Adaptive sampling and filtering of policy gradients

Anonymous Author(s)
Affiliation
Address
email

Abstract

This paper discusses the use of a Kalman filter to estimate the gradient and its expected error in an online fashion. The error estimate is used to inform the choice of batch size in policy gradient methods. The Kalman filter provides a natural means to determine the estimation error in a dynamical system but also makes various limiting assumptions about the problem structure. This work outlines when the Kalman filter provides an appropriate estimate of the policy gradient and its error and applies the result to simple control scenarios. Our approach consistently performs as well an optimally-tuned fixed batch size in terms of policy performance. When the batch size is not well-tuned our method also provides substantial savings in terms of sample complexity.

1 Introduction

Reinforcement learning (RL) approaches have recently made great strides in applications to control problems [7, 11, 12, 21]. In general, the applications of these ideas to real-world systems is limited by the poor *sample complexity* of the algorithms as such scenarios have additional associated costs for every trajectory executed on a robot or cyber-physical system. These costs, for example, could be a time cost for a human to monitor the system or a monetary cost in terms of electricity, wear-and-tear, or maintenance. Accordingly, improving the sample complexity of RL approaches is of great research interest. In this work, we focus on policy gradient methods, which have become increasingly popular in the last few years as gradient-based approaches have become the go-to optimization method with the development of neural network techniques. Significant effort has focused on reducing the variance of the gradient estimate of the likelihood ratio estimator or using more powerful optimization techniques. This is usually done by incorporating a baseline [20], introducing bias [13, 3], or using 2nd-order gradient information [12, 11, 21] and has yielded significant improvement in results. In contrast to existing work, we focus on a much neglected facet of policy gradient methods: how many samples are needed to approximate the gradient accurately? Most methods gather a batch of fixed size every iteration and this decision significantly affects the performance of the algorithm [4, 21]. Batch policy gradient methods make the assumption that the same number of sample are required to accurately estimate the gradient and improve the policy at each stage in training. If this is not the case, and fewer sample are, in fact, needed at any given iteration, then wasteful samples are drawn from the system. As an alternative to using a fixed sample size, we discuss an approach to determining online an appropriate number of samples to perform an update step with the goal of minimizing the total number of time steps of interaction with the environment. The main contribution of this work is an adaptive batch size algorithm. We propose to estimate the expected error in the gradient in an online fashion and use the expected error as a proxy to determine when we have sufficient confidence to perform a policy update. We find that adaptive batch sizes based on the Kalman filter expected error give as good performance as a well-tuned fixed batch size and significantly outperform large batch sizes in terms of sample complexity.

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

2 Motivation

Imagine you are presented with a simple convex objective function, $y = f(x)$. If this function is smooth and one has access to its derivative, $g = \frac{\partial f}{\partial x}$ any point x, the optimization becomes straight forward and in some cases the minimum can be found within a single step. Now consider the case where one only has access to noisy observations of the gradient, $\hat{g} = g + \epsilon$, where ϵ is some additive noise on the gradient. In practice, this noise may come from noise in the data or noise in a dynamical system. In this case, a single gradient observation may not be sufficient to make progress towards the minimum. Add to this uncertainty the inability to evaluate the objective exactly; instead one receives noisy observations of the objective as well. In turn, $\hat{y} = f(x) + \zeta$, where ζ is additive noise on the objective. This is the general challenge that RL approaches face. The gradient is noisily approximated by a form similar to finite-differences and the objective, defined as the expected return, $\mathbb{E}[G_\pi]$, under the current policy, π, must also be estimated by sampling under the policy. Given the considerable noise in the RL case, it is appealing to augment the standard gradient descent approach with a decision making component that aids in determining when we have confidently estimated the gradient. Figure 1 gives some intuition behind these ideas. The dashed line represents the true gradient of the function $y = \frac{1}{2}x^2$ at $x = 1$. Assuming standard Gaussian noise on the gradient, the orange curve represents the distribution of observed gradients.

3 Online estimation of the gradient and the gradient error

We assume access to noisy observations of the gradient over time and are tasked with estimating the true gradient. This is a natural fit for a recursive filtering approach. The goal is to estimate the true gradient, $g_{\text{true}} = \frac{\partial f}{\delta \theta}$ given noisy estimates of that quantity, $\hat{g}_i \sim g + \mathcal{N}(0, R)$, which we assume, for now, to be normally distributed around the true gradient with zero mean and covariance R. The covariance represents the variance in the estimate of the gradient. It is well known that the likelihood ratio policy gradient estimator can have high variance and that gradient variance affects the rate of convergence to the minimum [2, 1]. This work reaffirms this result relating variance to convergence rate.

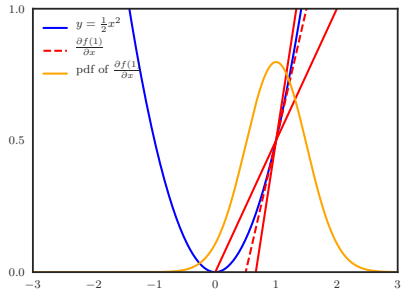


Figure 1: This figure provides intuition behind the optimization problem. The blue curve is the objective, $f(x) = \frac{1}{2}x^2$. The red dashed and solid lines are the true gradient sample noisy gradients at $x = 1$. The orange curve represents the distribution of observed gradients.

3.1 The discrete-time Kalman filter

We propose to formulate this estimation task as a variant of the Kalman filter [5]. The Kalman filter is optimal in that it minimizes the trace of the error covariance, assuming certain conditions. In particular, we require the noise to be Gaussian and the states to be Markovian.

There are varying notational conventions for Kalman filters. Since we will be estimating the gradient, we denote the true state by g_t and observations by z_t. The error covariance will be denoted by P and the observation covariance, R.

The Kalman filter addresses the general problem of estimating the state $g \in \mathbb{R}^n$ of a discrete-time controlled process that is governed by a linear stochastic difference equation.

$$g_t = Ag_{t-1} + Bu_t + q_t$$
$$z_t = Cg_t + r_t.$$

The variables q_t and r_t are process and measurement noise which are assumed to be independent of each other and Gaussian, $p(r) \sim \mathcal{N}(0, Q_t)$ and $p(q) \sim \mathcal{N}(0, R_t)$. The process noise covariance, Q, and measurement noise covariance, R, may change with time but are often assumed to be constant. In our case, R will be estimated given gradient observations and may change over time.

Filtering is logically decomposed into two steps, a time update and a measurement update. The time update projects the state and covariance estimates from step $t-1$ to t providing an *a priori* estimate $\bar{g}$, $\bar{P}$. The measurement update computes the gain matrix, K, obtains measurement z_t, and generates

$$\begin{aligned}\bar{g}_t &= Ag_{t-1} + Bu_t \\ \bar{P}_t &= AP_{t-1}A^\top + Q_t\end{aligned} \quad \Bigg| \quad \begin{aligned}K_t &= \bar{P}_t C^\top (C\bar{P}_t C^\top + R_t)^{-1} \\ \hat{g}_t &= \bar{g}_t + K_t(z_t - C\bar{x}_t) \\ P_t &= (I - K_t C)\bar{P}_t\end{aligned}$$

an *a posteriori* estimate $\hat{g}$, P. This recursion gives the optimal estimator of the state, $\hat{g}$, under the mean squared error loss for a linear system. The optimality of the Kalman filter is dependent on a few underlying assumptions that we present here.

Assumption 1 (The modeled system is, in fact, linear)**.** *The Kalman filter models the system in a linear fashion. If the true system is non-linear, optimality is no longer guaranteed, although the Kalman filter often performs well on linearized systems.*

Assumption 2 (The sequence of states form a Markov sequence)**.** *The sequence of states x_t, x_{t+1}, x_{t+2}, ..., must exhibit the Markov property, $P(x_{t+1} \mid x_t) = P(x_{t+1} \mid x_t, x_{t-1}, \dots x_0)$.*

Assumption 3 (The process and observation noise are Gaussian)**.** *The additive noise terms q_t and r_t must be normally distributed. If this is not the case the estimate given by the Kalman filter is no longer the optimal, least squares estimate.*

As we develop a practical application of this theory to gradient estimation in Section 4 we will revisit the validity of these assumptions in practice. We will find that Assumptions 1 and 2 are quite easy to satisfy in practice but that Assumption 3 is not guaranteed for non-trivial problems.

3.2 Online gradient estimation with the Kalman filter

To estimate the gradient of a function given noisy observations we model the gradient estimate as the state of a linear system. Since we do not wish to change our estimate of the gradient at each step before receiving a new observation, we will assume constant dynamics, hence $A = I$ and $Q_t = 0$. To give some intuition behind this simplified Kalman filter, imagine a stationary robot that is trying to estimate its pose based on sensor data. Its sensors, however, give noisy estimates of the true features of the environment. Given this noise the robot attempts to uncover the true underlying state. Our scenario is quite similar: we aim to estimate the true gradient from a noisy signal while the parameters remain fixed.

With the assumption of constant dynamics the state and error covariance pass through the prediction step of the Kalman filter unchanged, $\bar{g}_t = \hat{g}_{t-1}$ and $\bar{P}_t = P_{t-1}$, so $\hat{g}_{t-1}$ and P_{t-1} and pass directly into the data assimilation step. At each time step we obtain a new gradient estimate, z_t, which, in practice, is computed given newly available data. Accordingly the error is $E_t = \hat{g}_t - z_t$, where the full observation is assumed to be available, that is, we observe the gradient directly and the observation map is the identity, hence $C = I$. Given our assumptions of constant dynamics and complete observations the filter recursion simplifies to,

$$\begin{aligned}K_t &= P_{t-1}(P_{t-1} + R_{t-1})^{-1} \\ \hat{g}_t &= \hat{g}_{t-1} + K_t(z_t - \hat{g}_{t-1}) \\ P_t &= (I - K_t)P_{t-1}.\end{aligned} \tag{1}$$

The Kalman update relies on knowledge of the observation noise, R_t. Usually this quantity will not be known and must be estimated from the data. We estimate the noise using a windowed variant of Welford's algorithm for online variance estimation [19]. This algorithm maintains a sequential estimate of the mean and the sum of squares around the sample mean. Since the true observation noise is expected to change over time as the true gradient changes we maintain a windowed estimate of the mean and sum of squares by storing previous observations to fill a sliding window of size, m. At each time step when a new observation, z_t, is received the mean is updated by,

$$\mu_t = \mu_t + \frac{z_t - z_{t-m}}{m}, \tag{2}$$

and the sum of squares is updated by,

$$S_t = S_t + (z_t - \mu_{t-1})(z_t - \mu_t)^\top - (z_{t-m} - \mu_{t-1})(z_{t-m} - \mu_t)^\top. \tag{3}$$

The result is a numerically stable estimator that will track with general trends in the observation covariance.

3.3 Stability and convergence of this estimator

Since our goal is to determine when our confidence in the parameters is sufficiently high to perform an update to the parameters, it is natural to ask (i) "is this process stable" and, (ii) if so, "how quickly does it converge". First, the estimator is linear and hence, if the assumptions of the Kalman filter are met (see Section 3.1), the process is stable and will converge to the optimum under the squared error metric. It is important to note that the linearity of this estimator does not depend on linearity, or lack thereof, in the model. This is because the filter formulates the gradient as an unknown constant to be inferred and the dynamics and observation maps remain linear functions. If the estimation procedure was applied to the parameters themselves, as some researchers have investigated [9, 14], then linearity of the filter depends on linearity of the model. Second, the process will converge at an exponential rate that is dependent on the estimated variance of the error and the observations. Observe the error between the true and estimated state as $e_t = g_t - \hat{g}_t$. Then the true state evolves according to the model dynamics, which are constant $g_t = g_{t-1}$ and the estimated state evolves according to the Kalman estimate, $\hat{g}_t = \hat{g}_{t-1} + K_t E_t$. Now, the error evolves as follows:

$$\begin{aligned} g_t - \hat{g}_t &= g_{t-1} - (\hat{g}_{t-1} + K_t E_t) \\ &= g_{t-1} - \Big(\hat{g}_{t-1} + K_t(g_t + r_t - \hat{g}_{t-1})\Big) \\ &= e_{t-1} - K_t e_{t-1} - K_t r_t \\ e_t &= (I - K_t)e_{t-1} - K_t r_t. \end{aligned}$$

Since the observation noise, r_t, is assumed to be normally distributed with zero mean the expected error depends solely on the Kalman gain, K_t,

$$\mathbb{E}[e_t] = (I - K_t)\mathbb{E}[e_{t-1}]. \tag{4}$$

We know that $K_t = P_{t-1}(P_{t-1} + R_{t-1})^{-1}$ so the expression for the error expands to $\mathbb{E}[e_t] = \big(I - P_{t-1}(P_{t-1} + R_{t-1})^{-1}\big)\mathbb{E}[e_{t-1}]$. Because $P_{t-1}(P_{t-1} + R_{t-1})^{-1} < I$, the error estimate converges at an exponential rate. Additionally, P_t represents the covariance of this error estimate, e_t. As expected, if the observation variance is zero, the error in the estimated gradient error converges to zero in one step. If the gradient variance is high, convergence to the true gradient slows.

3.4 Adaptive sampling for stochastic gradient descent

With the tools we have developed we are now ready to derive an algorithm for adaptive sampling for gradient descent methods. The core idea behind our adaptive sampling method is to estimate the error, $\mathbb{E}[e_t] = (I - K_t)\mathbb{E}[e_{t-1}]$, as samples are received. Then, when the mean expected error is sufficiently small a parameter update is performed. The general outline of our algorithm is described in Algorithm 1, which describes the general case in which one wishes to minimize a function given noisy observations of the gradient. One item of importance to note is that the gradient and covariance estimates, $\hat{g}$ and R, are not reset between iterations. This means that, in general, the true error will already be quite small and the variance estimate will be close to the true gradient variance. Before moving on to the generalized reinforcement learning setup we briefly discuss the validity of the Kalman filter assumptions in this general case. Assumptions 1 and 2 are trivially satisfied. This can be seen since $g_t = I g_{t-1}$, the identity is a linear map, and $P(g_t \mid g_{t-1}) = P(g_t \mid g_{t-1}, \ldots, g_0)$ has all of its mass concentrated at g_{t-1}.

Assumption 3 is dependent on the model structure, the nature of the samples used to compute the gradient, and the loss function. This assumption becomes rather difficult to satisfy. As a simple counterexample, suppose we have a regression problem with data, X, that is uniformly distributed on some domain, and target y that is also uniformly distributed and employ a linear model, $\hat{y} = Wx$, with parameters, W. If we employ the squared error loss, $\mathcal{L} = \frac{1}{2}(y - \hat{y})^2$, then the gradient with respect to the parameters is $\hat{g} = (y - \hat{y})x$. If the errors, $(y - \hat{y})$, are normally distributed, as is often assumed, and the data is uniform, then the gradient, $\hat{g}$, is *not* normally distributed since the product of a uniform and normal random variable is not normally distributed. Despite this seemingly difficult assumption, in the policy gradient case the gradients are normally distributed with a mild assumption on the loss surface in the region around the current policy as long as the sampling distribution of the policy is also normal.

Algorithm 1 Kalman filter-based policy gradient estimation

Initialize model p with parameters θ
$\hat{g} \leftarrow \hat{g}_0$
$\epsilon \leftarrow$ error threshold
$\eta \leftarrow$ step size
while not converged **do**
    Reset the error estimate before gathering a batch of samples, $e_t \leftarrow e_0$
    while $e_t > \epsilon$ **do**
        Compute sample gradient, $z_t = \nabla_\theta \log p(z_t; \theta) f(z_t),\ z_t \sim p(z; \theta)$
        Compute the estimate of the gradient covariance, R_t, given z_t according to Eqs. 2,3
        Update the Kalman filter estimate according to Eq. 1
        Update the error estimate according to Eq. 4
    end while
    Update parameters with SGD using filtered gradient estimate, $\theta_{t+1} = \theta_t - \eta\hat{g}$
end while
return θ

4 Adaptive sampling for reinforcement learning

4.1 The continuous bandit case

In this case, we aim to solve a control problem of time horizon one. As before, we assume noisy observations of the gradient. In the one-step problem, the objective is not considered to be noisy since the true objective value is known for any given sample and does not need to be estimated by discounted rewards as in the episodic case.

In this case, the standard policy gradient approach is represented as an approximation of a score function where we aim to optimize the expected value of an objective with respect to some distribution, $\mathbb{E}_{p(z;\theta)}[f(z)]$. Applying the log-derivative trick [20], we can compute the derivative with respect to this expected value as $\hat{g} = \nabla_\theta \mathbb{E}_{p(z;\theta)}[f(z)] \approx \frac{1}{N}\sum_{t=1}^{N} \nabla_\theta \log p(z_t; \theta) f(z_t)$ where $z_t \sim p(z; \theta)$. The parameters are updated by gradient descent, $\theta_{t+1} = \theta_t - \eta\hat{g}$, for a learning rate η. This method is analogous to vanilla policy gradient approaches that many researchers have proposed [20, 8]. Policy gradient methods can be applied naturally to both discrete and continuous control problems. In the discrete case, the sampling distribution is generally chosen to be multinomial, $z_t \sim \text{Multinomial}(\theta)$, where θ_k gives the probability of class k. In the continuous case, the sampling distribution is usually Gaussian where the mean (and sometimes variance) are estimated, $p(z_t) \sim \mathcal{N}(\mu, \sigma^2)$, where $\theta = (\mu, \sigma)$.

We now examine the distribution of the gradient in these cases. In order to ensure normally distributed gradients, we require an additional assumption.

Assumption 4 (The objective is locally linear)**.** *The objective function f is locally linear around the mean of the distribution, $p(z_t)$.*

In the discrete case, the derivative of $\mathbb{E}_{p(z;\theta)}[f(z)]$ with respect to θ simplifies for a single sample as, $\hat{g} = \nabla_\theta \log p(z_t = k; \theta) f(z_t)$. The derivative is clearly non-normal since it is discrete for each class.

In the continuous score function case, we parameterize the mean and variance of a Gaussian distribution and must compute gradients for each. The gradient with respect to the mean is $\hat{g}_\mu = \frac{z_t - \mu}{\sigma^2} f(z_t)$. Since z_t is sampled from $\mathcal{N}(\mu, \sigma)$ the gradient, $\frac{z_t - \mu}{\sigma^2}$, is a whitened normal distribution. By Assumption 4, $\hat{g}_\mu$ is also approximately normal as a linear transformation of a normal distribution is normal. The gradient with respect to sigma is $\hat{g}_\sigma = \frac{(z_t - \mu)^2 - \sigma^2}{\sigma^3}$. This has the form of a shifted and scaled, squared normal distribution and, therefore, is clearly not normal. For the remainder of this work we will focus on the continuous case where the standard deviation is fixed so that the gradients are approximately Gaussian.

In cases where function approximation is used, this method can be extended with linear function approximation techniques without compromising the Gaussian property of the gradients. This follows from the fact that a linear transformation of a normal random variable remains normally distributed.

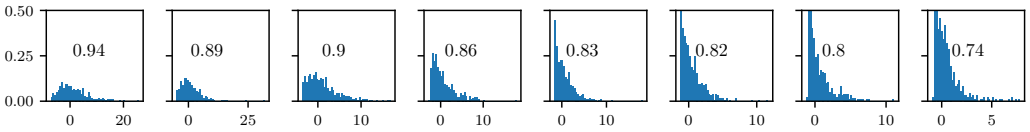


Figure 2: Histograms of observed gradients in the score function case over the several iterations of a sample run. The inset values represent Shapiro-Wilk test statistics for normality.

4.2 The episodic case

In general, this is a straight-forward adaptation of the simpler case we have already discussed. The core difference in this case is that the gradient is now computed over an entire trajectory, which we define analogous to a single sample in the bandit case. Now, the policy loss is now defined as $\mathcal{L} = \frac{1}{NT}\sum_i^N \sum_t^T \log \pi(a_{ij} \mid s_{it}; \theta) G(a_{it}, s_{it})$, where i runs over trajectories and t runs over time-steps. The policy, π, is used to sample actions, a, given a state, s, and the objective is the sum of discounted future rewards, $G(s_t, a_t) = \sum_{k=t}^T \gamma^{k-t} r_t$. Accordingly, the gradient is now a sum over the gradient at each time-step. Given NT normally distributed gradient observations the mean of the set is also normally distributed, even with correlated samples. Unlike the one-step case, in the episodic case the objective is considered noisy because for some state-action pair, s_t, a_t, if t is not the final time-step of the episode, the sample return, $G(s_t, a_t)$, is an approximation of the true return under the policy, $\mathbb{E}_\pi[G(s_t, a_t)]$.

5 Experimental evaluation

Now that we have developed the techniques to estimate the error in an effective manner and ensure that the Kalman filter assumptions hold approximately we evaluate the method on both a simple optimization case and a linear-quadratic regulator (LQR) problem. In this section we aim to answer a few specific questions about our method: (1) Does the distribution of sample gradients with respect to the mean parameter approximately fit a Gaussian distribution? (2) Does the Kalman filter error estimate of the expected gradient error accurately represent the true gradient error? (3) How does our method compare to a standard policy gradient method with a fixed batch size in an episodic control task?

5.1 A simple test case

In order to gain a deeper understanding of Kalman filter gradient estimation, we begin our experiments with analyzing how the algorithm behaves when minimizing a convex, quadratic function. We aim to find the minimum of the quadratic, $f(x) = \frac{1}{2}x^2$, with gradient descent given only noisy observations of the true gradient. Both standard score function optimization with a fixed batch size and our method are evaluated. For both methods the initial mean, μ_0, is set to 1 and the standard deviation, σ, is fixed at 0.5. Both methods were given a fixed learning rate of 0.01 and optimized with SGD. In the Kalman filter algorithm the window size is set to 20.

Naturally, since we require the gradient to be normally distributed we make this our first area of analysis. Figure 2 shows the distribution of gradients with respect to the mean parameter over the first 8 iterations in the score function case when minimizing a quadratic function, $f(x) = \frac{1}{2}x^2$. The histograms show an approximately Gaussian form and give high Shapiro-Wilk test scores for normality (Figure 2). Having satisfied our core assumption of approximately normally distributed gradients, we now examine how the error estimated by our algorithm compares to the true error. In the quadratic case we can determine the true gradient by numerically integrating the score function objective. Figure 3 shows the true error versus estimated error. In general, the estimate tracks the truth nicely; as before some deviation is expected due to sampling variability. Given our estimate of the error, we adaptively determine the batch size based on the mean expected error. Figure 4c shows the relationship between estimated gradient variance and batch size. When the variance is high, the Kalman filter error requires more samples to converge so updates are performed less frequently. The batch size tracks the estimated variance closely.

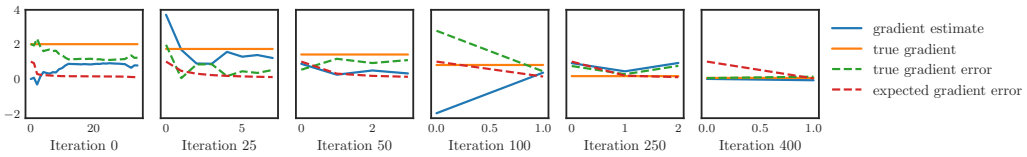
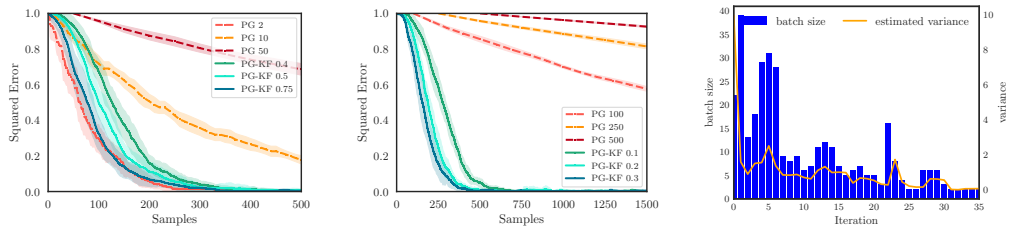


Figure 3: The true and expected gradient and the respective errors evaluated over several iterations. The number of samples per iteration becomes small as the gradient variance decreases.



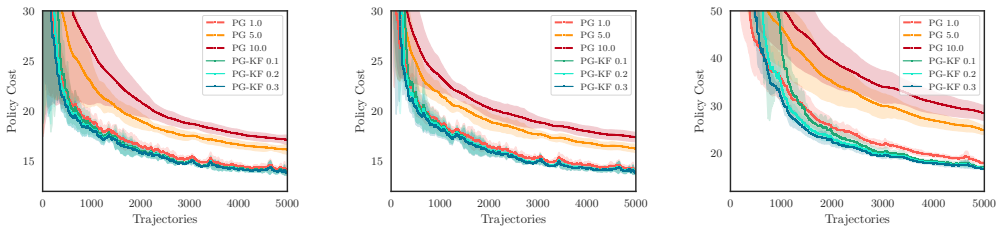
(a) High error threshold (b) Low error threshold (c) Variance versus batch

Figure 4: Figure (a) and (b) show the variation in sample complexity based on different error thresholds used to determine when to update. Each method is labeled by its respective batch size or error threshold. Note that the Kalman filter variant was not biased towards a smaller batch size and updates were based solely on expected error. Figure (c) shows the relationship between the variance and the selected batch size.

Finally, we observe the sample complexity of our method. We run several variants with different error thresholds for the update step and compare the results to a baseline score function method that uses a fixed batch size. We find that by setting the error threshold high, analogous to a small batch size, we can mirror the results of the smallest stable batch size (Figure 4a). Alternatively, when setting a low threshold we observe faster convergence than a large batch method (Figure 4b). As the mean of the sampling distribution, $p(z_t)$, nears the minimum the variance of the gradient decreases. The Kalman filter detects the low variance and perform updates rapidly.

5.2 A linear quadratic regulator task

Our goal in this section is to analyze the behavior of our method compared to a standard policy gradient algorithm. We define a task to stabilize a point mass at the origin following the recent work by Tucker et al. [17]. In this task, the objective is to learn a controller for an LQR system where the dynamics are not known. The true system is defined by the x and y positions and velocities of a point mass and the control is the force to apply in each dimension. The dynamics of our task exactly follow Tucker et al. [17] though we do not consider noise in the control. We set an initial state mean position and velocity, μ_0, with variance $0.0001I$. In our experiments we define a time horizon, $T = 100$, and parameterize the mean of a Gaussian policy with a linear model. The policy is optimized according to the REINFORCE algorithm [20] with a learning rate of 0.005 and the Adam optimizer [6]; we do not learn a critic for either version. For the vanilla policy gradient algorithm, we evaluate various batch sizes consisting of 1, 5 and 10 full trajectories. The Kalman gradient estimator receives a gradient observation per trajectory. As in the bandit case, we observe how our method performs with varying error thresholds. First, we observe that even in this more complicated task the gradients are still approximately Gaussian. Figure 5 shows the learning curves for each method. We find that a large batch variant, while very stable, exhibits slow convergence. On the other hand, a small batch variant initially converges very quickly. Our method strikes a balance of both. At the start of training our method samples large batches while the variance is high. As training progresses our method adapts to a smaller batch size and converges rapidly. We find that our method slightly outperforms the best performing static batch variant (Table 1).



(a) $\mu_0 = [0.5, 0.5, 0.0, 0.0]$ (b) $\mu_0 = [0.5, 0.5, 0.1, -0.1]$ (c) $\mu_0 = [0.5, 0.5, -0.25, 0.5]$

Figure 5: This Figure shows the learning curves for both the baseline policy gradient and our Kalman filter variant. Each run is labeled by the batch size in trajectories or the error threshold.

Table 1: Performance of learned policies on the LQR point mass task

Variant	Policy Cost	KF-0.1	KF-0.2	KF-0.3	PG-1	PG-5	PG-10	Nominal
(a)	best	13.54	**13.52**	13.53	13.57	15.34	16.37	∼11.5
	mean top-5	13.70	**13.67**	**13.67**	13.91	16.09	17.13	∼11.5
(b)	best	**13.60**	13.62	**13.60**	13.69	15.73	16.88	∼11.5
	mean top-5	13.73	13.75	**13.71**	13.92	16.18	17.35	∼11.5
(c)	best	16.47	**16.36**	16.47	17.45	23.70	26.34	∼10.4
	mean top-5	16.93	16.78	**16.66**	17.92	24.87	28.49	∼10.4

6 Related work

Previous work has studied the connections between Kalman filtering and gradient descent. It is generally understood that the Kalman filter can be interpreted as a pre-conditioned gradient descent where the error covariance, P_t, takes the place of the conditioning matrix. Thus, it is not surprising that the Kalman filter has been applied to parameter estimation problems, including neural network training [14] and online system identification [18]. Along these lines, Ollivier [9], recently showed that the extended Kalman filter applied to parameter estimation is exactly equivalent to an online natural gradient descent method. Despite this extensive body of research, we do not believe that the Kalman filter has previously been applied to gradient estimation. Adaptive batch sizes have been explored previously in several directions. Smith et al. [16] argues that one should increase the batch size as training progresses as opposed to decaying the learning rate. Smith and Le [15] also analyze SGD from a Bayesian perspective giving insight into optimal batch sizes and generalization behavior. In the policy gradient case, Papini et al. [10] derive an adaptive batch size method based on a probabilistic performance improvement guarantee.

7 Conclusion

Reinforcement learning promises systems that are able to operate in complicated, real-world environments. Unfortunately, the large sample complexity of existing RL methods still remains a critical bottleneck hindering many applications in the real world. In contrast to existing work in batch policy gradient methods, which employ a fixed batch size at every iteration, we focused on an online procedure for determining the number of samples required to confidently estimate the gradient. To this end, we introduced a Kalman filter formulation for estimating the error in the current gradient estimate and developed an algorithm for adaptively selecting the batch size. The approach avoids the evaluation of unnecessary samples that do not substantially change our estimation of the gradient. To employ the Kalman filter, special care needs to be put into meeting the respective assumptions, the most difficult being the assumption of normally distributed noise. We showed that in the policy gradient case this assumption is satisfied via the local linearity of the objective around the policy mean. We demonstrated that our method is on par with a well-tuned policy gradient method and consistently performs better in terms of sample complexity when compared to large batch variants.

References

[1] Alekh Agarwal and Ofer Dekel. Optimal algorithms for online convex optimization with multi-point bandit feedback. In *COLT*, pages 28–40. Citeseer, 2010.

[2] Peter W Glynn. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84, 1990.

[3] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E Turner, and Sergey Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. *arXiv preprint arXiv:1611.02247*, 2016.

[4] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560*, 2017.

[5] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[8] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[9] Y. Ollivier. Online Natural Gradient as a Kalman Filter. *ArXiv e-prints*, March 2017.

[10] Matteo Papini, Matteo Pirotta, and Marcello Restelli. Adaptive batch size for safe policy gradients. In *Advances in Neural Information Processing Systems*, pages 3594–3603, 2017.

[11] Aravind Rajeswaran, Kendall Lowrey, Emanuel V Todorov, and Sham M Kakade. Towards generalization and simplicity in continuous control. In *Advances in Neural Information Processing Systems*, pages 6553–6564, 2017.

[12] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[13] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[14] Sharad Singhal and Lance Wu. Training multilayer perceptrons with the extended kalman algorithm. In *Advances in neural information processing systems*, pages 133–140, 1989.

[15] Samuel L Smith and Quoc V Le. A bayesian perspective on generalization and stochastic gradient descent. In *Proceedings of Second workshop on Bayesian Deep Learning (NIPS 2017)*, 2017.

[16] Samuel L Smith, Pieter-Jan Kindermans, and Quoc V Le. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

[17] G. Tucker, S. Bhupatiraju, S. Gu, R. E. Turner, Z. Ghahramani, and S. Levine. The Mirage of Action-Dependent Baselines in Reinforcement Learning. *ArXiv e-prints*, February 2018.

[18] John Valasek and Wei Chen. Observer/kalman filter identification for online system identification of aircraft. *Journal of Guidance, Control, and Dynamics*, 26(2):347–353, 2003.

[19] BP Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4 (3):419–420, 1962.

[20] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer, 1992.

[21] Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, pages 5285–5294, 2017.