Towards Resolving Causal Confusion via Robust Imitation Learning

Rushit N. Shah
rshah231@uic.edu
University of Illinois at Chicago
Chicago, Illinois

ABSTRACT

Distributional shift in training and testing data may lead imitation learning models to learn false correlations from expert demonstrations, leading to a phenomenon known as 'causal confusion'. We show how the problem of causal confusion occurs in several imitation and reinforcement learning settings. To resolve causal confusion, we explore a technique which, through targeted interventions, attempts to learn the true underlying causal model that best explains the expert's actions. Building from this approach, we formulate a robust imitator which, given a set of causal and non-causal variables, learns a policy which attempts to mimic the expert, while still being robust to shifts in test distributions from training distributions.

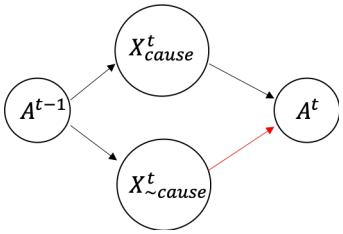

Figure 1: State variables may be causal X^t_{cause} or non-causal X^t_{cause}

1 INTRODUCTION

Humans have the remarkable ability to imitate complex behaviors just by merely observing the behavior of another person. Imitation learning enables robots to learn control policies directly from example demonstrations of an expert, precluding the need for the robot to interact extensively with its environment [Bojarski et al. 2016; Hussein et al. 2017; Pomerleau 1989]. But like any supervised, discriminative algorithm, the performance of imitation learning models suffer when distributional shift exists in the training and testing data [Daumé III 2009]. In statistical learning, the notion of a training and test distributions being different on the input space is known as distributional shift.

This distributional shift (also known as covariate shift) occurs since the robot's learned model is never perfect and it tends to visit states which are different from those visited by the expert—resulting in the observation of new, previously unseen states. Since robot's environment evolves over time as it executes its learned policy, the new unknown states may result in a causal misidentification and lead to compounding errors. A classic example is that of a neural network learning to drive a car by observing a human driver. The neural network might learn the correlation that brakes must be applied when a brake indicator light on the dashboard of the car turns on. Although the brake light is an effect of applying the brake and not the cause, learning this false correlation allows the neural network to achieve low training error. In this scenario, distributional shift occurs during test time with catastrophic outcomes since the neural network never applies the brakes because it does not see the brake light turning on; it has learnt to brake only after seeing the

brake light. This problem is referred to as causal confusion. In such cases, it is crucial to learn the underlying causal model from the expert demonstrations, so that the robot may take correct actions even under distributional shift.

Author's address: Rushit N. Shah, rshah231@uic.edu, University of Illinois at Chicago, Chicago, Illinois, 60607.

2 RELATED WORK

As stated before, distributional shift occurs since training and testing examples for imitation learning algorithms are sampled from different distributions i.e. that of the expert policy and the learned policy respectively. This results in large errors during testing which the robot cannot recover from. And it is exactly this problem of distributional shift which results in causal confusion. In this context, a few approaches have been proposed in existing literature [Bhattacharyya et al. 2018].

The authors in [Ross and Bagnell 2010; Ross et al. 2011] proposed DAgger, in which the robot queries the expert action at each state it encounters. This approach, however, was found to be computationally expensive since it requires the robot to constantly update its policy and it is inefficient for an expert to supervise the robot's learning process at each time step [Laskey et al. 2017a,b]. Another approach proposed in literature was that of noise injection; this involves perturbing the expert's demonstrations with random noise before showing them to the robot since this allows the robot to be exposed to states which deviate from the expert's policy, and is likely to encounter [Laskey et al. 2017b]. However, this approach too is tedious since it requires selection of the magnitude and direction of the noise to be injected.

The first approach we adopt in this study is one which focuses on uncovering the causal structure underlying the expert's observations [de Haan et al. 2019]. This involves first learning a causal-graph-parameterized policy using the expert observations, followed by the use of the agent's earned rewards for different causal graphs to learn the true causal graph i.e. distinguishing the causal variables from nuisance variables.

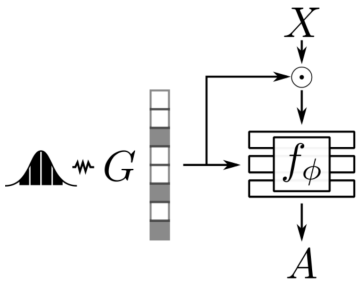

Figure 2: Procedure for graph-parameterized policy learning.

3 PROBLEM DESCRIPTION

The issue of causal confusion is particularly relevant in imitation and reinforcement learning settings. An agent's present state X_t is a direct result of its action in the previous state A_{t-1}. X_t denotes a set of state variables, which subsequently decide what action A_t the robot takes at time t. However, not all variables in this set are true causes (shown as X_{cause} in Figure 1) of the action—some of them are nuisance or non causal variables (shown as $X_{\sim cause}$ in Figure 1). Such non-causal state variables should not determine which action is taken at the following time step. The challenge then is to use the expert's observation data to discover this causal structure i.e. elimination of the causal arrow from all $X_{\sim cause}$ to the action. We try to address this issue by exploring a technique which enables the agent to learn the underlying causal structure of the data [de Haan et al. 2019].

We further explore how the knowledge of the causal structure may be leveraged to build an imitation learning agent which is robust to the covariate shift described in the previous section.

4 INITIAL SOLUTION

The search for the true causal model is a two-step process, as described in [de Haan et al. 2019]. The first involves jointly learning policies corresponding to different possible graphs, followed by a search over the hypothesis set of graphs for the true causal graph.

4.1 Causal-Graph Parameterized Policy Learning

In this step, we learn a policy corresponding to each candidate causal graph. For each expert training observation, a random set of state variables is set to zero, and the corresponding causal graph is concatenated with the observation and used to train a supervised learning model, with the output being the expert's action. This is shown in Figure 2.

4.2 Targeted Intervention

The graph-parameterized policy is executed for different hypothesis graphs, and the rewards the robot incurs for each episode are used to exponentially weight the corresponding graph. Each graph is sampled after reweighting using the gained rewards. After several iterations, the graph weights correspond to the actual causal graph.

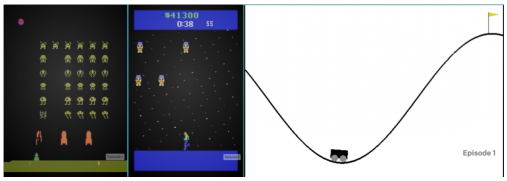

Figure 3: Screenshots of Atari games SpaceInvaders (left), Phoenix (middle), and MountainCar (right).

5 DATASET DESCRIPTION

We wish to use Atari games, on which imitation and reinforcement learning models are commonly trained, as our data sets of choice. Popular Atari games include SpaceInvaders, JourneyEscape, RiverRaid, and Phoenix. These data sets consist of RGB frames of size (210x160x3) captured during gameplay. Given that the data set consists of only images, all features are just pixels, which are inputs to the model. The outcome variable however is the action that the model outputs. Also, given that these are playable games, the data sets do not have a fixed size. The data sets are available as installable packages with environments which must be loaded. Further details are available at gym.openai.com/docs. Screenshots from some of these games are shown in Figure 3.

However, for initial experiments, the low-dimensional MountainCar control problem shown in Figure 3 was employed. As shown in the figure, the problem consists of a car situated on a one-dimensional track, in a valley between two hills. The goal is to drive up the mountain on the right and reach the flag; however, the car's engine is not strong enough to scale the mountain in a single pass. Therefore, the only way to succeed is to drive back and forth to build up momentum. An agent controlling the car can choose from three actions—push left, push right, or no action. To train a causal graph-parameterized policy, expert data is needed. The true state variables in the MountainCar problem are the position and velocity of the car. The action at the previous timestep was added as a third state variable to act as proxy non-causal variable. Expert data was generated by first generating an expert policy using a standard Q-learning-based reinforcement learning algorithm [Watkins and Dayan 1992]. The trained expert policy was then executed (rolled-out) in the MountainCar environment 10,000 times. Each observation comprises three features/covariates (position, velocity, and previous action) and one label (predicted expert action). Over the 10,000 expert policy roll-outs, a total of 1,214,339 such observations were collected. These observations, appropriately shuffled, comprised the data set for further training the causal graph-parameterized policy.

6 INITIAL EXPERIMENTS

6.1 Setup

The three-variable MountainCar problem described in the previous section has 2^3 potential causal graphs; each graph has a different combination of the three variables set to be causal/non-causal. The graph-parameterized policy was first trained using a logistic regression classifier following the procedure in section 4.1. For each

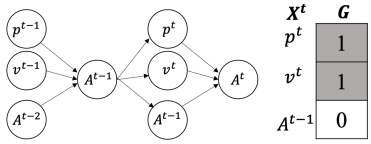

Figure 4: MountainCar problem: evolving state-action model(left); true causal graph (right).

expert observation, a candidate causal graph was generated by randomly setting one, two or all of the three variables to be non-causal; the variables in the observation corresponding to the non-causal components of the graph where also set to zero. The graph was then appended to the modified expert observation and used as input for the logistic regression classifier. For example, at time t, if a candidate graph was generated as $G^t = (1, 0, 1)$, the velocity v^t component of the expert observation at time t was set to 0. Thus, the final input observation at time t would be $(p^t, 0, A^{t-1}, 1, 0, 1)$

Table 1: Algorithm for targeted intervention after training of the graph-parametrized policy.

Algorithm 1 Policy execution intervention
Input: graph-parametrized policy f_ϕ
Initialize $w = 0, \mathcal{D} = \emptyset$
for $i = 1...N$ **do**
Sample $G \sim p(G) \propto exp\langle \text{w,G} \rangle$
Collect episode return R_G by executing f_ϕ
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(\text{G,R}_G)\}$
Fir w on $\mathcal{D}$ using linear regression.
end for
Return: $argmax_G p(G)$

Targeted intervention method in section 4.2 was used to learn weights of each component of the causal graph. The algorithm is provided in Table 1. Each component of the causal graph is initialized to weight of 0. A causal graph is sampled with a probability proportional to that weight. The graph-parameterized is executed for an entire episode and the reward obtained is stored. The components of the graph are then re-weighted by performing linear regression on the collected reward. As more and more episodes are observed, the weights of the graph components converges to the weight of the components in the true causal graph. The reasoning behind this is that policies which decide actions based on true causal variables yield higher rewards than other policies.

6.2 Results

The three state variables for the MountainCar problem are the position, velocity and action at previous time step. The state evolution diagram and the true causal graph corresponding to this problem are shown in Figure 4. For the graph G, 1's represent the causal variables, and 0's represent the non-causal variables. This is the expected result.

The expert-generated training observations consisted of a tuple of (p^t, v^t, A^{t-1}, G^t). These observations were used to train the logistic regression classifier. And finally the targeted intervention procedure was executed for 1000 iterations. As expected, the true causal graph was retrieved. The final weights of each component of the graph is shown in Table 2. The results are obtained as a probability distribution over all possible graphs, and the mode of this graph was found to be the true graph shown in Figure 4. These results are shown in Figure 5 as the output of the final few iterations of the targeted intervention procedure.

Table 2: Weights of graph components after targeted intervention procedure.

p^t	v^t	A^{t-1}
0.502	55.924	-56.125

While mode of the distribution is as expected, the actual weights of the graph components reveal an interesting find. While the position and velocity were recovered as the true causal variables, their weights differ by two orders of magnitude, which leads us to conclude that the velocity of the car contributes far more in determining the next action than the position. Intuitively, this makes sense since the previous velocities of the car are important in determining the acceleration of the car as well, which the imitation learning model seems to have discovered during training. Thus, it has been shown on a low-dimensional example that the adopted technique does indeed discover the underlying causal structure of the problem.

7 BUILDING A ROBUST IMITATOR

The approach considered in the previous sections relied on differentiating between the causal and non-causal variables. Indeed, the approach successfully recovered the true underlying causal graph by weighting variables proportional to their contribution to higher-reward-yielding policies. Building on the results of this methodology, we ask the following questions—what if the causal/non-causal variables in a problem are known, either *a priori* or from the approach in the previous section? With knowledge of such variables available at-hand, is it possible to build an imitating agent that is more robust in its predictions? Does knowing which variables are causal, and which ones are not, allow one to build an imitation agent that is privy to this information and yields more robust results under covariate/distributional shift?

In this and the following sections we attempt to build such an agent. Drawing from the fairness in AI literature, we formulate a classifier which, upon providing the non-causal variables as one of the inputs, trains a classifier which learns to choose its output independent of the non-causal variables. We first describe the formulation of the classifier in the next subsection, touching on the advantages and disadvantages of the proposed methodology. Then, we train the classifier on the MountainCar problem as in the previous sections. Finally, we compare the results of the two methodologies considered in this paper.

7.1 Formulation

Fairness in AI has garnered significant interest in recent years. Though maximizing accuracy has been the principal objective for

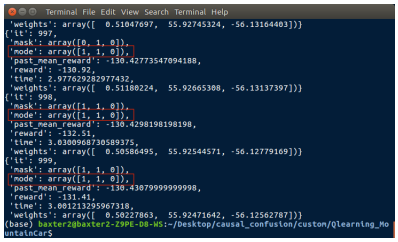

Figure 5: Screenshot of final few iterations of the targeted intervention method. The mode represents the graph sampled according to weights in a time step. The episode and previous mean reward are also shown.

classification tasks, competing priorities are also often of key concern in practice. Fairness properties that guarantee equivalent treatment to different groups in various ways are a prime example. These may be desirable— or even legally required—when making admissions decisions for universities [Chang 2006; Kabakchieva 2013; Lowry and Macpherson 1988], employment and promotion decisions for organizations [Lohr 2013], medical decisions for hospitals and insurers [Obermeyer and Emanuel 2016; Shipp et al. 2002], sentencing guidelines within the judicial system [Moses and Chan 2014; O'Neil 2016], loan decisions for the financial industry [Bose and Mahapatra 2001; Carter and Catlett 1987; Shaw and Gentry 1988], and in many other applications. Group fairness criteria generally partition the population based on a protected attribute into groups and mandate equal treatment of members across groups based on some defined statistical measures. One such group fairness measure is demographic parity [Calders et al. 2009]. A fair classifier that satisfies demographic parity outputs its decisions independent of some protected/sensitive attribute A

$$P(\hat{Y} = 1|A = a) = P(\hat{Y} = 1), \quad \forall a \in \mathcal{A}$$

where A is the protected/sensitive attribute. We extrapolate the idea of demographic parity (DP) to imitation learning scenario. We leverage our knowledge of the causal/non-causal variables in our problem. The protected sensitive attribute in [Rezaei et al. 2020] is thought of as the non-causal variable in the imitation learning scenario. We modify the DP constraint to our non-causal variables as

$$\Gamma : P(\hat{A} = 1|X_{\sim cause} = x) = P(\hat{A} = 1), \quad \forall x \in \mathcal{X}_{\sim cause}$$

Here A is the predicted action by the agent, and $X_{\sim cause}$ is the non-causal variable. Enforcing this constraint while minimizing the training loss ensures that the decision of the trained classifier is independent of the non-causal variables. While it is tempting to merely remove the non-causal variables from the data, given that we know which ones they are, it must be noted that in a higher-dimensional problem, there might be other intermediate attributes correlated with the non-causal variables, and information about the non-causal variables might be inferred from such attributes.

The authors in [Rezaei et al. 2020] develop a robust classifier which satisfies demographic parity by enforcing it as a constraint in the training optimization procedure which involves minimizing the logarithmic loss. Robustness is tackled from a robust estimation perspective by formulating the construction of the classifier as a minimax game between the classifier and a worst-case approximator of the data distribution as shown below

$$\min_{\hat{P} \in \Gamma} \max_{\check{P}} \mathbb{E}[-log\hat{P}(\hat{Y}|X_{cause}, X_{\sim cause}, Y)]$$

$\hat{P}$, minimizes the worst-case log loss—as chosen by approximator $\check{P}$ constrained to reflect training statistics—while providing an empirical guarantee that the decisions will be independent of the non-causal variable.

This, to a certain degree, ensures that the classifier does not incur higher losses under covariate shift—when the distribution of data during testing differs from that during training.

8 FINAL EXPERIMENTAL SETUP

The three-variable MountainCar problem employed in the initial experiments was once again employed in the final experiments to build the robust classifier. The same training data set with 1,214,339 expert observations was used. The data set was split into training (70%) and testing (30%) sets. The proxy non-causal variable added to the data set (action at previous time step) was flagged as the non-causal variable for the robust classifier. Training was performed was 20 iterations.

Also, in its current setup, the robust classifier only supports binary classification. For this reason, all training observations whose expert action corresponded to 'no action' were removed. These corresponded to about 5% of all observations. The data set was almost equally divided between the other two actions.

9 COMPARISON

The following section is dedicated to comparing results of the robust classifier and those of the graph-parameterized algorithm. First, the raw test accuracies of the two classifiers are compared; this corresponds to the accuracies obtained on the test set during training. The algorithms are then compared based on their actual scores; this corresponds to the actual rewards earned when agents trained using these approaches are deployed in the real environment. The graph-parametrized policy outperforms the robust classifier in both cases.

Table 3: Classification accuracies on test data (30%). GP: graph-parameterized policy algorithm; Robust: robust imitator

Algorithm	GP	Robust
Accuracy	0.74	0.57

Test Accuracy: The test accuracies of both, the graph-parametrized policy and the robust classifier, were recorded during training. These are reported in Table 3 above. The graph-parameterized algorithm significantly outperforms in the robust classifier in terms of the raw test accuracy.

Actual Scores: Once trained, both algorithms were deployed in the MountainCar environment. Starting at a random initial state, the policies were allowed to pick an action based on their learned rules. The chosen action was used to step the environment forward and the new state of the environment was observed. This was done until

Table 4: Average scores from 1000 policy execution episodes in the environment. GP: graph-parameterized policy algorithm; Robust: robust imitator

Algorithm	GP	Robust
Avg. Score	-130.43	-200

either, the goal was reached, or a specified number of maximum time steps had passed, whichever happened first. At the end of each such episode, the total reward accumulated by the policies was recorded. Each policy was executed for 1000 episodes. The average rewards accumulated over the 1000 episodes are reported in Table 4 above. The graph-parameterized policy once again outperforms the robust imitator by a wide margin. For reference, the best-reported score on the MountainCar problem in literature is -105.76.

Analysis: Clearly, the proposed robust approach is significantly inferior to the compared graph-parameterized approach. There may be a few reasons for this.

- Binary Restriction: As stated in the setup section, the robust imitator is currently restricted to only two classes; it only predicts the move left or move right actions. Before performing the experiments, it was hypothesized that this might benefit the robust imitator, even if it was more biased, since this agent would always make the car move either left/right which could only build momentum. However, based on the result it seems that the ignored 'no action' action is also important for the car to build enough momentum to reach its target on the hill to its right.
- Test Accuracy: Even before the two policies were deployed in the environment, it was evident that the robust classifier exhibited a lower test accuracy. One can imagine that while the car is trying to build momentum to reach its target, even a small number of misclassified actions would serve to break all the previously-built momentum; the robust classifier seems to misclassify almost every other action (based on its test accuracy being close to 0.5) and such poor performance might not even allow any momentum to built up at all.
- Parameter Tuning: The graph-parameterized policy was trained using hyperparameters known beforehand. The robust imitator on the other hand was not fine-tuned to the MountainCar problem. This could also explain the low test accuracies.
- Too Robust: One of the less likely reasons for the gap in performance of the two algorithms may be that the robust imitator has trained to be too robust. The manner in which the robustness materializes is that when presented with test data from a distribution significantly different from that of the training distribution, the classifier remains uncertain and assigns probabilities close to 0.5 to both candidate actions. This might serve to worsen the performance of an already poorly-tuned model.

10 FUTURE WORK

The current state of this work presents several opportunities to build upon, and improve this direction of research. In our opinion, the low-hanging fruits to reach for would be the drawbacks mentioned in the analysis section. For instance, hyperparameter tuning on the robust classifier might potentially work wonders. This could potentially alleviate the issue of low test accuracies subsequently leading to higher scores in the environment episodes. Extending the robust classifier to a multiclass setting to allow more than just two actions to be chosen is deemed to be a more involved and a long-term task.

The results obtained thus far only involve a very low-dimensional problem. The computer vision task of identifying causal variables from images is, on the other hand, very high-dimensional given that each pixel is a state variable whose value changes every frame (time step). It is desired that the methods which have successfully been applied on low-dimensional problems so far, be equally effective on high-dimensional problems. This involves the following steps as future work

- Training a deep Q-learning algorithm to act as the expert and generate training observations for the imitator.
- Preparation of vision data sets by watermarking images with additional previous state information as non-causal variables. In this case the pixels comprising the watermark form a group of non-causal variables.
- Training an imitation learning algorithm to obtain the graph-parameterized policy
- Execute the targeted intervention procedure to find the causal/non-causal state variables.

Of these steps, a deep Q-learning model has been trained. However, its performance is still far from what would be considered 'expert' behavior; longer training durations are desired. A procedure is also in place to watermark the data once it has been generated by the expert; the the watermark would comprise a letter/number indicating the action taken at the previous time step and that group of pixels would form a subset of the non-causal variables on-screen . Just the way in the MountainCar problem the non-causal variable was found to be the previous time step action, it is expected that upon successful implementation of this procedure on Atari games, non-causal pixels will be detected. For instance, we expect the algorithm to return the pixels with the previous action watermark as the non-causal pixels. We also expect to find which pixels contribute the most to the action at the next time step. For instance, in the case of the game SpaceInvaders (in Figure 3), it is expected that pixels corresponding to the lowest row of aliens on screen will achieve the highest weights, progressively decreasing towards the uppermost rows.

11 CONCLUSION

Two interesting approaches to resolving causal confusion in imitation learning have been explored in this paper. The first approach is useful when in an imitation learning setting it is not known which variables are causal/non-causal. The approach uncovers the underlying causal structure of the problem and trains a graph-parameterized policy which ignores information from the non-causal variables and bases its decision outcomes only on the causal variables.

The second approach is applicable when it is known beforehand which variables in a problem are causal and which ones are not. A robust imitator algorithm is proposed which attempts to base its

outcome decisions independent of the non-causal variables. The robust imitator is hypothesized to perform well under covariate shift conditions. A comparison of results show that the graph-parameterized policy currently significantly outperforms the robust imitator approach.

REFERENCES

Raunak P Bhattacharyya, Derek J Phillips, Blake Wulfe, Jeremy Morton, Alex Kuefler, and Mykel J Kochenderfer. 2018. Multi-agent imitation learning for driving simulation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1534–1539.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* (2016).

Indranil Bose and Radha K Mahapatra. 2001. Business data mining—a machine learning perspective. *Information & management* 39, 3 (2001), 211–225.

Toon Calders, Faisal Kamiran, and Mykola Pechenizkiy. 2009. Building classifiers with independency constraints. In *2009 IEEE International Conference on Data Mining Workshops*. IEEE, 13–18.

Chris Carter and Jason Catlett. 1987. Assessing credit card applications using machine learning. *IEEE expert* 2, 3 (1987), 71–79.

Lin Chang. 2006. Applying data mining to predict college admissions yield: A case study. *New Directions for Institutional Research* 131 (2006), 53–68.

Hal Daumé III. 2009. Unsupervised search-based structured prediction. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 209–216.

Pim de Haan, Dinesh Jayaraman, and Sergey Levine. 2019. Causal Confusion in Imitation Learning. In *Advances in neural information processing systems*.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. 2017. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)* 50, 2 (2017), 21.

Dorina Kabakchieva. 2013. Predicting student performance by using data mining methods for classification. *Cybernetics and information technologies* 13, 1 (2013), 61–72.

Michael Laskey, Caleb Chuck, Jonathan Lee, Jeffrey Mahler, Sanjay Krishnan, Kevin Jamieson, Anca Dragan, and Ken Goldberg. 2017a. Comparing human-centric and robot-centric sampling for robot deep learning from demonstrations. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 358–365.

Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. 2017b. Dart: Noise injection for robust imitation learning. *arXiv preprint arXiv:1703.09327* (2017).

Steve Lohr. 2013. Big data, trying to build better workers. *The New York Times* 21 (2013).

Stella Lowry and Gordon Macpherson. 1988. A blot on the profession. *British medical journal (Clinical research ed.)* 296, 6623 (1988), 657.

Lyria Bennett Moses and Janet Chan. 2014. Using big data for legal and law enforcement decisions: Testing the new tools. *UNSWLJ* 37 (2014), 643.

Ziad Obermeyer and Ezekiel J Emanuel. 2016. Predicting the future—big data, machine learning, and clinical medicine. *The New England journal of medicine* 375, 13 (2016), 1216.

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

Dean A Pomerleau. 1989. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*. 305–313.

Ashkan Rezaei, Rizal Fathony, Omid Memarrast, and Brian Ziebart. 2020. Fairness for Robust Log Loss Classification. *Proc. 24th AAAI Conference on Conference on Artificial Intelligence (New York, New York, USA, February 2020)* (2020).

Stéphane Ross and Drew Bagnell. 2010. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 661–668.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 627–635.

Michael J Shaw and James A Gentry. 1988. Using an expert system with inductive learning to evaluate business loans. *Financial Management* (1988), 45–56.

Margaret A Shipp, Ken N Ross, Pablo Tamayo, Andrew P Weng, Jeffery L Kutok, Ricardo CT Aguiar, Michelle Gaasenbeek, Michael Angelo, Michael Reich, Geraldine S Pinkus, et al. 2002. Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. *Nature medicine* 8, 1 (2002), 68.

Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8, 3-4 (1992), 279–292.