Online Machine Learning for Solving a Sequence of Linear Systems

Mikhail Khodak, Edmond Chow, Maria-Florina Balcan, Ameet Talwalkar

Abstract

Machine learning is often presented as an alternative to well-established and effective numerical methods. In this work, we present an example where machine learning is used to augment existing numerical methods.

Consider solving a sequence of linear systems

$$A_t x = f_t, \quad t = 1, \ldots, T$$

with SOR(ω), or some other preconditioner–solver combination in general, where we need to choose a parameter for the preconditioner or solver for each system. We are to solve each system before the next system is presented to us. Our goal is to choose the SOR parameter ω for each system to minimize the total number of iterations. To accomplish this, we can make use of the information about the number of iterations used to solve previous systems.

There must be some assumptions for us to do anything interesting. We could assume, for example, that the sequence of matrices $\{A_t\}$ changes slowly. This type of assumption could be useful if we are further allowed to use a method to obtain a good estimate of ω, when needed. Then, we could use this value of ω for solving several linear systems until the number of iterations required for a system becomes so large that it becomes profitable to estimate a new value of ω. This kind of strategy has appeared in various guises in the literature and is perhaps the best competitor strategy to what we will present here.

In this work, we consider using multi-armed bandits from online machine learning to select the value of ω for solving each system. Such algorithms are very effective for the following class of practical problems. Suppose every time a user visits your web page, you have the choice of showing an advertisement in one of four locations: top, bottom, left, and right. You wish to choose the location each time to maximize the total number of times users visiting your web page will click on the advertisement. The underlying assumption is that there is an unknown probability that a user will click on the advertisement in each of the four cases. Your problem is to discover the case that has the highest such probability (exploration), while also trying to maximize the number of clicks (exploitation) by not wasting time on low probability cases, and possibly not knowing the number of users your web page will ultimately have. Formally, the multi-armed bandit problem is the following:

Multi-armed bandit problem

for $t = 1, \ldots, T$ **do**
  Choose and perform *action* a_t from $\{1, \ldots, d\}$
  Receive *reward* (or *loss*) y_t
    Different actions lead to different rewards.
    Do not see rewards for actions not taken.
end for

The actions are choosing among the four locations where we can place the advertisement. For our sequence of linear systems, the actions are choosing a (discretized) value of ω. The goal is to choose the actions such that the *cumulative regret* is minimized. The cumulative regret is the difference between the expected reward for the single best action and the expected reward for our choice of

actions, summed over t rounds. In particular, the goal is to obtain strategies that give cumulative regret that is sublinear in t.

We address two types of assumptions about our sequence of linear systems: (1) the optimal ω follows a fixed distribution and (2) the optimal ω follows a distribution that changes. Case (1) can be handled with stochastic bandits such as UCB1, an upper confidence bound algorithm. Case (2) can be handled with adversarial bandits such as Exp3, the exponential-weight algorithm for exploration and exploitation. We further look at sequences of matrices of the form $A_t = A + c_t I$ where the scalar shift c_t is known before ω is chosen. This case can be handled by contextual bandits.

The simplest contextual bandit algorithm will discretize the contexts (shifts) into intervals and use an adversarial bandit separately on each interval. However, we want an approach that exploits the smoothness of the optimal mapping from the context (shift c) to the action (ω). For this, we reduce the online contextual bandit problem to a problem of online regression to finding a weight vector w given observations that arrive in sequence:

Online regression protocol for $y = f(x; w)$

Initialize regression weights w
for $t = 1, \ldots, T$ **do**
  Observe x_t
  Predict $\hat{y}_t = f(x_t; w)$
  Observe y_t and suffer loss $(\hat{y}_t - y_t)^2$
  Update w
end for

In online regression, the goal is to choose the weights to minimize the cumulative loss. For our contextual bandit, we assume we have a good method for solving this problem (the oracle). In particular, in our contextual bandit, we use online regression to fit the loss vs. (context, action), i.e., y = number of iterations vs. $x = (c, \omega)$.

An example of such an approach is the SquareCB algorithm (Foster and Rakhlin, 2020):

SquareCB algorithm

Input: learning rate $\eta > 0$, exploration parameter $\mu > 0$
for $t = 1, \ldots, T$ **do**
  Observe context c_t
  Compute $\hat{y}_{t,a} = f(c_t, a; w)$ for all possible a
  $b_t = \arg\min_a \hat{y}_{t,a}$
  $p_{t,a} = \frac{1}{\mu + \eta(\hat{y}_{t,a} - \hat{y}_{t,b_t})}, \quad \forall a \neq b_t$
  $p_{t,b_t} = 1 - \sum_{a \neq b_t} p_{t,a}$
  Sample $a_t \sim p_t$ and perform action a_t
  Observe actual loss y_t
  Update the online regression oracle with example $((c_t, a_t), y_t)$
end for

Above, the action a can be associated with possible values of ω for our setting of solving a sequence of linear systems.

We develop a contextual bandit called ChebCB, a contextual bandit using Chebyshev regression. For each action (possible ω) separately, we fit the loss vs. context c using regularized polynomial regression. In particular, we use polynomials in a Chebyshev basis with coefficients for each

Chebyshev polynomial constrained to be small.

We do not show the results here, but tests on a 2-D heat equation with time-dependent coefficients and time-dependent forcing show that the ChebCB contextual bandit method asymptotically achieves the performance of the instance-optimal policy, which selects the best ω for each instance.

In summary, this work shows the potential of using well-understood learning algorithms to augment and speed up linear system solvers, without sacrificing the ability to obtain high accuracy. Additional information can be found in the reference below.

[1] M. Khodak, E. Chow, M.-F. Balcan, and A. Talwalkar, Learning to Relax: Setting Solver Parameters Across a Sequence of Linear System Instances, Proceedings of the 12th International Conference on Learning Representations (ICLR), 2024. Spotlight. https://arxiv.org/abs/2310.02246