Warfarin Dosage Prediction with LinUCB and Logistic Regression

Haojun Li [1]

Abstract

Predicting the correct dosage of Warfarin usually requires many trial-and-errors, and correct dosage found for one patient cannot be immediately used to predict the correct dosage for the next patient using traditional methods. However, due to Reinforcement Learning's fast ability to learn and adapt to new data, we can make more accurate decisions as we see more and more patients and incorporate them immediately in predicting the dosage for the next patient. In this report I model this problem as a Stochastic Linear Bandit problem and uses Disjoint LinUCB algorithm to predict the dosage range given patient features. This method outperforms 2 baselines (Fixed Dosage and Clinical Dosing Algorithm) in terms of regret. I also provide a supervised version of the algorithm, and shows that it outperforms all other methods in terms of regret, but not fraction of incorrect dosages against Clinical Dosing Algorithm due to problem construction.

1. Introduction

Warfarin is a blood thinner commonly used to treat a variety of blood clots. Since it is difficult to accurately prescribe the appropriate dose of Warfarin to patients, many researchers have attempted this problem with a variety of algorithms and data. (Consortium, 2009) However, most of these approaches are based on a supervised dataset obtained from existing patients with known correct dosage of Warfarin. These gold labels of correct Warfarin dosage is obtained through trial and error (give a low dosage in the beginning and gradually increase the dosage until we find the correct one). These methods perform very well in practice, but a main drawback of these algorithms is that it requires an initial period of trial and error. This means that the next patient who needs to be treated with Warfarin cannot benefit from the information learned from the patient immediately before it. Reinforcement Learning, on the other hand, allows us to constantly update our estimation based on observations of the previous predicted dose. Thus, it is an attractive algorithm in finding the correct dosage of Warfarin in an ongoing medial trial.

Reinforcement Learning has applications in many fields ranging from robotics (Peters et al., 2003) to advertisement bidding (Cai et al., 2017). Its ability to learn information in real-time is particularly attractive to problems where an agent can interact with an environment and gather real time data. It has allowed researchers to move beyond rule-based systems and instead use a data-driven approach to improve systems in real time. Another advantage of these algorithms is that it is sometimes able to achieve super human performance, especially in video games (Mnih et al., 2013). However, many challenges remain for these algorithms. For example, most of them require substantial amount of trial and error in order for the agent to learn an optimal policy. In Mnih et al. (2013) for example, the agent has to play millions of games before achieving reasonable performance, and eventually achieving super human performance. An attempt to address this problem is using imitation learning (Codevilla et al., 2018) to learn a policy by mimicking human behavior. Another approach is to bound the regret theoretically, such as the UCB algorithm (Lai & Robbins, 1985), which balances exploration and exploitation, achieving better results in situations where trying millions of times is not an option. These so called "multi-arm bandit" problem formulation is widely used in online advertising due to its ability to (eventually) behaving optimally while exploring a reasonable amount of time. A specific type of bandit problem called "stochastic linear bandits" (Abe et al., 2003) allows us to also incorporate context in decision making, making bandit problems more realistic in problems where observations and actions can be featurized and incorporated into the agent's decision making process.

2. Related Works

There are many algorithms introduced over the years in providing better bounds for the regret for stochastic linear bandit problems. The classic algorithms such as one introduced in Auer (2002) as well as later works to improve upon

[1]Department of Computer Science, Stanford University, CA, USA. Correspondence to: Haojun Li <haojun@stanford.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

the upper bound in Dani et al. (2008) both attempted to find a good algorithm that gives polylog(T) regret. Other works adapted the algorithms to other practical domains, such as Li et al. (2010), which introduces the LinUCB algorithm that we will be experimenting with in this report. Other methods that were adapted to classic multi-arm bandits such as Thompson Sampling have been adapted to contextual bandits with linear payoffs as well (Agrawal & Goyal, 2013). For more information related to more algorithms in linear bandits, please refer to the book Lattimore & Szepesvári (2018) as well as the default project handout.

3. Approach

3.1. Problem Definition

The problem definition is the same as the default project handout. Please refer to the handout for more information.

3.2. Baselines

The baselines are as defined in the default project handout. We will chose the first 2 baselines to compare, namely a *Fixed dose* baseline as well as a *Warfarin Clinical Dosing Algorithm* baseline. The former is a baseline that prescribes medium dose (21-49 mg/week) while the latter is an algorithm that predicts the dosage based on some features of the patient. To make comparison fair to the baseline, all feature-based algorithm introduced below as well as regret calculations will be using the same set of feature classes: **Race, Age in decades, Height (cm), Weight (kg), Carbamazepine (Tegretol), Amiodarone (Cordarone), Phenytoin (Dilantin), Rifampin or Rifampicin**.

3.3. LinUCB

The LinUCB algorithm that we will adapt is introduced in Li et al. (2010). We will adapt the "LinUCB with disjoint linear models". We did not experiment with Hybrid models since there is no additional context shared among different arms, but rather the decision to pull each arm can be seen as independent. According to the original paper, the benefit of the hybrid model is that it will understand context similarities between arms, which does not apply to our problem.

Since we know that there are only 3 arms to pull, we can initialize the matrix beforehand. The rest of the algorithm is exactly the same as in the original paper. However, the original author of the paper provided $\alpha = 1 + \sqrt{\ln(2/\delta)/2}$ as the input of the algorithm given δ. However, further investigation below shows that different α's yield different results given our horizon. In particular, smaller α leads to earlier exploitation, which yields a worse regret bound in exchange for minimizing short term regret. This is particularly useful given our problem since we have a short horizon (only 5528 patients). This also leads us to adopt smaller set of features since we hypothesize that a higher set of features will lead the agent to explore more due to its optimistic nature.

Algorithm 1 Adapted Disjoint LinUCB

Input: $\alpha \in \mathbb{R}^+$
Initialize $\mathbf{A}_a \leftarrow \mathbf{I}_d$ for all $a \in \{\text{low, medium, high}\}$
Initialize $\mathbf{b}_a \leftarrow \mathbf{0}_{d\times 1}$ for all $a \in \{\text{low, medium, high}\}$
for $T = 1, 2, 3, \cdots, T$ **do**
  for $a \in$ {low, medium, high} **do**
    $\hat{\beta}_a \leftarrow \mathbf{A}_a^{-1}\mathbf{b}_a$
    $p_{t,a} \leftarrow \hat{\beta}_a^\top \mathbf{x}_{t,a} + \alpha\sqrt{\mathbf{x}_{t,a}^\top \mathbf{A}_a^{-1}\mathbf{x}_{t,a}}$
  end for
  $a_t = \arg\max_{a\in\{low,medium,high\}} p_{t,a}$ and observe r_t
  $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_{t,a_t}\mathbf{x}_{t,a_t}^T$
  $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t\mathbf{x}_{t,a_t}$
end for

3.4. Supervised Learning

We also adopt a supervised learning approach. Since we have the correct dosage category for each patient, it seems like a waste to not be able to use it. We will use Softmax Regression to fit the features defined in section 3.2 in bold with a batch size of 50. In essence, for every 50 patients we will add them to our memory and refit the model on all the patients we have seen so far. The generic algorithm that works with any supervised model class is described in detail below

Algorithm 2 Supervised Learning

Input: Batch Size b
Initialize A list of patient features X_0 and labels y_0
for $t = 1, 2, 3, \cdots, T$ **do**
  if $T < b$ **then**
    $X_t \leftarrow X_{t-1} + [x_t]$
    select $a_t =$ medium, observe r_t and label y_t
    $y_T \leftarrow y_{t-1} + [y_t]$
  end if
  if $t \bmod b == 0$ **then**
    Fit supervised predictor $f_{\lfloor t/b \rfloor}$ to X_{t-1} and y_{t-1}
  end if
  select $a_t = f_{\lfloor t/b \rfloor}(x_t)$, observe r_t and label y_t
  $X \leftarrow X + [x_t]$
  $y \leftarrow y + [y_t]$
end for

The batch size can be tuned depending on how frequently we want to update. Faster updates means that we can adapt quicker to new data, but also means that we might have more noisy predictions due to this fast adaptation.

3.5. Regret

To calculate regret, we first fit a linear model on the features selected in section 3.2 in bold text as well as the actual reward (-1 for a failed prediction and 0 otherwise) for each action (low, medium, high). Then, we use the coefficients derived from the linear model to get the predicted reward for each dosage category for each patient in our dataset $\beta_a^\top x_i$. To calculate empirical regret, we simply sum the regret for each predicted action against the action with the highest $\beta_a^\top x_i$. Specifically, the total regret for a sequence of actions $\mathbf{a}$ on the patient features X is

$$\text{regret}_{X,\mathbf{a}} = \sum_{t=1}^{T} \max_{a' \in \{low, medium, high\}} \beta_{a'}^\top x_t - \beta_{a_t}^\top x_t \quad (1)$$

4. Experiment Results

4.1. LinUCB

4.1.1. Alpha Search

Finding the correct exploration bonus term is important since we have a shorter horizon. Thus, it may be better to commit early to a policy that we believe will do sufficiently well rather than keep exploring hoping for a better policy. Thus, a lower exploration bonus means lower average total regret but could lead to sub-optimal solutions. Here we compare the 2 metrics (average fraction of incorrect dosages and average total regret) of each α value.

If the α value is 0.0, which means that we have no exploration bonus, then we are basically running a greedy strategy. We estimate the value for each patient and immediately apply that information to the next patient. Thus, we have a huge variation of fraction of incorrect cases and average total regret as seen in figure 1 and 2. The confidence interval of the average of the 2 metrics across 20 trials is very large. In addition to that, as seen in figure 3, we committed to a policy too early and was unable to find the optimal solution later on, achieving much worse average fraction of incorrect doses as the algorithm sees more patients.

Another important result is that lower α does usually means better average total regret but also means that we could have committed to a policy too early (which explains the lower average fraction of incorrect doses). As seen in figure 3, a lower α not only lead to sub-optimal decisions (like $\alpha = 0.0$) but even when we eventually achieve optimal parameters, it committed to a strategy too early and thus it takes more time to rectify that mistake since we are exploring so little. Thus, finding the correct α is a balance between the 2. The best alpha found is 0.612 which achieved an average total regret of 51.45 and fraction of incorrect cases at 0.3692. These results comparing to the baseline results will be shown later.

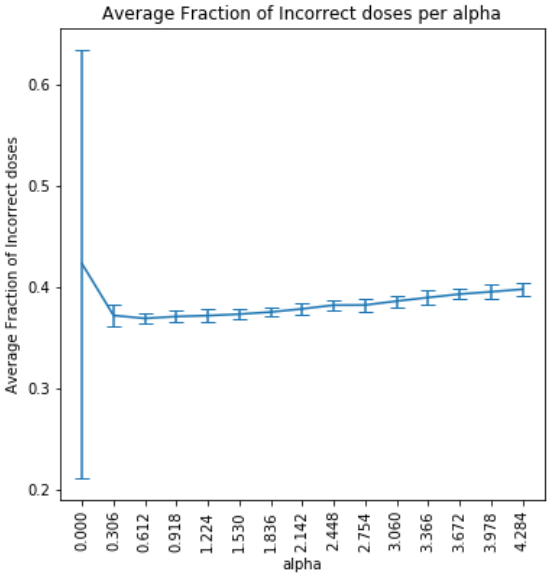

Figure 1. Average fraction of incorrect doses for each α value. There are 20 trials run for each alpha value. Fraction of incorrect cases are plotted with 95% confidence interval

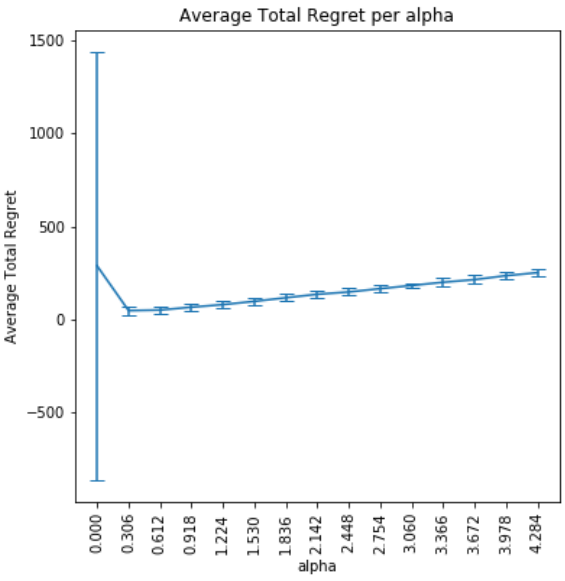

Figure 2. Average total regret for each α value. There are 20 trials run for each alpha value. Averages of total regret are plotted with 95% confidence interval

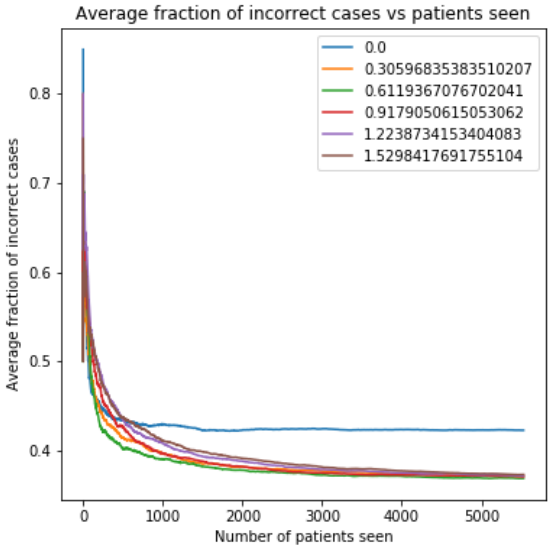

Figure 3. Average fraction incorrect cases as the algorithm sees more patients. Each α value simulation is run 20 times and the fraction of incorrect dosages is averaged across these 20 runs at each "time stamp". 95% confidence interval not displayed to minimize clutter

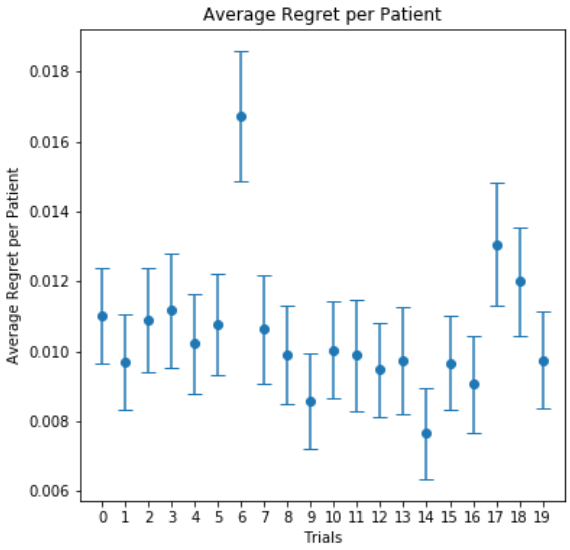

Figure 4. Expected regret per patient across 20 trials for $\alpha = 0.612$. Plotted with 95% confidence interval with t-statistic

4.1.2. Robustness

In order to verify that the model is robust to the permutation of patients, we have run all LinUCB experiments for 20 trials with different permutations of patients. As we can see in figure 4 and 5, we achieved fairly stable average regret per patient of around 0.009 (average total regret of 51.45) and fraction of incorrect dosages around 0.37. This not only mean that our results are statistically sound, it also means that our model is very robust across different permutation of patients judging by the stable results.

4.2. Supervised Methods

4.2.1. Upper limit of performance

For supervised methods, we used 2 different kind of supervised learning model classes, namely SVM and Multinomial Logistic Regression (aka Softmax Regression). Since the algorithm can receive the correct Warfarin dosage as part of its training and not just rewards of correct/incorrect dosages, these algorithms should in theory do much better than the RL-based methods. Indeed, they do very well comparing to the results shown above. However, we are first interested in seeing the potential optimal of different model classes in order to chose the best one. These upper limits are derived by fitting the model on the entire dataset with correct labels,

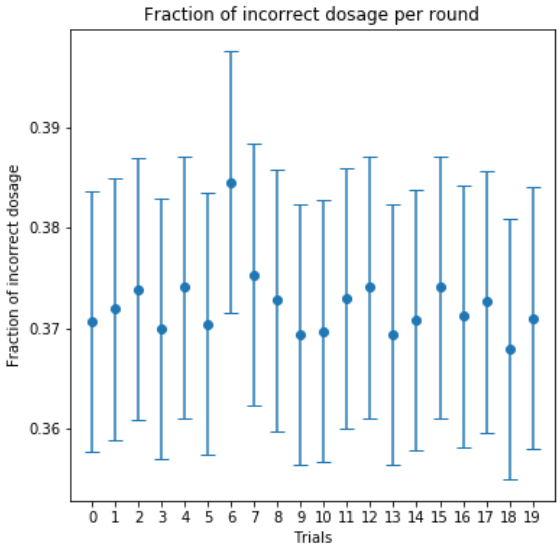

Figure 5. Expected value of incorrectness of dosage across 20 trials $\alpha = 0.612$. Plotted with 95% confidence interval with t-statistic

Table 1. Table of Model Class and their optimal regret and Opitmal fraction of incorrect doses. Obtained by training the model on the entire dataset and evaluating it on the entire dataset

MODEL CLASS	OPTIMAL REGRET	OPTIMAL FRACTION
LOGISTIC	**30.93**	**0.3538**
SVM	61.53	0.3882

Table 2. Table of comparison across methods

METHOD	AVG TOTAL REGRET	AVG FRACTION
FIXED DOSE	61.53	0.3882
CLINICAL DOSING	54.227	**0.3572**
LINUCB	51.45	0.3692
LOGISTIC	**33.49**	0.3598

and evaluating them on the same dataset and observe the regret and fraction of incorrect Warfarin doses. In tabel 1 we can clearly see that logistic regression does so much better in terms of both optimal regret as well as optimal fraction of incorrect warfarin doses. We suspect the reason being that logistic regression is a more powerful model in this specific case because the correct dosage does seem to have a linear relationship with the features.

4.2.2. SUPERVISED RESULTS

We have also run this model several times and we found that with a batch size of 50, the supervised model described in section 3.4 achieve a total regret of 33.49 and a fraction of incorrect cases 0.3598. Interestingly this does beat the clinical algorithm in terms of regret, but it does not beat the clinical algorithms in terms of total incorrect cases. In terms of regret, since the supervised model is linear, it is able to find better parameters for each dosage category than a linear regression predictor such as the Clinical Dosing Algorithm. However, as shown in section 4.2.1 and table 1, it is not a limitation of the model but rather a limitation of the batched method. The agent can only see 50 patients at a time, and only after certain amount of patients does it have a good estimate for each of the parameters.

4.3. Comparison Across All Methods

Here we compare results across all methods. They are summarized in table 2. As we can see from the table, all the new methods tried are able to beat the Fixed Dose baseline method in terms of average total regret and average fraction of incorrect doses. The supervised logistic regression method with a batch size of 50 achieved the least regret. However, it is still not able to beat the Clinical Dosing algorithm in terms of average fraction of incorrect dosages. Still, as shown in section 4.2.2, this is not a limit of the model class but rather a limit to the problem setting where patients come one at a time while Clinical Dosing algorithm does not have this "exploration period". The LinUCB Disjoint model is able to beat both baselines in terms of average total regret, but it was not able to achieve better fractions of incorrect dosages due to its exploration period.

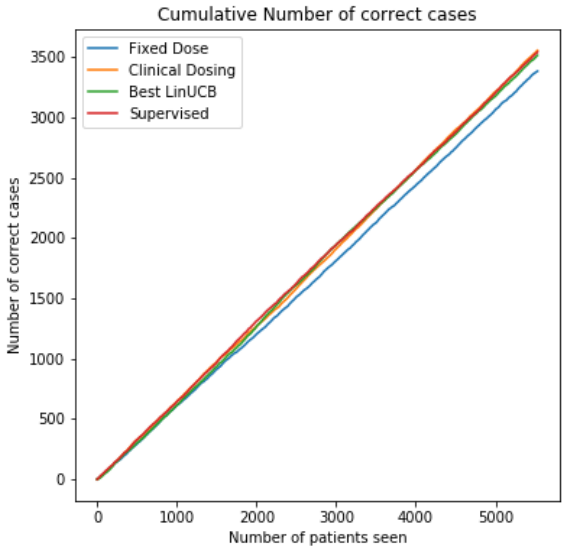

Figure 6. A comparison of algorithms and their trajectory in terms of number of correct dosage predictions. We picked the best performing run of each algorithm in composing this figure

4.3.1. PERFORMANCE ANALYSIS

A better way of understanding the performance of these algorithms is to see the types of decisions they make for each patient as it learns more about the patients. Plotted in figure 6 is a comparison of algorithms in terms of cumulative number of correct decisions made by the algorithm as it sees more patients and gain a better understanding of the patients. These are plotted by selecting the run with the highest rewards in order to compare their maximum performances. As we can see from the plot, the fixed dose method performed very poorly and the number of correct cases grows linearly with time. The Supervised learning method with Logistic regression had the most consistent trajectory across all algorithms. As you can see from the Green line (the best run of a LinUCB algorithm), it spends some time initially performing below the Clinical Dosing and Supervised algorithms because it is exploring rewards for different dosages. Notably, it performs worse than fixed

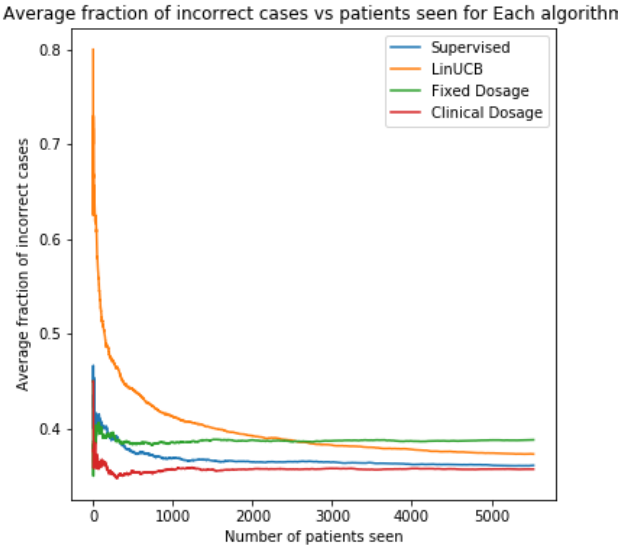

Figure 7. A comparison of algorithms and the fraction of correct cases as they start seeing more patients. Each algorithm is run 20 times and the fraction of incorrect dosages is averaged across these 20 runs at each "time stamp". 95% confidence interval not displayed to minimize clutter

dosages in the beginning. Then, after finding good parameters, it quickly caught up to other algorithms and sometimes surpassing them. This is what we expected to see in this graph.

Another way of showing this is to display the average fraction of incorrect cases as we see more patients across 20 different permutation of patients much like those in section 4.1.1. As we can see from figure 7, clinical dosage start correctly predicting cases almost immediately due to the fact that it is a fixed policy with good parameters. Fixed Dosage also perform very well in the beginning because they have no exploration period. This also means that their performance is fixed throughout the entire run. On the other hand, LinUCB algorithm had an initial period of exploration (which is why it is doing poorly) and eventually the fraction of incorrect cases drops after it has a good estimate of the parameters. However, the initial period of exploration means that we will have a lot of failed cases in the beginning, and that inflates the number of total incorrect cases. The supervised learning approach also did fairly well. After an initial period of uncertainty, it quickly converged to very good parameters, but still fell short in beating the Clinical Dosage baseline.

5. Conclusion

Both the LinUCB method and the supervised methods performed well comparing to very strong baselines. Both methods beat both baselines in terms of average total regret, but both fell short in beating the clinical dosing algorithm in terms of average fraction of incorrect cases. Due to the fact that these algorithms require a short exploration period before being able to converge to good parameters, while clinical dosage algorithms is able to make correct predictions right away. Still it did do sufficiently well in approximating the clinical dosage baseline. Analysis showed that for short horizon problems like these, it is better to chose a smaller α value so that we commit to a strategy early instead of exploring more in order to find a better policy.

6. Team Contributions

I myself completed this project by myself with extra support from Haojun Li (myself)

References

Abe, N., Biermann, A. W., and Long, P. M. Reinforcement learning with immediate rewards and linear hypotheses. *Algorithmica*, 37(4):263–293, 2003.

Agrawal, S. and Goyal, N. Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning*, pp. 127–135, 2013.

Auer, P. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

Cai, H., Ren, K., Zhang, W., Malialis, K., Wang, J., Yu, Y., and Guo, D. Real-time bidding by reinforcement learning in display advertising. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pp. 661–670, 2017.

Codevilla, F., Miiller, M., López, A., Koltun, V., and Dosovitskiy, A. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–9. IEEE, 2018.

Consortium, I. W. P. Estimation of the warfarin dose with clinical and pharmacogenetic data. *New England Journal of Medicine*, 360(8):753–764, 2009.

Dani, V., Hayes, T. P., and Kakade, S. M. Stochastic linear optimization under bandit feedback. 2008.

Lai, T. L. and Robbins, H. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1): 4–22, 1985.

Lattimore, T. and Szepesvári, C. Bandit algorithms. *preprint*, pp. 28, 2018.

Li, L., Chu, W., Langford, J., and Schapire, R. E. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pp. 661–670, 2010.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Peters, J., Vijayakumar, S., and Schaal, S. Reinforcement learning for humanoid robotics. In *Proceedings of the third IEEE-RAS international conference on humanoid robots*, pp. 1–20, 2003.