Control of Multi-Robot Systems: A Survey. *SICE Journal of Control, Measurement, and System Integration* 10(6):495–503.

Diaz-Mercado, Y.; Lee, S. G.; and Egerstedt, M. 2015. Distributed dynamic density coverage for human-swarm interactions. In *2015 American Control Conference (ACC)*, 353–358. Chicago, IL, USA: IEEE.

Low, K. H.; Dolan, J. M.; and Khosla, P. 2008. Adaptive Multi-Robot Wide-Area Exploration and Mapping. In *Proceedings of 7th International Conference on Autonomous Agents and Multiagent Systems*, 8. Estoril, Portugal: International Foundation for Autonomous Agents and Multiagent Systems.

McDonald, A. 2019. OpenSwarm: An Open-Source Swarm Robotics Control Framework. Distributed Cyber Physical Human Systems (D-CYPHER) Lab, Michigan State University: https://github.com/andrewmcdonald27/OpenSwarm.

Ouyang, R.; Low, K. H.; Chen, J.; and Jaillet, P. 2014. Multi-Robot Active Sensing of Non-Stationary Gaussian Process-Based Environmental Phenomena. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014)*. Paris, France: International Foundation for Autonomous Agents and Multiagent Systems.

Rasmussen, C. E., and Williams, C. K. I. 2006. *Gaussian processes for machine learning*. Adaptive computation and machine learning. Cambridge, Mass: MIT Press. OCLC: ocm61285753.

Todescato, M.; Carron, A.; Carli, R.; Pillonetto, G.; and Schenato, L. 2017. Multi-robots Gaussian estimation and coverage control: From client–server to peer-to-peer architectures. *Automatica* 80:284–294.

Generating Crochet Patterns With Recurrent Neural Networks

Proscovia Nakiranda [1], **Engineer Bainomugisha** [2], **Daniel Mutembesa** [2]
Makerere University
nakirandaproscovia@gmail.com,
mutembesadaniel@gmail.com,
baino@cis.mak.ac.ug

Abstract

Crocheting is a process of creating fabric by interlocking loops of yarn, thread, or strands of other materials using a crochet hook.It has been used to make clothing, décor, accessories, and recently in architecture, and mathematics to explain different geometrical concepts . It can be used to come up with design objects of up to three dimensions. To come up with a given object shape, one has to follow algorithmic instructions called crochet patterns. Coming up with new innovative patterns is a non-trivial task and challenging especially for beginners. Recently software tools for creating crochet patterns have emerged. However, most existing software are limited to generating two-dimensional patterns and require a sketch from a user as first input. This paper proposes the use of a machine learning method, specifically recurrent neural networks (RNN) to generate three-dimension crochet patterns. The model is trained with existing crochet patterns and it learns generate new crochet patterns automatically. In particular the model is based on a special kind of RNN known as a Long-Short-Term-Memory (LSTM) network. LSTM networks have enhanced memory capability, creating the possibility of using them for learning and generating crochet patterns. Keywords—Crochet patterns, cultures, LSTM RNNs, crochet generation.

Background

Crocheted handcrafting is a cultural tradition, predominantly a cottage art form, practiced by the rural youth of both genders, but mainly by women. In most developing countries where tourism contributes to local incomes, crocheted crafts form the bulk of art works sold locally. Crochet is a creative art that has been used in a variety of ways: to entertain, in fashion, to delight, to challenge, to give meaning, to interpret, and to raise awareness. For example, Dr. Hinke Osinga and Professor Bernd Krauskopf(Osinga and Krauskopf 2004) crocheted a magnificent Lorenz manifold using 25,511 crochet stitches. Crochet has also been used to illustrate shapes in hyperbolic (Henderson and Taimina 2001) space that are difficult to reproduce using other media or are difficult to understand when viewed two-dimensionally. However, many crocheters have difficulty in coming up with new and better patterns since like any other craft, it is handed down in tradition of craftsmanship and generational mentoring.

Furthermore, novice crocheters need simplified algorithms for studying patterns. In this study, we propose the use of Recurrent Neural Networks (RNNs) called LSTM(Hochreiter and Schmidhuber 1997) to automate the crochet pattern generation, the model is trained on a number of sophisticated patterns and it learns to generate new crochet patterns. In literature, RNNs have had success in generation of text (Sutskever, Martens, and Hinton 2011) , music lyrics generation (Potash, Romanov, and Rumshisky 2015) and poetry generation(Yan 2016)

Problem Statement

Crochet instructions are superficially language like but obviously not written in natural language. They are sequence like poetry and lyrics because they follow a given convention but crochet patterns are complicated because:

- they might include stitch sequences that might be repeated multiple times in a row
- consistency is key, inconsistency can lead to change in curvature.
- change in color can cause change in the design of the object

Approach

We propose the use of LSTM (Hochreiter and Schmidhuber 1997) for generating crochet patterns. LSTM are capable of learning long term dependencies and solve the problem of vanishing gradient. We use a stacked LSTM model.We then compare our custom model to a pretrained RNN model called textgenrnn (Woolf 2018).textgenrnn is a python 3 model on top of Keras/Tensorflow that uses a CuDnn implementation of RNN when trained on a GPU,which helps speed up training time as opposed to typical LSTM implementations.However,we use custom model to detect bias from the pretraining to the model.

Methodology

Data Properties

Crochet patterns are based on repeated stitches. basic stiches are chain stitch(ch),double crochet(dc),single crochet(sc),slip stitch(sl st),treble or triple crochet.

chain 12. Row 1:sc in 2nd ch from hook,sc in each across. Means make 12 chain stitches, skip the first chain away from the hook. Single crochet in the 2nd chain away from the hook and in each ch across

Figure 1 below shows word cloud of most used stitches and words to come up with patterns in the dataset.

Figure 1: word cloud of the patterns

Model

Our custom model was trained on crotchet pattern text containing average 13,152 sentences downloaded from existing public online resources and from crochet books giving a total of Training on 1,327,516-character sequences.

We used a stacked lstm with three lstm layers, added dropout(Srivastava et al. 2014) layer to prevent overfitting. A dropout rate of 0.4 was applied. The model had a Dense layer, Activation softmax, Loss categorical cross entropy, Optimizer Adam (Kingma and Ba 2014)and to widen network we used 200 units per LSTM layer We trained for a given number of epochs to try and minimize the loss function. The model was developed in keras using TensorFlow backend. For textgenrnn we specified the size and complexity of the neural network to be the same as the custom LSTM network. We used 3 RNN layers each with 200 units, dropout of 0.4.

Results

We are in the process of generating results and report our work in progress. Figure 2 below shows some of the patterns generated. The accuracy of the models increased as the number of epochs increased.We generated patterns using varying temperatures. The temperature is a setting that controls the extent to which the generated patterns deviate from the model predictions. Lower temperatures cause the model to make predictions that are similar to the training data while higher temperatures cause the model to generative creative patterns.

However,the pretrained model trains faster and performs better than the custom-made model. Although the LSTM approach is not yet successfully in learning the syntax of crochet expressions, the networks come up with new interesting words like ncith, bediwld, repafreat, bowwetickeos, rotnd, stefcheas and much more which confirms Graves'(2013) statement about character level networks being able to invent new words. The number of stitches in most patterns generated are not consistent hence shapes are curved and bend at some stage.As reported by Daina Taimina(2001), the curvature of a crochet surface can be changed by adding few or more stitches as you crochet.However, adding too many stitches causes a negative curvature creating shapes like a hyperbolic plane and too few will lead to negative curvature causing dome forms like a sphere.

Chain 12.

Row 1: Sc in 2nd ch from hook, sc in each ch across (8 sc), turn.

Row 2: Ch 1, sc in each st across, turn.

Row 3: Ch 2 (does not count as st throughout), dc in each st around, ss to

Figure 2: Sample patterns generated by our RNN method

Discussion/challenge/lessons

We now discuss the challenges we have encountered with the crochet pattern learning process:

One of the challenges is that there are no automatic syntax checkers for crochet patterns.

Crocheters have different writing styles.Some crochet patterns are wordy because some crocheters add in extra textual information to explain instructions they find complex and this makes it difficult for the model to learn.

Developing an automatic syntax checker that weeds out explanations and comments not necessary for machine learning is important to this work.

The training process is computation and time intensive.We had access to limited computational power which prevented us from running large-scale training runs.We plan to study alternate algorithms that can produce novel patterns with limited computational costs while still complying with the constraints for producing correct patterns.

Conclusion

This work presents preliminary results to understand ways of developing new crochet patterns that are unique and sensitive to the culture of the community from which the patterns originated. This is a promising method for promoting the uniqueness of community artforms. In future, we hope to collect a dataset from different regions of the world, develop a model that takes the crochet pattern and represents the diagrammatically.In addition to looking into automatic syntax checkers for the patterns and computationally tractable algorithms,we also hope to build a model with some form of a discriminator so that no human intervention is needed to tell which patterns are syntax and semantically correct

Acknowledgments

We would like to thank Professor Anita Raja and anonymous reviewers for their feedback

References

Graves, A. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Henderson, D. W., and Taimina, D. 2001. Crocheting the hyperbolic plane. *Mathematical Intelligencer* 23(2):17–27.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Osinga, H. M., and Krauskopf, B. 2004. Crocheting the lorenz manifold. *Mathematical Intelligencer* 26(4):25–37.

Potash, P.; Romanov, A.; and Rumshisky, A. 2015. Ghostwriter: Using an lstm for automatic rap lyric generation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1919–1924.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15(1):1929–1958.

Sutskever, I.; Martens, J.; and Hinton, G. E. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 1017–1024.

Woolf, M. 2018. textgenrnn. *https://github.com/minimaxir/textgenrnn*.

Yan, R. 2016. i, poet: Automatic poetry composition through recurrent neural networks with iterative polishing schema. In *IJCAI*, 2238–2244.

Nonparametric Vehicle Following Model

John Nguyen[1]
[1]University of Minnesota, Twin Cities
200 Union Street SE
Minneapolis, MN, 55455
nguy2539@umn.edu

Abstract

Classic car following models are used to model the driving behavior of human drivers following another vehicle. Due to the limited availability, these models tend to follow a simple explicit functional form. This project aims to utilize this new dataset to develop nonparametric models for human driving.

Introduction and Background

In order to better train autonomous vehicles, it is necessary to understand how humans drive. Due to limitations on available data, it has been difficult to study driving behavior, particularly in busy streets and highways. Consequently, simple parametric models have been used for the past decade (Gunter et al. 2019a). For instance, some models assumed a constant reaction time for every vehicle, or that there is an optimal inter-vehicle distance irrespective of driving conditions such as weather. These models fail to completely capture driving behavior because they assume drivers select acceleration according to a pre-specified formula. This project attempts to develop a model based only on vehicle data.

We focus our study on predicting both velocity and acceleration from the given data. The HighD dataset (Krajewski et al. 2018) includes data from 60 recordings of highways, where position, velocity, acceleration, vehicle type and other factors have been determined for each vehicle in every frame of the recording. As a result, the HighD dataset creates new opportunities for further research in vehicle following models. This is the most comprehensive car following dataset to date.

This project was Dr. Raphael Stern's original idea, which was inspired by the recent publication of large traffic datasets. My primary contribution to this project is to develop a nonparametric regression model which could accurately predict driving behavior. I wrote all of the code, including the necessary preprocessing, training/testing the model and creating the figures. Dr. Stern analyzed the results and advised me on what to do next, but I wrote all the code.

Model and Data

The HighD dataset has 60 recordings, each with associated metadata and data files. Using the metadata, we were able to focus on vehicles which exhibit our desired behavior: vehicles which drove from one end of the frame to the other without lane changing. This is because lane changing is another dimension of driving which we will study later. After finding the vehicles which exhibited this behavior from the metadata file, we extracted our desired features from the associated data file. We focused on the current (follower) vehicle and the lead vehicle. For a given frame, the features we used are:

- Follower Vehicle Type (Car or Truck)
- Follower Vehicle Velocity
- Follower Vehicle Acceleration
- Inter-vehicle distance
- Leader Vehicle Type (Car or Truck)
- Leader Vehicle Velocity
- Leader Vehicle Acceleration

Our model takes in the data for a given frame and attempts to predict the acceleration of the follower vehicle in the next frame. When a vehicle does not have a lead vehicle (there is no vehicle in front of it), we are not able to generate predictions because leader data is a required part of our features. We will remove this reliance later in the project.

It is sufficient to model driving behavior in a follower-leader model because the most immediate application of our research is in adaptive cruise control systems. These system are effectively follower-leader model which attempt to maintain safe driving distance in dynamic environments. Furthermore, we do not consider vehicles which change lanes at the moment, so it is sufficient to model traffic as a sequence of follower-leader pairs, where the follower of one pair is the leader in the next pair.

Our motivation is to develop a vehicle follower model that is based purely on data and not any predefined formulas or structures. Thus we aim to utilize nonparametric regression methods. At this stage, we have obtained our proof of concept results. The results were obtained from kernel ridge regression (KRR) model. We chose to use KRR for our proof

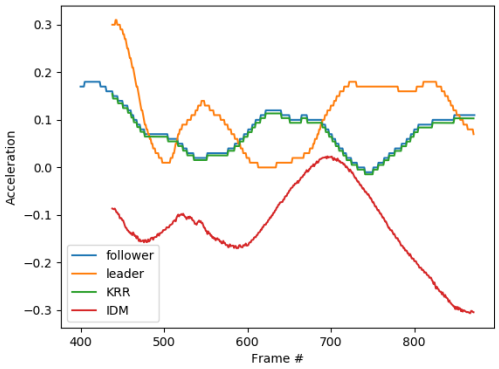


Figure 1: Plot of follower acceleration (blue), predicted (green) acceleration, IDM acceleration and leader acceleration (yellow).

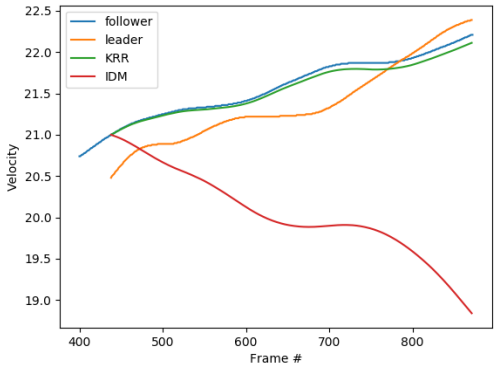


Figure 2: Plot of follower velocity (blue), predicted (green) velocity, IDM velocity (red) and leader velocity (yellow).

of concept because it is readily available in the scikit-learn (Pedregosa et al. 2011) python package. We fitted a KRR model to predict acceleration. We then calculated predicted velocity by calculating the Riemann sum of the predicted acceleration and velocity in he previous frame.

We trained our model on 1,000 data points from the first set of files in the HighD dataset. We wanted to obtain some initial results and move forward with the project based on our results.

Results and Analysis

In Figure 1 and Figure 2, we plot the acceleration and velocity of a vehicle, our model's predicted acceleration and velocity and the leader acceleration and velocity against the Intelligent Driver Model using parameters from (Gunter et al. 2019a). We have 20 other similar plots with similar results. Notice how it appears that our model almost perfectly predicts acceleration.

Our model seems to predict acceleration too well, so we started suspecting that they are overfitting. While this is likely the case, there are a few things that suggest otherwise. To begin, the training and testing data comes from two different recordings of the HighD dataset. Therefore, our model appears to be overfitting on data it has never seen before. Furthermore, we are only training on a very small fraction of our data. Thus it is confusing how our model may overfit data when it has only been trained on less than 1% of the HighD dataset.

Conclusions and Future Work

Our project is ongoing, though we have obtained promising results. We are beginning to replicate our experiments done on the HighD dataset with a similar dataset based on adaptive cruise control systems (Gunter et al. 2019b). If we are able to produce similar results, we will proceed assuming our model is not overfitting and begin to study how a KRR model can learn human driving behavior so well. If we fail to produce similar results to our HighD experiment, we will begin to study how to avoid overfitting. Ultimately, though our results require further analysis, we have demonstrated the potential use of nonparametric regression in vehicle follower models.

We expect that results from the new dataset will be obtained within the next month. Afterwards, it will take a few months to program some classic parametric driver models to compare our model against. We plan to begin submitting our results to conferences early next summer.

Our research so far is meant to be a proof of concept for the use of nonparametric regression in follower models. More robust and advanced models can be tested on driving data. Another approach may be using unsupervised learning methods used to classify similar driving behaviors.

Acknowledgments

I would like to thank Dr. Raphael Stern for his continued mentorship and advising throughout this project. Furthermore, I would like to thank Robert Krajewski, Julian Bock, Laurent Kloeker and Lutz Eckstein for allowing me to use their HighD dataset (Krajewski et al. 2018).

References

Gunter, G.; Janssen, C.; Barbour, W.; Stern, R. E.; and Work, D. B. 2019a. Model based string stability of adaptive cruise control systems using field data. *CoRR* abs/1902.04983.

Gunter, G.; Gloudemans, D.; Stern, R. E.; McQuade, S. T.; Bhadani, R.; Bunting, M.; Monache, M. L. D.; Lysecky, R.; Seibold, B.; Sprinkle, J.; Piccoli, B.; and Work, D. B. 2019b. Are commercially implemented adaptive cruise control systems string stable? *CoRR* abs/1905.02108.

Krajewski, R.; Bock, J.; Kloeker, L.; and Eckstein, L. 2018. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. *CoRR* abs/1810.05642.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Robustness in Macro-Action-Based Deep Multi-Agent Reinforcement Learning

Joshua Hoffman
Khoury College of Computer Sciences, Northeastern University
360 Huntington Ave, Boston, Massachusetts 02115
hoffman.j@husky.neu.edu

Abstract

Previous multi-agent deep reinforcement learning methods assumed synchronized action execution and updates by the agents. To allow for more complex action execution, we proposed two algorithms and frameworks that allow for asynchronous macro-actions to be used. These approaches learn policies that have agents act under a centralized controller and a decentralized controller respectively. We have evaluated our approaches both in simulation and via real robots. Beyond this published work, I have begun considering the case where a robot becomes inoperable while completing the task. Can the other robots learn to plan in this new dynamic and stochastic environment? In addition to the current work and research directions, future research aspirations are addressed and directions I will strive for in my doctoral studies and beyond.

Introduction

In the real world, robots, distributed systems, and self driving cars do not act synchronously. As such, multi-agent reinforcement learning (MARL) must account for this in order to behave optimally and perform collaborative heterogeneous tasks in larger, stochastic, and uncertain environments. Formally, multi-agent asynchronous decision making under partial observability in fully cooperative tasks is modeled as *Macro-Action Decentralized Partially Observable Markov Decision Processes* (MacDec-POMDPs) [Amato, Konidaris, and Kaelbling 2014]. While several state-of-the-art deep-MARL methods have achieved impressive results under both cooperative and competitive domains [Omidshafiei et al. 2017, Lowe et al. 2017, Foerster et al. 2018], they all assume that each agent acts and performs primitive actions synchronously.

Our work bridges this gap by introducing principled ways of learning asynchronous, macro-action-based policies [Xiao, Hoffman, and Amato 2019]. In many cases, agents are unable to learn a good policy as the state-action space is too large to efficiently explore. One method of overcoming this is well-designed macro-actions defined as a tuple containing a termination condition, an initiation set, and the policy of that macro-action. Through macro-actions, we are able to shrink the state-action space to learn more-optimal policies that can still achieve high-quality solutions.

Approaches and Significance

First, we propose a decentralized macro-action-based framework with a new experience replay buffer; this new buffer maintains separate yet concurrent macro-action-based transitions for each agent. Second, we introduce a centralized macro-action-based learning framework to learn a joint macro-action-value function using a conditional prediction method [Xiao, Hoffman, and Amato 2019]. The conditional prediction method allows for better asynchronous macro-action selection that ultimately increases the returned rewards. Next, we improve the decentralized macro-action methods by presenting *Macro-Action-Based Decentralized Multi-Agent Double Deep Recurrent Q-Net* (MacDec-MADDRQN) that enables each agent's Q-net update to consider and reason about the other agents' macro-actions. Moreover, without knowledge of the domain properties, it is unknown whether a centralized or decentralized exploration method generates accurate Q-values. Thus, MacDec-MADDRQN also performs centralized and decentralized exploration in parallel during training (Parallel-MacDec-MADDRQN) [Xiao et al. 2019].

Ultimately, these methods allow a team of agents to perform highly cooperative tasks under various domains via centralized *or* decentralized Q-nets. In particular, we have even shown that these methods are effective and applicable to the real world by deploying the trained networks on physical robots.

Learned Macro-Action-Based Decentralized and Centralized Policy

During execution in the decentralized case, agents collect macro-action-observation experiences into the buffer and each agent i has its own accumulated reward. During training, we combine Decentralized Hysteretic DRQN (Dec-HDDRQN) [Omidshafiei et al. 2017] with Double DQN to update each agent's individual macro-action-value function $Q_{\theta_i}(h_i, m_i)$. Updates use a concurrent minibatch of sequential experiences sampled from the buffer. We minimize the

loss of a double DQN style loss function that takes into account macro-action-observation history of each agent. In the individual macro-action-value function, h_i represents the macro-action-observation history of agent i and m_i represents the macro-action of agent i.

At each time-step in the centralized case (Cen-DDRQN), the buffer collects a joint macro-action-observation experience with a shared joint accumulated reward. The centralized macro-action-value function $Q_\phi(\vec{h}, \vec{m})$ is then optimized by minimizing a centralized loss function. The loss function incorporates a conditional target-value prediction method, $\vec{m}^{\text{undone}}$, that represents which agents have not yet finished their macro-actions. This allows the centralized Q-net to consider the execution status of agents' asynchronous macro-actions, and return more optimal macro-actions to each agent when they finish.

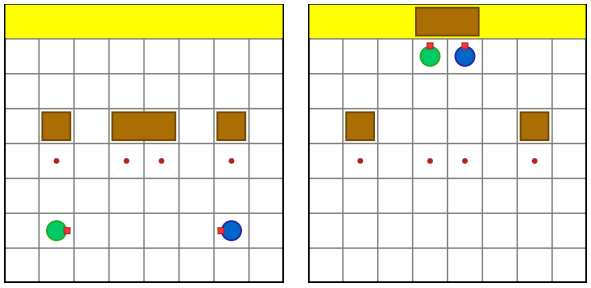
Macro-Action-Based Decentralized Multi-Agent Double Deep Recurrent Q-Net

This second method adopts "centralized training, decentralized execution" for deep Q-learning to learn the decentralized Q-nets, Q_{θ_i}, for each agent i using the centralized Q-net, Q_ϕ. During training, each agent iteratively samples a minibatch of sequential experiences to first optimize the centralized macro-action-value function Q_ϕ using Cen-DDRQN, introduced above. It then trains each decentralized macro-action-value function by performing gradient descent on the loss. Here, the decentralized target Q-value is calculated by using the centralized Q-net for the next macro-action selection of agent i while considering other teammates' behaviors and their current macro-action execution status. Parallel-MacDec-MADDRQN also applies the double Q-update rule for training purely based on decentralized experiences, while the centralized Q-net is trained only using the experiences generated by the centralized exploration in another environment.

Beyond evaluating these methods in simulation, I deployed the trained Q-nets in a tool delivery domain. The warehouse task involves a human requiring three tools to build an object. We showed that a team of robots, a Fetch robot and two Turtlebots, were able to autonomously pass the tools between themselves and deliver them in the correct order to a human who is building something.

Robust MARL - Moving Forward

A problem we commonly encountered in the robot experiments was that if a robot malfunctioned or died, the whole process fell apart. Agents could not plan around this and continue to finish the task at hand. As such, my next research direction is RL based approaches for when an agent "dies" (a robot fully stops working) in a multi-agent domain. This will hopefully enable an agent to still act in an intelligent fashion even when other agents fail. I have already implemented "agent death" in simulation over a certain likelihood under the box-pushing domain as shown in Fig. 1 and using primitive actions. This implementation is the most general case with no assumptions regarding what the other agents can observe about the other. In this scenario, agents are rarely able to learn any policy that has any meaningful results.



(a) Box Pushing Domain Setup (b) Big Box Push

Figure 1: Agents are tasked with pushing boxes, and an episode ends when any box reaches the yellow area. a) They can individually push a small box or cooperatively push the big box. b) The highest reward is received when pushing the big box to the yellow goal line.

I am investigating future directions like sharing the liveliness state of other agents in the observation space. Hopefully this additional observation will allow for different agents to act more optimally dependent on their observations. I am also investigating modeling belief of the status of the other agents like in POMDPs, and training a set of Q-nets for the different scenarios. When an agent believes the other is dead, it will act as if it is the only agent in the environment. Finally, I hope to eventually train n-agents (where $n > 2$) with the goal of having the set of agents learn more complex behaviors especially when an agent malfunctions. Ultimately, this work hopes to create robustness in multi-agent reinforcement learning to account for the stochasticity within robots acting in the physical world.

Explainability - Doctoral Studies

In moving past current research topics as an undergraduate and into doctoral studies, I hope to continue to study MARL along with human-centered AI. Even in the warehouse domain described above, the overall goal and application is to aid a human. Research must be done into human-AI teaming, and methods must be created to develop trust between humans and assistive agents. One promising research direction is into explainability and summarizing behavior as a mechanism of building trust between an agent and end-users. [Amir, Doshi-Velez, and Sarne 2018]. While nascent, this area of AI has the potential to further explode in it's usefulness to the real world. Moreover in connection to my current work, I would personally like to explore and research agent summarization in a MARL setting.

I look forward to pursuing a doctoral degree and penning a dissertation on multi-agent explainable artificial intelligence. Ultimately, I strive towards academia to teach and advise my own students in explainability, MARL, and even the future of AI that is yet to be discovered.

References

Amato, C.; Konidaris, G. D.; and Kaelbling, L. P. 2014. Planning with macro-actions in decentralized POMDPs. In Proceedings of the Conference on Autonomous Agents and Multiagent Systems.

Amir, O.; Doshi-Velez, F.; and Sarne, D. 2018. Agent strategy summarization. In the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018 blue sky track).

Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2018. Counterfactual multi-agent policy gradients. In AAAI 2018: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. Neural Information Processing Systems (NIPS).

Omidshafiei, S.; Pazis, J.; Amato, C.; How, J. P.; and Vian, J. 2017. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, 2681–2690.

Xiao, Y.; Hoffman, J.; Xia, T.; and Amato, C. 2019. Multi-robot deep reinforcement learning with macro-actions. In CoRR.

Xiao, Y.; Hoffman, J.; and Amato, C. 2019. Macro-action-based deep multi-agent reinforcement learning. In 3nd Annual Conference on Robot Learning (CoRL).

Various Perspectives of Studying Signals for Evaluating the Credibility of Online Sources of Information

Khonzodakhon Umarova
Department of Computer Science
Wellesley College
kumarova@wellesley.edu

Introduction

Misinformation, in its various forms and interpretations, has been part of societies for perhaps as long as actual information has: from gossip to myths to government issued propaganda. As technologies for mass communication evolved, so did the ways in which misinformation travels.

In the present day, people from all over the world query Google Search to access the information they need. Previous studies have shown that people automatically trust search results without realizing the work of algorithms behind the scenes in identifying and ranking "relevant" information (Pan et al. 2007). However, blind dependence on algorithms is problematic. Users who are disincentivized from critically evaluating information they see online are vulnerable to false, incomplete, or misleading results that slip through the net of algorithms. Therefore, it is integral for web users to develop web literacy skills that would empower them to evaluate web sources through so-called credibility signals.

CredLab (Credibility Lab) at Wellesley College, led by Computer Science Professor Eni Mustafaraj, is a research lab that studies web sources and credibility signals associated with them. As a member of CredLab, I have contributed on several branches of this multi-year project. The overarching goal of the project is to

1. Identify human-understandable signals that can be used to evaluate the credibility of a web source, and
2. Use AI tools and other computational methods to determine "values" of these signals for different web sources.

Misinformation lives under many appearances and flavors, and in my research at CredLab, I looked into three different types of misinformation:

- Science/health/medicine, or so called "pseudoscience";
- Problematic information targeting women and other minority groups;
- Information/disinformation related to political discourse.

In the following sections, I will explain each sub project in detail.

The Observatory of Pseudoscience

I was introduced to the CredLab through a Summer Research program sponsored by the Wellesley College Science Center in 2017. Motivated by news stories of problematic search results in featured snippets (Pick 2015), we investigated their prevalence for science and health claims, known as pseudoscience.

To do this systematically, I built a SERP Observatory – an automated infrastructure that allows to monitor Google SERPs (Search Engine results pages) for various queries, including important page elements (e.g. featured snippet). Using the observatory, I recorded and monitored over time results that appear on the first pages of Google Search for a predefined set of false scientific search phrases. The queries were compiled based on the claims that were debunked by the Snopes.com fact-checking website in Science and Medical topics. Then, two students and I assigned labels to sources that appeared among the top search results as either "reliable", "pseudoscience"(promoting alternative scientific claims), "dubious" (fake news and conspiracy theory sites), or "misc" (ex. platforms). The ultimate goal was to understand what categories of sites prevail in search results and how the rankings fluctuate over time.

The Observatory of Pseudoscience gave interesting insights into the kinds of websites that are ranked as most relevant by Google for false scientific claims. In particular, we saw that sometimes the top ranks include alternative medicine and pseudoscience websites that promote and confirm claims that are already debunked by fact-checkers.

At the same time, for the bigger goal of credibility research, the Observatory of Pseudoscience was a case study that served as the first interaction with the information ecosystem and some of its actors. Through the process of inspecting and labelling web sources from SERPs, we also got ideas for potential credibility signals (e.g., the presence and kind of advertisements hosted on the website). Lastly, the SERP Observatory turned out to be an extremely useful tool for data gathering and parts of its infrastructure continue being used by the CredLab members for other aspects of Credibility Signals research (Lurie and Mustafaraj 2019).

Exemplifying Gender Bias with Word Embeddings

Objectivity is considered an important criteria to consider when assessing the credibility of web sources of information (Metzger 2007). Objectivity in this context encompasses not only the extent to which presented information constitutes facts as opposed to opinions but also the understanding of intentions and agenda of the author/publisher. Hence, the source's position on certain topic(s) (or in other words, its bias), if known, might be a valuable credibility signal. However, identifying the bias of the source/publisher is not trivial. First, there are many different forms of bias: political, racial, age, gender. Further, bias is often implicit and can be disguised under the use of coded language.

Bolukbasi et al. (2016) in their *Debiasing Word Embeddings* paper displayed the extent to which word embedding, in particular the **word2vec** model, may amplify biases present in the corpus. Inspired by this research, we decided to investigate the possibility of exemplifying bias of the source using word embedding. If one trains a word2vec model on the text data from the source, can it serve as a tool to display various implicit and explicit biases of the source? In order to answer this question, the focus was narrowed down to gender bias. For the proof of concept model of such tool, I collected data from three exemplar sources: a feminist blog, a website from manosphere (i.e. a network of websites that focus on a new vision of masculinity[1], including movements like men's rights and Men Going Their Own Way), and Wikipedia's featured articles as control ("neutral") set. I trained word vector representations using Tomas Mikolov's word2vec with the continuous bag-of-words architecture for each of the sources (Mikolov et al. 2013). Experimenting with the three models showed that source's gender bias can be deduced from the words, whose vector representation is close to vectors of the keywords that are associated with gender (feminist, men, girl, etc.). Finally, in a poster at AAAI FLAIRS 2018 conference, I presented the proposed design of the tool as well as it usage scenarios. Even though only explored through the gender bias perspective, with appropriate indicator keywords, it can be extended to other forms of bias.

An important contribution of this research is that such tool would empower users to decide which aspect of bias to focus on. In addition, with the current surge in demand for transparent AI, the output produced by such tool (i.e. the words that are "close" to the bias-indicator keywords) can be used as an explanation for decision made about the bias signal.

Political Bias of News Sources

Another signal to evaluate credibility of a news source is its reputation: how it is perceived by others. There are many factors that establish reputation, but one of the first steps recommended when assessing the reputation of an unfamiliar source is to "google it" – a strategy also known as lateral reading (Caulfield 2017). User studies conducted by the CredLab indicate that the Knowledge Panel (KP)–the box on the right-hand side of SERP–play an important role in this assessment (Lurie and Mustafaraj 2018). In particular, references to political bias have been identified as particularly helpful to users (Rothshild, Lurie, and Mustafaraj 2019).

However, these reference are often extracted from the first paragraphs of corresponding Wikipedia pages. Observing SERPs over time (with the SERP Observatory I built) led to the discovery of frequent changes in the portrayal of news sources in KPs (knowledge panels). It turns out, these changes are often associated with repeated addition and removal of political labels (such as "alt-right", "far-left", etc.). In order to understand this phenomenon, we did an investigation of Wikipedia revisions. By obtaining all revisions for Wikipeda pages of $\tilde{1}300$ news source, I used Google's diff-match-patch library to study changes in the text. The results indicate that Wikipedia pages for sources that are perceived as strongly biased (both on the right and left sides of political spectrum) often experience intense "political labelling" edit wars (Umarova and Mustafaraj 2019).

The importance of this finding is that demand for "political bias" label as a credibility signal turned Wikipedia space into the arena for the new kind of edit wars. Hence, when studying this information ecosystem, one needs to be aware of different actors and consequences of various attempts to polish or tarnish a new source's reputation.

Broader Impact

Research that I have engaged in over the past three years at CredLab contributes to the overall long-term goal of building AI that works on behalf of the people. In order to achieve this, the AI should not only support information tasks, but also increase transparency and trust in the information ecosystem. In a way, this echoes Tim Berners-Lee's vision of Semantic Web and intelligent agents (Berners-Lee et al. 2001) that work together with humans. Findings about signals that we make as part of various branches of this project, useful techniques that we identify for obtaining "values" of important signals, and data collection/parsing tools that we build move us closer towards the goal of creating a system that augments search results pages with useful signal information. Presenting such information before a user would empower them to make informed decisions about the information they consume daily.

References

Berners-Lee, T.; Hendler, J.; Lassila, O.; et al. 2001. The semantic web. *Scientific american* 284(5):28–37.

Bolukbasi, T.; Chang, K.-W.; Zou, J. Y.; Saligrama, V.; and Kalai, A. T. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in neural information processing systems*, 4349–4357.

Caulfield, M. 2017. *Web literacy for student fact-checkers*. Michael Arthur Caulfield.

Lurie, E., and Mustafaraj, E. 2018. Investigating the effects of google's search engine result page in evaluating the credibility of online news sources. In *Proceedings of the 10th ACM Conference on Web Science*, 107–116. ACM.

[1] https://rationalwiki.org/wiki/Manosphere

Lurie, E., and Mustafaraj, E. 2019. Opening up the black box: Auditing google's top stories algorithm. In *Proceedings of the... International Florida Artificial Intelligence Research Society Conference*, volume 32.

Metzger, M. J. 2007. Making sense of credibility on the web: Models for evaluating online information and recommendations for future research. *Journal of the American Society for Information Science and Technology* 58(13):2078–2091.

Mikolov, T.; Chen, K.; Corrado, G.; Dean, J.; Sutskever, L.; and Zweig, G. 2013. word2vec. *URL https://code. google. com/p/word2vec*.

Pan, B.; Hembrooke, H.; Joachims, T.; Lorigo, L.; Gay, G.; and Granka, L. 2007. In google we trust: Users' decisions on rank, position, and relevance. *Journal of computer-mediated communication* 12(3):801–823.

Pick, R. 2015. Go ahead, ask google 'what happened to the dinosaurs'. *Vice. URL https://www.vice.com/en_us/article/pga4wg/go-ahead-ask-google-what-happened-to-the-dinosaurs*.

Rothshild, A.; Lurie, E.; and Mustafaraj, E. 2019. How the interplay of google and wikipedia affects perceptions of online news sources. In *Computation+ Journalism Symposium*.

Umarova, K., and Mustafaraj, E. 2019. How partisanship and perceived political bias affect wikipedia entries of news sources. In *Companion of The 2019 World Wide Web Conference, WWW 2019, San Francisco, CA, USA*, 1248–1253.