Analysis of Machine Truth Serum

Doron Hazan*
Massachusetts Institute of Technology
Cambridge, MA
`doronh@mit.edu`

Abstract

Machine Truth Serum [8] was proposed to offer an alternative way of predicting the right answer in classification tasks in the domain of machine learning, where there are number of voters (classifiers) and the answer is unknown. It entails the idea of replacing majority voting, when the majority might be wrong, by incorporating the fundamentals of Bayesian Truth Serum [10]. In this paper, I will extend the ideas proposed in [8], implement its proposed algorithms, replicate its results, analyze its drawbacks, and propose improvements. The experiments and results I provide support the idea that Machine Truth Serum is indeed a promising alternative to majority voting. In addition, they shed light on potential improvements of Machine Truth Serum, which might lead to better classification results, and better machine learning performance overall.

1 Introduction

Wisdom of the crowd [14] harnesses the power of aggregated opinion of a diverse group rather than a few individuals. What captures the essence of democracy, it represents a concept that acts as a commonsense to us. Though initially proposed for mainly aggregating human judgements in the social domain, this idea has been successfully implemented in the context of machine learning. This idea is commonly known as ensemble models, which were proposed and studied to suggest better prediction performance [3]. As of today, ensemble models are prevalent in domains of supervised machine learning, both classification and regression, and are widely used both in research and industry. Some of the more popular ensemble methods include Boosting (such as XGBoost [2], AdaBoost [4]), Bootstrap aggregating (bagging), Stacking [1], and Random Forest [6]. The simplest and perhaps most popular way to perform aggregation is via majority voting rule. There are ways to perform even more sophisticated inference that aim to use smarter aggregation which seeks to outperform majority-voted answers [12], [17]. For example, a method that considers a 'soft' weighting scheme between voters, such that prediction of the class label is based on the argmax of the sums of the predicted probabilities instead of just the majority ('hard') call. However, these aforementioned aggregation methods rely on the assumption that the majority answer is more likely to the correct - they will fail in settings where special knowledge is needed to infer the truth, but it is owned by few individuals when they are not widely shared [11]. This includes more sophisticated inference techniqies, since their inferences will mostly likely initiate based on majority-voted answers.

The problem of aggregating human judgements is shown to be similar to the challenge of aggregating classifiers' predictions in machine learning. This is particularly important nowadays - the advent of deep learning [5] we can have very complex classification models. In ensemble settings, one very capable deep learning model, which performs the best among multiple models in the ensemble, would fail to predict the right answer since it is the expert minority, thus applying majority voting leads to wrong answers.

This is where the authors of Machine Truth Serum (MTS) come to play. They offered two algorithms that incorporate the notion of Bayesian Truth Serum [10] into machine learning classification tasks.

Their proposed algorithms, Heuristic Machine Truth Serum (HMTS) and Discriminative Machine Truth Serum (DMTS) demonstrated that better classification performance can be obtained compared to always trusting the majority voting. In addition, these methods outperformed popular ensemble algorithms, suggesting that they can be generically applied as a subroutine in ensemble methods to replace majority voting rule.

In reading and analyzing their paper, I highlighted various key points that to me seem significant. Their overall results seem promising, both computationally and intuitively. Since the authors did not provide a coding supplement, I wanted to re-implement their algorithms. My agenda in doing so was to (1) deepen my understanding of their assumptions and approaches, (2) replicate their results for future use, (3) provide a working framework of their logic, (4) identify drawbacks, and ultimately (5), hopefully to contribute and offer improvements and implement them.

1.1 Paper Organization

Section 2 outlines related work and provides a brief qualitative summary of the two algorithms proposed in [8]: Heuristic Machine Truth Serum (HMTS) and Discriminative Machine Truth Serum (DMTS). Section 3 provides a detailed analysis of these two algorithms, and their drawbacks, coupled with my alternative to each drawback mention in the previous section. In section 4 I present experimental results of the proposed changes. I conclude with a summary and a discussion of future work in Section 5.

2 Related Work - from BTS to MTS

Since incorporating ideas such as Bayesian Truth Serum from the social domain to machine learning were not proposed before the paper by Luo et al. (and I could not find evidence that there were any new ones after their paper), I will focus in this related work section on their approach, combined with the ideas coming from Bayesian Truth Serum.

In MTS, the authors wanted to capture the idea that "a surprisingly more popular answer is more likely the true answer" to classification problems. The core idea behind a surprising answer/hypothesis is that its posterior is higher than its prior, which makes an intuitive sense: an informative hypothesis will likely increase (in probability) after experiencing a change in the system (like seeing new data - coming up with a posterior). For example, if my hypothesis that it's going to rain tomorrow with probability 60%, and then I run an experiment, collect results, and update my belief to be 80% that it is going to rain tomorrow–there is probably some information about the fact that it is about to rain tomorrow. More formally, Theorem 2, taken from MTS and [11] helps elucidating this:

Theorem 2: For any x_i, the average estimate of the prior prediction for the correct classification answer will be underestimated if not every classifier provides the correct classification prediction.

This shows that the minority should be the final answer instead of the majority if the prior (estimated prediction) is less than the posterior. Their ideas are heavily based on the ideas taken from Bayesian Truth Serum, particularly when calculating the prior. Notably, the notion of *peer prediction information*, which constitutes the percentage of other agents (classifiers) one agent believes will agree with it. Formally,

$$p_i = \mathbb{E}_{\mathcal{B}_i} = (\frac{\sum_{j\neq i} \mathbb{1}(l_j = l_i)}{K-1}) \tag{1}$$

$$Prior(1) = \frac{\sum_{i=1}^{K} p_i^{\mathbb{1}(l_i=1)}(1-p_i)^{1-\mathbb{1}(l_i=1)}}{K} \tag{2}$$

$$Posterior(1) = \frac{\sum_i \mathbb{1}(l_i = 1)}{K} \tag{3}$$

These equations provide a framework to calculate the prior of a hypothesis among a crowd of voters: the average predicted percentage of the responding label as the approximation of the priors, such that the prior of hypothesis 1 is the average peer prediction information of the voters regarding hypothesis 1.

These are the building blocks the authors of MTS took from BTS to construct their algorithm that are going to be analyzed in the following section.

3 Analysis of HMTS & DMTS

The main challenge with applying ideas from the social domain to the machine learning domain is that unlike the social domain, in machine learning settings we do not deal with human beings, regardless of the fact that both domain tackle similar prediction problems. To tackle this, the authors of MTS designed HMTS, which was meant to capture the ideas presented in BTS. HMTS is based on calculating the prior and posterior of each hypothesis, and comparing them to determine the right answer, according to BTS. The authors designed an machine learning version of peer prediction information, presented in equation 4. To facilitate my descriptions of the algorithms' analysis, HMTS 1 is presented below.

$$\bar{y_i}^j = \left(\frac{\sum_{j \neq k} \mathbb{1}(f_j(x_i) = f_k(x_i))}{K-1}\right) \tag{4}$$

Algorithm 1: HMTS

```
Input:
$\mathcal{D} = \{(x_1, y_1), ...(x_N, y_N)\}$: training data
$\mathcal{T} = \{(x_1, y_1), ...(x_T, y_T)\}$: testing data
$\mathcal{F} = \{f_1, ...f_K\}$: classifiers
1  Train K classifiers ($\mathcal{F}$) on the training data
2  for $j = 1$ to $K$ do
3      for $i = 1$ to $N$ do
4          Compute $\bar{y_i}^j$ according to equation 4
5      end
6      Train machine belief $g_{j,0}, g_{j,1}$ on training dataset $\mathcal{D}_j^H := \{(x_i, \bar{y_i}^j\}_{i=1}^N$
7  end
8  for $t = 1$ to $T$ do
9      Compute $Prior(x_i, l = 1)$ and $Posterior(x_i, l = 1)$
10     if $Prior(x_i, l = 1) < Posterior(x_i, l = 1)$ then
11         Output "surprising" answer 1 as the final prediction
12     end
13     if $Prior(x_i, l = 1) > Posterior(x_i, l = 1)$ then
14         Output "surprising" answer 0 as the final prediction
15     end
16 end
```

As we can see here, the authors captured the idea of peer prediction information by two "belief regressors", $g_{j,0}, g_{j,1}$, which resembled human judgement and tried to estimate the agreement of other classifiers with the output of the classifier whom they are attached to. To my understanding, this is the only major difference this algorithm has from the principles presented in BTS. It's worth to note that the prior here is calculated by simply taking the average of the belief regressors, or machine peer prediction information.

Discriminative Machine Truth Serum, or DMTS, follows a simpler logic and thus is not presented here. In DMTS, the authors used again an ensemble of votes (classifiers) to make predictions. They created a new dataset that replaces the response variable with a binary outcome: whether the majority was right or not. Then, they trained an additional classifier whose job was to predict correctly whether we should trust the majority or not, per each data point. This design allowed DMTS to be much less computationally heavy than HMTS, but it is worth noting that DMTS does not really incorporate ideas of BTS, it simply performs a version of boosting: using prediction, and then correcting that prediction error using another prediction. This process can go recursively on and on, similarly to algorithms such as [9] and [2]).

Overall, these algorithms were able to demonstrate consistent improvements over majority voting, since I believe they:

- follow intuitive logic.
- are easy to implement.
- are not computationally heavy (especially DMTS).
- can operate over each data point separately without assuming homogeneous assumptions across a massive dataset (such as other popular algorithms like Random Forests).
- were able to demonstrate consistent improvement over majority voting both in binary and multi-class predictions.
- outperformed popular algorithms, such as Adaboost, Random Forests, and Weighted Majority.

3.1 Critical Review of MTS

Whilst these algorithms seem to perform greatly and propose a new way to tackle classification tasks using ensemble models, I noted few points that might be considered as areas of improvement. In this section, I provide key challenges and my way to approach them.

Challenge: although the notion of peer prediction information using belief regressors seems to be working, it poses some challenges, especially when **weak classifiers** are part of the ensemble. In those scenarios, which I experienced after re-implementing HMTS, weak classifiers predict just one output, thus the belief regressor for the other output will not have the chance nor the data to be fitted. This is particularly problematic when dealing with small, complex (i.e. following non-linear correlation) datasets using 'weaker' classifiers (such as perceptron [13] or logistic regression).

Proposed Solution: a potential way to overcome this is to include one belief regressor, instead of two, that will be trained on the entire newly peer-prediction dataset. This is contrary to the original approach, which has each belief regressor train on a seprarate peer-prediction dataset, based on if the output that was used to calculate peer-prediction was zero or one (i.e. $g_{j,0}$ will be trained on the data that was calculated using only outcomes of zero, and similarly for $g_{j,1}$). This proposed solution not only eliminates the threat of facing no data to train on (i.e., the challenge), but also performs similarly and sometimes even better than the two regressors, which is elucidated in the experiments section. This will be particularly useful in scenarios of very **imbalanced** dataset, in which one belief regressor will have plenty of data to train on, while the other will face scarcity. This might lead to a major performance difference between the estimations of these two belief regressors, potentially leading to biased estimates from the belief regressor which was trained on little data. This, consequentially, will perhaps lead to worse prediction results via HMTS.

Challenge: the authors give equal weights to each peer-prediction vote (prediction), when calculating the prior. While this is a conservative approach, it might be problematic in real-settings, especially nowadays, where some advanced, expert-like models can be part of the ensemble. Today there are models such as ResNet [15] or Transformer based [16] deep learning models, which perform exceptionally well in complex tasks and with lots of data. In settings such as an ensemble of classical machine learning models with one very advanced model - where the task is a complex classification prediction - treating each classifier equally might lead to the wrong answer. In such scenarios, there might be a very small expert minority that will hold the right answer, and while BST aims to expose that minority via the definition of "surprising answers", it might fail if that minority affects the calculation of the prior like the other voters do. In this settings, I believe they should be identified and perhaps have higher weights when making predictions.

Proposed Solution: Followed by the ideas presented in our last lecture, I propose detecting scenarios such as a small minority, which predicts the outcome with very high probability, while having very low peer-prediction estimation. In those scenarios, I believe we can identify this minority as an expert minority, and use its prediction as the right answer. Unfortunately, besides from my recollection from our last lecture, I do not have a profound evidence to back this idea up, but I choose this due to two reasons: first, while experimenting with the implemented code, I noticed that one classifier correctly predicted the right answer, while the vast majority predicted the wrong answer. Second, it is intuitively right: asking a quantum physicist a simple question regarding quantum physics will most likely result with the physicist correctly predicting the right answer with very high probability, while estimating that the rest of the world would know it with a very low probability. This resembles scenarios where the classification task requires specific domain knowledge; my analogy to these problems in machine learning are highly complex classification problems. While this is just a proposition, I believe this can constitute a set of future experiments.

Challenge: The authors of MTS experimented with "high disagreement" scenarios. For example, in binary settings with 15 classifiers, they noted that a high disagreement scenario will have 8-7 or 9-6 majority to minority ratio. I did not observe a thorough guide of finding that optimal split, thus I believe "high disagreement" is loosely defined. I experienced this while running the code implementation of HMTS and DMTS: there were scenarios where the vast majority was wrong, and an expert minority was right. These scenarios were not captures, because they were not labeled as "high disagreement". While I understand the intuition that BST will mostly work on "tight situations", where there is no vast majority agreeing on one prediction, I believe not having a formal disagreement threshold will cause the algorithm to not fulfill its predicting potential.

Proposed Solution: I propose formally introducing a **disagreement threshold**. I believe this threshold should be treated like a hyper-parameter, which will be unique to each dataset and its length, and the number of classifiers presented in the ensemble. One way of doing so is via grid search [7] and cross validation. The threshold will be determined by the best performance of the predictions.

Challenge: If HMTS and DMTS work separately, why not combining them? While this might not technically be considered as a challenge, I think that due to the nature of HMTS and DMTS, they work via self-improving, this specifically applies to DMTS. Boosting algorithms do not suffice with just one 'boost', but rather they go on until a stopping criterion is met.

Proposed Solution: I crafted two modifications to HMTS and DMTS that would potentially lead to improved results:

1. Perform majority voting between HMTS, DMTS, and the majority. In this approach, DMTS will predict whether the majority should be trusted or not (traditional DMTS), but this time it will utilize HMTS's prediction to finalize a decision. Thus, if HMTS agrees with the majority (while DMTS disagrees), the majority answer will be taken, and when DMTS agrees with HMTS that the majority was wrong, the other answer will be taken (binary settings).

2. Perform 'boosted DMTS': having a classifier that will predict whether DMTS was right, and an additional classifier that will base its prediction on whether the previous classifier was right, and so on. This will be done in a recursive manner and will stop via a stopping criteria (such as error threshold). The intuition behind this proposition is taken from boosting techniques: if one boost worked, why won't additional boosts work?

4 Experimental Results

In this part, I used my implementation of the authors' algorithms to replicate their results and try running some of my proposed ideas. I ran my experiments on five binary classification datasets, SpamBase Dataset, Breast Cancer Wisconsin (Diagnostic) Dataset, German Credit Dataset, Australian Credit Approval Dataset, and Hill-Valley Dataset. These datasets were taken from the UCI ML repository. Table 1 provides the statistics of those datasets. (Note, I could not locate the "Movie Review" dataset the authors mentioned, it did not have supporting documentation and there are plenty moview datasets online).

Dataset Statistics			
Dataset	# Data Points	# Attributes	% Majority
Spam	4601	57	60.6%
Breast Cancer	569	30	62.7%
German	1000	20	70%
Australian	690	14	56%
Hill-Valley	1212	100	50.7%

Table 1: Dataset Statistics

After obtaining these datasets, I conducted numerous experiments. I had two settings: (1), 15 classifiers which consisted of decision trees, naive bayes, support vector machines, logistic regression, MLP, Perceptron, and random forests. I made each different by tweaking their paramets (such as MLP with different threshold, svm with different kernels, random forests with different number of estimators, etc.). (2) a small sub-set of 5 unique classifiers from the set above. I experimented with adding random noise to the data (like the authors did), but decided not to include it in this report. The experiments below summarize the investigations I made to the MTS approach via my implementation of its algorithms:

Experiment #1: In this experiment, I ran HMTS, DMTS, majority voting on the datasets using the bigger subset of classifiers (15 classifiers). I included the performance of popular ensemble methods such as Random Forest and adaboost, each with 15 estimators. Table 2 summarizes my results.

Experiment #1 Results					
Dataset/Method	Random Forest	Adaboost	Majority	DMTS	HMTS
Spam	95.54%	92.50%	94.78%	**94.89**%	**94.89**%
Breast Cancer	94.73%	97.36%	98.24%	98.24%	97.36%
German	78%	72.5%	77.5%	**78**%	76.5%
Australian	86.21%	89.13%	86.23%	86.23%	**86.95**%
Hill-Valley	97.94%	90.94%	94.23%	**95.47**%	92.59%

Table 2: Results with 15 classifiers

These results are on par with the results presented in [8], I believe that the slight differences are due to noise adding, different selection of classifiers, different implementation, and slightly different datasets that were used. The results in bold highlight scenarios that either HMTS or DMTS outperformed majority voting. I did not observe a scenario in this experiment in which majority voting defeated both HMTS and DMTS, but did observe the opposite. This supports my intuition that HMTS and DMTS can be combined to perform even better.

Experiment #2: This experiment is identical to experiment #1, except that here I tried using one belief regressor instead of two. This follows the challenge I mention in the previous section, which is particularly stressed when few of the classifiers in the ensemble are very weak, or when the data is extremely imbalanced. Table 3 summarizes the results.

Experiment #2 Results		
Dataset/Method	HMTS (two regressors)	HMTS (one regressor)
Spam	94.89%	94.89%
Breast Cancer	97.36%	97.36%
German	76.5%	**77**%
Australian	86.95%	86.95%
Hill-Valley	92.59%	**93.41**%

Table 3: Results with 15 classifiers and one regressor

The bold entries correspond to scenarios in which HMTS performed better via using one regressor, as opposed to two. These results support my proposition: using one regressor can potentially improve results and avoid pitfalls which might happen when very weak classifiers are within the ensemble, those which might predict just one label and will disable one of the regressors. This can also benefit dealing with heavy imbalanced datasets: in those scenarios the peer prediction of one regressor might be better than the other, due to the difference in its training data.

Experiment #3 and #4: In those trials, I experimented with the disagreement threshold. I tried both relaxing and strengthening the threshold. Thus in experiment #3 I defined "high disagreement" as the majority having 8, 9, 10 votes (more relaxed, as opposed to 8, 9, in the original paper). In experiment #4, I defined "high disagreement" as the majority having only 8 votes (stricter). The results are summarized in tables 4.

The results shown in this experiment highlight the importance of finding the optimal disagreement threshold. In the table above, we observe different optimal results based on different high-disagreement thresholds. I believe that with with real case scenarios, where the number of classifiers

Experiment #3 & #4 Results						
Dataset/Method	HMTS (N)	DMTS (N)	HMTS (S)	DMTS (S)	HMTS (R)	DMTS (R)
Spam	94.89%	94.89%	94.89%	94.89%	**95.01**%	**95.54**
Breast Cancer	97.36%	98.24%	97.36%	98.24%	97.36%	98.24
German	76.5%	78%	**77**%	78%	76.5%	78
Australian	86.95%	86.23%	86.23%	86.23%	86.23%	86.23
Hill-Valley	92.59%	95.47%	**93.00**%	95.06%	92.59%	94.23

Table 4: Results with 15 classifiers and different high disagreement threshold. N: denotes normal disagreement threshold, S: denotes a strict disagreement threshold and R: denotes a relaxed disagreement threshold

might be much higher, it is important to treat the high-disagreement threshold as a hyper-parameter, and optimize the algorithm based on it.

5 Conclusion

In this document, I reviewed the ideas proposed in Machine Truth Serum by Luo et al.; I highlighted their contribution, critically assessed their method and assumptions, and offered potential improvements. In addition, I re-implemented their algorithms, which can be used now for further research. I designed and ran experiment to test my implementation and test my proposed ideas of improvements. To summarize, I believe Machine Truth Serum is a promising idea: it incorporates Bayesian Truth Serum, a strong idea from the social domain, to provide an alternative to majority voting. Besides showing improvement in performance, it also adds in more clarity and interpretability to the algorithms, which in my view is crucial in the domain of machine learning, especially nowadays with deep learning, where we treat models as black boxes. I think that incorporating ideas from the social domain such as Bayesian Truth Serum not only can improve the results of machine learning models, but also provide more intuition and clarity to the models, which can ultimately benefit the understanding of artificial intelligence (and humans, as well). For future work, I might (1) experiment with expert minorities: classifiers with high prediction probability and low peer prediction estimation, hoping this method of identifying and picking them will improve results. (2) Perform grid search with high disagreement threshold. (3) Design and experiment a combination of HMTS, DMTS, Majority as I proposed, and a boosted version of DMTS.

6 Acknowledgements

Thank you for a wonderful semester!

References

[1] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[2] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4, 2015.

[3] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[4] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[6] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[7] Petro Liashchynskyi and Pavlo Liashchynskyi. Grid search, random search, genetic algorithm: A big comparison for nas. *arXiv preprint arXiv:1912.06059*, 2019.

[8] Tianyi Luo and Yang Liu. Machine truth serum. *arXiv preprint arXiv:1909.13004*, 2019.

[9] Alexey Natekin and Alois Knoll. Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7:21, 2013.

[10] Drazen Prelec. A bayesian truth serum for subjective data. *science*, 306(5695):462–466, 2004.

[11] Dražen Prelec, H Sebastian Seung, and John McCoy. A solution to the single-question crowd wisdom problem. *Nature*, 541(7638):532–535, 2017.

[12] Vikas C Raykar, Shipeng Yu, Linda H Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *Journal of machine learning research*, 11(4), 2010.

[13] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[14] James Surowiecki. The wisdom of crowds. anchor; 2005. *Randall, Stephen: The Social Web and your Business's Five-year Survival Rate, article in Design Management Institute Review*, 21(1), 2010.

[15] Sasha Targ, Diogo Almeida, and Kevin Lyman. Resnet in resnet: Generalizing residual architectures. *arXiv preprint arXiv:1603.08029*, 2016.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[17] Yuchen Zhang, Xi Chen, Dengyong Zhou, and Michael I Jordan. Spectral methods meet em: A provably optimal algorithm for crowdsourcing. *Advances in neural information processing systems*, 27, 2014.