Final Exam, CIS700 – Monday @ 8:00pm

Kevin Martin

Syracuse University

Section 1

Q1 – Data Related

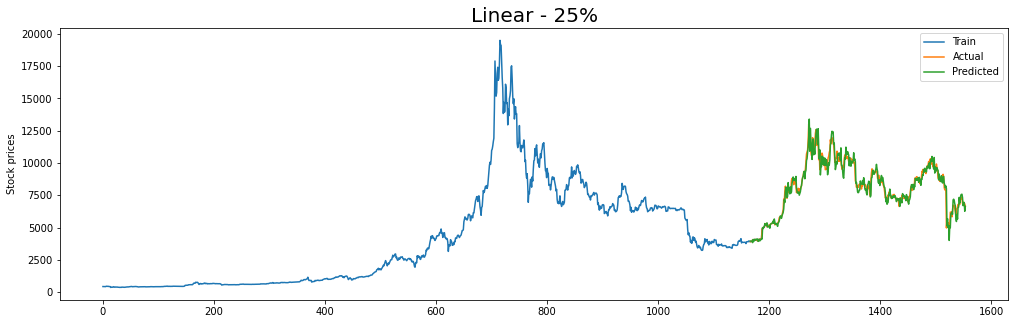
In order to best train my models, I’ll be using a several of methods and test them individually. Then, I will combine them in a couple of different ensemble methods using different techniques. I’ll be using bagging as a hold out approach, allowing my models to only see a certain percent of the available records to train on. Then, I will test on the remaining portions. For this dataset, the most obvious choice of a target value is that of price (or “price\_usd” as the official column name). In Question 2 I will look more at classification algorithms, but as we’ll see regression algorithms are much more suitable for this task. The price is a continuous variable, and the goal is to continually forecast it for a specified number of days. This is exactly the use case for a regression approach.

As far as training goes, I am going to use three different batches of train/test split on three different models. So at the end I’ll have nine differently trained approaches. I believe this will effectively demonstrate how different subsampling approaches can create better or worse models.

For the ensemble portion, I will combine all nine models in two different ways. First, I’ll use hard voting where I’ll take the mode of the price for every single day in the test dataset. Second, I’ll use a weighting approach. With less models, I believe tuning the weights is an excellent hyperparameter to investigate. However, for demonstrative purposes I just used equal weights.

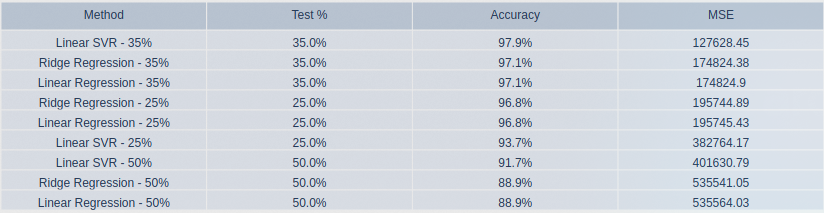
Regarding the data itself, I’ll be focusing specifically on Bitcoin. As I mentioned previously in my milestones, Bitcoin is the most popular and notable digital coin. It also has one of the highest number of records from available coins, which is helpful for training. In milestone 1 I removed unnecessary columns, missing records, and checked for outliers. So in the code I’ll be importing the CSV file, coverting it to a pandas data frame, creating a frame just for Bitcoin, and then remove those items. Thus the starting data frame will be nice and clean to work with.

My approach is to create a driver function where I can run different models through. It makes it easier and cleaner to read, but also guarantees the same train/test split, so the models will be apples to apples in comparisons. For the code file (submitted separately), I will comment out most of the lines that print graphs, but they should be able to run if desired. I’m going to introduce an example now, just to show what the rest look like if uncommented. All nine models (and the two ensemble methods) can generate the same graph format:



While I found visually inspecting the graphs interesting and almost pleasing, it was not enough to compare results. So for this exercise, I will be focusing on minimising mean squared error (MSE) and judging my models on that. I will also be looking at accuracy, but as it is defined and implemented through scikit-learn, I do not believe it is as relevant for a continuous dataset like this.

**Results**: I found it interesting that performance of the models was influenced more by my choice in training sets than the algorithm. To elaborate, the three models trained on the first 65% (and tested on 35%) performed better than any of the others. Next the three trained on 75%, and then finally, as expected, the models trained on only 50%. In general, Linear SVR was the best performer, follwed by Ridge Regression, and finally Linear Regression. This also makes sense in that the more sophisticated algorithms generally performed better. But again, the choice was *less important* than the train/test split:

*Note: figure can be recreated from code file if desired*

**Ensemble**: After running these nine approaches, I combined them in two ways. Hard voting was the better of the two options, which makes sense. With a large amount of members in the ensemble, the more that agree on a price, and more likely that price is to be correct. However, interestingly, the MSE was actually higher (worse) with this approach than most of the individual approaches. It falls in just behind the Linear Regression – 25%. I suspect this is due to some confusing days that the poorer models just couldn’t understand.

Next, the weighted approach. Equal weighting across nine models turned out to be **much** worse. The MSE was an order of magnitude off even the worst 50% algorithm. Here I believe the problem is just too many models and too much input. To perform a successful weighted ensemble approach, I believe less models and more specific weights would yield more optimal results.

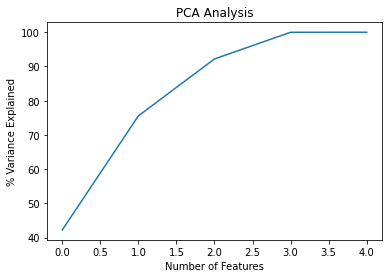
In summation, I believe this is a great showing, start to finish, of how different data-related choices can really impact model performance. Furthermore, I believe this was a successful showing of the ensemble approach. Even though the performance was not quite as good as the best performing individual models, it still demonstrates how multiple models can be incorporated to potentially optimize performance.

Q2 – Features Related

As discussed in Question 1, this dataset and the target variable of price are more suited to regression techniques. Also, as a quick aside, I focused my final milestone on regression techniques after understanding their fit for the situation. However, for feature selection, I found this to be a very fun exercise. The question is, can price be predicted from the features of a trade alone? The answer is no, at least not with any degree of accuracy that someone would fine useful. But by running some common feature reduction techniques and algorithms, we can still generate some sort of predictive power.

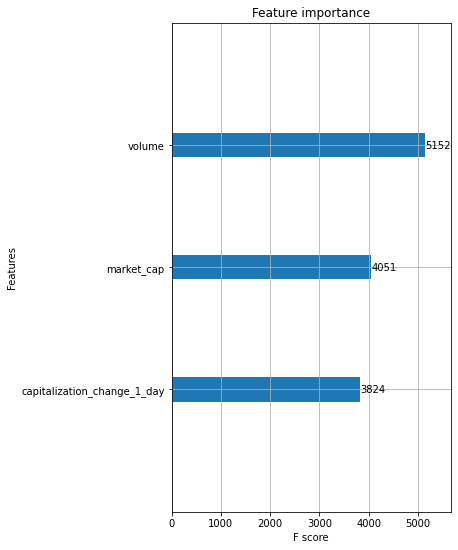
The biggest issue for me was the choice of features. When I first settled on this dataset, I saw that it had 17 columns to choose from. I thought that would be more than enough, but it turns out many were irrelevant, and a couple redundant. Things like URL of the coin itself, or coin/token flag proved to be unusable. Also there were several columns like change in price and change in Bitcoin price. So after chipping away at all these columns, I was really only left with five: 'volume', 'market\_cap', 'capitalization\_change\_1\_day', 'max\_supply', and 'minable'. That last one, minable, is simply a 1/0 binary flag if the coin can be mined by users. In the case of Bitcoin it doesn’t matter, it would only be helpful in trying to differentiate between coins, which was not the goal here.

But still, with five featuers, I could still run the checks and support I wanted, and was able to get the results I was looking for. First, I used a package known as XGB (eXtreme Gradient Boosting) to help with my PCA analysis. The package has a ‘dmatrix’ fnction which will train a model (gradient boosting specifically, tuned by the XGB package) and allow for the covariances to be run. From here, I was able to recreate the steps from Assignment 2 and determine how many feautres would give me 90% variance explanation. The result was three, which was actually encouraging. I was afraid I wouldn’t be able to get that high. When I graphed the results, one of my earlier statements was confirmed:

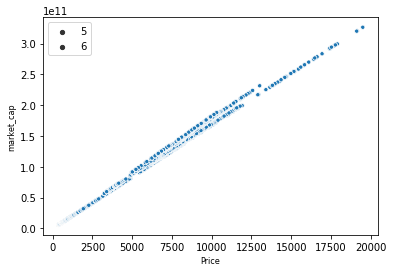
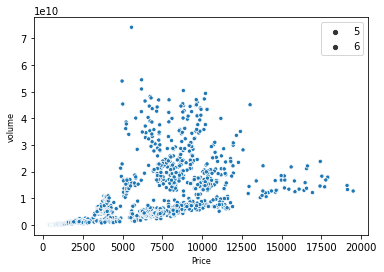


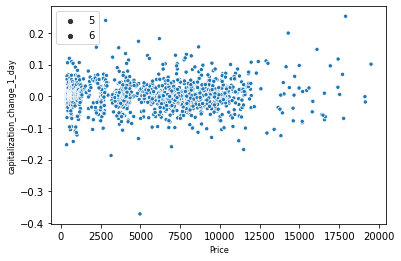
The fifth feature contributed statistically nothing to the variance explanation. That fifth feature was the ‘minable’ flag. While intuitively I thought it didn’t add anything, it was interesting to see it visually.

Next, I used another package in XGB to plot the relative importance of each. Of the five, only three made enough statistical impact to make it onto the chart:



While it appears that “volume” adds the most, I do not believe that is the correct way to interpret this graph. My understanding is that it is showing the contribution to the total variance explanation, and not the most statistically significant items. So by allowing volume to be first, it contributed the largest percentage of the three. To confirm this, I ran a simple scatterplot of each to evaluate:

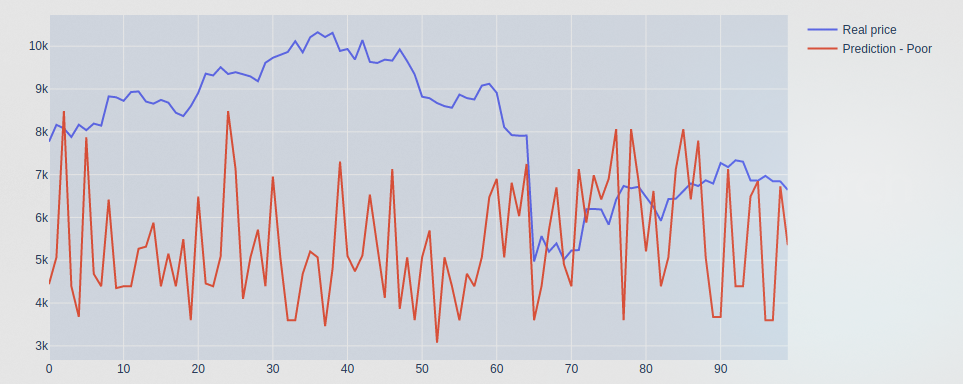




This makes more sense. The attribute ‘market\_cap’ is more statistically significant, and that should be true. Market Capitalization is the total volume of stock (or coins) outstanding multiplied by that day’s price. Price, the target variable, is sort of built in. So a change in market cap should indicate a change in price, and indeed it does. For completeness, I also ran the same three attributes through a partial dependence mdoel and graphed the results, but they mostly mirrored what was seen above (but can be viewed in the code if uncommented).

Next, the models. To really test the impact of feature selection, I ran three different ones. First, a gradient boosting regressor with just the optimal three features found above. Next, one with all the features that could be considered relevant. Even though there were a couple of duplicate/redundant columns, I was hoping to see a difference in performance. Finally, for demonstration purposes, a “bad” model with three very unhelpful attributes.

**Results:** I used MSE again to judge (and also looked at mean average error, MAE as a backup), and the results were pretty much as expected. Model 1 slightly underperformed model 2, and I believe this was due to a couple of those extra fields just tipping the prediction a bit better. Model 3 was terrible and almost incomparable to the other two. But that was exactly the point of desinging it that way, and the attribute selection process does work. If desired, the graphs for all three can be run, and it is almost humorous in comparison to the results of Question 1. Look how bad Model 3 was:



**Ensemble**: To complete the exercise, I ran all three models through the same two ensemble approaches as before: hard voting and (equal) weighting. Because of the inclusion of Model 3, the performance was awful in both circumstances. Interestingly, in contrast to the findings of Question 1, here the weighted ensemble performed better than the hard voting. I believe this to be due to the wild nature of Model 3’s predictions, so by giving it only 33% of the pricing weight the results could only be skewed so much.

In summation, I feel that this was a very good demonstration of how feature selection can help improve a model. Also, with the ensemble approach, tuning it and understanding the constituants is incredibly important in order to achieve meaningful improvements.

Section 2: Research

Q1 – Adversarial Machine Learning at Scale

In general, machine learning models required for security applications are often supervised and need to be told what the right outcomes are as they make the connections required for predictions. Each record contains a set of attributes that get distilled into a single outcome: a measure of quality, a binary yes or no, a suggested location, etc. After reviewing hundreds and thousands of records, patterns begin to emerge and the outcome can start to be predicted.

However, it is possible to fool these models with the right type of malicious inputs. In short, by slowly injecting carefully designed records, the predictive prowess of the models can begin to be manipulated. In the worst case, and the direction this paper alludes to, this can have serious implications for critical security systems. So there is a strong impetus to understand and prevent this type of attack. The word “attack” is appropriate here as this type of activity could not stem from accidental or random data, but only from malicious intent. In order to fool the model, an attacker must have some knowledge about the setup and deployment. Additionally, the paper “Adversarial Machine Learning at Scale” (Kurakin et al., 2017) introduces the idea of how it is possible for attackers to create “black box attacks” that don’t require knowledge of the model’s hyperparameters. In either case, the types of adversarial effects discussed are not the result of data anomalies, but rather specific intent.

To combat this problem, researchers have begun to get ahead of these attacks by injecting some malicious data of their own. By carefully training models with some amount of corrupted data, it is argued that these models can become more resistant to future attacks. In concept, this makes sense: a sophisticated model should be able to learn and distinguish from attacks. In practice, this is not an easy feat to accomplish because striking the correct balance of how much and what kind of adversarial records to introduce is quite challenging. Different models require different types of adversarial training, and are naturally going to be met with different amounts of success.

The core approach used by Kurakin et al. is on image classification. By utilizing a clean data set and an adversarial sets, the accuracies can be compared and the results contrasted. The adversarial set will change depending on the model and type of attack being simulated, and this helps illuminate how the right tool for the right circumstance can create positive results. Jumping to the conclusion of the paper, we can see that the use of adversarial training can indeed imporve robustness against future attacks, though this is not a guarantee across the board. The paper covers two categories, with varied amounts of success: one-step methods and iterative methods By looking at not only the relative successes but also the reasons behind the successes, we can get a better picture of how this type of training provides value.

Beginning with the most successful training, one-step, we see strong results. Simply put, the one-step method focuses on maximizing the probability of the least likely class predicted by the network. This will help minimize the impact of adversarial perturbations by reducing the value of the loss function. Another related approach was to use a random class as the target class. Here the idea is the same, to reduce the value of the loss function, but it is implemented differently. The first approach proved to have better results, and the article continues to focus on this approach. By comparing this first approach to the clean dataset, the researchers found that adversarially trained models became more resilient to malicious examples, but did lose some overall accuracy on the clean data set. As another test, a “deeper” model (one with more hidden layers) was trained, and the results turned out even better. There was an overall gain in robustness towards corrupted data, but the loss on the clean examples was reduced. Thus a successful guard toward these types of attacks has been demonstrated.

The researchers tried to implement an additional techniques to see if the results could be improved upon. The iterative approach, which applies the team’s base model (fast gradient sign method), but on a much smaller step scale. In theory, the iterative approach should perform better than the base model. However, by injecting adversarial examples, no derived benefit to corrupted data could be observed. The team also found that this approach is more computationally expensive, so any gains that could have been observed may not have been worth the training efforts.

The two takeaways here are that it is possible to train a model on adversarial data to prepare it for malicious attacks, and the more sophisticated/deeper a model is, the more it will benefit from this approach. Given the nuanced nature of the task, this makes intuitive sense, but seeing the successful results are very encouraging. A second paper, Towards Deep Learning Models Resistant to Adversarial Attacks (Madry et al., 2019) expands upon this topic by conducting a similar study. Here the researchers focus more on deep neural networks, but also through the lens of image recognition.

Reassuringly, the researchers came to the same conclusion as the original paper, that neural networks can be made resistant to these types of attacks. Here the definitions of success and robustness are expanded upon which help enforce the conclusions. The papers discusses the concept of optimization whereby parameters are chosen to both increase classification accuracy and decrease (or minimize) risk. Striking a balance is a common paradigm in machine learning, with many scenarios involving concessions in one area to achieve improvement in another (consider the very basic idea of computational expense in which more computations often result in better results but to the detriment of train time and resource usage).

In defining optimization, the paper describes a “saddled” approach with a balance between an inner maximization problem and an outer minimization problem. The inner maximization problem is in regards to maximizing loss, which is the same focus as the first paper. The outer minimization problem is finding those parameters that reduces the adversarial loss given by the inner attack problem. A strong balance between the two creates a robust model. Interestingly, the article criticizes the first paper directly by stating how, while it does employ multiple methods to maximize loss, it *only* focuses on maximizing loss. By ignoring this second paradigm, the paper claims that this approach and ones like it do not allow for “adversarial perturbations” because the loss becomes so small for all perturbations. Thus an unbalanced model results and would be suitable in these academic situations and not in real world situations.

It should be noted that efforts of the first paper are not entirely dismissed. Quite the opposite, the work done on maximizing loss is the jumping off point for the research of this second paper. Only by examining the shortcomings of previous research can we continue to expand the field and develop better and safer models.

A third paper, Ensemble Adversarial Training: Attacks and Defenses (Tramer et al., 2020), makes a similar critique of the findings from the original paper. One notable fact about this third paper is that two of the authors from the original paper are also authors of this paper (Kurakin and Goodfellow). I appreciate this fact both because it adds to the validity of the criticism but also shows the value of continuous learning and improvement.

The criticism here is levied at the vulnerabilities which still exist even after a model has been successfully trained using the best performing approach, the single-step (or one-step from the first paper). The researchers here suggest a black-box or white-box attack would be sufficient to exploit these problems. Again the goal is to champion the notion of robustness and not focus on too narrow an attack type. A black-box attack is one where perturbations are computed on one model (source) and transferred to others (targets). A white-box attack is one where the source and target are the same. The paper find that adversarial training (similar to the first paper) increases robustness to white-box attacks, but creates more errors for a black-box attack. Thus the models create a false robustness that is only apparent in an isolated setting.

To encourage true robustness, the idea of Ensemble Adversarial Training was introduced. In this approach, a more classic machine learning concept, generalization, is emphasized. Intuitively, this sounds like a good metric to focus on especially if the issue success in an over-specific setting. To elaborate, a model that only performs well in a certain instance may be subject to overfitting and needs to be more generalized to allow for flexible results. Here the same concept applies, it is just implemented differently. The ensemble portion is the use of static pre-trained models, where the source of adversarial examples is rotated between the currently trained model and one of the pre-trained ones. The source is selected at random which diversifies examples of malicious data for each epoch.

The authors conclude that this approach creates true robustness beyond what was previously accomplished by simple adversarial training. While both this and the second paper find issues with the original paper’s premise, the complaints are on entirely different grounds. One suggests the metric itself isn’t complete (balance) and the other states the training doesn’t go far enough. While both have their own merit, they continue to exemplify the theme of growth in the field.

Finally, I would like to introduce one final paper to help drive home the point about robustness, and specifically the black-box and white-box attacks. Boosting Adversarial Attacks with Momentum (Dong et al., 2018) levy similar criticisms to the training methods from the original paper, namely that of poor “transferability” (i.e., poor generalization). The solution here is similar to the third paper with a focus on an ensemble approach. Each of the models in their group are momentum-based iterative approaches, which they claim can fool not only white-box attacks but also black-box ones.

An interesting feature of this study is that these core models (that make up the final ensemble) stem from the original approaches in the first paper, namely one-step gradient methods as well as (the less effective) iterative approach. Again, we find an intuitive appeal whereby combining relatively successful models in a more sophisticated ensemble should increase not only performance but also transferability. Indeed, the conclusion of this paper illuminates the success of higher generalization being achieved through this approach.

After reading these papers and familiarizing myself with the topic, I have reached several conclusions. First, it is certainly possible to frog-boil a trained model and skew the results. I still believe that is is very difficult and requires a diligent attacker to truly understand neural networks. However that does not rule out the possibility, and the concept of a black-box attack certainly brings that possibility to life. Second, the core machine learning concepts continue to be useful even in advanced applications such as these. Balancing trade-offs, focusing on generalization, the right model for the right data, and the value of finely tuned ensemble approaches are all present throughout these papers. Having a good foundation in these concepts made this type of research easier to grasp. Last, the continued growth of this field is needed to stay one step ahead of malicious attackers. The first paper was written in 2017, and seeing the growth of approaches and increase in understanding of the field was quite eye opening. I look forward to seeing what the industry does going forward, and what the next iteration of security focused machine learning algorithms unfold.

https://arxiv.org/pdf/1611.01236.pdf

https://arxiv.org/pdf/1706.06083.pdf

https://arxiv.org/pdf/1705.07204.pdf

http://openaccess.thecvf.com/content\_cvpr\_2018/papers/Dong\_Boosting\_Adversarial\_Attacks\_CVPR\_2018\_paper.pdf

Q3 – The Security of Machine Learning

It is fair to say that a trained machine learning model represents a very sophisticated approach to solving a problem. To create an effective one, a programmer must have a solid understanding of the business case, or the real world problem that is trying to be solved. Then, the programmer must have enough skills to translate those thoughts into a program that fits the specific need. That program must be completely sound from a programming perspective, but it also must be accurate from a business perspective. Successfully implementing a machine learning algorithm is not an easy feat as the prerequisites are quite exhaustive. However, the results are unparalleled in their effectiveness. Arguably, no other technique exists that could deliver the same outcome, and certainly not for the same required computational power. As such, developing these models has become a highly desirable skillset with a wide range of applications.

One area of great importance is that of cybersecurity. Machine learning algorithms are perfect candidates because of how flexible and adaptable they are. All different types of cybersecurity can benefit from some sort of trained algorithm. Difficult situations are broken down and examined by a well designed model and enormous amounts of data can be synthesized and interpreted to generate predictive decisions. Unfortunately, some of these key areas of success can also become weaknesses. The paper The Security of Machine Learning (Barreno et al., 2010) highlights the balance between adaptability of a model and its vulnerabilities. To fully understand how these two dynamics interact, one must understand a few core concepts such as the definitions of success, adaptability, attacks, and vulnerability. Fortunately the article provides a thorough framework from which to examine these concepts and to ultimately determine how safe even an advanced machine learning algorithm is.

To simplify the experiment, the researchers focused on the most basic of machine learning examples: binary classification. Here the model only needs to make a yes/no categorical decision (i.e. does this example belong to a class or not). It should be noted that this not detract from any sophistication of the algorithm behind the decision, rather only the outcome becomes simplistic. This is also useful to define success. The paper looks at the measure false positives and false negatives. These are very common metrics for machine learning, but here they work well to define malicious activity. Here a the positive class indicates a corrupt instance while the negative class represents a normal or benign instance. A false positive will be generated if a normal instance is classified as positive (malicious when it should not be) and a false negative when an intrusion is classified as negative (benign when it should have been malicious). Through this lens we can examine successful implementations.

Next, the paper defines several layers and gradients of how an attack can be classified. We will take a quick look at these and then combine it with the above definitions to help describe the scenarios available before we can begin to measure them. In the first layer, there is both causative and exploratory attacks. Causative attacks are those that alter the training process which are analogous to the frog-boil situation. Conversely, exploratory attacks exploit an existing weakness. To me, both sound difficult to implement as the attacker would have to have some understanding of the model in question. I suspect exploratory attacks would most likely be the easier of the two to implement, but my speculations are irrelevant.

The second layer focus on the scoring system discussed previously. Attacks on integrity try to increase false negatives (a hostile record being marked as benign). The other type, attacks on availability seek to do the reverse (prevent a valid record from entering). Of the two types, attacks on integrity are certainly more dangerous, but that decision also depends on the situation. Some security systems are designed to minimize false positives, so attacks on availability could theoretically be more damaging to that specific system. It is important to keep that in mind when analyzing the results, because each use case is unique.

The last layer discussed relates to the inputs. Targeted attacks target a specific input, while indiscriminate attacks don’t care which inputs they go after. Once again, we see one type being (intuitively) more difficult to implement, the targeted attack. The attacker must have knowledge of the attributes in order to specifically attack one. One can imagine that a well designed indiscriminate attack could actually be more harmful. A successful attack that works without a *priori* knowledge of the system is more flexible and easier to implement.

In order to measure the success of a machine learning system is to attacks, the paper focuses on defenses against these different type of attacks. A successful system should be able to be defensive towards a variety of situations to help ensure the results are accurate. Before investigating further, the paper briefly mentions the resiliency of existing systems by examining other works published previously. This exposition is used primarily to as the jumping off point, a theme similar to the works in Question 1. By identifying past successes and failures, the researchers can point their ideas in the right direction. The short takeaway is that past systems have had varying degrees of success but, as always with machine learning, it depends on the situation at hand.

Regarding defenses, the paper does an excellent job of elaborating on both currently available and previously studied approaches as well as some postulation about potentially effective approaches. For each of the attacks presented previously, multiple solutions are at least presented. This gives a lot of ammunition to someone designing a particular system as several combinations of these defenses could be employed. One of the most important takeaways of this section does not actually have to do with the success of the approaches, but rather a unifying concept that we have seen presented time and time again within this discipline: the push-pull dynamic of trade-offs. The article states that by allowing an algorithm to focus more on worst-case attacks they will (generally) be less effective on average. So the designer must determine how finely to tune the algorithm and what sort of misses are allowable. Of course, this will again depend on each specific implementation.

The article concludes its analysis with a case study by leveraging the works of another paper which introduced an email spam filter referred to as the SpamBayes filter. In short, the model examines the body of an email and counts tokens which represent the likelihood that email is spam. Ultimately, this should result in a binary classification prediction: spam or not spam, but he model is allowed a third option of unsure. Using this model, the researchers implement several of the proposed defenses. The benefit of this approach is the consistency between results by using the same base model. Indeed, in comparing results we see that implementing these types of attacks cause the SpamBayes to fail, thus illustrating its weakness. However, the researchers point out that by understanding these attacks and why they are successful, more robust defenses can be developed.

Earlier I remarked that in order for a successful attack on a trained algorithm to be mounted, the attacker must have some knowledge of the algorithm itself. Given the speed at which technology moves and how information is shared, it should come as no surprise that this prior knowledge is no longer a necessity to nefariously manipulate a model. The paper Practical Black-Box Attacks against Machine Learning (Papernot et al., 2017) expands on this idea by explaining how a black-box attack works. In short, they are sophisticated attacks that do not require prior knowledge of the system they are trying to thwart.

This paper also looks at classification algorithms, so in that sense there is a bridge to the first paper. The difference is in the sophistication of the models being examined. Whereas in the first paper much simpler classifiers were being implemented (recall the SpamBayes token counter), here deep neural nets are the model of choice. I believe the comparison is still appropriate because the level of sophistication of these types of attacks is substantially elevated, so it is only fair the level of security system is also elevated. Also, the core concepts of fooling a sophisticated algorithm can also be applied to a simpler one, so the lessons are still valuable in multiple circumstances.

The authors do an excellent job in laying out the necessary background pieces needed to understand black-box attacks against these types of models. Deep neural nets in this context are simply multi-layered activation functions focused on calculating the weights between neurons, a system we have seen many times. Adding the multiple hidden layers generally helps reduces errors by giving more options to correctly determine the optimal weights. The end result is the same as the simpler model described previously, looking only for a binary prediction outcome.

Before introducing the main black-box adversarial strategies, a brief mention of less complex threat models are discussed. Similar attacks are discussed that we’ve actually seen before, namely targeted and adversarial, and the conclusions reached are also similar. The crux of this argument is that knowledge of the model is required. The authors summarize the concept perfectly: “Because evaluating these functions and their gradients requires knowledge of the DNN architecture and parameters, such an attack is not possible under our black-box scenario. It was shown that adversaries **with access** to an independently collected labeled training set from the same population distribution than the oracle **could** train a model with a different architecture and use it as a substitute” (emphasis added). Access is required, and model *could* theoretically be trained.

The black-box attack is shown to be successful at overcoming these restrictions. The approach is to enable an adversary to train a substitute model without a real, labeled dataset and instead use the target model to help construct a “synthetic dataset.” With these new records, the adversary observes the labels generated and constructs a substitute network where it can begin to do its own testing to discover the parameters of the target model. From here, the adversarial model can create malicious records that are specifically designed to fool the target model.

After expanding on these concepts, the article runs a test on a remote neural network to test the effectiveness of a black-box attack. The previous articles mentioned so far (including in Question 1) have focused more on the defensive capabilities, so to see the actual adversarial mechanisms put into action was quite interesting. I have brought up the idea of how, at least on intuition alone, a black-box model would have to be quite sophisticated to be successful. Looking at the experiments and results of these tests, my conjecture is confirmed. While it is certainly possible to create and fool a deep neural net, it is not easy.

Evolution continues in the field of course, and once these sorts of attacks were discovered the next step would be to design components to detect malicious inputs. The paper Adversarial Examples are not Easily Detected: Bypassing Ten Detection Methods cuts right to the chase and challenges ten previous examples of adversarial input detection. This is important because, as the paper itself puts it, having an accurate classifier is imply not enough for secure machine learning. Understanding attacks (both type and intent) are crucial in creating a secure system.

Unfortunately, none of the ten detection methods that were evaluated by this study proved to be sufficient at detecting a white-box attack. Recall that a white-box attack is less dangerous than a black-box attack, but still the widespread failure at this stage is troubling. An additional conclusion, and one that is perhaps more interesting, is that no clear intrinsic properties that would allow a model to differentiate between a regular example or an adversarial one. The implication is that the detection stage must be enhanced in its thoroughness to overcome this hurdle.

Finally, I would like to introduce one last concept to sort of shift the narrative. Thus far, we have examined the growth of adversarial attacks from basic to complex and the ensuing arms race that has begun in order to combat this. Success and failure at this point are determined by the best and most advanced model (be it a security model and its detection or an adversarial model and its attacks). However, there is another way in which chaos can enter the realm of even the best-designed security system: accidents. The paper Concrete Problems in AI Safety (Amodei et al., 2016) discusses how a poorly designed system can exhibit unintended behavior.

Again, we see the concept of robustness discussed, where it is not enough to just have a model that gives accurate results, but one must also be designed to withstand attacks (and accidents). This issue has been given rise due to the fact that implementing a machine learning algorithm has gotten substantially easier. Indeed, in 10 short weeks I have mastered the art and am ready for full-scale enterprise deployment[[1]](#footnote-2). The paper does not discredit the authors of these algorithms, simply the rise in number which leads to more observable anomalies. Because large scale or catastrophic accidents are difficult to define (and thus test) the paper focus on smaller, more pervasive issues.

These problems generally take the form of an unintended consequence that violates the spirit of the original design. Such is often the case with the literal nature of computers. An example presented is one of a learning algorithm that seeks to maximize its rewards, and in the case of a cleaning robot. If the robot is rewarding for cleaning up more messes, it begin to create messes so it will have more to clean up. This is know as “reward hacking” and is present in reward-based learning systems. The paper notes that as it stands, this type of behavior is relatively easy to correct by tuning the parameters of the reward. Although, as systems become more complicated and reward functions and agents have more time to interact, the situation could be come more difficult to address.

The point of this final paper was simply to complete the idea of how difficult it is to design a modern security system. Not only are there directed attacks, but now there are advanced black-box methods which circumvent the need for basic model knowledge. While detection methods are in place, they have shown to be (mostly) ineffective. And finally, the rise of accidental behavior can skew a model even in complete isolation, without malice intent from a third party. All of these components highlight the need for continued research in the field and advancement in the battle against adversarial attacks.

https://link.springer.com/content/pdf/10.1007/s10994-010-5188-5.pdf

https://dl.acm.org/doi/pdf/10.1145/3052973.3053009

https://dl.acm.org/doi/pdf/10.1145/3128572.3140444

<https://arxiv.org/pdf/1606.06565.pdf>

1. Perhaps not quite yet [↑](#footnote-ref-2)