Hidden Risks of Machine Learning Applied to Healthcare: Unintended Feedback Loops Between Models and Future Data Causing Model Degradation

George Alexandru Adam ALEX.ADAM@MAIL.UTORONTO.CA
University of Toronto, Vector Institute

Chun-Hao Kingsley Chang KINGSLEY@CS.TORONTO.EDU
University of Toronto, Vector Institute

Benjamin Haibe-Kains BENJAMIN.HAIBE.KAINS@UTORONTO.CA
University of Toronto, Vector Institute, University Health Network

Anna Goldenberg ANNA.GOLDENBERG@UTORONTO.CA
University of Toronto, Vector Institute, The Hospital for Sick Children

Abstract

There is much hope for the positive impact of machine learning on healthcare. In fact, several ML methods are already used in everyday clinical practice, but the effect of adopting imperfect predictions from an ML system on model performance over time is unknown. Clinicians changing their decisions based on an imperfect ML system changes the underlying probability distribution $P(Y)$ of future data, where Y is the outcome. This effect has not been carefully studied to date. In this work we tackle the problem of model predictions influencing future labels (which we refer to as the *feedback loop*) by considering several supervised learning scenarios, and show that unlike in the no-feedback-loop setting, if clinicians fully trust the model (100% adoption of the predicted label) the false positive rate (FPR) grows uncontrollably with the number of updates. We simulate the feedback loop problem on a real-world ICU data (MIMIC-IV v0.1) as the distribution shifts over time. Among our scenarios, we consider how the clinician's trust in the model over time impacts the magnitude of the FPR increase due to a feedback loop. Finally, we propose mitigating solutions to the observed model degradation using heuristics that discard potentially incorrectly labeled samples. We hope that our work draws attention to the existence of the feedback-loop problem resulting in both theoretical and practical advances for ML in healthcare.

1. Introduction

Machine learning has a great potential to revolutionise the delivery of healthcare via in-home diagnoses, reduction of workload for clinicians, and improved outcomes for patients through the use of personalized medicine (Wiens et al., 2019; Gulshan et al., 2016; Bzdok and Meyer-Lindenberg, 2018). However, when the clinicians follow the ML predictions and give the patients different care, the future data distribution is changed toward the belief of these ML predictions and thus reinforce the ML's belief that could be wrong. For example, Challen et al. (2019) indicates that very ill patients predicted by the model to have a high probability of dying may receive palliative care (e.g. given painkillers) instead

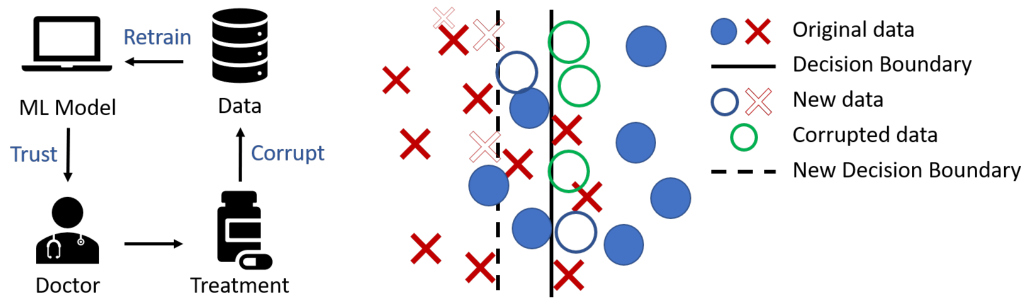


Figure 1: Predictive feedback loop. (Left) When we deploy a ML model in the clinics, if the model wrongly predicts such patient will die, and the doctor trusts the model, the patient would receive palliative care instead of treatment which subsequently leads to the mortality of the patient. When we update our model based on the new data, the model would reinforce its belief and result in higher false positive rates. (Right) In a binary classification setting, we train a model on the original data (solid circle and cross) and deploy into the clinics. Since the feedback loop makes the new label as positive (empty green circle) if predicted as positive by the original classifier, this causes the moving of the decision boundary to the left and thus mis-classify more negative data into positive and having higher false positive rate under the original distribution.

of alternative treatment, and this eventually leads to their mortality. Thus, the model's potentially incorrect predictions, in the case of falsely predicting the positive class (death), become the labels for future data. These incorrect labels are then used to update the model resulting in an increased false positive rate (FPR), and this repeats with every update hence why it is referred to as a feedback loop. We emphasize that in the curative vs. palliative care setting, or in an ICU where resources are limited and patients have to be prioritized, only false positive predictions by the model are able to influence future labels. The reason is that a false negative prediction in either of these settings means that a patient who will die regardless of treatment will end up receiving treatment due to the incorrect prediction by the model, and will still die. Hence, we focus on false positive predictions in this work.

On one hand, we want to include newer data as it may be more reliable and represent a better understanding of the disease. On the other hand, it is dangerous to rely on new data that might have been affected by the model's predictions. We are left with contradictory advice: update the model, and its performance may degenerate; do not update the model, and it will never reach the best possible performance. The goal of our work is to empirically demonstrate the existence of and characterize feedback loops, and present potential solutions.
Generalizable Insights and Clinical Relevance:

- *Insight:* We show that the common metric AUC is misleading for comparing deployed models when feedback loops exist. We show that False Positivie Rate (FPR) is a more reliable metric to detect the feedback loop phenomenon. *Relevance:* More thorough evaluations of models using multiple metrics must be done to detect model degradation due to a feedback loop.

- *Insight:* We compare how different model updating schemes affect FPR increase. We find that online updates of the model is more vulnerable than the refitting with all data when a feedback loop is present. *Relevance:* Special attention must be paid when using online learning/updating for medical ML systems as online updates are most vulnerable to model degradation.
- *Insight:* We examine the effect of clinician's trust on the pervasiveness of the feedback loop. We show that being skeptical of the model's predictions can reduce the effect of the feedback loop by a large degree. *Relevance:* Human-computer interaction can play an important role in preventing feedback loops, so training programs instructing clinicians on how to interact with ML models, and making clinicians aware of model limitations are essential.
- *Insight:* We propose a simple detection method for feedback loops and demonstrate the effectiveness of various sample removal techniques in reducing FPR increase in the presence of a feedback loop when updating the model. *Relevance:* We propose several ways to detect and mitigate this important clinical problem.

2. Related Work

Since the deployment of ML models in healthcare settings is a fairly recent application, there has not been much work on addressing feedback loops. What makes this scenario unique is the interaction between the physician and model. Both make false positive decisions, so the intersection of their incorrect decisions is an important quantity in determining false positive rate increase as we will later demonstrate. A seemingly related problem to that of feedback loops is the label shift problem where $P(x|y)$ is assumed to be stable but $P(y)$ changes between training and deployment (Azizzadenesheli et al., 2019; Lipton et al., 2018; Storkey, 2013; Garg et al., 2020). A canonical healthcare example is the cyclical nature of the flu where if we consider $y = 1$ to indicate that a patient has the flu, and a single feature $x = 1$ to indicate the presence of a cough, then $P(y = 1|x = 1)$ increases during flu season, yet $P(x = 1|y = 1)$ does not necessarily change. However, the assumption that $P(x|y)$ remains stable is not true in our scenario because model predictions can change $P(x|y)$ via the feedback loop. Knowing that the model is responsible for generating corrupted labels gives rise to other assumptions which can help address the problem. These assumptions are explored in Section 5 where we provide heuristics to prevent feedback loops.

Label shift is similar to another phenomenon known as concept drift where changes in missing context can change a concept gradually over time. For example, predicting the sentiment of book reviews can depend heavily on genre, so if genre context is missing, and the genre of the reviews being classified by the model changes over time, performance will degrade. The application of a concept drift detection method in our setting would tell engineers when data is stable enough to safely update the model, and alert them of a potential feedback loop when model performance changes past a certain threshold. One possible method to detect such a change was introduced by (Harel et al.) where the difference in the risk of an estimator trained and tested on sequential data is compared to that of an estimator trained and evaluated on random train/test splits. By setting a threshold on this difference, a hypothesis test can be conducted to detect a concept drift. (Yu et al.,

2016) use a more sophisticated method which tracks four relevant classification performance metrics and tests if these change significantly over time due to concept drift by comparing to a monte-carlo simulated confusion matrix. An observed entry that exceeds the entries in the simulated matrix is considered a possible sign of concept drift. The feedback loop scenario limits the applicability of such methods since the concept drift here is induced by the model, and agrees with the model's predictions . Lastly, the class imbalance, large relative increases/decreases in positive class proportion from year to year, and covariate shift present in MIMIC-IV adds yet another layer of complexity to detecting when a feedback loop has occurred via existing concept drift methods.

While there is no thorough quantification of the effect that feedback loops can have on model performance over time, there is some discussion about their existence (Challen et al., 2019) where the issue is considered to be a medium term challenge for ML systems along with automation complacency. Automation complacency is when clinicians rely more on an ML system than they should due to the system having made correct predictions in that past. This is a matter of human psychology, and thus beyond the scope of our work, but the trust experiments we perform aim to simulate this issue.

Feedback loops have also been mentioned in an important position paper by Sculley et al. (2015) under the framework of technical debt which is used to analyze the cost of rapid development in software engineering. Specifically, technical debt in the context of software engineering is accumulated in the form of undocumented code, absence of unit tests, and unnecessary dependencies. Failure to pay this debt back results in bugs, unmaintainable code, and an overall decrease in product utility over time. Their work extends upon the traditional software engineering version of technical debt since machine learning systems have system-level rather than code-level debt. To make this distinction explicit, the architecture and weights of a neural network are not sufficient to predict the effects of using the network, particularly when its outputs are unknowingly fed into other systems. To address the fact that an operational ML model can affect future data it sees, the authors propose the use of bandit algorithms in an attempt to encourage exploration in decisions. We emphasize that stochastically making decisions can lead to clinicians not trusting ML models since predictions might be inconsistent for two similar patients, so randomization would not be accepted in the clinic even if it could mitigate the feedback loop problem.

Ensign et al. (2018) look at the effect of feedback loops in predictive policing for a model that uses only past crime rates to allocate resources to a given area in proportion to how many crimes are discovered/reported there. To our knowledge, this is the only work that actually numerically analyzes feedback loops rather than simply mentioning their existence. The authors use a Polya urn model to demonstrate that if crime rates in two areas are even slightly different, eventually all resources will be sent to the area with the higher crime rate if only discovered crimes are taken into account. This is a classic rich get richer phenomenon, but its solutions do not apply to our scenario since the Polya urn model cannot be applied directly to our scenario becuase we use per-patient features to make predictions, not just prior disease rates.

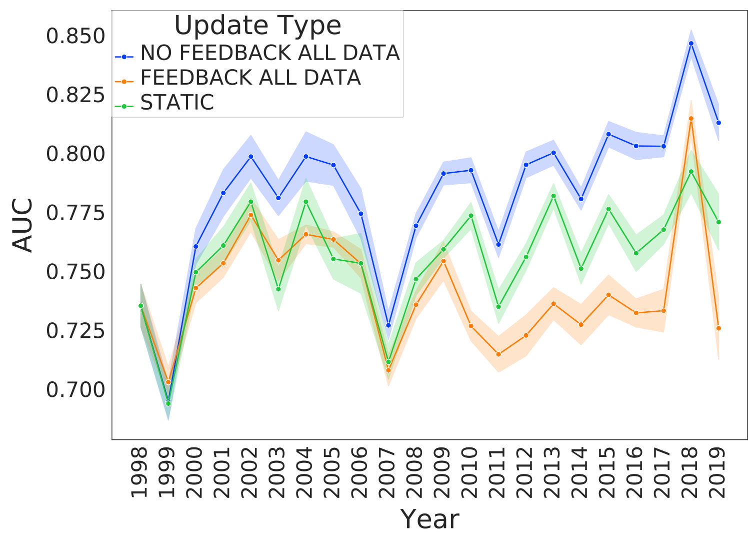


Figure 2: AUC of random forest models on MIMIC-IV data over time. The *static* model (green) is initially trained on years 1996 to 1997 and then never updated, while *feedback* (orange) and *no feedback* (blue) models are updated on data available up to the year before and evaluate on that year, under the corrupted and clean labels respectively.

3. Datasets

In this study we use MIMIC-IV v0.1 (Johnson et al., 2016) data. MIMIC-IV v0.1 is an ICU dataset gathered from the Beth Israel Deaconess Medical Center from 1996 to 2019. We chose MIMIC-IV to simulate the feedback loop effects across time, although the year of visit in the dataset is not completely accurate in order to preserve patient anonymity. We only take patients having data for more than 48 hours resulting in 31,269 samples and 47 features. The mortality rate is $\approx 10\%$. The details and the script for feature extraction can be found in Appendix A. For plots showing number of updates rather than year, we use a shuffled version of MIMIC-IV by shuffling and splitting all data into train, update, and test splits.

Note that we do not observe the same extreme AUC drop at the year 2008 shown by Nestor et al. (2019) in Figure 3 of their work since we remap the features across the EHR systems changing from Carevue to MataVision.

4. Experiments

To motivate the danger of feedback loops on real data, we visualize the AUC of random forest models on MIMIC-IV over time in Figure 2. We show that a *static* model trained on data up to 1997 and then never updated (green line) performs much worse than a *no feedback* model (blue line) which is constantly updated over the years. However, when the feedback loop is present, the static model outperforms the updated model *feedback* (orange line). It presents a dilemma that having a model constantly updated might not be the best practice when the feedback loop might occur.

Algorithm 1: The simulation of updating the model with a feedback loop present

```
Input: Trained Model $M$, # batches $T$, Batch Size $B$, Specified FPR $r$, Trust Level $\gamma$
for $t = 1$ to $T$ do
  Sample update data $\{(x_i, y_i)\}_{i=1}^{B} = \mathcal{X}_{\text{update}}^{(t)}$
  $\hat{\mathcal{X}}_{\text{update}}^{(t)} = \{\}$
  for $i = 1$ to $B$ do
    Compute $\hat{y_i} = M(x_i)$
    Sample $z \sim \text{Bernoulli}(\gamma)$
    if $\hat{y_i} = 1$ and $y_i = 0$ and $z = 1$ then
      $\hat{\mathcal{X}}_{\text{update}}^{(t)} = \hat{\mathcal{X}}_{\text{update}}^{(t)} \cup \{(x_i, \hat{y_i})\}$
    else
      $\hat{\mathcal{X}}_{\text{update}}^{(t)} = \hat{\mathcal{X}}_{\text{update}}^{(t)} \cup \{(x_i, y_i)\}$
    end if
  end for
  $M$.update($\hat{\mathcal{X}}_{\text{update}}^{(t)}$)
  Update the threshold $\tau$ for model $M$ on $(\mathcal{X}_{\text{train}} \cup \hat{\mathcal{X}}_{\text{update}}^{(t)})$ to have the given FPR $r$
end for
```

4.1. Experimental Setup

The way we have formulated the feedback loop problem is that false positive (FP) predictions by a deployed model are able to switch labels that would have otherwise been negative to positive. To simulate this effect, we choose an initial threshold τ such that the resulting model has a fixed false positive rate (FPR) on a held out test set. We choose τ separately for each model such that our models all have a comparable initial FPR across random initializations and data splits. All experiments are run with 10 different random seeds to obtain error bars.

To simulate an online feedback loop problem, we consider data as being partitioned into the following (ordered) sets :

- $\mathcal{X}_{\text{train}}$ the original training set used to train the initial model
- $\mathcal{X}_{\text{update}}$ the new data that becomes available as the model is in use
- $\mathcal{X}_{\text{test}}$ a held out test set to evaluate the performance of original/updated models

Note that a given model can only change labels in $\mathcal{X}_{\text{update}}$ but not $\mathcal{X}_{\text{train}}$ or $\mathcal{X}_{\text{test}}$. More precisely, models are updated according to Algorithm 1.

Reselecting threshold τ in each model's update In all of our experiments, to follow clinical requirements that the model has a required FPR, we reselect the threshold τ after each update of our model based on joint data $\mathcal{X}_{\text{train}} \cup \mathcal{X}_{\text{update}}$, and then report AUC and FPR on the $\mathcal{X}_{\text{test}}$. We also explored fixing τ after training and never updating it after, though that amplified the effect of the feedback loop.

4.2. Meaningful Comparison of Deployed Models

The way we have formulated the feedback loop problem is that false positive predictions by a deployed model are able to switch labels that would have otherwise been negative to positive. Thus, we are interested in analyzing model FPR change with each batch of update data on a held out test set. We train a logistic regression model on the shuffled version of MIMIC-IV under a feedback loop with τ originally selected such that the initial FPR is 0.2. The data is split into 10% training, 70% update, and 20% test. As can be seen in Figure 3a, AUC remains stable while FPR increases.

This means that the decision boundary shifts closer to samples in the negative class over time like in figure 3b on Gaussian data, yet the ordering of probabilities (distances to decision boundary) remains roughly the same, so a threshold that maintains the original FPR still exists even though FPR for the updated τ increases. Using AUC to compare deployed models is therefore not an effective approach since deployed models are tied to a particular τ which is reselected after every update. Since τ is reselected for a particular FPR using both training and update data, the desired FPR on held-out data will be different because some of the update samples are mislabeled. Using just the training set instead to reselect τ assumes that update data is distributed identically to training data, and that the samples in the training data are not in any way biased which is also an unreasonable assumption in practice. We also explored fixing τ after training and never updating it after, though that amplified the effect of the feedback loop.

Furthermore, model calibration on Gaussian data gets worse after updating. Figures 4a show the original model's calibration and 4b shows the updated model, and we find that the updated model has is more poorly calibrated than the original model..

4.3. Effects of Update Type

In cases where model training requires many computational resources, or where data privacy is of particular concern, it may be impractical to update/refit models on a cumulative dataset. Instead, online learning must be employed in order to both save training time, and avoid storing sensitive data on computers/networks that were not designed with the best security principles in mind. We consider the 3 following update types:

- **Single Step of GD Using Most Recent Update Data Only (SSRD)**:

  We update the model weights w by $w^{(t)} = w^{(t-1)} - \alpha \frac{\partial \mathcal{L}(\mathcal{X}^{(t)}_{\text{update}})}{\partial w^{(t-1)}}$ where $\mathcal{X}^{(t)}_{\text{update}}$ is the update batch at time t. This is the most computationally efficient update type because it performs just a single gradient step on the current update data batch. Since it ignores the training data, it adapts quickly to new data, and is most effective when the data distribution shifts without the presence of a feedback loop.

- **Single Step of GD Using All Data Gathered So Far (SSAD)**:

  We update the model weights w by $w^{(t)} = w^{(t-1)} - \alpha \frac{\partial \mathcal{L}(\mathcal{X}^{(t)}_{\text{cumulative}})}{\partial w^{(t-1)}}$. We consider two versions of this, `SSAD-T` (with training data) where $\mathcal{X}^{(t)}_{\text{cumulative}} = \mathcal{X}_{\text{train}} \cup \mathcal{X}^{(1)}_{\text{update}} \cup \ldots \cup \mathcal{X}^{(t-1)}_{\text{update}}$, and `SSAD-NT` (without training data) where $\mathcal{X}^{(t)}_{\text{cumulative}} = \mathcal{X}^{(1)}_{\text{update}} \cup \ldots \cup \mathcal{X}^{(t-1)}_{\text{update}}$. This update is still considered to be online, but is less efficient since it is performed on

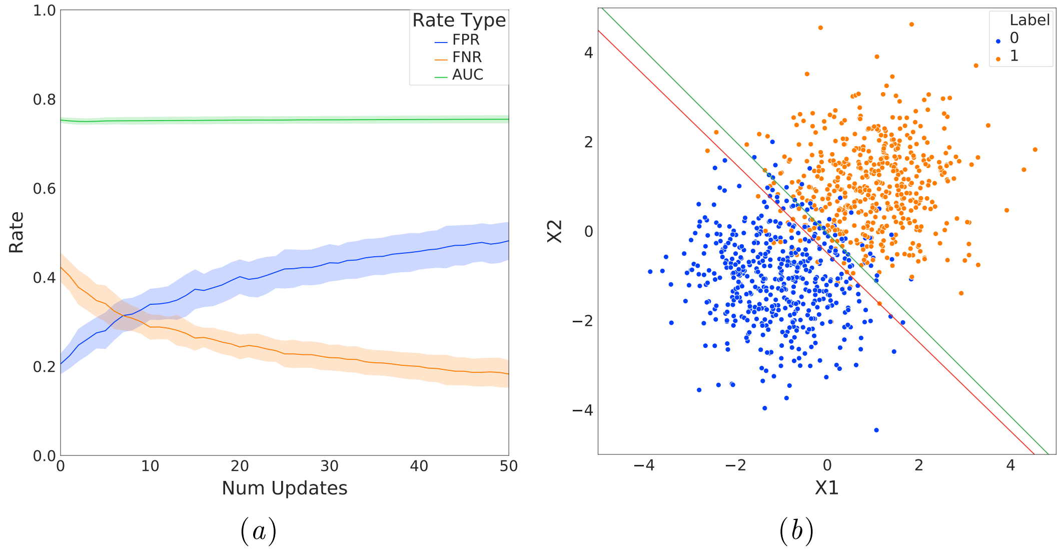


Figure 3: AUC is not indicative of feedback loop under linear model. (a) Logistic regression model updated with feedback loop on MIMIC-IV shuffled data. AUC remains stable while FPR keeps increasing. (b) The decision boundaries (DBs) of a 2-D Gaussian dataset before (Green) and after (Red) update under feedback loops. The DB shifts to the bottom left and increases the FPR. However, the AUC remains unchanged after update since the ranking of distance from each sample to the new DB remains the same.

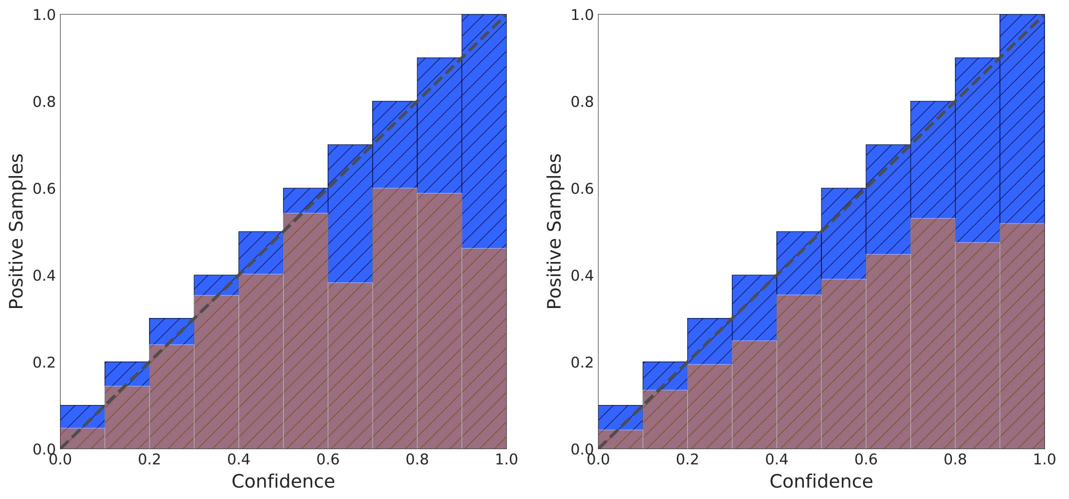
all data observed so far. The inclusion of training data and past update data can act as a regularizer that prevents FPR from increasing as quickly as with `SSRD` updates because there are many samples which are correctly labeled, and fitting those samples can prevent the model from fitting the mislabeled samples.

- **Complete Model Refit (Refit)**
  At update number t, let $w_{\text{temp}}^{(0)} \leftarrow w^{(t-1)}$. For $j \in 1 \dots e$

$$w_{\text{temp}}^{(j)} = w_{\text{temp}}^{(j-1)} - \alpha \frac{\partial \mathcal{L}(\mathcal{X}_{\text{cumulative}}^{(t)})}{\partial w_{\text{temp}}^{(j-1)}} \tag{1}$$

  and $w^{(t)} \leftarrow w_{\text{temp}}^{(e)}$. We consider two verisons of this. First, `Refit All Data` where $\mathcal{X}_{\text{cumulative}}^{(t)} = \mathcal{X}_{\text{train}} \cup \mathcal{X}_{\text{update}}^{(1)} \cup \dots \cup \mathcal{X}_{\text{update}}^{(t-1)}$ and `Refit Past Year` where $\mathcal{X}_{\text{cumulative}}^{(t)} = \mathcal{X}_{\text{update}}^{(t-1)}$. While this update can be impractical for large datasets, it is the most complete since the updated parameters are obtained via an entire optimization process starting at the previous parameters, not just a single step of gradient descent.

We compare all update types under the same learning rate, which means the difference lies in the number of optimization iterations per update, and the data used to reset τ after each update. In Figure 5, as expected, `SSRD` has the worst FPR increase since it



(*a*) Calibration of a logistic regression on MIMIC-IV before updating.

(*b*) Updated logistic regression model on MIMIC-IV has worse calibration.

Figure 4: Calibration plot before and after model updates under feedback loop. The blue is the perfect calibration and the brown is the measured result.

does not utilize any past data when updating. `SSAD-NT` has a sharper increase than `Refit All Data` initially, but then approaches the `Refit All Data` curve as more update data is accumulated. `SSAD-T` and `Refit All Data` have lowest FPR increase due to the use of all available data when fitting the model. Contrary to the advice given by Nestor et al. (2019) where the best update type was `Refit Past Year` since that adapted the most quickly after the EHR system change in 2008 at the Beth Israel Deaconess Medical Center, `Refit Past Year` is one of the worst update types when a feedback loop is present. Thus, we emphasize that best practices in terms of updating/refitting models on clean data may actually exacerbate the FPR increase due to feedback loops.

4.4. Clinician Trust

Any ML model implemented in a healthcare setting will likely not be running with complete autonomy. Rather, it will be supporting a clinician to increase the clinician's productivity. We consider two scenarios of varying levels of clinician trust to simulate how a clinician might adapt to working with an ML model:

- Constant trust: The clinician uses the model's prediction rather than their own judgement with some percentage γ which remains the same as the model is updated.
- Conditional trust: The clinician's trust in the model at time t depends on the model's FPR at that time $\gamma^{(t)} = 1 - FPR^{(t-1)}$. This represents the clinician becoming increasingly aware of the model's faults and relying more on their own judgements when necessary.

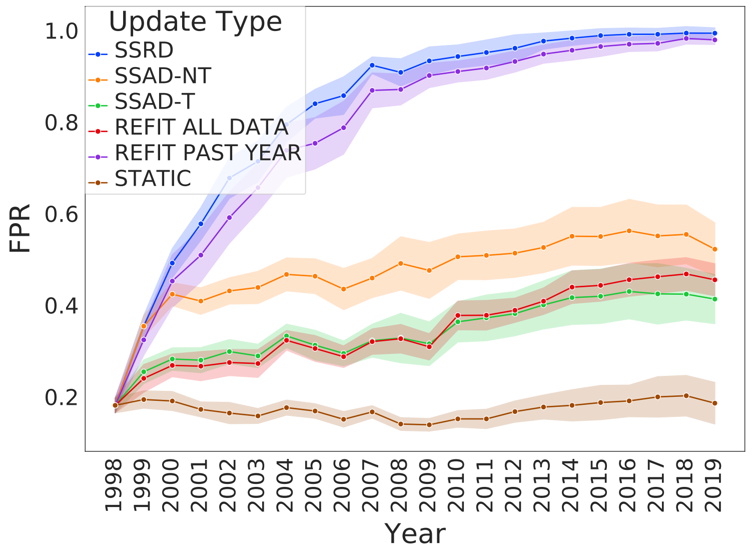
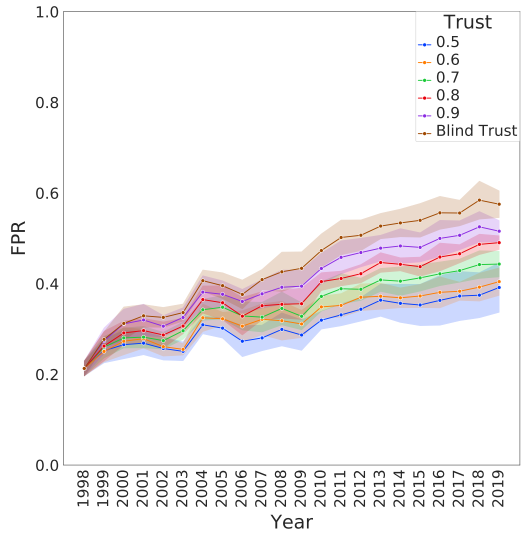


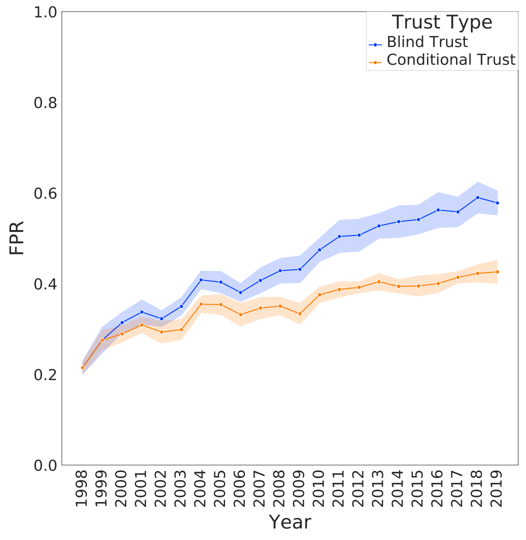
Figure 5: Comparison of update types for logistic regression models. `SSRD` is a single step of gradient descent on the most recent update batch only. `SSAD-NT` is a single step of gradient descent on all cumulative data without training data, and `SSAD-T` includes training data. `Refit` performs full gradient descent starting at the previously initialized parameters. The static model was only trained on data from 1996-1997 and never updated from then on.

We use constant trust to characterize how the interaction between clinician FPR and model FPR affects the strength of the feedback loop. If a plausible level of clinician trust exists that can prevent model FPR from increasing as quickly as blind trust, then the clinician-model interaction itself can be regarded as a regularization that mitigates the feedback loop effect. Figures 6a and 6c show that this is indeed the case when model and clinician FPR are both initially 0.2. Similar curves can be seen in appendix section B for clinician FPR levels of 0.01, 0.05, and 0.1 which suggests that the model's FPR is much more important in determining feedback-loop induced FPR increase than the physician's FPR. This is a positive message since any delay of feedback-loop-induced FPR increase gives users more opportunity to see if the model is behaving reasonably, and reduces cumulative incorrect predictions.

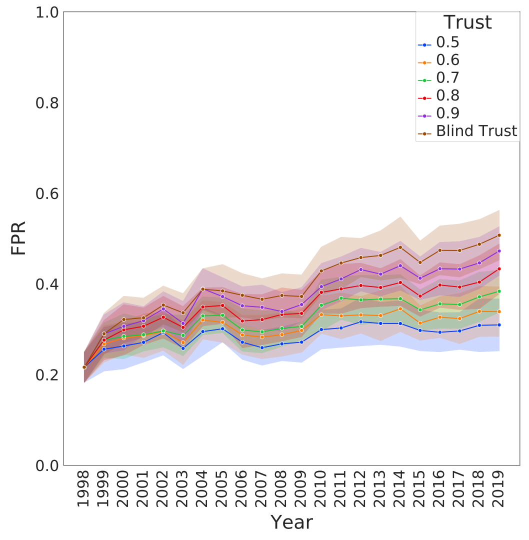
Conditional trust is a much more realistic representation of how human trust in black-box machine learning models varies over time as demonstrated by Yin et al. (2019). The authors show that initial reported model accuracy on a held out dataset has a significant effect on human trust, and that lower observed accuracy in practice reduces this trust such that the initial reported accuracy is no longer relevant. The way in which humans update their trust seems rather intuitive: by comparing the model's observed accuracy to their own accuracy, if the model is more accurate than they are, then their trust in the model increases. Since update data in our setting is corrupted by the model, it does not make sense to use the model's FPR on update data to inform the clinician about its performance. Thus, we compute model FPR on the uncorrupted training set in order to set the clinician's trust level dynamically. Figure 6b shows this is effective at reducing FPR increase for an XGBoost



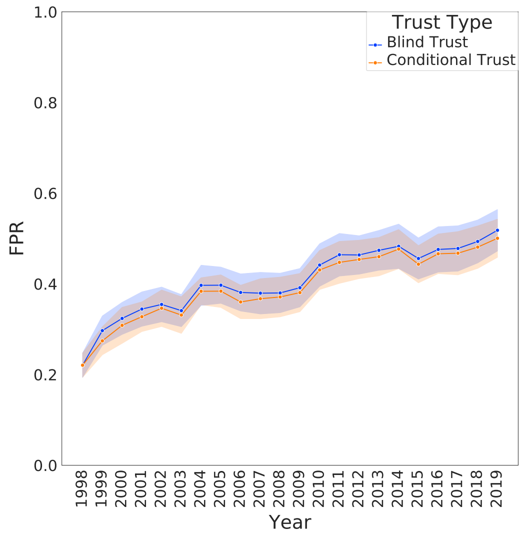
(a) Constant Trust XGBoost



(b) Conditional Trust XGBoost



(c) Constant Trust Random Forest



(d) Conditional Trust Random Forest

Figure 6: Effect of clinician trust on feedback loops. The prediction threshold for random forest models increases rapidly on new data, so naturally the false positive rate on the training set decreases due to this.

model, but this is not the case for a random forest as seen in Figure 6d. The discrepancy is explained by the fact that the decision threshold τ for the random forest model increases rapidly in order to maintain the same FPR of 0.2 on all cumulative data. Considering that the original τ_{train} selected for an FPR of 0.2 on the training set is smaller than the updated τ_{update}, the FPR on the training set for τ_{update} decreases, so trust increases. τ_{update} for the XGBoost model remains rather stable, so the FPR on the training set increases due to the degradation of the model.

5. Preventing Feedback Loops

We first propose a way of detecting if a feedback loop exists, then we experiment with several ways of preventing the feedback loops in the model's update.

5.1. Detecting if feedback loops exist

To detect if feedback loops exist, we can measure if there is a significant increase in the true positive rate (TPR) in the update data (i.e. proportion of positively labeled samples predicded as positive) before and after the ML model is deployed. Given the original TPR r_1 and the sample size n_1, and the new data with r_2 and n_2, we can calculate the p value by independent two-sample t-test to detect if the ratio significantly changes.

$$t = (r_1 - r_2) / \left(s_p \sqrt{\left(\frac{1}{n_1} + \frac{1}{n_2}\right)} \right) \quad \text{where} \quad s_p = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}}$$

s_1 and s_2 is the standard deviation of the r_1 and r_2 and s_p is the pooled standard deviation. Even detection of a feedback loop is valuable since users can be made aware of the model's flaws, and not rely on its predictions or stop using it altogether. Once the use of the model is suspended, engineers can determine how to change the interaction between the model and clinicians to prevent a feedback loop.

5.2. Mitigating feedback loops in the model's update

If we detect feedback loops exist, we propose 3 methods to reduce the effect.

Dropping All Positively Predicted Samples Dropping all predicted labeled samples when predicted as positive is the most extreme procedure that can be applied to ensure that false positive predictions do not compound over time via an unknown feedback loop. It also increases class imbalance as more update data is available since the only positive samples are those from the training set, and update samples which are predicted to be false negatives by the model prior to update.

Randomly Dropping Samples Let $p_1^{(0)}$ be the fraction of positively labeled samples in the training data, and $p_1^{(1)}$ be the fraction of positively labeled samples in the first update data batch. Given a large enough sample size N, if we expect $p_1^{(1)} \in [p_1^{(0)} - \epsilon, p_1^{(0)} + \epsilon]$, yet it is observed that $\delta = p_1^{(1)} - p_1^{(0)} > \epsilon$, then $N_{\text{fp}} = \lfloor \delta \cdot |\mathcal{X}^{(1)}| \rfloor$ samples are likely to be mislabeled. A simple procedure to perform is randomly drop N_{fp} positively labeled samples. We do this by sampling N_{fp} indices without replacement from an urn containing the indices of all positively labeled samples. While this is far from optimal and does not leverage any information about the model itself, it serves as another baseline.

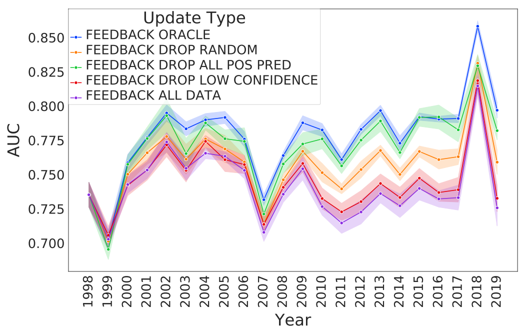
Dropping Low-Confidence Samples A more clever way of dealing with knowingly corrupted data is to leverage the predictions of the model itself. We make the following simplifying assumptions to motivate the heuristic to drop positively labeled samples:

- The model's initial predictions are well calibrated
- There is no underlying data distribution shift

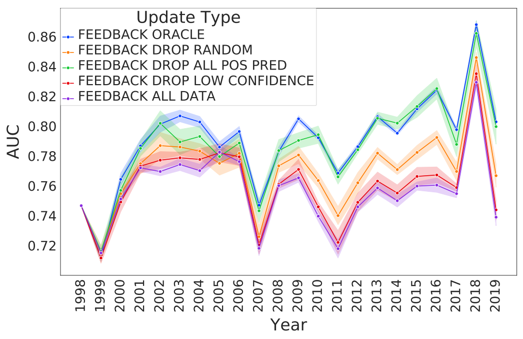
Given these assumptions, it is more likely that a sample observed to have a positive label is incorrectly labeled due to previous model predictions if the model's confidence is low on that sample than if the confidence is high. i.e. $f(x) - \tau < 1 - f(x)$ where it is assumed that the model's prediction confidence $f(x)$ on the sample is greater than the threshold τ, otherwise could not have been falsely labeled by the model. To this end, we propose the following procedure:

- Let $\mathcal{X}_{\text{update}}^{(pos)} = \{x \in \mathcal{X}_{\text{update}} : f(x) > \tau\}$ be an arbitrarily ordered list of samples in the update data that are already predicted to be positive by the model. These are the only samples which are at risk of having been mislabeled due to the model's predictions.
- Sort samples $x \in \mathcal{X}_{\text{update}}^{(pos)}$ according to $f(x)$ in ascending order
- Compute $N_{\text{fp}} = \left\lceil \delta \cdot |\mathcal{X}_{\text{update}}^{(pos)}| \right\rceil$ as in `Randomly Dropping Samples`
- Drop N_{fp} lowest confidence samples from $\mathcal{X}_{\text{update}}^{(pos)}$ to obtain $\mathcal{X}_{\text{update}}^{(reg)}$
- Let $\mathcal{X}_{\text{update}}^{(neg)} = \{x \in \mathcal{X}_{\text{update}} : f(x) < \tau\}$
- Use $\mathcal{X}_{\text{update}}^{(neg)} \cup \mathcal{X}_{\text{update}}^{(reg)}$ to update the model

Results Figure 7 shows the effect of the sample dropping techniques previously mentioned for random forest and XGBoost models. The oracle (blue line) is the best any method can hope to achieve and is obtained by dropping all known false positively labeled samples. Surprisingly, dropping all positively predicted samples (green line) is the best approach that most closely mimics the performance of the oracle. Randomly dropping excess positive samples (orange) nearly as effective as dropping all positively labeled samples. This could be because the update data has more mislabeled positive samples than actual positive samples due to the high initial model FPR of 0.2 relative to the fraction of positive samples 0.1 in MIMIC-IV. However, the ranking between the curves remains the same even for a smaller FPR of 0.05 as shown in appendix C, so this does not explain the discrepancy. Lastly, dropping low confidence positively labeled samples performs eve worse than randomly dropping samples which indicates that the assumptions are being violated and/or that further restrictions need to be imposed.



(*a*) Regularization Random Forest



(*b*) Regularization XGBoost

Figure 7: Effect of regularization on feedback loops. Reducing the effect of feedback loops by removing samples that are suspected to be mislabeled.

6. Discussion

While the scenario we have explored in this work may seem grim, it is a worst-case analysis, so all is not lost. The purpose of this work is by no means to discourage the implementation of ML in healthcare, rather to raise awareness about what might possibly go wrong when models are able to have too much influence in settings where decisions affect future data. The user plays an important role in mitigating feedback loops by becoming less trusting of model predictions when it is clear that the model is making increasingly more incorrect predictions. Eventually, the user should pause use of the model, and reconsider if the data being provided to the model is missing context like in the work by Caruana et al. (2015) where a model predicting risk of death due to pneumonia counterintuitively learned that patients with asthma have lower risk, or if the chain of influence between model predictions and future labels can be broken. Additionally, the development of new regularization methods is crucial in mitigating feedback loops in systems that eventually have complete autonomy and little human oversight. Existing techniques such as elastic weight consolidation (EWC) Kirkpatrick et al. (2017) designed to overcome catastrohpic forgetting have the potential to slow feedback loops by preventing updates from significantly lowering performance on the original training data. However, EWC was designed for neural networks and cannot be applied directly to models such as random forests or SVMs. Another possible mitigation strategy for feedback loops is to give models the ability to defer samples to clinicians. Rejection learning as proposed by Cortes et al. (2016) is one such possibility, but ignores the interaction between the model and clinician, and fails to capture how the model might be better at classifying some subgroups better than the clinician and vice versa. Madras et al. (2018) introduced learning to defer to address this limitation by making the model aware of clinician decisions when predicting whether or not to pass on predicting a given sample. This framework was evaluated on important high-risk tasks such as predicting recidivism in convicted criminals, and was effective in learning which samples to defer to human experts. What limits the use of learning to defer is that we are specifically trying to identify FPs, whereas learning to defer just identifies samples that the model might get wrong, regardless of direction. A method to identify FPs would essentially be an oracle.

We hope that our work motivates machine learning engineers, clinicians, and researchers to be aware of the feedback loop problem, and design systems that are robust to it.

References

Kamyar Azizzadenesheli, Anqi Liu, Fanny Yang, and Animashree Anandkumar. REGULARIZED LEARNING FOR DOMAIN ADAPTATION UNDER LABEL SHIFTS. Technical report, 2019.

Danilo Bzdok and Andreas Meyer-Lindenberg. Machine Learning for Precision Psychiatry: Opportunities and Challenges, mar 2018. ISSN 24519030. URL https://www.sciencedirect.com/science/article/abs/pii/S2451902217302069.

Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1721–1730, 2015.

Robert Challen, Joshua Denny, Martin Pitt, and Luke Gompels. Challen R, et al. BMJ Qual Artificial intelligence, bias and clinical safety. *Saf*, 28:231–237, 2019. doi: 10.1136/bmjqs-2018-008370. URL http://qualitysafety.bmj.com/.

Corinna Cortes, Giulia Desalvo, and Mehryar Mohri. Learning with Rejection. Technical report, 2016.

Danielle Ensign, Sorelle A Friedler, Scott Neville, Carlos Scheidegger, Suresh Venkatasubramanian, and Christo Wilson. Runaway Feedback Loops in Predictive Policing *. Technical report, 2018. URL https://github.com/algofairness/.

Saurabh Garg, Yifan Wu, Sivaraman Balakrishnan, and Zachary C Lipton. A unified view of label shift estimation. *arXiv preprint arXiv:2003.07554*, 2020.

Varun Gulshan, Lily Peng, Marc Coram, Martin C. Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, Ramasamy Kim, Rajiv Raman, Philip C. Nelson, Jessica L. Mega, and Dale R. Webster. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *JAMA - Journal of the American Medical Association*, 316(22):2402–2410, dec 2016. ISSN 15383598. doi: 10.1001/jama.2016.17216. URL https://jamanetwork.com/journals/jama/fullarticle/2588763.

Maayan Harel, Koby Crammer, Ran El-Yaniv, and Shie Mannor. Concept Drift Detection Through Resampling Koby Crammer Ran El-Yaniv Shie Mannor. Technical report.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Zachary C Lipton, Yu-Xiang Wang, and Alexander J Smola. Detecting and Correcting for Label Shift with Black Box Predictors. Technical report, 2018.

David Madras, Toniann Pitassi, and Richard Zemel. Predict Responsibly: Improving Fairness and Accuracy by Learning to Defer. Technical report, 2018.

Bret Nestor, Matthew B A Mcdermott, Willie Boag, Gabriela Berner, Tristan Naumann, Michael C Hughes, Anna Goldenberg, and Marzyeh Ghassemi. Caveats to Deployable Model Performance in Common Clinical Machine Learning Tasks. Technical report, 2019. URL https://mimic.physionet.org/mimicdata/carevue/.

David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. In *Advances in neural information processing systems*, pages 2503–2511, 2015.

Amos J Storkey. When Training and Test Sets are Different: Characterising Learning Transfer 1 Overview. Technical report, 2013.

Jenna Wiens, Suchi Saria, Mark Sendak, Marzyeh Ghassemi, Vincent X Liu, Finale Doshi-Velez, Kenneth Jung, Katherine Heller, David Kale, Mohammed Saeed, et al. Do no harm: a roadmap for responsible machine learning for health care. *Nature medicine*, 25 (9):1337–1340, 2019.

Ming Yin, Jennifer Wortman Vaughan, and Hanna Wallach. Understanding the Effect of Accuracy on Trust in Machine Learning Models. 2019. doi: 10.1145/3290605.3300509. URL https://doi.org/10.1145/3290605.3300509.

Shujian Yu, Zubin Abraham, Heng Wang, Mohak Shah, Yantao Wei, and José C Príncipe. Concept Drift Detection and Adaptation with Hierarchical Hypothesis Testing. Technical report, 2016.

Appendix A. MIMIC-IV v0.1 preprocessing

We only take those patients who stay at least 48 hours, which results in 31,270 ICU stays. We extract 8 features from chart events tables: `Heart rate, Systolic Blood Pressure, Diastolic Blood Pressure, Mean Blood Pressure, Respiratory rate, Temperature, SpO2 and Glucose`. We filter out the outliers value by their respective resasonable ranges, normalize their units and map different itemids to the same attributes if they represent the same measurement. Due to this itemid mapping, we do not observe the same extreme AUC drop shown by Nestor et al. (2019) at the year 2008 where the EHR system was changed from Philips CareVue to MetaVision. We then take the maximum, mean and minimun value of the first 48 hours for each feature, resulting in total $8 \times 3 = 24$ features.

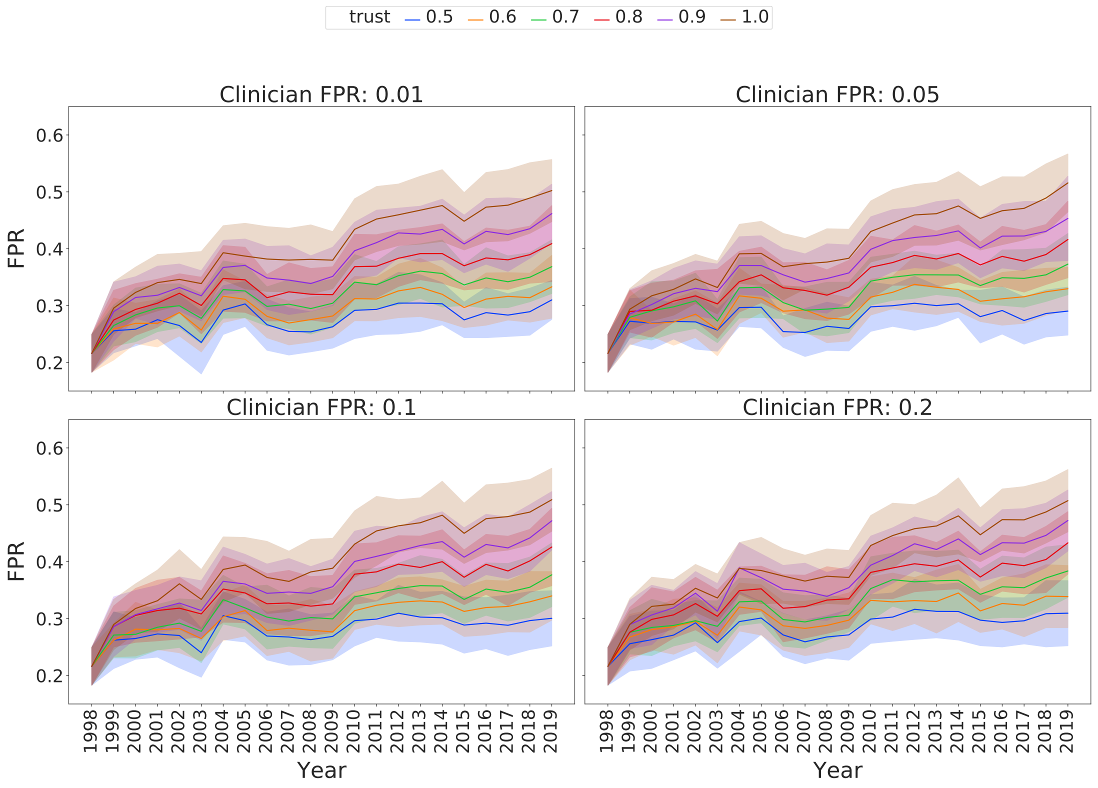


Figure 8: Comparison of varying levels of clinician FPR in the constant trust scenario.

We also extract 20 features from lab events tables: `ANION GAP`, `ALBUMIN`, `BICARBONATE`, `CREATININE`, `CHLORIDE`, `GLUCOSE`, `HEMATOCRIT`, `HEMOGLOBIN`, `LACTATE`, `MAGNESIUM`, `PHOSPHATE`, `PLATELET`, `POTASSIUM`, `PTT`, `INR`, `PT`, `SODIUM`, `BUN`, `WBC`. We take the values measured between -6 to 48 hours where 0 represents the time they get into the ICU, and take the maximum value if there exists multiple measurements within the period.

We impute the missing value with the populational mean of the same age and gender. We share our preprocessing scripts anonymous here [1].

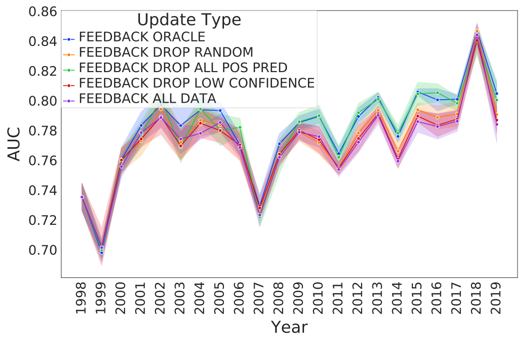
Appendix B. Constant Trust With Varying Clinician FPR

Here we show that when the model FPR is most important in determining FPR increase as the curves for varying levels of clinician FPR are very similar.

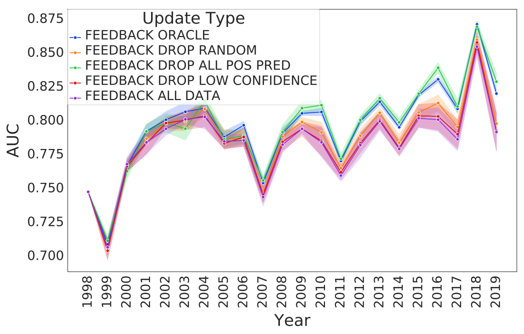
1. https://drive.google.com/file/d/1kOQA7Qw_W2gmRTOYdUkhyYFET3rRu3cB/view?usp=sharinghttps://drive.google.com/file/d/1Y4c6mat1X2DPTpjNMr6eAVIafHKUtXF-/view?usp=sharing

Appendix C. Sample Dropping Effectiveness Does not Depend on Model FPR

Here show that the ranking between the different sample dropping techniques does not depend on model FPR since for a smaller model FPR of 0.05, the ranking is the same as for an FPR of 0.2.



(a) Regularization Random Forest



(b) Regularization XGBoost

Figure 9: Reducing the effect of feedback loops by removing samples that are suspected to be mislabeled for models with 0.05 initial FPR.

Appendix D. Additional Rates

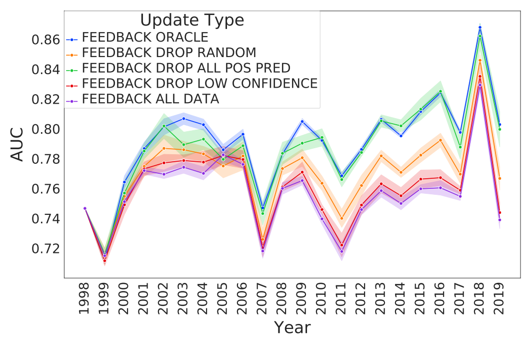
Here we show the effect of the discussed regularization methods on performance metrics other than AUC in order to understand potential tradeoffs. Dropping all positively predicted samples is still one of the most effective regularization methods, though we note that this is highly data-dependent because even without a feedback loop present, dropping positively predicted samples from update data improves AUC.

Appendix E. Detecting Feedback Loops

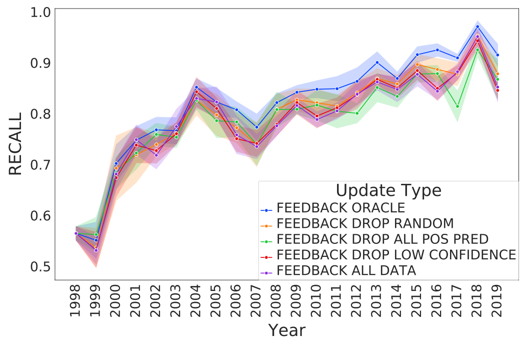
To investigate the effectiveness of our hypothesis test for feedback loop detection, we use various false positive rates, and a shuffled version of MIMIC-IV to control for the amount of training and update data. Table 1 shows how much update data was necessary for a given FPR to detect a feedback loop. The model used is XGBoost, and in this case we did not reset the threshold after initial training in order to also report how much FPR has increased by the time the feedback loop was detected. Note that the final FPR when using a larger number of training samples is very close to the initial FPR since the model is given many more correcly labeled samples in compared to false-positive labeled samples.

Appendix F. MIMIC-IV AUC Spikes

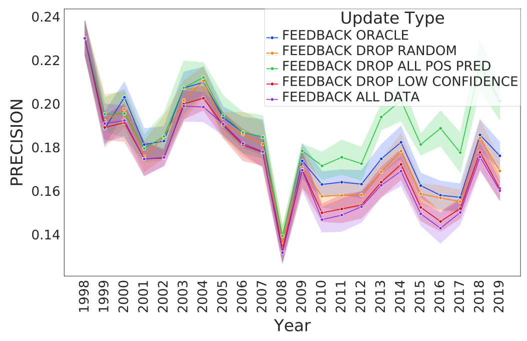
There are several noticeable spikes in the AUCs of our models, with a particularly large one being in 2018. To explain this, we investigated the data to check for obvious drifts in



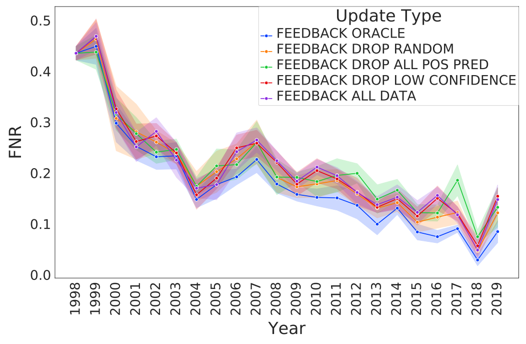
(a) AUC



(b) Recall



(c) Precision



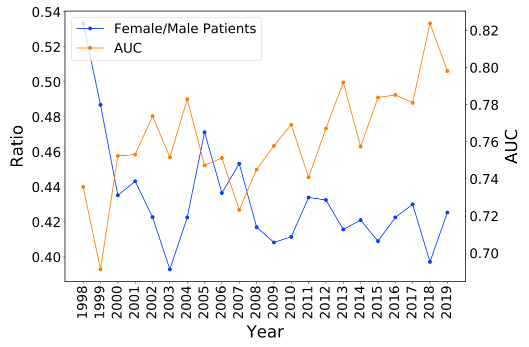
(d) FNR

Figure 10: Multiple performance metrics for XGBoost models with various regularization methods.

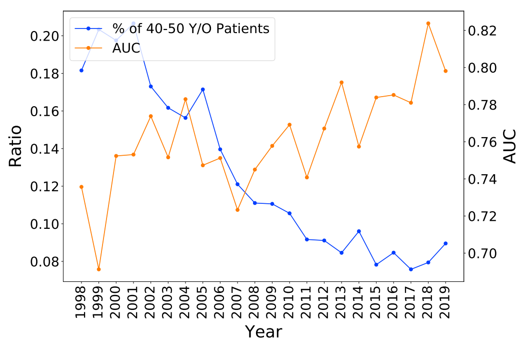
Table 1: How the amount of training data and initial FPR affects the number of update samples required to detect a feedback loop, and the model's FPR at that point.

Initial FPR	# Training Samples	# Update Samples	Final FPR
0.05	100	40	0.254
0.05	1000	440	0.051
0.1	100	40	0.265
0.1	1000	60	0.1
0.2	100	30	0.456
0.2	1000	20	0.192

demographic features over time. Figure 11a shows the ratio of female to male patients each year, with the AUC of a static (trained only on 1996-1997 data and never updated after) random forest model overlaid as well. There is a noticeable relationship between the two curves, and the Spearman correlation between the two is -0.61 which indicates a moderate relationship. We also investigated the relation between patient age and model AUC, and found an even stronger Spearman correlation of -0.65 between the percentage of patients in the age group 40-50 and AUC as seen in figure 11b.



(a) AUC and Gender



(b) AUC and Age

Figure 11: Effect of regularization on feedback loops. Reducing the effect of feedback loops by removing samples that are suspected to be mislabeled.

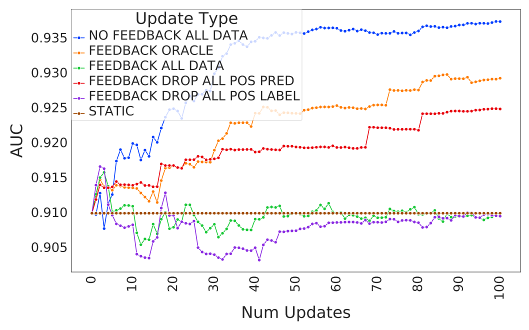
Appendix G. Experiments on Simulated Data

Our experiments on MIMIC-IV were originally motivated by investigations on simulated data. We show some of these experiments below on the following multivariate Gaussian data:

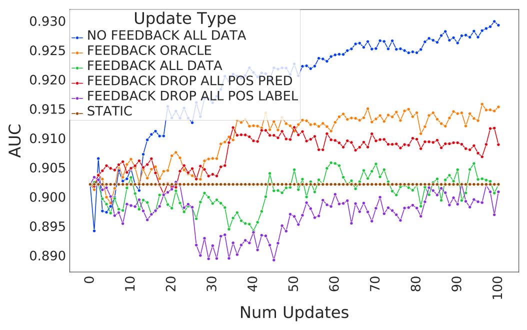
$$\begin{aligned} x &\sim \mathcal{N}(\mathbf{0}, \Sigma) \\ w &\sim \mathcal{N}(\mathbf{0}, 2I) \\ y &= x^\top w + \epsilon \end{aligned}$$

where Σ is a random covariance matrix with all non-zero entries to simulate correlated features, and $x, w \in R^{10}$. $\epsilon \sim \mathcal{N}(0, \sigma)$ where σ is set to the standard deviation of $x^\top w$ computed across all samples and divided by two.

We use 100 samples to train the model, 1000 samples to update the model, and evaluate the model on 5000 samples. This ratio train/update/test samples is uncommon in practice since training data is limited, but here we simply wanted to have a large enough amount of samples to reliably evaluate the model, and a high enough amount of update samples to simulate a model in production for a long period of time. As is observed in figures 12a and 12b, the feedback loop degrades model performance, but on this simple dataset, it eventually recovers. This recovery is due to how simple the data is: even as more corrupted labels are gathered and used to train the model, clean data is also gathered. A binary Gaussian classification tasks does not require many samples to learn a good solution, so once that threshold sample size is achieved, the optimization algorithms ignore the corrupted samples.



(*a*) LR



(*b*) XGBoost

Figure 12: Regularization effect on feedback loops for simulated data.