Comparing AMSGrad with Adam

Jason Woo
Tandon School of Engineering
New York University
Brooklyn, NY 11201
jwoo@nyu.edu

Kevin Xu
Tandon School of Engineering
New York University
Brooklyn, NY 11201
sx670@nyu.edu

Abstract

This report provides an overview of two gradient descent methods. Adam, the ubiquitous optimizer used in neural networks, as well as a follow up method known as AMSGrad, which aims to correct errors made in the original Adam analysis. Along with an overview of the two optimization methods, we provide experimental results using simple convex optimization problems, such as linear and logistic regression, to compare the convergence rates of both Adam and AMSGrad to see if, in practice, the corrected version of Adam can actually result in worse run time.

1 Introduction

In neural networks, there are a host of optimizers that aim to quicken convergence rate to a global minimum. While much of the analysis of these methods reside in the convex optimization setting, many of the real world applications using optimization techniques deal with non convex problems. Due to an explosion of the amount of data, as well as the number of trainable parameters, most modern methods use stochasticity to achieve tenable convergence rates. To achieve even faster convergence rates, accelerative methods "that scale coordinates of the gradient by square roots of some form of averaging of the squared coordinates in the past gradients" [1], such as Nesterov's accelerated gradient descent have been proven to achieve better performance.

For example, in class we proved that in the convex setting, assuming the loss function is α strongly convex and β smooth, if the total number of iterations (T) for gradient descent is $T = O(\beta/\alpha log(R\beta/\epsilon))$ such that R represents initial bounded distance and ϵ is an error term, then we have $f(x^T) - f(x^*) \leq \epsilon$. However, using Nesterov's accelerated gradient descent, if $T = O(\sqrt{\beta/\alpha} log(\beta/\alpha/\epsilon)$, then we get the same convergence guarantee, in a smaller number of iterations. Stochastic methods allow for similar convergence guarantees, except that the per iteration cost can be significantly less, resulting in faster overall run time. This is because each iteration of gradient descent requires looking at all the training data for one step, and stochastic gradient descent only requires looking at one example for a gradient step. Variations on this stochastic method include mini-batch gradient descent where each gradient update step is made using several examples from the training data, not just one in the stochastic setting or all the data in regular gradient descent. Adam (adaptive moment estimation) and AMSGrad utilize stochastic gradient descent with acceleration, but also use other techniques to result in faster convergence.

AMSGrad corrects a mistake made it the analysis of Adam, but Adam still is still widely used over AMSGrad. The two are so similar that in TensorFlow, AMSGrad is defined as a parameter to Adam.

```
tf.keras.optimizers.Adam(
    learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07, amsgrad=False,
    name='Adam', **kwargs
)
```

A potential reason for this being the case is because it was discovered that Adam fails to converge in a very simple one-dimensional convex setting. However, in practice, Adam is not used in such a simple and specific setting. It is widely used in deep neural networks with complicated, non-convex loss functions.

Problem The problem we are trying to tackle is when does AMSGrad do worse than Adam, in terms of convergence? There have been several experimental studies conducted outlining the convergence rates of Adam and AMSGrad on different loss functions on real world data and networks, such as the Vgg netowrk on CIFAR-10 data and Logistic Regression on MNIST data. In these real world examples, the benefits of AMSGrad has been negligible, but these studies have not shown AMSGrad doing worse than Adam. We will run experiments to see if this is possible.

2 Adam

It will be difficult to discuss the methods involved with Adam without first discussing RMSprop (Root Mean Square Prop), the algorithm that contributed to the motivation for Adam. At the core of RMSProp is exponential weighted averages of the squared gradients. Given weights, W, bias, b and derivative, d, RMSProp's update rule is defined as follows:

$$\begin{aligned} S_{dW} &\leftarrow \beta \cdot S_{dW} + (1-\beta) \cdot dW^2 \\ S_{db} &\leftarrow \beta \cdot S_{db} + (1-\beta) \cdot db^2 \\ W &\leftarrow W - \alpha \frac{dW}{\sqrt{S_{dW}} + \epsilon} \\ b &\leftarrow b - \alpha \frac{db}{\sqrt{S_{db}} + \epsilon} \end{aligned} \tag{1}$$

β is a hyper-parameter that controls how far back the moving average is affected by previous gradient information. It is generally fixed at .9. ϵ is located in the denominators just in case the derivative is equal to zero.

Adam is a modification to RMSProp, such that it uses the squared gradient as well as the regular gradient in its update step. Adam still uses exponential moving averages, and its update step at iteration t for parameter θ is defined below.

$$\begin{aligned} g_t &\leftarrow \nabla_\theta f_t(\theta_{t-1}) \\ m_t &\leftarrow \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t \\ v_t &\leftarrow \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2 \\ \hat{m}_t &\leftarrow m_t / (1-\beta_1^t) \\ \hat{v}_t &\leftarrow v_t / (1-\beta_2^t) \\ \theta_t &\leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon) \end{aligned} \tag{2}$$

The main differences between Adam and RMSProp is that the bias update is deprecated, it uses both the squared (element wise) gradient as well as the non squared gradient, and does bias correction. Also, according to Kingma and Lei Ba in the original paper for Adam, "RMSProp with momentum generates its parameter updates using a momentum on the re-scaled gradient, whereas Adam updates are directly estimated using a running average of first and second moment of the gradient." [1]

2.1 Adam Convergence Analysis

Just as we proved online gradient descent method in class, such that regret is ϵ and $f(x^T) - f(x^*) \leq \epsilon$ for some satisfactory number of iterations, T, Adam is also proved using a regret bound. Regret is formally defined as follows:

$$R(T) = \sum_{t=1}^{T} [f_t(\theta_t) - f_t(\theta^*)] \tag{3}$$

such that θ_t is the weight parameter at iteration t and $\theta^* = argmin_{\theta \in X} \sum_{t=1}^{T} f_t(\theta)$. Regret captures the difference of each iteration's loss from the optimal solution's loss. If $R(T) < O(T)$, then on average the optimization will converge to an optimal solution because this will result in $R(T)/T \leq c/T^x$ for some constant, c, and some $x < 1$. As $T \to \infty$ then $R(T) \to 0$. Given the complexity of their results, I will omit the derivation of the theorems used to prove Adam's convergence and instead provide them without proof. Given the necessary assumptions:

$$R(T) \leq \frac{D^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} \sqrt{T \hat{v_{T,i}}} + \frac{\alpha(1+\beta_1) G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} ||g_{1:T,i}||_2 + \sum_{i=1}^{d} \frac{D_\infty^2 G_\infty \sqrt{1-\beta_2}}{2\alpha(1-\beta_1)(1-\lambda)^2} \tag{4}$$

Making the necessary assumptions, this results in:

$$\frac{R(T)}{T} \leq O\left(\frac{1}{\sqrt{T}}\right) \tag{5}$$

3 AMSGrad

As mentioned earlier, stochastic optimization algorithms that rely on the exponential moving averages of squared past gradients such as Adam fail to converge to the optimal solution in some settings despite their excellent practical performance. To fix the convergence issue, AMSGrad is proposed as a new variant of Adam that keeps a long-term memory of past gradients and can also be implemented in the same time and space complexity as the original Adam algorithm.

The following quantity:

$$\Gamma_{t+1} = \left(\frac{\sqrt{V_{t+1}}}{\alpha_{t+1}} - \frac{\sqrt{V_t}}{\alpha_t}\right) \tag{6}$$

measures the change in the inverse of learning rate with respect to time. Γ_t needs to be positive semi-definite in order to obtain a non-increasing learning rate that is critical for the convergence of the algorithm. In the original paper of Adam, it is incorrectly assumed that Γ_t is always positive semi-definite. In reality, unlike the traditional SGD method, the update rule for Adam does not guarantee a non-increasing learning rate , which leads to convergence issue in some settings.

It is shown that Adam fails to converge even in a one-dimensional convex setting. Consider the following functions for $\mathcal{F} = [-1, 1]$ and $x \in \mathcal{F}$:

$$f_t(x) = \begin{cases} Cx, & \text{for } t \bmod 3 = 1 \\ -x, & \text{otherwise} \end{cases} \tag{7}$$

where $C > 2$, and we set $\beta_1 = 0$ and $\beta_2 = 1/(1 + C^2)$. Adam converges to a sub-optimal solution of $x = +1$, but the minimum regret is found at the point $x = -1$. An intuitive explanation would be that the large gradient C is only obtained every 3 steps and scaled down by a factor of almost C, while the other two steps move x in the wrong direction with gradients -1. As a result, the algorithm gradually moves x into the opposite direction and converges to 1 instead of -1.

As a variant of Adam, AMSGrad is also an algorithm based on exponential moving averages of past gradients with slight changes to the update rules to guarantee convergence. To see the difference between Adam and AMSGrad, let's put their update rules side by side for a better comparison:

Algorithm 1 ADAM

while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
$\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$
$\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$
end while

Algorithm 2 AMSGrad

for $t = 1$ to T **do**
$g_t = \nabla f_t(x_t)$
$m_t = \beta_{1t} m_{t-1} + (1 - \beta_1) g_t$
$v_t = \beta_{2t} v_{t-1} + (1 - \beta_2) g_t^2$
$\hat{v}_t = max(\hat{v_{t-1}}, v_t)$ and $\hat{V}_t = diag(\hat{v}_t)$
$x_{t+1} = \Pi_{\mathcal{F}, \sqrt{\hat{V}_t}}(x_t - \alpha_t m_t / \sqrt{\hat{v}_t})$
end for

As we can see, the key difference between the two algorithms is that AMSGrad keeps the maximum value of v_t up to the present time step, and the maximum value of v_t is used to update our parameters instead of the bias-corrected v_t used in Adam. By doing that, we would have a non-increasing learning rate regardless of the gradient at the next next time step. To better illustrate, consider a case at time t and $i \in [d]$ where we have $v_{t-1,i} > g_{t,i}^2 > 0$. With Adam, the learning rate would increase because the larger value $v_{t-1,i}$ is scaled down and, therefore, decreases v_t. In contrast, the learning rate of AMSGrad is maintained since the maximum value of v_t is kept in memory.

AMSGrad preserves the intuition of Adam to gradually decay the effect of past gradients on the learning rate while successfully achieving an non-increasing step size by keeping a "long-term memory" of past gradients.

3.1 AMSGrad Convergence Analysis

A similar analysis from section 2.1 on the regret bound of AMSGrad is provided in this section. As mentioned before, we are looking for a regret better than $O(T)$ because if $R(T) < O(T)$, the average regret converges, and the algorithm converges to an optimal solution as a result. Given the necessary assumption:

$$R(T) \leq \frac{D_\infty^2 \sqrt{T}}{\alpha(1-\beta_1)} \sum_{i=1}^{d} \hat{v_{T,i}^{1/2}} + \frac{D_\infty^2}{(1-\beta_1)^2} \sum_{t=1}^{T} \sum_{i=1}^{d} \frac{\beta_{1t} \hat{v_{T,i}^{1/2}}}{\alpha_t} + \frac{\alpha\sqrt{1+logT}}{(1-\beta_1)^2(1-\gamma)\sqrt{1-\beta_2}} \sum_{i=1}^{d} ||g_{1:T,i}||_2 \tag{8}$$

If we set $\beta_{1t} = \beta_1 \lambda^{t-1}$ in equation 8, then we have:

$$R(T) \leq \frac{D_\infty^2 \sqrt{T}}{\alpha(1-\beta_1)} \sum_{i=1}^{d} \hat{v_{T,i}^{1/2}} + \frac{\beta_1 D_\infty^2 G_\infty}{(1-\beta_1)^2(1-\lambda)^2} + \frac{\alpha\sqrt{1+logT}}{(1-\beta_1)^2(1-\gamma)\sqrt{1-\beta_2}} \sum_{i=1}^{d} ||g_{1:T,i}||_2 \tag{9}$$

When $\sum_{i=1}^{d} \hat{v_{T,i}^{1/2}} \ll \sqrt{d}$ and $\sum_{i=1}^{d} ||g_{1:T,i}||_2 \ll \sqrt{dT}$, the above bound is better than the $O(\sqrt{dT})$ bound of SGD, and if we use a decay rate of $\beta_{1t} = \beta_1 / t$, a regret bound of $O(\sqrt{T})$ can be obtained. Then we can prove that the average regret of AMSGrad converges:

$$\frac{R(T)}{T} \leq O\left(\frac{1}{\sqrt{T}}\right) \tag{10}$$

The performance of AMSGrad with real world data is still doubted by many people since the non-convergence of Adam in practice is relatively rare. Some found that AMSGrad yields similar results to Adam, while others believe that AMSGrad actually performs worse than Adam in a lot of cases.

4 Experiment: Reddi et al. Loss Function

In the paper proposing AMSGrad by Reddi et al., an example of a convex loss function which Adam fails to converge is provided in equation 7. The deterministic nature of this piece wise loss function is concerning because as we proved stochastic gradient descent in class, the loss function requires a finite sum structure: $f(x) = \sum_{i=1}^{n} f_i(x)$ with $f_1, ..., f_n$ all being convex. The function parameters, x are updated at iteration $i + 1$ by choosing a random $j_i \in 1, ..., n$ and setting $x^{(i+1)} = x^{(i)} - \eta \nabla f_{j_i}(x^i)$.

The proposed setting where Adam fails to converge does not satisfy the assumptions required to prove stochastic gradient descent using an online method. As exploratory analysis, we change the the proposed loss function to be random to see if Adam still fails to converge. The script used for this was adapted from code provided in a Medium article: Adam — latest trends in deep learning optimization, written by Vitaly Bushaev.

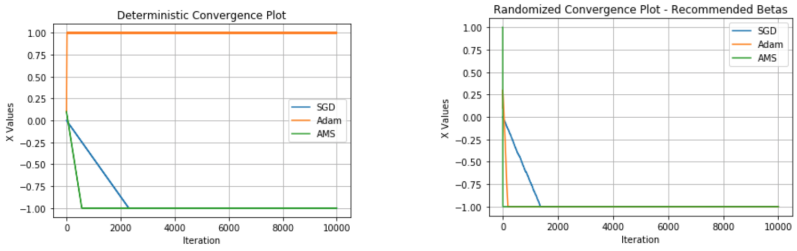
```
def grad_determ(t, C):

  if t % 3 == 1:
      return C
  else:
      return -1
```

Listing 1: Gradient to Loss Function

```
def grad_random(C):
    rand_int = random.randint(1, 2)
    if rand_int > 1:
        return C
    else:
        return -1
```

Listing 2: Gradient to Random Loss Function

The difference in code to implement the two methods is slight. The original code requires on every third iteration, t, the gradient to be C and at every other iteration the gradient to be -1. To make this random, we make a random variable that dictates the output to the function. A notebook outlining the full methodology can be found at on my **GitHub**. It is known that Adam will not converge using the deterministic method, plot shown below. This was also supported in the randomized method as well. However, when setting the parameters $\beta_1 = .9, \beta_2 = .999$, their recommended values, Adam manages to converge, in both the deterministic on non deterministic methods.



(a) Deterministic

(b) Random

Figure 1: Convergence Plots

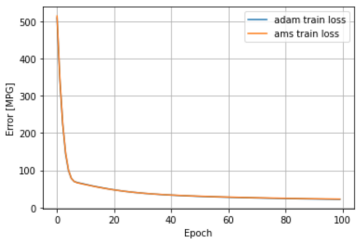
The purpose of this experiment was to see if the proposed loss function would fail in a random setting. If it would fail in the random setting, then I would question the results of Reddi et al. The results of this experiment are still interesting. It provides a simple example of why Adam still performs just as well AMSGrad in practice. The recommended values $\beta_1 = .9$ and $\beta_2 = .999$ ensure enough previous gradient information is taken into account in the update step. In exponential moving averages, the β terms affect how many previous gradients we average over. With $\beta_1 = .9$, the previous 10 gradients are averaged and with $beta_2$, the previous 1000 squared gradients are averaged. In the simple loss function proposed by Reddi et al., these hyper-parameters are enough for the algorithm to converge. In the following sections, we provide more experiments to find if there are instances where AMSGrad performs worse than Adam in very specific situations.

5 Experiment: Linear Regression

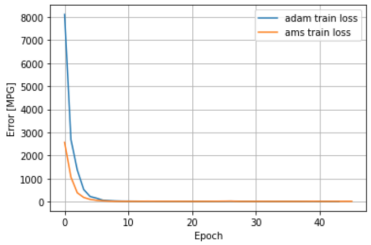
After running the above experiment, we became curious if we can synthesize a situation where AMSGrad does worse than Adam. We move away from the simple 1-dimensional setting and run Adam and AMSGrad on linear regression using mean squared error as the loss function, which is still convex. Using TensorFlow, we create a network with one fully connected layer with a single output. This represents linear regression. An example of the model is provided below and the notebook can be found in my notebook on **Google Colab**.

```
linear_model = tf.keras.Sequential([
    layers.Dense(units=1)
])

linear_model.compile(
    optimizer=tf.optimizers.Adam(learning_rate=0.1, amsgrad=True),
    loss='mse')
```

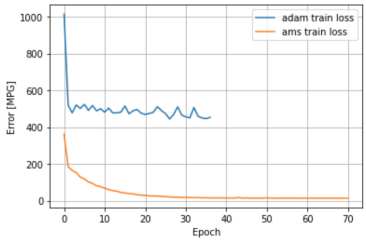
In this synthesized problem, we manipulate P so that the inputted data, AP, has varying condition number. We use the Jacobi preconditioner, P, as a baseline. The data matrix is defined with A, $D = diag(A^T A)$ and $P = \sqrt{D^{-1}}$. We also create P using various other methods. A complete guide can be referenced in the notebook linked above. Two plots are included:



(a) Jacobi Preconditioner



(b) Random Gaussian P



(c) Random Gaussian P - Bad Betas

Figure 2: Convergence Plots for Different P - Linear Regression

From the figures, we can see that Adam and AMSGrad have almost identical convergence using the Jacobi preconditioner. The two gradient methods also produce nearly identical results using a random Gaussian preconditioner, with AMSGrad resulting in slightly faster convergence than Adam. These were taken in only one iteration, an overall converegence rates were compared over many different random instances of P.

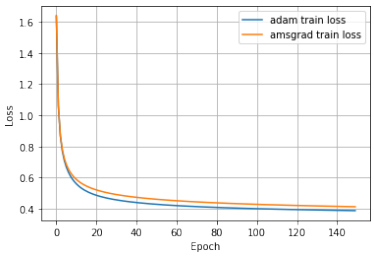
Very similarly to experiment 1, if the hyper-parameters for Adam and AMSGrad are set to not recommended values, $\beta_1 \neq .9$ and $\beta_2 \neq .999$, then Adam again fails to converge and AMSGRad again manages to converge. From this experiment, we could not find an instance where AMSGRad does worse than Adam. We are only reassured that AMSGrad is more robust than Adam, in terms the setting of its hyperparameters.

6 Experiment: Logistic Regression

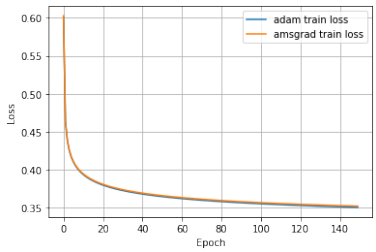
We will run experiments similar to that of linear regression on logistic regression models. Logistic regression was implemented and applied on the Fashion MNIST dataset. Using TensorFlow, a single layer network was created where flattened image vectors are fed into a Softmax layer and output the classified label. An example of the model is provided below and the complete code can be found on **Google Colab**.

```
logistic_model = tf.keras.Sequential()

numUnits_L1 = 10
layer1 = tf.keras.layers.Dense(
    units=numUnits_L1, activation=tf.nn.softmax, use_bias=True, name='Logistic')
logistic_model.add(layer1)

logistic_model.compile(
    optimizer=tf.optimizers.Adam(learning_rate=0.005, amsgrad=True),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy'])
```

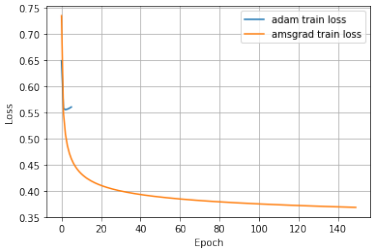
Like we did in the previous section, P was manipulated and the Jacobi preconditioner was calculated as a baseline. Different P were used to evaluate the convergence of Adam and AMSGrad, and the results were plotted and shown below:



(a) Jacobi Preconditioner



(b) Random P



(c) Random P - Bad Betas

Figure 3: Convergence Plots for Different P - Logistic Regression

As we can see from the figures, Adam and AMSGrad have almost identical convergence when P is a randomly generated diagonal matrix. Using the Jacobi preconditioner, Adam converges slightly faster than AMSGrad and obtains a better loss.

When the betas are set to undesired values, we also observed that Adam can fail to converge in logistic regression.

7 Conclusion

From our exploratory analysis, we failed to find a setting where AMSGrad can perform worse than Adam. We showed this by first looking at the loss function proposed by Reddi et al. in their paper

outlining Adam's mistake. We were originally skeptical of the proposed loss function where Adam fails to converge because of its deterministic nature. Stochastic gradient descent assumes that the loss function has a finite sum structure and that a random element of this finite sum is chosen at time step t of the algorithm. From a simple experiment, we show that it does not matter whether the proposed loss in deterministically or randomly chosen. Adam fails to converge given certain values of β_1 and β_2 in either setting.

This led us to look at other loss functions, such as mean squared error, using linear regression. In class we discussed preconditioning as a way to reduce the condition number of data. Using this idea, we try to find instances of preconditioning that would "break" AMSGrad and favor Adam. Similarly as in out first experiment, we could not show AMSGrad doing worse than Adam, instead, we provide another example of how Adam is brittle, in terms of its hyper-parameters. Adam and AMSGrad perform almost identically when its hyper-parameters are set as suggested, $\beta_1 = .9$ and $\beta_2 = .999$. However, when choosing $\beta_1 = 0$ and $\beta_2 = .05$, only AMSGrad manages to converge while Adam fails to do so. An intuitive explanation of this is because the β terms dictate how far back in the past the algorithm uses previous gradient information. With high values of β, the algorithm averages over many past iterations. This is where the error in convergence guarantee for Adam is highlighted.

When we run the same experiments with logistic regression models, we found that when using the Jacobi preconditioner, Adam does obtain a slightly better loss after the same number of iterations. However, AMSGrad converges at almost the same rate and the advantage is not significant enough to conclude that AMSGrad performs worse in this setting.

References

[1] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Proceedings of 3rd International Conference on Learning Representations, 2015.

[2] Reddi, S.J.; Kale, S.; Kumar, S. On the Convergence of Adam and Beyond. arXiv 2014, arXiv:1904.09237

[3] Alexandre Défossez, Léon Bottou, Francis Bach, and Nicolas Usunier. On the convergence of adam and adagrad. arXiv preprint arXiv:2003.02395, 2020