Offline Contextual Bandits for Wireless Network Optimization

Miguel Suau[1,2,*], **Alexandros Agapitos**[1], **David Lynch**[1], **Derek Farrell**[1], **Mingqi Zhou**[1], **and Aleksandar Milenovic**[1]

[1]Huawei Ireland Research Center
[2]Delft University of Technology
[*]m.suaudecastro@tudelft.nl

Abstract

The explosion in mobile data traffic together with the ever-increasing expectations for higher quality of service call for the development of AI algorithms for wireless network optimization. In this paper, we investigate how to learn policies that can automatically adjust the configuration parameters of every cell in the network from a static log of past real interactions. We propose a hybrid method that combines a policy network learned via importance sampling to suggest preliminary actions, followed by gradient-based local fine-tuning on a differentiable model of the reward. Empirical results suggest that our method will achieve important performance gains when deployed in the real network while satisfying practical constrains on computational efficiency.

1 Introduction

The evolution of *wireless networks* from 2G, 3G, 4G and now to 5G, in response to users' insatiable appetite for greater capacity and lower latency, has been driven by continuous innovations in technology and infrastructure. New data hungry services now available to the user (such as online gaming, video streaming and virtual reality) are causing an explosion in mobile data traffic. Furthermore, the usage growth translates into increasingly complex wireless networks. These factors render network operation and maintenance infeasible. Currently, network operators adjust the configuration parameters (CPs) of every cell in the network through a trial and error process. Manually adjusting CPs is time consuming process and typically results in highly suboptimal settings. Long lead times cause previously performant CPs to become stale relative to dynamic usage demand and meteorological conditions. Moreover, the scale of the network, with up to 10K inter-dependent cells and many CPs per cell makes ad hoc solutions impractical.

Here we investigate a data-driven approach to wireless network optimization. We learn policies that select the appropriate CPs for each cell, given information characterizing the current network state. Our end goal is to optimize the average *throughput* (TP) per user using only a *static dataset* of past interactions (Metevier et al., 2019; Sachdeva et al., 2020; Levine et al., 2020) with the real network. The key challenges addressed in this paper are summarized below:

1. **Large action space:** Our goal is to control 14 CPs in total. Any combination of CP values within the allowed range is a candidate solution. Optimizing TP given only a static data set is thus not trivial since the resulting action space is very large and the CPs may have complex, possibly non-linear, effects on TP.
2. **Data sparsity and covariate shift:** Due to the large combinatorial action space, it is impractical to obtain a densely sampled dataset. Certain parts of the state-action space

Offline Reinforcement Learning Workshop at Neural Information Processing Systems, 2021.

will be inevitably under-represented in the logged data. Moreover, given that we want to improve over the logging policy, the action distribution induced by the learned policy should be different from the logged distribution. This phenomenon, known as covariate shift (Bickel et al., 2009; Quiñonero-Candela et al., 2009), only exacerbates the problem of sample sparsity since the actions we are interested in will likely be poorly supported in the data.

3. **Computational efficiency:** The number of cells in the network is large (2945) and updated CPs must be pushed to all cells simultaneously every hour. As such, due to hardware constraints, it is essential that our solution is computationally efficient.

The main contributions of this paper are:

1. We formulate of the wireless network optimization problem as an **offline contextual bandit** (Auer et al., 2002; Langford and Zhang, 2008).
2. We propose a **hybrid method** that combines a **policy network** learned via importance sampling to suggest preliminary actions, followed by **gradient-based local fine-tuning** on a differentiable model of the TP.
3. We introduce a **modular neural network architecture** for TP prediction that ensures a balanced representation of all input modalities and **enhances the sensitivity** of the model to the CPs.
4. Finally, to guarantee the reliability of the method against model bias, we perform **counterfactual data augmentation** to help our model generalize over unseen actions, and apply a **penalty** to the objective function to prevent the optimization algorithm from finding solutions for which the TP predictions are **uncertain**.

The rest of the paper is organized as follows. Section 2 introduces the problem formulation and mathematical notation. We also provide a short description of the dataset used to train and evaluate our policies. Section 3 reviews the two most popular offline learning approaches for the contextual bandit setting. In section 4, we present our method and describe how we addressed the above challenges. Finally, the results of our experiments are reported in Section 5, from which key conclusions are drawn.

2 Preliminaries

2.1 Problem definition

Let us call $s \in \mathcal{S}$ the *network state* (sometimes referred to as *context* in the bandits literature), which is specified by performance management (PM) counters, engineering parameters (EP) and other contextual information describing the network environment. $\mathcal{A}$ is the action space of 14 dimensions, one for each of the CPs to be optimized, such that each of the infinitely many actions $a \in \mathcal{A} = \mathcal{A}_1 \times \ldots \mathcal{A}_{14}$ represents a particular parameter configuration at a specific time. The task consists in finding the policy $\pi(s, a) = P(a|s)$ that maximizes a certain objective function,

$$J(\pi) = \mathbb{E}_{s\sim\mathcal{S}, a\sim\pi(s,a)}\left[R(s, a) = r\right] \tag{1}$$

and r is the feedback provided by the system. In our case, $R(s, a)$ is an unknown function that models the TP per user given the action a and the network state s. As opposed to the more general reinforcement learning framework, the bandit formulation assumes that actions have no temporal dependencies. While this might introduce approximation errors, it also simplifies the optimization problem. Moreover, this simplification is motivated by the fact that CP changes occur only once per day in our static dataset, which might dilute temporal dependencies from hour-to-hour.

2.2 Dataset Description

As mentioned in the introduction, we are given a static dataset of past system interactions collected using a fixed policy π_0, known as the logging policy,

$$\mathcal{D}_{\pi_0} = \{s_i, a_i, r_i\}_{i=1}^{N}. \tag{2}$$

The dataset was retrieved from a real 4G network. The network contained 2945 cells served by 1030 radio base stations. The dataset covered a period of 9 days during which changes to 14 CPs were

made *daily* to each cell between midnight and 2 AM. The 14 CPs were chosen because they have the largest impact on TP. A total of 295K CP changes were registered. The effect of these changes on the network was monitored by hourly performance management (PM) counters. Other information regarding the engineering parameters (EP) of the antennas, their physical location, and the time of the day was also included in the dataset.

3 Existing Approaches

In the offline learning setting, we are not allowed to explore and assess actions on the real network as is normally assumed for standard contextual bandits. This poses a fundamental challenge for the learning algorithm. On the one hand, we want the new policy to outperform the logging policy, which requires it to take different actions from those observed in the dataset. On the other hand, the outcomes for these hypothetical actions are not present in the dataset. All we can do is learn from $\mathcal{D}$ those actions that lead to high-payoffs in throughput with the expectation that our policy will be able to generalize effectively over unseen contexts. See Levine et al. (2020) for an extensive discussion on this matter. Below we describe the most popular solutions for the offline learning problem in the contextual bandit setting.

3.1 Direct Method

The *direct method* (Beygelzimer and Langford, 2009) consists of learning a regression model that for every state s can predict the reward for taking any of the actions available, $\hat{R}(s,a) \approx R(s,a)$ for all $a \in \mathcal{A}$. This can be then used to compute a policy $\pi(s)$ as

$$\pi(s) = \arg\max_{a \in \mathcal{A}} \hat{R}(s,a). \tag{3}$$

It is clear that the performance of DM depends very much on the accuracy of the reward model. Unfortunately, since this is only trained on $\mathcal{D}$, its estimates might be biased from the true reward values when the number of samples is too small. Note that a biased model is particularly problematic in this case, because the *argmax* in equation (3) forces the policy to select the actions for which the reward estimates are the largest, which may or may not correspond to those actions that maximize the true reward.

3.2 Inverse Propensity Score

An alternative to the DM is to obtain an estimate of the objective function $J(\pi)$ by applying importance sampling (Horvitz and Thompson, 1952) on experiences drawn from $\mathcal{D}$,

$$J(\pi_\theta) \approx \frac{1}{N} \sum_{(x_i,a_i,r_i) \in \mathcal{D}} \frac{\pi(s_i,a_i)}{\pi_0(s_i,a_i)} r_i, \tag{4}$$

where $\frac{\pi_\theta(s_i,a_i)}{\pi_0(s_i,a_i)}$ is the importance ratio, which corrects for the fact that the actions in D were sampled from π_0 rather than π. Equation (4) is known as inverse propensity score (IPS) (Strehl et al., 2010; Li et al., 2014) and although it is an unbiased estimator, it can have very high variance especially when π deviates from π_0.

4 Methods

We now describe how the above techniques were adapted for wireless network optimization. We will focus on the three main challenges we described in the introduction; namely, large action space, covariate shift, and computational efficiency.

We first train a neural network to predict the reward (TP) given the network state s and the action a. The modular neural network architecture is displayed in Figure 1. Separate input heads balance the influence of temporal context, engineering parameters (EPs) characterising a cell, and counters by mapping them to tensors of equal dimensionality (50 neurons). The 'Covariate Representation Network' merges the resulting tensors to yield a representation of the network state s. Explicitly balancing the representations of s and a ensures the influence of CPs is not diluted in the presence other high dimensional covariates, such as counters. Sensitivity to CPs is further enhanced by

introducing them near the output layer. Finally, all representations are concatenated and passed through linear output layers to give the predicted TP. Once the network is trained, we can specify a greedy policy that selects the action a^* that maximizes the predicted reward given network state s.

4.1 Challenge 1: Large Action Space

The optimization problem specified in (3) can be solved using exhaustive search if the action space is finite and small. However, this is prohibitively expensive in our setting due to the high-dimensional and continuous action space. The proposed solution is to perform gradient ascent (GA) using the reward model, which is point-wise differentiable. We make use of software for automatic differentiation (Paszke et al., 2019) to compute $\nabla_a \hat{R}(s, a)$. Then, starting from a random initial a_0 we iteratively update its value following the direction of steepest ascent,

$$a_{k+1} = a_k + \alpha \nabla_a \hat{R}(s, a), \qquad (5)$$

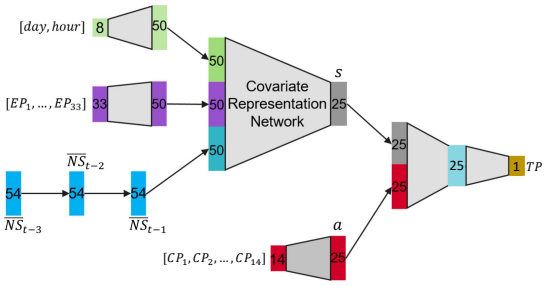

Figure 1: Reward model with modular architecture. Modularity ensures a balanced representation of all input modalities and sensitivity to CPs. Coloured rectangles indicate input covariates and representations produced by the network, where the number of neurons is specified. Trapezoids are linear layers. Trend in the counters is modelled using a Gated Recurrent Unit network (cyan blocks).

where $\alpha \in \mathbb{R}$ constrains the steps size in the action space. If $\hat{R}(s, a)$ is non-convex (as it is normally the case for multi-layer neural networks) gradient ascent is only guaranteed to find a local optimum. The process is repeated multiple times with different initial values for a_0 to avoid poor local optima.

4.2 Challenge 2: Sparse Data and Covariate Shift

As mentioned in Section 3.1, since the DM is prone to bias, deploying the resulting policy directly on the real network may be unsafe. In general, we cannot guarantee that the reward model will generalize well to regions of the state-action space that are underexplored by the logging policy (Janner et al., 2019). A biased reward model may confidently predict that an action will attain a high TP, even if this action was not well supported in the dataset. As a result, the policy might not necessarily select the actions for which the true reward is the largest but also those for which the model is merely highly overconfident. Two complementary solutions are introduced to alleviate bias and improve the reliability of the DM: data augmentation and uncertainty regularization.

4.2.1 Counterfactual Data Augmentation

Our ability to train accurate and reliable reward models is hindered by selection bias in the factual dataset. Selection bias arises because CPs are not set uniformly at random, but rather they are generated by the logging policy, when it maps network state information to optimized CPs. The resulting dataset may lack diversity (if some CPs are never selected) and contain spurious correlations between CPs and TP (due to confounding).

Challenges arising from selection bias motivate the proposed causal inference matching algorithm. Our method draws inspiration from Schwab et al. (2018, 2020). We augment the factual dataset with hypothetical examples obtained via matching. A hypothetical example is synthesized from each factual example in the dataset. A single hypothetical example is obtained by replacing the *factual* CPs (given by the logging policy) with new *hypothetical* CPs sampled uniformly at random from their allowed range of values. Ten factual nearest neighbours are identified based on Euclidean distance. These nearest neighbours should have similar CPs and covariates to those of the hypothetical. The unknown TP that would have been observed if the hypothetical CPs had been executed is estimated as the mean (factual) TP of the nearest neighbours. An augmented dataset is produced in this fashion and used to train the reward model. The resulting model will generalize over arbitrary CPs, even if such CPs would not be selected by the logging policy.

4.2.2 Uncertainty Regularization.

An uncertainty penalty is added to the objective in (3) in order to prevent the optimization process from converging to those regions of the state-action space for which the model may be biased. To do so we first create K different bootstrap samples $\mathcal{D}_k$ of size N by sampling with replacement from $\mathcal{D}$. A neural network ensemble Hansen and Salamon (1990); Heskes (1996); Khosravi et al. (2011) is then trained by fitting K models $\hat{R}_k(s,a)$ (with the identical architectures) to each of the K datasets. Reward estimates are computed as the average prediction of all models in the ensemble. The uncertainty penalty is given by the standard deviation of those predictions,

$$\pi(s) = \arg\max_{a \in \mathcal{A}} \left(\hat{\mu}(s,a) - \beta\hat{\sigma}(s,a)\right) \tag{6}$$

with μ being the empirical mean, σ the standard deviation and $\beta \in \mathbb{R}$ a hyperparameter controlling the magnitude of the uncertainty penalty. Since the objective (6) is still differentiable, gradients can be backpropagated through each of the models in the ensemble to compute $\nabla_a \left(\hat{\mu}(s,a) - \beta\hat{\sigma}(s,a)\right)$.

4.3 Challenge 3: Computational Efficiency

One important caveat of the above described method is its computational complexity. Running multiple iterations of GA every time an action is chosen is extremely time consuming, especially if a neural network ensemble is used to estimate the uncertainty. An alternative, is to optimize a policy network $\pi_\theta(s,a)$ directly by following the off-policy policy gradient (OPPG) (Precup et al., 2000). An estimate of the OPPG can be easily derived from (4),

$$\nabla_\theta J(\pi_\theta) \approx \frac{1}{N} \sum_{(x_i,a_i,r_i)\in\mathcal{D}} \frac{\pi_\theta(s_i,a_i)}{\pi_0(s_i,a_i)} r_i \nabla_\theta \log \pi_\theta(s_i,a_i). \tag{7}$$

Unfortunately, the importance weight $\frac{\pi_\theta(s_i,a_i)}{\pi_0(s_i,a_i)}$ has unbounded variance. Gradient estimates may grow arbitrarily large when certain state-action pairs for which $\pi_0(s,a) \approx 0$, become more likely under π_θ. The most straight-forward solution is to control the size of $\frac{\pi_\theta(s_i,a_i)}{\pi_0(s_i,a_i)}$ by setting a maximum threshold (Ionides, 2008; Swaminathan and Joachims, 2015),

$$\nabla_\theta J(\pi_\theta) \approx \frac{1}{N} \sum_{(x_i,a_i,r_i)\in\mathcal{D}} \min\left\{\frac{\pi_\theta(s_i,a_i)}{\pi_0(s_i,a_i)}, M\right\} r_i \nabla_\theta \log \pi_\theta(s_i,a_i), \tag{8}$$

where the hyperparameter M lets us trade variance for bias. Notice that, if we set $M = 1$ we get back the standard policy gradient estimator.

4.3.1 The Hybrid Method.

The variance problem becomes more severe when the action space is high-dimensional. This is because, as we increase the number of dimensions the likelihood of certain actions under the logging policy π_0 decreases exponentially, thus limiting the quality of the policies we can learn. Nonetheless, we can still benefit from these policies, even if they improve the performance of π_0 only slightly. We propose a *hybrid method* whereby GA is executed on the reward model, but rather than starting from a random initial point, we start from the actions suggested by a pre-trained policy network π_θ. Our experiments suggest that this approach not only decreases the number of random starts needed, but also helps GA converge much faster, since the actions sampled from π_θ are often good enough, and only need to be fine-tuned.

5 Evaluation

The method is tested on 10K samples drawn from the logged data that were excluded from the training set. We follow a similar evaluation protocol to that of Brookes et al. (2019) and use two separate reward model ensembles each of them consisting of 5 neural networks trained on different bootstrap samples. We use the first ensemble $\hat{R}_\theta$ to optimize the actions and report the TP predicted by the second ensemble $\hat{R}^*_\theta$. The plot on the left of Figure 2 shows the true TP (green) and the predicted TP given by $\hat{R}^*_\theta$ realized when CPs were set by the logging policy (red). The purple box plot shows the predicted TP when the CPs were set by the policy network π_θ (see Section 4.3).

Figure 2 (middle) compares the performance of running GA on $\hat{R}_\theta$ when choosing the initial actions at random (orange) against the hybrid method (blue), which starts from the actions suggested by the policy network (GA $+\pi_\theta$). The box plots suggest that both GA and GA $+\pi_\theta$ outperform the logging policy (red). As expected, the performance gain grows when we increase the number of starts.

On the other hand, even though the policy network π_θ trained via OPPG as a stand-alone solution (purple) improves over π_0 (red) only slightly, when combined with GA (GA $+\pi_\theta$) it matches the performance of standard GA with random starts while mitigating the computational burden. This is supported by the horizontal box plots on the right of Figure 2. The plot on the top shows the total wall clock time of running a full loop of GA on the reward model. The plot on the bottom shows the number of GA steps it takes to converge to an optimum. The results clearly indicate that the actions sampled from π_θ are often good enough and only need to be fine-tuned. Hence, the improvement in computational efficiency.

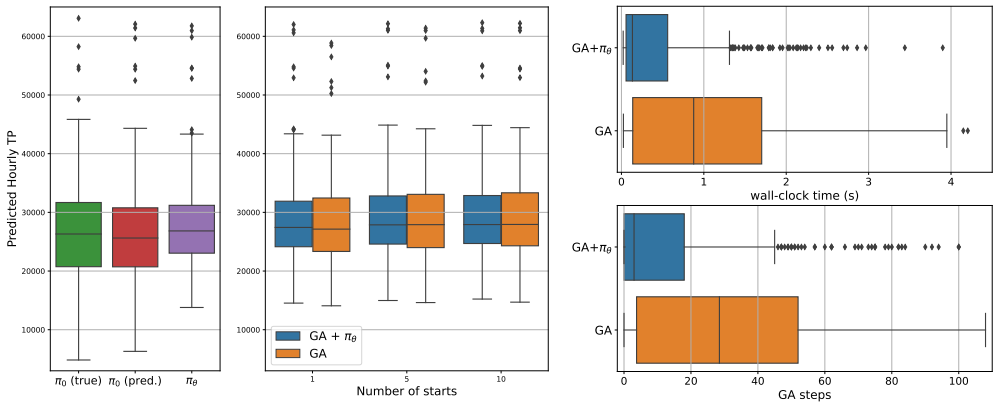

Figure 2: **Left:** True (green) and predicted (red) hourly TP for the logging policy π_0. Predicted TP for the policy network π_θ (purple). **Middle:** Predicted TP for GA with 1, 5, and 10 starts when sampling the initial actions at random from a uniform distribution (orange) and from π_θ (blue). **Top right:** Wall-clock time of running a full GA loop on the reward model with (blue) and without (orange) π_θ. **Bottom right:** Number of GA steps to convergence with (blue) and without (orange) π_θ.

The performance of GA $+\pi_\theta$ when adding the uncertainty penalty term to the objective function (6) is reported in Figure 3. The plot on the left shows how the uncertainty of the TP predictions decreases as we increase the penalty coefficient β, which implies that GA is avoiding the regions were the reward predictions might be overconfident. On the other hand, the plot on the right reveals that, decreasing the uncertainty comes at the expense of potentially lower TP gains.

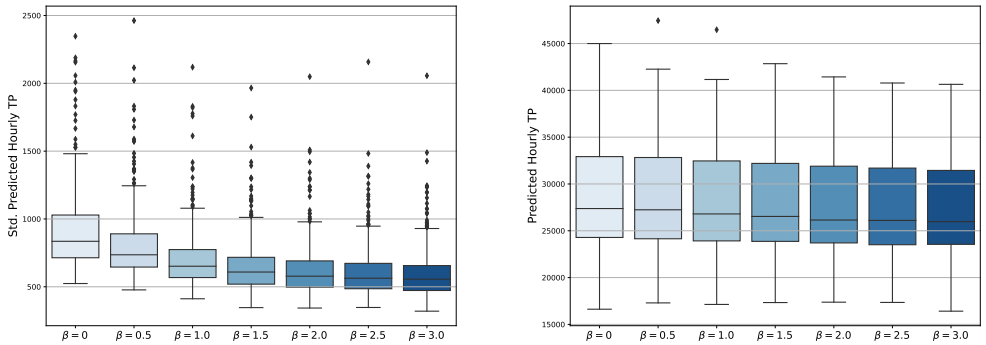

Figure 3: **Left:** Uncertainty of the predicted TP after optimizing (6) with GA for different values of penalty coefficient β. **Right:** Predicted TP after optimizing (6) for different values of β.

6 Conclusion

In this paper we presented a data-driven solution to the wireless network optimization problem. We first showed how we formulated the problem as a contextual bandit, and explained how we learned policies offline from a static dataset. Our method combines a policy network trained via OPPG followed by gradient-based local fine-tuning. To guarantee robustness against model bias we augmented the dataset with hypothetical counterfactual examples. We also applied an uncertainty penalty to the optimization objective. Our results suggest that the proposed hybrid method can handle large action spaces while being computationally efficient. Moreover, pending online evaluation, we expect important gains in TP when this solution is deployed in the real wireless network.

References

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. (2002). The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32:48–77.

Beygelzimer, A. and Langford, J. (2009). The offset tree for learning with partial labels. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 129–138.

Bickel, S., Brückner, M., and Scheffer, T. (2009). Discriminative learning under covariate shift. *Journal of Machine Learning Research*, 10(9).

Brookes, D., Park, H., and Listgarten, J. (2019). Conditioning by adaptive sampling for robust design. In *Proceedings of the 36th International Conference on Machine Learning*, pages 773–782.

Hansen, L. and Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001.

Heskes, T. (1996). Practical confidence and prediction intervals. In *Proceedings of the 9th International Conference on Neural Information Processing Systems*, NIPS'96, page 176–182, Cambridge, MA, USA. MIT Press.

Horvitz, D. G. and Thompson, D. J. (1952). A generalization of sampling without replacement from a finite universe. *Journal of the American statistical Association*, 47(260):663–685.

Ionides, E. L. (2008). Truncated importance sampling. *Journal of Computational and Graphical Statistics*, 17(2):295–311.

Janner, M., Fu, J., Zhang, M., and Levine, S. (2019). When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, volume 32.

Khosravi, A., Nahavandi, S., Creighton, D., and Atiya, A. F. (2011). Comprehensive review of neural network-based prediction intervals and new advances. *IEEE Transactions on Neural Networks*, 22(9):1341–1356.

Langford, J. and Zhang, T. (2008). The epoch-greedy algorithm for multi-armed bandits with side information. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc.

Levine, S., Kumar, A., Tucker, G., and Fu, J. (2020). Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*.

Li, L., Chen, S., Kleban, J., and Gupta, A. (2014). Counterfactual estimation and optimization of click metrics for search engines. *CoRR*.

Metevier, B., Giguere, S., Brockman, S., Kobren, A., Brun, Y., Brunskill, E., and Thomas, P. S. (2019). Offline contextual bandits with high probability fairness guarantees. In *Advances in Neural Information Processing Systems*, volume 32.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035.

Precup, D., Sutton, R. S., and Singh, S. P. (2000). Eligibility traces for off-policy policy evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, page 759–766, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Quiñonero-Candela, J., Sugiyama, M., Lawrence, N. D., and Schwaighofer, A. (2009). *Dataset shift in machine learning*. Mit Press.

Sachdeva, N., Su, Y., and Joachims, T. (2020). Off-policy bandits with deficient support. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 965–975.

Schwab, P., Linhardt, L., Bauer, S., Buhmann, J. M., and Karlen, W. (2020). Learning counterfactual representations for estimating individual dose-response curves. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34.

Schwab, P., Linhardt, L., and Karlen, W. (2018). Perfect match: A simple method for learning representations for counterfactual inference with neural networks. *arXiv preprint arXiv:1810.00656*.

Strehl, A. L., Langford, J., Li, L., and Kakade, S. M. (2010). Learning from logged implicit exploration data. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2*, NIPS'10, page 2217–2225.

Swaminathan, A. and Joachims, T. (2015). Batch learning from logged bandit feedback through counterfactual risk minimization. *Journal of Machine Learning Research*, 16(52):1731–1755.