Journal of Machine Learning Research 23 (2022) 1-13 Submitted 5/24; Published N/A

Exploring the Dynamics of Adam

Alexandre Kaiser AMK1004@NYU.EDU
Department of Computer Science
Courant Institute, NYU
NY, NY, USA

Editor: None

Abstract

This project explores some of the possible reasons for why Adam dominates the market for training deep neural networks. First we inspect how Adam deals with the noise of stochastic gradients, and find that although the variance of its steps may converge to zero, it does so much slower than Adam's own typical convergence time. Then we analyze the dynamics of Adam in three different optimization settings: (i) crossing flat regions of the loss, (ii) avoiding slow time around saddle points, (iii) converging quickly in valleys. We find that Adam outperforms other competitive optimizers when it comes to saddle dynamics, however its performance in valleys appear to be dominated by that of RMSProp. Lastly, although we find Adam can cross flats to a certain extent, we find serious reasons to doubt the practical importance this ability.

Keywords: adam, dynamics, flat, saddle, valley

1 Introduction

Adam, short for Adaptive Moment Estimation Kingma and Ba (2017), is a stochastic gradient descent algorithm that has become the gold standard optimizer for training neural networks. Adam was initially designed to merge concepts from Adagrad and RMSprop to create a stable first order method that works with sparse gradients and requires little-to-no fine-tuning. In practice, Adam is so reliable that if training is unsuccessful the researchers will change everything else before thinking about changing Adam. Nevertheless, it has very little theoretical backing.

There are many explanations that people refer to for why Adam is superior to SGD, for this project we will explore some of the most prominent claims both theoretically and empirically. Starting with a simple claim, the addition of a moving average for the first moment enables Adam to cross flat regions of the loss surface. Practically every other gradient descent method steps in the direction of an unbiased estimate of the gradient at that step, thus their dynamics end once reaching a flat. In convex optimization, reaching a flat is good news because you have reached the global optima. However, the loss landscape of neural network is not convex, which warrants the ability to explore past these local optima.

Next, motivated by the work of Jacot et al. A. Jacot et al. (2022) on the saddle-to-saddle dynamics of neural networks, we will attempt to characterize Adam's dynamics near saddle points. In non-convex optimization, saddle-point dynamics are crucial to understand because the dynamics are slowest the closer you are to one. This fact means that you can

approximate the time to converge to a local optima by aggregating the time spent near a saddle. If Adam could avoid these slow-downs, it could possibly explain why Adam empirically has the fastest convergence speed.

Lastly, motivated by visualizations of the loss landscape of neural networks by Li et al. Li et al. (2018) and Chaudhari et al. Chaudhari et al. (2017), we conclude our exploration by analyzing the dynamics of Adam in valleys. In the field of partial differential equations, which often study non-linear dynamics, valleys cause quite the headache because of the oscillating behavior that many solvers are susceptible to. If Adam could quickly travel through valleys, it could also possibly explain Adam's convergence time.

2 Preliminary

2.1 Adam's historical connections

In the original paper Kingma and Ba (2017), the authors introduced Adam as a mix of AdaGrad and RMSProp, which are both stochastic gradient descent (SGD) methods. The standard form of an SGD algorithm is

$$w_{t+1} = w_t - \eta \nabla \mathcal{L}(w_t, (x_t, y_t)) \tag{1}$$

Assuming that $\mathcal{L}$ is convex with respect to w_t, each gradient step, for some training data pair (x_t, y_t), should converge the global minimum of the loss. Unfortunately do to possible resonances explored in the field of Partial Differential Equations, we scale the gradient step by some learning rate λ.

AdaGrad innovated the standard SGD equation by "adapting the learning rate", that is η was standardized by a measure of deviation of past gradients using their second moment, seen in Equation 2.

$$w_{t+1} = w_t - \frac{\eta}{\sum_i = 1^t \left(\nabla \mathcal{L}(w_i, (x_i, y_i))\right)^2} \nabla \mathcal{L}(w_t, (x_t, y_t)) \tag{2}$$

RMSprop innovated on that idea to form a regret minimization algorithm robust to non-stationary distributions, and they did so by changing the measure of deviation to be a moving average.

$$\begin{cases} w_{t+1} = w_t - \dfrac{\eta}{\sigma_t + \epsilon} \nabla \mathcal{L}(w_t, (x_t, y_t)), \\ \sigma_t^2 = \beta_2 \sigma_{t-1}^2 + (1 - \beta_1) \left(\nabla f(w_t)\right)^2 \end{cases} \tag{3}$$

Inspired by the real world performance of those two algorithms, the authors for Adam had seen the trend towards momentum and decided to apply the same concept to the gradients as well. In addition, to avoid the bias of initializing both momentum terms at zero, there is an attempt at bias correction.

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

Figure 1: The original definition for Adam

2.2 Defining Adam

In the original paper Kingma and Ba (2017), the authors define Adam as in Figure 1. We can reformulate Adam's weight update to appear similar to the previous SGD methods in Equation 4.

$$w_{t+1} = w_t - \eta \frac{g_t}{\sigma_t + \hat{\epsilon}} \frac{\sqrt{1 - \beta_2^t}}{1 - \beta_1^t} \tag{4}$$

where,

$$\begin{cases} g_t = \beta_1 g_{t-1} + (1 - \beta_1) \nabla f(w_t), \\ \sigma_t^2 = \beta_2 \sigma_{t-1}^2 + (1 - \beta_1) \left(\nabla f(w_t) \right)^2, \\ \hat{\epsilon} = \epsilon \sqrt{1 - \beta_2^t}. \end{cases}$$

Note that g_t and σ_t can also be expressed explicitly in terms of the gradient history.

$$\begin{cases} g_T = (1 - \beta_1) \sum_{t=1}^{T} \nabla f(w_t) \beta_1^{T-t}, \\ \sigma_t^2 = (1 - \beta_1) \sum_{t=1}^{T} \nabla \left(\nabla f(w_t) \right)^2 \beta_2^{T-t}. \end{cases}$$

For most of the analysis in this project, we analyze the long term behavior of Adam, thus the bias correction has negligible effect on the gradient update. In this case, the update rule simplifies into Equation 5.

$$w_{t+1} = w_t - \eta \frac{g_t}{\sigma_t + \epsilon} \tag{5}$$

Remark, Adam is rarely tuned in practice, so our analysis can rely on the default values for the hyperparameters as listed in the original paper.

$$\begin{cases} \beta_1 = 1 - 10^{-1}, \\ \beta_2 = 1 - 10^{-3}, \\ \eta = 10^{-3}, \\ \epsilon = 10^{-8}. \end{cases}$$

2.3 Dimensionality

Remark that since the gradient step $\eta \frac{g_t}{\sigma_t + \epsilon}$ is an element-wise division of tensors with the same units, the gradient step has no units. This is a very important point because it reinforces the need to normalize data before training on Adam, or RMSProp, or even Adagrad, otherwise the solution has no physical meaning. This point is doubly important because future sections of this paper will find that important properties of Adam are purely dependent of the hyperparameters which are very rarely tuned in practice, which would only be reasonable if all problems are non-dimensionalized.

3 Noise profile

The first claim we want to explore is that of smoothing the gradients by reducing the noise. Let's start by assuming that each gradient can be written as follows.

$$\nabla \mathcal{L}\left(w_t, (x_t, f(x_t))\right) = \nabla \mathcal{L}(w_t, f) + \mathcal{N}_t \tag{6}$$

Where the noise is unbiased $\mathbb{E}[\mathcal{N}_t = 0$. In this way, we can solve for the mean and variance of g_t.

$$\begin{aligned} \mathbb{E}[g_T] &= \mathbb{E}\left[(1-\beta_1)\sum_{t=1}^{T} \nabla \mathcal{L}\left(w_t, (x_t, f(x_t))\right) \beta_1^{T-t}\right] \\ &= \mathbb{E}\left[(1-\beta_1)\sum_{t=1}^{T} \nabla \mathcal{L}(w_t, f) \beta_1^{T-t}\right] + \mathbb{E}\left[(1-\beta_1)\sum_{t=1}^{T} \mathcal{N}_t \beta_1^{T-t}\right] \\ &= (1-\beta_1)\sum_{t=1}^{T} \nabla \mathcal{L}(w_t, f) \beta_1^{T-t} \\ \mathrm{Var}[g_t] &= \mathbb{E}\left[\left((1-\beta_1)\sum_{t=1}^{T} \nabla \mathcal{L}\left(w_t, (x_t, f(x_t))\right) \beta_1^{T-t} - (1-\beta_1)\sum_{t=1}^{T} \nabla \mathcal{L}(w_t, f) \beta_1^{T-t}\right)^2\right] \\ &= (1-\beta_1)^2 \mathbb{E}\left[\left(\sum_{t=1}^{T} \mathcal{N}_t \beta_1^{T-t}\right)^2\right] \end{aligned}$$

The mean and variance of σ_t^2 can be solved in the same way. Note that if the noise of each gradient step is sampled independently of past gradients, then by the central limit theorem we get that variance of g_t tends to zero as $\frac{1}{\sqrt{T}}$ for T steps. The convergence towards

zero variance is thus slower than the unbiased terms converging exponentially fast to the biased terms. Thus, the need for noiseless gradient steps is not terribly important early in training. On top of that, in practice mini-batches $(x_t, f(x_t))$ are not necessarily independent from past samples, this would add to the time it takes to converge towards a variance of zero.

In the original paper Kingma and Ba (2017), the authors presented the following proof for the bias correction of g_t and σ_t^2.

$$\begin{aligned}\mathbb{E}[g_T] &= \mathbb{E}\left[(1-\beta_1)\sum_{t=1}^{T}\nabla\mathcal{L}(w_t, f)\beta_1^T\right] \\ &= (1-\beta_1)\mathbb{E}\left[\nabla\mathcal{L}(w_t, f)\right]\sum_{t=1}^{T}\beta_1^T \\ &= \mathbb{E}\left[\nabla\mathcal{L}(w_t, f)\right](1-\beta_1^T)\end{aligned}$$

Notice how they factored out $\mathbb{E}\left[\nabla\mathcal{L}(w_t, f)\right]$, which you can only do if it is constant with respect to time, which is clearly an unreasonable assumption. In particular, it almost contradicts the second proof in the paper which states that Adam converges to a global minima for $\mathcal{L}$ convex. This is because if $\mathbb{E}\left[\nabla\mathcal{L}(w_t, f)\right]$ is a constant in time then we can simplify the weight update in the following way.

$$\begin{aligned}\mathbb{E}[w_{t+1} - w_t] &= -\eta\frac{(1-\beta_1)\nabla\mathcal{L}(f)\sum_{t=1}^{T}\beta_1^t}{\sqrt{(1-\beta_2)\left(\nabla\mathcal{L}(f)\right)^2\sum_{t=1}^{T}\beta_2^t} + \epsilon} \\ &= -\operatorname{sgn}(\nabla\mathcal{L}(f))\eta\frac{1-\beta_1^t}{\sqrt{1-\beta_2^t}} + \mathcal{O}(\epsilon)\end{aligned}$$

The original convergence proof assumed that gradients were bounded, but given the same assumption they made to remove the bias from g_t and σ_t, the magnitude of the gradient is irrelevant until the momentum is the same order as ϵ. Each step exists approximately on the corners of a hypercube. To achieve convergence, the global optima needs to be flat enough for the momentum to converge to order ϵ before leaving the optima (see the next section for more details). In all generality, assumption that $\mathbb{E}[\nabla\mathcal{L}(w_t, f)]$ would not lead to convergence.

Not only that, we will cover explicit loss surfaces in which $\mathbb{E}\left[\nabla\mathcal{L}(w_t, f)\right]$ is far from constant. In all, even though the gradient steps have a vanishing variance, it is quite a mystery that Adam is capable to perform empirically even though it's gradient step is biased.

4 Crossing a flat

The previous sections paints a very critical image of Adam, particularly for its use of a moving average for the gradient. Nevertheless, Adam is widely preferred in practice over

Adagrad and RMSProp for training deep neural networks, so for the remainder of this paper, let's uncover optimization scenarios that could explain the practical relevance of Adam.

This first section considers Adam's ability to cover flat regions of the loss. Although a flat region for a convex loss signifies convergence to the global optima, the motivation for studying flats stems from the common observation that the loss for deep networks can have these great plateaus, particularly near the origin. Unlike every other SGD method that use the real gradient, which is zero on a flat, Adam's use of momentum could give it the ability to cross these sub-optimal flat regions of the loss function.

It is difficult to contextualize a distance in parameter space, especially given the that the parameters are dimensionless. Nevertheless, we could compare a distance in parameter space to the largest step that Adam can take along any one direction. In this way, we are making a psuedo-conversion from unitless distance to Adam-steps.

Proposition 1. *The gradient step for each weight is bounded by a function of the hyperparameters*

$$|w_{t+1} - w_t| \leq \eta \frac{\sqrt{\beta_2}(1-\beta_1)}{\sqrt{(1-\beta_2)(\beta_2 - \beta_1^2)}} \approx 7.3\eta$$

Proof Let δ_i be the sequence of gradients observed by time t. The gradient step can be written explicitly in terms of δ_i.

$$w_{t+1} - w_t = -\eta \frac{(1-\beta_1)\sum_{i=0}^{t} \delta_{t-i}\beta_1^i}{\hat{\epsilon} + \sqrt{(1-\beta_2)\sum_{i=0}^{t} \delta_{t-i}^2 \beta_2^i}} \cdot \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t}$$

Lets first show by induction that $\frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t} \leq 1$ for all $t \geq 1$.
At $t = 1$, we have $\frac{\sqrt{1-\beta_2}}{1-\beta_1} = 10^{-\frac{1}{2}} < 1$.
Now let $1 - \beta_2^t \leq (1-\beta_1^t)^2$

$$\begin{aligned} 1 - \beta_2^t &\leq (1-\beta_1^t)^2 \\ 1 - \beta_2^t &\leq 1 - 2\beta_1^t + \beta_1^{2t} \\ \beta_2^t &\geq \beta_1^{2t} - 2\beta_1^t \end{aligned}$$

By induction, since $\beta_2 > \beta_1^2$, we show that $1 - \beta_2^{t+1} \leq (1-\beta_1^{t+1})^2$

$$\beta_2^{t+1} \geq \beta_2(\beta_1^{2t} - 2\beta_1^t) \geq \beta_1^{2t+2} - 2\beta_1^{t+2} \geq \beta_1^{2t+2} - 2\beta_1^{t+1}$$

Thus the bias correction factor is bounded by 1. Due to the symmetry about the sign of g_t, let $g_t \geq 0$ for the following analysis. We can make an initial bound by removing the bias corrections and ϵ.

$$|w_{t+1} - w_t| \leq \eta \frac{1-\beta_1}{\sqrt{1-\beta_2}} \cdot \frac{\sum_{i=0}^{t} \delta_{t-i}\beta_1^i}{\sqrt{\sum_{i=0}^{t} \delta_{t-i}^2 \beta_2^i}}$$

We can bound the step by its supremum to find the following bound.

$$|w_{t+1} - w_t| \leq \sup_{\delta_i\ \forall i} \eta \frac{1-\beta_1}{\sqrt{1-\beta_2}} \cdot \frac{\sum_{i=0}^{t} \delta_{t-i}\beta_1^i}{\sqrt{\sum_{i=0}^{t} \delta_{t-i}^2 \beta_2^i}}$$
$$= \eta \frac{1-\beta_1}{\sqrt{1-\beta_2}} \sup_{\delta_i\ \forall i} \frac{\sum_{i=0}^{t} \delta_{t-i}\beta_1^i}{\sqrt{\sum_{i=0}^{t} \delta_{t-i}^2 \beta_2^i}}$$

To bound the ratio of sums we can utilize the Cauchy-Schwartz inequality, which states

$$\left(\sum_i u_i v_i\right)^2 \leq \left(\sum_i u_i^2\right)\left(\sum_i v_i^2\right)$$

We can rearrange the Cauchy-Schwartz inequality in the following way to identify our maximization problem.

$$\frac{\sum_i^t u_i v_i}{\sqrt{\sum_i^t u_i^2}} \leq \sqrt{\sum_i^t v_i^2}$$

To apply the Cauchy-Schwartz inequality, let

$$\begin{cases} u_i = \sqrt{\beta_2^i}\delta_{t-i}, \\ v_i = \dfrac{\beta_1^i}{\sqrt{\beta_2^i}}. \end{cases} \Rightarrow \begin{cases} u_i v_i = \beta_1^i \delta_{t-i}, \\ u_i^2 = \beta_2^i \delta_{t-i}^2. \end{cases}$$

Applying the Cauchy-Schwartz inequality, we get the following bound for the Adam's step size.

$$|w_{t+1} - w_t| \leq \eta \frac{1-\beta_1}{\sqrt{1-\beta_2}} \sqrt{\frac{1 - \left(\frac{\beta_1^2}{\beta_2}\right)^{t+1}}{1 - \frac{\beta_1^2}{\beta_2}}}$$

Since $\beta_2 > \beta_1^2$, the bound converges to its supremum as $t \to \infty$. Thus the general bound for the step size of Adam is

$$|w_{t+1} - w_t| \leq \eta \frac{\sqrt{\beta_2}(1-\beta_1)}{\sqrt{(1-\beta_2)(\beta_2 - \beta_1^2)}} \approx 7.3\eta$$

■

Now that we have a reference to contextualize distance, let's solve for the maximum distance covered in a flat.

Proposition 2. *If the loss $\mathcal{L}(w_t, f)$ is Lipschitz in ts first argument, then the distance each weight can cross a flat is bounded by*

$$\lim_{s \to +\infty} |w_s - w_0| \leq \eta \frac{\beta_2(1-\beta_1)}{(\sqrt{\beta_2} - \beta_1)\sqrt{(1-\beta_2)(\beta_2 - \beta_1^2)}} \approx 73\eta$$

Proof To begin, let g_0 and σ_0 be the values of g_T and σ_T when Adam reaches the flat at time T. To avoid confusion of time scales, let s denote the time since reaching the flat.

$$\begin{cases} w_s - w_0 = -\eta \sum_{i=0}^{s-1} \frac{g_i}{\sigma_i + \epsilon} \frac{\sqrt{1-\beta_2^{T+s}}}{1-\beta_1^{T+s}} \\ g_i = \beta_1 g_{i-1}, \\ \sigma_i^2 = \beta_2 \sigma_{i-1}^2. \end{cases}$$

Firstly, since $\frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t} \leq 1$ for all $t \geq 1$, we can upper bound the bias correction in the same way as the previous proof.

$$\begin{aligned} |w_s - w_0| &\leq \left| \eta \sum_{i=0}^{s-1} \frac{g_0 \beta_1^i}{\sigma_0 \sqrt{\beta_2^i} + \epsilon} \right| \\ &\leq \eta \operatorname{sgn}(g_0) \, \frac{g_0}{\sigma_0} \cdot \sum_{i=0}^{s-1} \left(\frac{\beta_1}{\sqrt{\beta_2}} \right)^i \\ &= \eta \operatorname{sgn}(g_0) \, \frac{g_0}{\sigma_0} \cdot \frac{1 - \left(\frac{\beta_1}{\sqrt{\beta_2}} \right)^s}{1 - \frac{\beta_1}{\sqrt{\beta_2}}} \end{aligned}$$

For default β_1 and β_2, we have that $\beta_1 < \sqrt{\beta_2}$, thus the series converges to

$$\lim_{s \to +\infty} |w_s - w_0| \leq \eta \operatorname{sgn}(g_0) \, \frac{g_0}{\sigma_0} \cdot \frac{1}{1 - \frac{\beta_1}{\sqrt{\beta_2}}} \tag{7}$$

To maximize the distance in the flat, we need to maximize the ratio $\frac{g_0}{\sigma_0}$. Notice that this is the same problem as Proposition 1. Thus we already know that the ratio $\frac{g_0}{\sigma_0}$ is maximized by

$$\max_{\delta_i \, \forall i} \frac{g_0}{\sigma_0} = \frac{\sqrt{\beta_2}(1-\beta_1)}{\sqrt{(1-\beta_2)(\beta_2 - \beta_1^2)}}$$

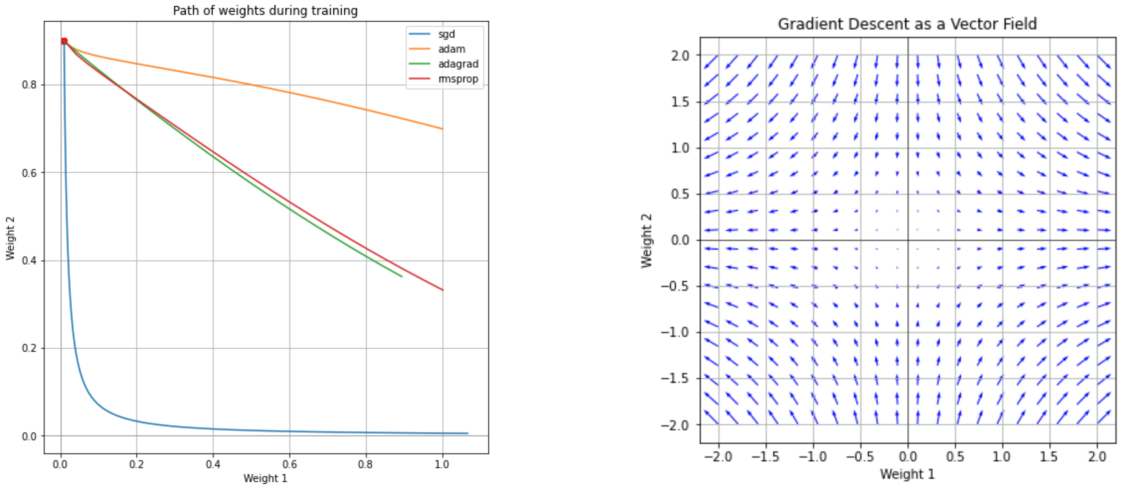
By combining the bound for the ratio $\frac{g_0}{\sigma_0}$ and the previous bound, we get a complete bound for the maximum crossing distance in a flat region.

$$\lim_{s \to +\infty} |w_s - w_0| \leq \eta \, \frac{\beta_2(1-\beta_1)}{(\sqrt{\beta_2} - \beta_1)\sqrt{(1-\beta_2)(\beta_2 - \beta_1^2)}} \approx 73\eta$$

■

Proposition 2 finds that Adam can cross a distance equivalent to approximately 10 steps once it reaches a flat. It is particularly unusual that the distance it can travel has no relation to the properties of the loss surface. This begs the question, if it were relevant to cross the flat, then wouldn't the hyperparameters need to be fine-tuned to each new loss surface?

In practice, even small neural networks are trained for a few thousand epochs (steps). In comparison, 10 steps seems relatively insignificant.



(a) 2-dimensional dynamics near a saddle

(b) Gradient descent visualized as vector field

Figure 2: Various visualizations of dynamics

5 Dynamics near saddle

Research into the loss surface of neural networks suggests that overparameterized models have loss function for which the only fixed points are either global optima, or saddles. This implies that saddle-point dynamics are responsible for most of the time to converge, since being near a saddle results in slow dynamics. In this section, we will inspect the behavior or Adam near saddle points to observe whether it outperforms the other standard optimizers, namely SGD, AdaGrad and RMSProp.

The intuition for why Adam might outcompete other methods comes down to the approximately flat region near saddle points. It is this flat nature that creates slow dynamics, however Adam's use of momentum might be able to escape the saddle disproportionally quickly due to its ability to cross flat regions.

Figure 2a compares the dynamics of Adam, SGD, AdaGrad and RMSProp on a typical saddle point characterized by the hessian

$$\mathcal{H} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

The Figure clearly shows that Adam is the most favorable out of the optimizers tested, however it does in an unintuitive way, by avoiding the near-saddle dynamics entirely. The reason for this comes from normalizig the gradients by the second moment, as we see all of AdaGrad, RMSProp and Adam behave similarly in this respect. As we remarked in Section 3, if the gradients are similar to one another, then the gradient step will exist on the corners of a hypercube, i.e. a diagonal. This is why AdaGrad and RMSProp act very similarly in this setting. However, to explain why Adam outperforms those two, we must prove the following statement.

Proposition 3. *Adam's gradient step is largest when the mass of the gradient history is more recent, and smallest when the mass of the gradient history was earlier.*

Proof To prove this statement, let's return to a familiar expression for the update size when t is large enough for $1 - \beta_1^t \approx 1$ and $\sqrt{1 - \beta_2^t} \approx 1$.

$$|w_{t+1} - w_t| \approx \eta \frac{1 - \beta_1}{\sqrt{1 - \beta_2}} \cdot \frac{\sum_{i=1}^{t} \delta_t \beta_1^{t-i}}{\sqrt{\sum_{i=1}^{t} \delta_{t-i}^2 \beta_2^{t-i}}}$$

Let's drop the approximation and assume equality for the sake of simplicity. If the entire gradient history were zero, and we could add mass to only one gradient, which one would maximize the step?

$$\begin{aligned}\max_{i \geq 1} |w_{t+1} - w_t| &= \max_{i \geq 1} \eta \frac{1 - \beta_1}{\sqrt{1 - \beta_2}} \cdot \frac{\delta_t \beta_1^{t-i}}{\sqrt{\delta_t^2 \beta_2^{t-i}}} \\ &= \eta \frac{1 - \beta_1}{\sqrt{1 - \beta_2}} \max_{i \geq 1} \left(\frac{\beta_1}{\sqrt{\beta_2}} \right)^{t-i}\end{aligned}$$

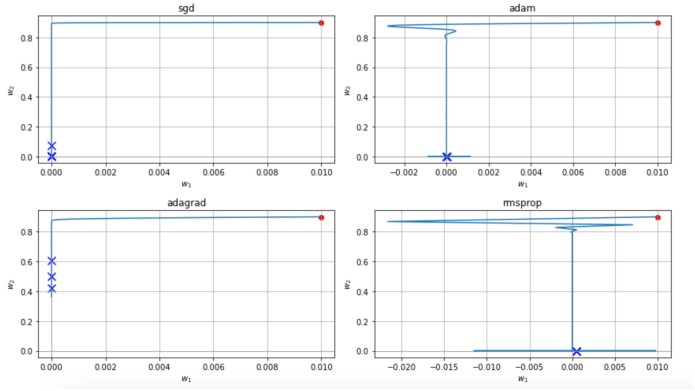
Since $\beta_1 < \sqrt{\beta_2}$, the largest step would occur for $i = t$, and the smallest would occur for $i = 1$ ■

We can now explain why Adam outperforms the other methods n Figure 2a. As Adam moves in a diagonal from its initial position, its gradient along w_1 increases and its gradient along w_2 decreases. If the gradients along w_1 are increasing, then most of the mass of its history is recent, as opposed to the decreasing gradients of w_2 which have most of its mass at the beginning of the gradient history. Proposition 3 explains that Adam's gradient step will thus favor w_1 over w_2.

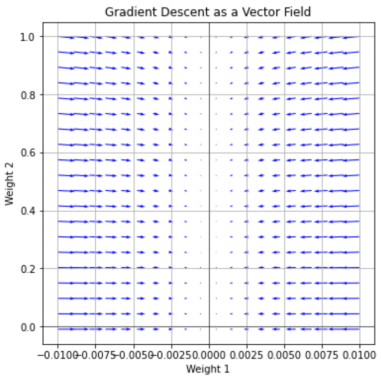
6 Stability in valleys

The last environment that we will look at is narrow valleys. A narrow valley is similar to a saddle in that there is a direction along which points are being attracted, however along a second direction there is a smooth monotonic slope (as opposed to a repulsive direction in the case of a saddle). A valley is considered narrow if the attracting direction dwarfs the monotonic slope of the second direction. The reason that valley are challenging to optimize over is because of the possible oscillations produced by taking gradient steps along the very attractive direction, thus making disproportionally less progress along the direction of the valley.

Let's consider a first valley for a smooth function $f(x, y) = x^2 + 10^{-3}y^2$. The movement along the x direction is orders of magnitude faster than that of the y direction. In Figure 3, we plot the dynamics of SGD, Adam, AdaGrad and RMSProp, as well as the associated vector field. Each path has the same length of 10^6. To get a sense of convergence time, each plot has 3 x-markers used to mark the quarter, half and 3-quarter mark of each path. From a gradient flow perspective, SGD avoided having any unnecessary oscillating behavior, however it doesnt appear to be the fastest to find the origin. Adam and RMSProp both find the origin within the first quarter of their paths. Further analysis reveals that Adam

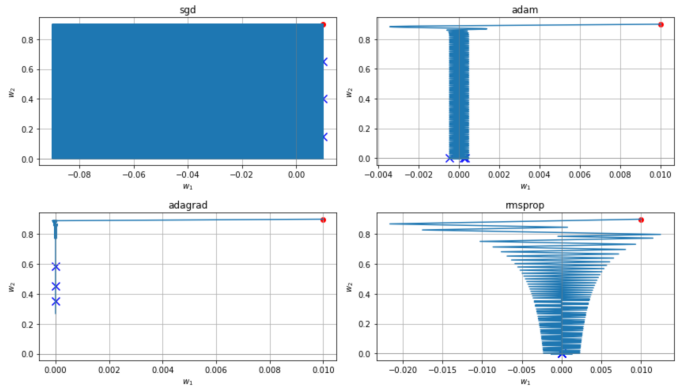


(a) 2-dimensional dynamics near a smooth valley

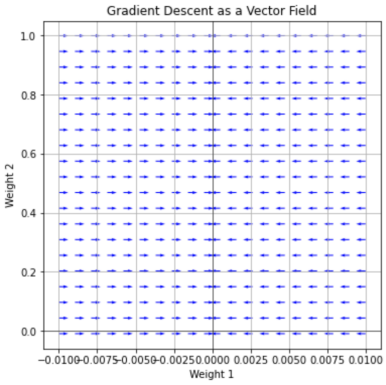


(b) Associated vector field

Figure 3: Comparing optimizers on a smooth valley



(a) 2-dimensional dynamics near a sharp valley



(b) Associated vector field

Figure 4: Comparing optimizers on a sharp valley

was within 10^{-3} of the origin in 3113 steps, whereas RMSProp achieved the same feat in 1086 steps. However, Adam doesn't converge.

Similarly for Figure 4, the only difference is that the function uses absolute values instead of quadratics to form what we term a *sharp* valley, $f(x, y) = |x| + 10^{-4}|y|$. The result finds that SGD is uselessly oscillating about its initial point, AdaGrad follows closest to the gradient flow solution but runs out of time, and both Adam and RMSProp find their way to the origin very quickly. Adam quickly falls into a stable oscillating behavior along w_1, whereas RMSProp has decaying oscillations all the way to the origin. In that way, RMSProp resembles the dynamics of an underdamped mechanical system, which we know converges eventually, whereas Adam gets stuck in its stable periodic solution and thus doesn't converge. Further analysis also reveals that RMSProp again converges faster to within 10^{-3} of the origin, doing so in 220 steps, compared to Adam which needed 900 steps.

In Figures 3 and 4, it is difficult to tell whether Adam, RMSProp or SGD in the sharp vase converge. To attempt to visualize the oscillatory behavior of each method, Figure 5 plots the evolution of w_1 for each optimizer in both types of valleys. It seems that Adam is the only optimizer that doesnt appear to converge to $w_1 = 0$ in either setting. We capture this observation with Proposition 4.

Proposition 4. *Adam has a 2-periodic fixed point if*

$$\exists w, C \ s.t. \quad \nabla\mathcal{L}\left(w - \eta\frac{C(1-\beta_1)}{(C^2+\epsilon)(1+\beta_1)}, f\right) = -\nabla\mathcal{L}(w,f) = C$$

Proof The intuition behind this proof is that the drift which enables Adam to cross flats also gives leads to an oscillating behavior. First let's show that Adam has a periodic point of period 2. Let $\nabla\mathcal{L}(w_t, f) = (-1)^t C$, for some constant C.

$$\begin{cases} g_t = (1-\beta_1)\sum_{i=0}^{\infty}(-1)^{t+i}C\beta_1^i = (-1)^t C\frac{1-\beta_1}{1+\beta_1}, \\ \sigma_t^2 = (1-\beta_1)\sum_{i=0}^{\infty} C^2\beta_2^i = C^2. \end{cases}$$

Thus we see that σ_t tends to a constant, and g_t is 2-periodic. This would mean

$$w_{t+2} - w_t = -\eta\frac{g_t}{\sigma_t+\epsilon} - \eta\frac{g_{t+1}}{\sigma_{t+1}+\epsilon} = -\frac{\eta}{C^2+\epsilon}(C-C)\frac{1-\beta_1}{1+\beta_1} = 0$$

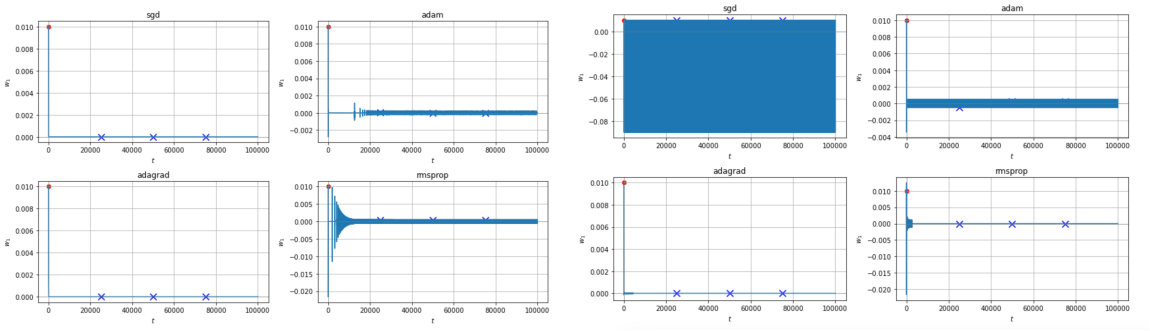
For these dynamics to be possible, there needs to exist a pair of points that are one step from each other such that their gradients are opposites, i.e. if

$$\exists w, C \text{ s.t.} \quad \nabla\mathcal{L}\left(w - \eta\frac{C(1-\beta_1)}{(C^2+\epsilon)(1+\beta_1)}, f\right) = -\nabla\mathcal{L}(w,f) = C$$

■

I only had enough time prove the existence of a 2-periodic fixed point, however it is worth noting that all other gradient descent methods are susceptible to 2-periodic oscillation, which is why the PDEs community uses robust solvers with adaptive time steps. Nevertheless, Proposition 4 serves as a prototype for the k-periodic solutions that Adam is susceptible to precisely because the moving average is not an unbiased estimate of the gradient. In Figure 5a is 2-periodic but in Figure 5b it is 12-periodic. That could be a deal-breaker in some applications but, again, there is substantial empirical support for using Adam to train neural networks, so the impact of these large periodic issues must be smaller than the other errors associated with deep learning.

Since RMSProp actually converges and does so faster than Adam, valley's can not explain the use of Adam over RMSProp.



(a) Oscillations in the smooth valley

(b) Oscillations in the sharp valley

Figure 5: Oscillations of each optimizer

7 Conclusion

The reason for Adam's dominance in the market of optimizers for deep neural networks remains a mystery. In this paper, we explored how the bias of the moving average of the gradients could benefit Adam in cases where it might need to cross a flat, or to escape a saddle. However, other explanations such as the noise reduction and its speed in valley's appear unfavorable.

References

F. Geb A. Jacot, B. Şimşek, C. Hongler, and F. Gabriel. Saddle-to-saddle dynamics in deep linear networks: Small initialization training, symmetry, and sparsity. 2022.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys, 2017.

D.P. Kingma and J. Lei Ba. Adam: A method for stochastic optimization. 2017.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets, 2018.