Anticipating Concept Drift in Online Learning[*]

Michał Dereziński
Computer Science Department
University of California, Santa Cruz
CA 95064, U.S.A.
mderezin@soe.ucsc.edu

Badri Narayan Bhaskar
Yahoo Labs
701 1st Ave, Sunnyvale
CA 94089, U.S.A.
bbhaskar@yahoo-inc.com

Abstract

Adapting to changing environment is a key challenge in online learning. Many state of the art algorithms have been shown to incur losses that are bounded using the variability of the learned concept over time. However, what if this concept drift can be anticipated? We propose online optimization algorithms that use the sequence of past predictive models to estimate this drift, allowing them to converge to a moving comparator which are supported by theory and simulations.

1 Introduction

Online convex programming provides an effective model for adaptive online prediction [BT03, Zin03, NY83, CBL06]. The notions of tracking and adaptive regret are often employed to evaluate algorithms in changing environments [HS09, LW94, AKCV12, HW01, HW98, BW02]. However, most of the approaches focus on adapting to the changes after they occur, so the bounds have to contain some penalty that is proportional to the variability of the underlying concept. If those changes follow some predictable pattern or trajectory, we should be able to anticipate them, thus avoiding this penalty [HW13]. We propose to introduce dynamic concept drift estimation into online optimization algorithms by modeling the evolution of the concept, which can be viewed as combining filtering [BC09, XSdS94, TS96, KWH06] with optimization. In the following sections, we present theoretical background for analyzing online optimization algorithms capable of learning and anticipating concept drift [GvB$^+$14]. We also propose several algorithms supported by partial theoretical results and simulations, including a simple extension of gradient descent which exhibits exact convergence to a moving optimum under certain assumptions.

2 Background

We consider an online learning setup, where at each time step t algorithm proposes a model $\widehat{\theta}_t \in \Theta$, then receives loss $f_t(\widehat{\theta}_t)$. The goal is to minimize the cumulative loss over all examples until some time T. To evaluate the algorithm for a given experiment, it is useful to consider a comparator sequence of models $\boldsymbol{\theta} = \{\theta_t\}_{t=1}^T$, which would represent the *true* concept drift. From this, we obtain the regret incurred by the algorithm:

$$R_T(\boldsymbol{\theta}) := \sum_{t=1}^T f_t(\widehat{\theta}_t) - \sum_{t=1}^T f_t(\theta_t).$$

The classical Mirror Descent (MD) algorithm, and its special case, Gradient Descent (GD), use the gradient of the most recent loss function f_t to update the model $\widehat{\theta}_t$ so that it better fits the corresponding example [BT03, NY83]. For convex losses, the tracking regret of MD can be bounded by

[*]This research was supported by the NSF grant IIS-1118028.

$R_T(\boldsymbol{\theta}) \leq O(\sqrt{T}(1+V(\boldsymbol{\theta})))$, where $V(\boldsymbol{\theta}) = \sum_{t=1}^{T} \|\theta_{t+1} - \theta_t\|$ is the variability of the comparator sequence, which effectively measures the amount of concept drift. The $O(\cdot)$ notation hides some constants describing the convexity of losses and properties of the parameter space Θ. The $\sqrt{T}$ term means that, without concept drift, the regret is sublinear with respect to T, so the model has to converge to an optimal solution. On the other hand, drift variability $V(\boldsymbol{\theta})$ may often grow linearly, in which case this algorithm will no longer exhibit convergence.

Suppose that we believe the concept drift follows a dynamical model $\Phi_t : \Theta \to \Theta$ that comes from some continuous family $\mathcal{D}$. A sequence $\boldsymbol{\theta}$ follows drift $\Phi \in \mathcal{D}$ if for all t we have $\theta_{t+1} = \Phi_t(\theta_t)$. For example, $\mathcal{D}$ may consist of all linear drift models, i.e., each Φ corresponds to a vector v, so that $\Phi_t(\theta) = \theta + v$. For any comparator $\boldsymbol{\theta}$, we can measure its cumulative average deviation from a drift model Φ by generalizing the notion of variability

$$V_{\Phi}^{\langle k \rangle}(\boldsymbol{\theta}) = \sum_{t=k+1}^{T} \left\| \frac{1}{k} \sum_{i=t-k}^{t-1} (\theta_{i+1} - \Phi_i(\theta_i)) \right\|$$

We use a sliding window of width k to average out any independent noise, measuring only the persistent variability, which will be useful in further analysis. Similarly, a sequence $\boldsymbol{\theta}$ can be described with respect to a drift family $\mathcal{D}$ by looking at the drift model $\Phi^* \in \mathcal{D}$ that minimizes $V_{\Phi^*}(\boldsymbol{\theta}) := V_{\Phi^*}^{\langle 1 \rangle}(\boldsymbol{\theta})$. Can we achieve regret $O(\sqrt{T}(1 + V_{\Phi^*}(\boldsymbol{\theta})))$?

3 Proposed Algorithms

An online optimization system capable of sublinear regret in the presence of concept drift needs to incorporate a component responsible for selecting a good estimate of the drift, $\widehat{\Phi}_t \in \mathcal{D}$, which is updated at each time-step. Recently, Dynamic Mirror Descent (DMD) [HW13] algorithm has been proposed as a way to incorporate a predetermined drift dynamic into the optimization procedure by alternating the gradient step and the drift step. Instead of fixing the drift from the start, we propose to learn it using the sequence of models $\widehat{\theta}_t$ as a proxy for how the *true* concept evolves, as if treating $\widehat{\theta}_t$ as a noisy measurement of θ_t in an online filtering task. The key difference between this task and the techniques used in Kalman filters [WB95] is that in optimization we don't have any observed variable that describes the latent state, but rather the algorithm must query the domain and generate the estimates itself by relying on the loss function. Naturally, those are only reasonable when the optimization has already converged to some degree, so we might not want to apply the drift estimate throughout the process, but only when we expect it to be beneficial. We will use linear drift family as a test case for analysis. A natural way to estimate it is by taking the average of last k update steps:

$$\widehat{\Phi}_t(\theta; \widehat{\boldsymbol{\theta}}_{t-k}^{t}) = \theta + \frac{1}{k} \sum_{i=t-k}^{t-1} (\widehat{\theta}_{i+1} - \widehat{\theta}_i) = \theta + \frac{1}{k}(\widehat{\theta}_t - \widehat{\theta}_{t-k}). \tag{1}$$

We can now introduce this to the DMD algorithm, obtaining as a special case the algorithm we call Average Momentum Gradient Descent (AMGD):

$$\widehat{\theta}_{t+1} = \widehat{\theta}_t - \eta_t \nabla f_t(\widehat{\theta}_t) + \frac{1}{k}(\widehat{\theta}_t - \widehat{\theta}_{t-k}).$$

Adding the drift term indiscriminately to every update is dangerous from a practical stand point, because it can lead to instability. However, AMGD is a good starting point for convergence analysis, and even in this case some interesting guarantees appear to be obtainable.

A common approach for ensuring stability of an optimization algorithm is through an ensemble method. We propose a simple *Two Track* algorithm, with two sequences of models updated separately. The first sequence $\widehat{\boldsymbol{\theta}}^{(1)}$ is regular Mirror Descent (without the drift step). The second one $\widehat{\boldsymbol{\theta}}^{(2)}$ uses Dynamic Mirror Descent with a drift step based on an estimated model $\widehat{\Phi}_t$. The final prediction at each time-step is a weighted average of the two tracks:

$$\widehat{\theta}_t = (1 - w_t)\, \widehat{\theta}_t^{(1)} + w_t\, \widehat{\theta}_t^{(2)}.$$

Here, $w_t \in [0, 1]$ represents our confidence in the drift estimation. To update this weight, we can use one of the standard methods applied in the ensemble setting, like the Fixed Share Forecaster (FSF)

[CBL06]. Notice, that with this algorithm we have more options for estimating the drift. Namely, we can use either of the two tracks, or the final mixed prediction, as the proxy for the concept drift. This choice has a significant impact on the behavior of the algorithm, which will be discussed later.

4 Analysis

In this section, we propose some techniques for analyzing the effectiveness of drift estimation. We focus on the linear drift, that is, when family $\mathcal{D}$ represents all linear trajectories (along a fixed direction), but we believe the results can be generalized. The model updates depend on the loss function and determine the rate of convergence. For instance, for gradient descent with loss function $f_t(\theta) = \frac{1}{2}\|\theta - \theta_t\|_2^2$, the update takes the form $G_t^{\boldsymbol{\theta}}(\widehat{\theta}_t) = \alpha(\theta_t - \widehat{\theta}_t)$ which leads to a linear convergence in the absence of any concept drift. We use the function $G_t^{\boldsymbol{\theta}}$ as a reference for the convergence behavior in our analysis, but our experiments show that we can substantially relax the conditions on the model updates and perhaps extend our theoretical argument. If a comparator is drifting along a specific linear trajectory, then a prediction sequence converging to it will follow a very similar trajectory once it gets close enough. This suggests a natural strategy for the Two Track algorithm, where initially we put weight on the no-drift track, then we gradually switch to the drift-enhanced predictions. In fact, with the right weight update scheme we can be almost as good as the best instantiation of this strategy. The FSF ensemble method is shown to achieve regret $O(\sqrt{T}(1 + \min_t\{V(\boldsymbol{\theta}_1^t) + V_{\widehat{\Phi}}(\boldsymbol{\theta}_{t+1}^T)\}))$. To understand the performance of drift estimation in this scenario, we show the following bound for arbitrary sequences $\boldsymbol{\theta}$ and $\widehat{\boldsymbol{\theta}}$:

$$\min_t\{V(\boldsymbol{\theta}_1^t) + V_{\widehat{\Phi}}(\boldsymbol{\theta}_{t+1}^T)\} \le \frac{d}{\alpha}\log\frac{M}{d} + V_{\Phi^*}(\boldsymbol{\theta}) + \frac{2+\alpha}{\alpha}V_{\Phi^*}^{\langle k\rangle}(\boldsymbol{\theta}) + \frac{2}{\alpha}V_{G^{\boldsymbol{\theta}}}^{\langle k\rangle}(\widehat{\boldsymbol{\theta}}), \tag{2}$$

where $M = \max_{\theta,\theta'\in\Theta}\|\theta - \theta'\|$ and $d = \|\Phi^*(\theta) - \theta\|$ is the speed of the optimal linear drift $\Phi^* \in \mathcal{D}$ for comparator $\boldsymbol{\theta}$. This bound holds for estimation $\widehat{\Phi}$ defined in Equation (1). First term is responsible for the drift of the initial sequence $\boldsymbol{\theta}_1^t$, the next two depend only on the comparator's deviation from the best matching drift, while the last one controls the convergence behavior of $\widehat{\boldsymbol{\theta}}$ (that is, if $\widehat{\boldsymbol{\theta}}$ follows update $G_t^{\boldsymbol{\theta}}$, this term goes away). Note, that increasing k improves the bound by averaging out the deviations, but it also requires maintaining a larger estimation window.

The inequality (2) unfortunately does not lead to a proper adversarial regret bound, because of the last term's dependence on the sequence $\widehat{\boldsymbol{\theta}}$. It also does not capture an intriguing self-reinforcing effect of the drift estimation, that can be observed experimentally. Namely, the accuracy of drift estimation depends on the distance from the comparator. Adding drift to the predictions is expected to reduce that distance. If we then use those new $\widehat{\theta}_t$'s to estimate the drift, the feedback loop can result in convergence to a moving comparator. By using stability theory of complex polynomials [Cho11] we show the convergence of an idealized version of the AMGD algorithm. Here the updates exactly follow the linear convergence step $G_t^{\boldsymbol{\theta}}$:

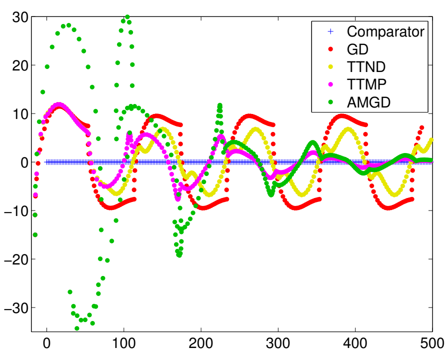
$$\widehat{\theta}_{t+1} = \widehat{\theta}_t + \alpha(\theta_t - \widehat{\theta}_t) + \frac{1}{k}(\widehat{\theta}_t - \widehat{\theta}_{t-k}). \tag{3}$$

Theorem 1 *For $k \ge 2$ and $\alpha \in (0, 1)$, sequence $\widehat{\theta}_t$ defined by (3) converges to any comparator of the form $\theta_t = \theta_0 + t \cdot v$.*

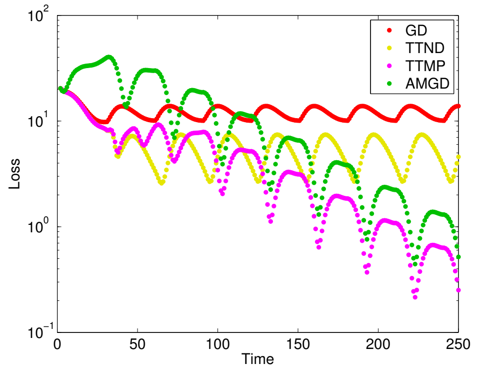
We conjecture that this result extends to the AMGD algorithm for strongly convex and Lipschitz smooth loss functions, thereby generalizing classical Gradient Descent convergence guarantees.

5 Experiments

To evaluate and compare the proposed algorithms, we ran various simulations. Our goal was to establish the best way of introducing drift estimation into online optimization, so instead of focusing on a specific optimization step (like Gradient Descent), we looked at various types of static convergence behavior. That is, similarly as in Section 4, we look at a convergence function $G_t^{\boldsymbol{\theta}} : \Theta \to \Theta$, which at time-step t can transform a point $\widehat{\theta}$ into some $G_t^{\boldsymbol{\theta}}(\widehat{\theta})$ in such a way, that would guarantee



(a) Trajectories



(b) Losses

Figure 1: As seen on plot (a), the comparator is moving across space $\Theta = \mathbb{R}^2$ along the X axis from left to right. The losses (b) show that algorithms AMGD and TTMP achieve linear convergence.

convergence to a static optimum. However, in this case we will allow any sequence of functions satisfying static linear convergence: $\|G_t(\widehat{\theta}) - \theta_t\| \leq (1-\alpha)\|\widehat{\theta} - \theta_t\|$ for all $\widehat{\theta} \in \Theta$ and all $t \in [1, T]$. With this constraint in mind, we can now treat function $G_t(\cdot)$ as an adversary, aiming to hinder the drift estimation. In the simulations, we look at the following approaches: a simple single track algorithm corresponding to AMGD (where G_t replaces the gradient step), and the Two Track algorithm as described in Section 3, with two variants of drift estimation. The first variant uses the no-drift track $\widehat{\boldsymbol{\theta}}^{(1)}$ for estimation (TTND), while the second one uses the final mixed prediction for that purpose (TTMP). Notice that using the first track for estimation means there is no feedback loop effect in estimation. Multiple values of parameter k were tested, but the results shown used $k = 13$. As a baseline, we use the optimization step $G_t(\widehat{\theta}_t)$ by itself (denoted GD).

Multiple simulations were ran to find the most adversarial function sequences G_t. Intuition suggests that a large angle between the update vector $G_t(\widehat{\theta}_t) - \widehat{\theta}_t$ and the ideal convergence step $\alpha(\theta_t - \widehat{\theta}_t)$ would affect the trajectory of predictions, thereby corrupting the drift estimation. Simulations support this claim. We used two-dimensional model space in the experiments so that it is possible to plot the trajectories, as seen on Figure 1a. The comparator is moving from left to right at the speed of 2 units per time step. To properly read the trajectories we have to keep in mind the hidden time dimension. The losses of the algorithms (loss is measured as the distance from the comparator) at each time step are shown on Figure 1b. The adversarial function G_t was chosen so that it maximizes the angle of the step from the direction of the comparator, while still preserving a prespecified static convergence rate. Every 30 time steps the angle is flipped symmetrically (observe the GD trajectory for a better intuition). Other adversarial sequences were also tested. Observe that all three drift estimation approaches provide a significant benefit compared to GD. Looking at the right end of the trajectories we see that only GD lags behind the comparator horizontally. However, only TTMP and AMGD achieve convergence, whereas TTND keeps oscillating at a fixed distance. The convergence is only possible with the drift feedback loop, because otherwise the estimation remains noisy. However, the feedback loop introduces a possibility of "overshooting" the comparator, which leads to instability, as seen in the early trajectory of AMGD. In fact, for a small enough value of k, instead of converging, this algorithm may (under adversarial conditions) fall into a diverging spiral. This can be avoided by selecting a sufficiently wide estimation window. This issue is also alleviated in TTMP, because even if it does not converge, as an ensemble method it will not do much worse than GD (and in fact, it did better in all the simulations).

It should be noted that the linear drift family is just a test case. Analyzing estimation of other families of smooth drift trajectories is an interesting future research direction. Moreover, in many practical tasks each example is labeled with a precise time stamp, which can be used instead of simple step enumeration. In some learning problems, a batch learning solution can be used, which includes time stamp as a feature. A better understanding is needed of when this is more appropriate than online approaches. Similar drift estimation techniques could also be applied to optimization tasks where the loss gradient is not available [BB04] and batch learning is impossible.

References

[AKCV12] Dmitry Adamskiy, Wouter M. Koolen, Alexey Chernov, and Vladimir Vovk. A closer look at adaptive regret. In *Proceedings of the 23rd International Conference on Algorithmic Learning Theory*, ALT'12, pages 290–304, Berlin, Heidelberg, 2012. Springer-Verlag.

[BB04] Tim Blackwell and Jrgen Branke. Multi-swarm optimization in dynamic environments. In *Applications of Evolutionary Computing*, volume 3005 of *Lecture Notes in Computer Science*, pages 489–500. Springer Berlin Heidelberg, 2004.

[BC09] A. Bain and D. Crisan. *Fundamentals of stochastic filtering*. Springer, 2009.

[BT03] Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Oper. Res. Lett.*, 31(3):167–175, May 2003.

[BW02] O. Bousquet and M. K. Warmuth. Tracking a small set of experts by mixing past posteriors. *Journal of Machine Learning Research*, 3:363–396, 2002.

[CBL06] Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA, 2006.

[Cho11] Younseok Choo. An elementary proof of the jury test for real polynomials. *Automatica*, 47(1):249–252, January 2011.

[GvB$^+$14] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, March 2014.

[HS09] Elad Hazan and C. Seshadhri. Efficient learning algorithms for changing environments. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 393–400, New York, NY, USA, 2009. ACM.

[HW98] Mark Herbster and Manfred K. Warmuth. Tracking the best expert. *Mach. Learn.*, 32(2):151–178, August 1998.

[HW01] Mark Herbster and Manfred K. Warmuth. Tracking the best linear predictor. *J. Mach. Learn. Res.*, 1:281–309, September 2001.

[HW13] E. Hall and R. Willett. Dynamical models and tracking regret in online convex programming. In *Proceedings of The 30th International Conference on Machine Learning*, pages 579–587, 2013.

[KWH06] J. Kivinen, M.K. Warmuth, and B. Hassibi. The p-norm generalization of the lms algorithm for adaptive filtering. *Trans. Sig. Proc.*, 54(5):1782–1793, October 2006.

[LW94] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Inf. Comput.*, 108(2):212–261, February 1994.

[NY83] A. Nemirovski and D. B. Yudin. *Problem complexity and method efficiency in optimization*. Wiley-Interscience series in discrete mathematics. Wiley, Chichester, New York, 1983. A Wiley-Interscience publication.

[TS96] Yahali Theodor and Uri Shaked. Robust discrete-time minimum-variance filtering. *IEEE Transactions on Signal Processing*, 44(2):181–189, 1996.

[WB95] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA, 1995.

[XSdS94] Lihua Xie, Yeng Chai Soh, and Carlos E. de Souza. Robust kalman filtering for uncertain discrete-time systems. *Automatic Control, IEEE Transactions on*, 39(6):1310–1314, Jun 1994.

[Zin03] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proc. Int. Conf. Mach. Learning (ICML)*, pages 928–936, 2003.