

Backpropagation and Optimizers

Due: Friday, November 7th at 11:59 pm

Deliverables. Submit a PDF of your write-up to Gradescope *HW3 Written*

Note: we highly discourage very long answers, your free response answers should be 1-3 sentences.

Jincheng Gu.

Honor Code

Write and sign the following statement:

"I certify that all solutions in this document are entirely my own and that I have not looked at anyone else's solution. I have given credit to all external sources I consulted."

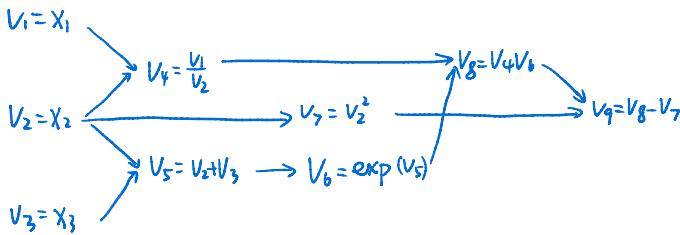
Backpropagation Practice

Let's practice calculating the gradient using adjoints. We're given the following function and want to find the gradient of the function with respect to $\mathbf{x} = [x_1 \ x_2 \ x_3]$:

$$f(\mathbf{x}) = \frac{x_1}{x_2} e^{x_2+x_3} - x_2^2$$

Q1. Constructing the Computation Graph Draw the computation graph (check out lecture 12 for reference). We recommend handwriting it instead of using LATEX.

$$\begin{aligned} v_1 &= x_1 \\ v_2 &= x_2 \\ v_3 &= x_3 \\ v_4 &= \frac{v_1}{v_2} \\ v_5 &= v_2 + v_3 \\ v_6 &= \exp(v_5) \\ v_7 &= v_2^2 \\ v_8 &= v_4 v_6 \\ v_9 &= v_8 - v_7 \end{aligned}$$



Q2. Computing Adjoints Now, compute the adjoint for each node in your computation graph with respect to $\mathbf{x} = [x_1 \ x_2 \ x_3] = [1 \ 2 \ 3]$. What is the gradient with respect to \mathbf{x} ? We recommend that you use your computation graph and write it out instead of typing.

$$\begin{array}{ll} x_1=1 \rightarrow v_1=x_1 & v_4=\frac{v_1}{v_2} \\ (v_1=1) & \\ x_2=2 \rightarrow v_2=x_2 & v_7=v_2^2 \\ (v_2=2) & \\ x_3=3 \rightarrow v_3=x_3 & v_5=v_2+v_3 \rightarrow v_6=\exp(v_5) \\ (v_3=3) & \\ \end{array} \quad \begin{array}{l} v_8=v_4 v_6 \\ v_9=v_8-v_7 \end{array} \quad \begin{array}{l} v_4=\frac{1}{2} \\ v_5=5 \\ v_6=e^5 \\ v_7=4 \\ v_8=\frac{1}{2}e^5 \end{array}$$

$$f = v_9 \quad \frac{\partial f}{\partial v_9} = 1 \text{ (Base Case)}$$

$$\frac{\partial f}{\partial x_1} = \frac{\partial f}{\partial v_1} = \frac{\partial f}{\partial v_4} \left(\frac{\partial v_4}{\partial v_1} \right) = \left(\frac{1}{v_2} \right) \frac{\partial f}{\partial v_8} \left(\frac{\partial v_8}{\partial v_4} \right) = \left(\frac{1}{v_2} \right) \frac{\partial f}{\partial v_9} \left(\frac{\partial v_9}{\partial v_8} \right) = \frac{v_6}{v_2} \cdot 1 = \frac{v_6}{v_2} = \frac{e^5}{2}$$

$$\frac{\partial f}{\partial x_2} = \frac{\partial f}{\partial v_2} = \frac{\partial f}{\partial v_4} \left(\frac{\partial v_4}{\partial v_2} \right) + \frac{\partial f}{\partial v_5} \left(\frac{\partial v_5}{\partial v_2} \right) + \frac{\partial f}{\partial v_7} \left(\frac{\partial v_7}{\partial v_2} \right) = \left(\frac{v_1}{v_2^2} \right) \frac{\partial f}{\partial v_8} \left(\frac{\partial v_8}{\partial v_4} \right) + (2v_2) \frac{\partial f}{\partial v_9} \left(\frac{\partial v_9}{\partial v_8} \right) + \frac{\partial f}{\partial v_9} \left(\frac{\partial v_9}{\partial v_7} \right) = -\frac{v_1 v_6}{v_2^2} - 2v_2 + v_6 e^{v_5} = -\frac{e^5}{4} - 4 + \frac{1}{2} e^5 = \frac{e^5}{4} - 4$$

$$\left(-\frac{v_1 v_6}{v_2^2} \right) \frac{\partial f}{\partial v_8} \left(\frac{\partial v_8}{\partial v_4} \right) = 2v_2 \cdot 1 \cdot (-1) = -2v_2$$

$$e^{v_5} \frac{\partial f}{\partial v_8} \left(\frac{\partial v_8}{\partial v_4} \right)$$

$$v_6 e^{v_5} \frac{\partial f}{\partial v_8} \left(\frac{\partial v_8}{\partial v_4} \right)$$

$$\frac{\partial f}{\partial x_3} = \frac{\partial f}{\partial v_3} = \frac{\partial f}{\partial v_5} \left(\frac{\partial v_5}{\partial v_3} \right) = \frac{\partial f}{\partial v_6} \left(\frac{\partial v_6}{\partial v_5} \right) = e^{v_5} \cdot \frac{\partial f}{\partial v_8} \left(\frac{\partial v_8}{\partial v_6} \right) = v_6 \cdot e^{v_5} = \frac{e^5}{2}$$

$$\therefore \nabla f = [v_1 \ v_2 \ v_3] = [\frac{e^5}{2} \ \frac{e^5}{2} - 4 \ \frac{e^5}{2}]$$

Optimizers Overview

Adaptive gradient methods are among the most widely used optimization algorithms in deep learning. In this assignment, we focus on two main algorithms: **Adam** and **AdamW** (briefly AdaGrad, RMSProp). Because the primary sources are dense, please start with the short blog posts below, then (optionally) consult the original papers.

Recommended reading (short):

- Understanding Deep Learning Optimizers: Momentum, AdaGrad, RMSProp, Adam — quick refresher on the progression from SGD to Adam.
- Adam vs. AdamW (A. Bennaceur) — concise comparison highlighting why decoupled weight decay matters.
- **Supplement (Section 3):** CS182 Discussion 2 Solutions (Spring 2021) — see Section 3 for a compact optimizer summary.

Optional primary sources:

- Adam: A Method for Stochastic Optimization (Kingma & Ba, 2015)
- Decoupled Weight Decay Regularization (Loshchilov & Hutter, 2017)

Before answering the questions, also briefly review the original Adam algorithm (Algorithm 1 from Kingma & Ba, 2014) and AdamW, reproduced below for reference.

Algorithm 1 Adam, our proposed algorithm for stochastic optimization (Kingma & Ba, 2014)

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

```

1:  $m_0 \leftarrow 0$                                 ▷ Initialize 1st moment vector
2:  $v_0 \leftarrow 0$                                 ▷ Initialize 2nd moment vector
3:  $t \leftarrow 0$                                  ▷ Initialize timestep
4: while  $\theta_t$  not converged do
5:    $t \leftarrow t + 1$ 
6:    $g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$           ▷ Get gradients
7:    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$  ▷ EMA of gradients
8:    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$  ▷ EMA of squared gradients
9:    $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$            ▷ Bias-corrected 1st moment
10:   $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$            ▷ Bias-corrected 2nd moment
11:   $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$  ▷ Update
12: end while
13: return  $\theta_t$ 

```

Algorithm 2 Adam with L2 regularization and Adam with decoupled weight decay (AdamW)

Require: $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, $\lambda \in \mathbb{R}$

- 1: **initialize** $t \leftarrow 0$, parameter vector $\theta_{t=0} \in \mathbb{R}^n$, first moment $m_{t=0} \leftarrow 0$, second moment $v_{t=0} \leftarrow 0$, schedule multiplier $\eta_{t=0} \in \mathbb{R}$
- 2: **repeat**
- 3: $t \leftarrow t + 1$
- 4: $\nabla f_t(\theta_{t-1}) \leftarrow \text{SELECTBATCH}(\theta_{t-1})$
- 5: $g_t \leftarrow \nabla f_t(\theta_{t-1}) + \lambda \theta_{t-1}$ ▷ (L2-regularized Adam) add penalty inside gradient
- 6: $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
- 7: $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
- 8: $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$
- 9: $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$
- 10: $\eta_t \leftarrow \text{SETSCHEDULEMULTIPLIER}(t)$
- 11: $\theta_t \leftarrow \theta_{t-1} - \eta_t \left(\alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} + \lambda \theta_{t-1} \right)$ ▷ (AdamW) decoupled decay
- 12: **until** stopping criterion is met
- 13: **return** optimized parameters θ_t

Q3. Reviewing Previous Optimizers. We have learned how variants of gradient descent improve training speed and stability. For each method — **Momentum**, **AdaGrad**, and **RMSProp** — write the update equations, state the key idea (one concise sentence), and describe the problem it solves (one concise sentence). We have provided SGD for you.

SGD

Update: $\theta_t \leftarrow \theta_{t-1} - \alpha g_t$, where $g_t = \nabla_{\theta} f_t(\theta_{t-1})$.

Key idea: Use a random minibatch each step for computational efficiency.

Problem it solves: Full-batch gradient descent is slow and redundant; minibatching makes updates cheaper.

Momentum: $\theta_t \leftarrow \theta_{t-1} - \alpha m_t$, where $m_t = \beta m_{t-1} + (1-\beta) \nabla_{\theta} f_t(\theta_{t-1})$

Key idea: Maintain a exponentially weighted moving average of past gradients "指数加权过去梯度"

Problem it solves Reduces noisy, oscillatory updates, and accelerates progress along consistent gradient directions

AdaGrad: $v_t \leftarrow v_{t-1} + g_t^2$, $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \frac{g_t}{\sqrt{v_t + \epsilon}}$

key idea: Accumulate squared gradients to adapt the learning rate per parameter.

Problem it solves: Frequent large gradients shrink their steps, while sparse ones retain large steps

RMSProp: $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$, $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \frac{g_t}{\sqrt{v_t + \epsilon}}$

Key idea: Use EWMA (指数加权过去梯度) of squared gradients for adaptive step sizes.

Problem it solves: Prevents AdaGrad's steps from decaying too quickly by tracking recent gradient magnitudes.

Q4. Handling Large Gradients. What happens when gradients are very large in plain gradient descent, and how does Adam's update rule address this issue?

When gradients are very large, in plain GD, parameter updates can be very large, leading to instability or divergence of training.

Adam mitigates this by $\frac{m_t}{\sqrt{v_t + \epsilon}}$. Since it grows with squared gradient, large gradient get scaled down, preventing excessively large updates. It keeps the optimization stable even when gradients are very large.

Q5. Introducing AdamW. How does AdamW decouple weight decay from the gradient update?

Why does this help generalization?

AdamW applies weight decay directly as a parameter update step.

This preserves the desired regularization effect, regardless of adaptive scaling, leading to better generalization.

Q6. AdamW Weight Decoupling. How does decoupling weight decay affect hyperparameter sensitivity?

In original Adam, adaptive scaling and L_2 regularization made weight decay unpredictable and sensitive to learning rate.

In AdamW, it's decoupling makes weight decay stable and independent. This makes tuning easier and results more consistent across tasks.

Q7. Tracing Adam's Origins (Line Mapping). As seen in lecture, Adam builds on ideas from several earlier optimizers. Now let's explore exactly how these ideas are used in Adam. Below are the key lines (7–11) from the Adam algorithm.

For each line, label which previous algorithm inspired it (or indicate if it was newly introduced) using one of the following categories:

- Momentum
- Adaptive Scaling (RMSProp/AdaGrad)
- Newly Introduced by Adam
- Combination (merges two or more earlier ideas into a single update rule)

Briefly explain your reasoning for each label.

Algorithm 3 Adam: Core Update Lines (Kingma & Ba, 2014)

- | | |
|--|-----------|
| 1: $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$ | ▷ Line 7 |
| 2: $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ | ▷ Line 8 |
| 3: $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ | ▷ Line 9 |
| 4: $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ | ▷ Line 10 |
| 5: $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ | ▷ Line 11 |
-

Line 7: Momentum

Line 8: Adaptive Scaling

Line 9: Newly Introduced by Adam: Bias Correction

Line 10:

Line 11: Combination: Combines Line 7 and Line 8 into a single adaptive step.

Q8. Step 1 Gradient: Adam and AdamW updates. Consider the scalar objective

$$f(\theta) = \theta^2, \quad \nabla_\theta f(\theta) = g_t = 2\theta_t.$$

$$\begin{cases} M_t = \beta_1 M_{t-1} + (1-\beta_1) g_{t-1} \\ V_t = \beta_2 V_{t-1} + (1-\beta_2) g_{t-1}^2 \end{cases}$$

Use the following hyperparameters unless otherwise specified:

$$g_0 = 2\theta_0 = 20$$

$$\theta_0 = 10, \quad \alpha = 0.1, \quad \lambda = 0.01, \quad \beta_1 = 0.9, \quad \beta_2 = 0.99, \quad \epsilon = 10^{-8}, \quad m_0 = v_0 = 0$$

Compute all quantities for $t = 1$.

$$\theta_1 = \theta_0 - 0.1 \times \frac{20}{20+10^{-8}} = 10 - 0.1 \times 9.9$$

$$(a) \text{ Compute } g_1, m_1, v_1. \quad \begin{aligned} m_1 &= 0.9 \times 0 + 0.1 \times 20 = 2 \\ v_1 &= 0.99 \times 0 + 0.01 \times 400 = 4 \end{aligned} \quad \begin{aligned} \hat{m}_1 &= \frac{2}{0.9} = 20 \\ \hat{v}_1 &= \frac{4}{0.99} = 400 \end{aligned} \quad \therefore g_1 = 2\theta_1 = 19.8$$

(b) Compute bias-corrected \hat{m}_1, \hat{v}_1 (feel free to refer to Algorithm Block provided in Q1). $\hat{m}_1 = 20$

(c) Compute $\theta_1^{\text{Adam}} = \theta_0 - \alpha \frac{\hat{m}_1}{\sqrt{\hat{v}_1} + \epsilon}$ (step 1 update of Adam). $\theta_1 = 9.9$

(d) Compute $\theta_1^{\text{AdamW}} = \theta_1^{\text{Adam}} - \alpha \lambda \theta_0$ (step 1 update of AdamW).

$$(\text{AdamW}) \theta_1 = 9.9 - 0.1 \times 0.01 \times 10 = 9.89$$

Q9. Step 2 Gradient: Adam continuation ($\beta_2 = 0.99$). Compute the second update step for Adam (you don't have to compute for AdamW). In other words, compute θ_2^{Adam} .

$$\text{(Adam)} \quad \theta_2 = \theta_1 - \alpha \cdot \frac{\tilde{m}_2}{\sqrt{\tilde{v}_2} + \epsilon} \quad \alpha = 0.1 \quad m_1 = 2 \quad v_1 = 4 \quad g_1 = 19.8$$

$$\left\{ \begin{array}{l} m_2 = \beta_2 \cdot m_1 + (1-\beta_2) \cdot g_1 \\ v_2 = \beta_2 \cdot v_1 + (1-\beta_2) \cdot g_1^2 \end{array} \right. \Rightarrow \left\{ \begin{array}{l} m_2 = 0.99 \cdot 2 + 0.01 \cdot 19.8 = 3.78 \\ v_2 = 0.99 \cdot 4 + 0.01 \cdot 19.8^2 = 7.88 \end{array} \right. \quad \begin{array}{l} \tilde{m}_2 = \frac{m_2}{1-\beta_2} = 19.89 \\ \tilde{v}_2 = \frac{v_2}{1-\beta_2} = 395.98 \end{array}$$

$$\therefore \theta_2 = 9.9 - 0.1 \cdot \frac{19.89}{\sqrt{395.98} + \epsilon} = 9.8$$

Q10. Conceptual reflection. Keep the baseline at $\beta_2 = 0.99$. Now imagine we set $\beta_2 = 0.5$ (all else unchanged). Without redoing the full calculation, explain qualitatively what changes at $t = 1$ and $t \geq 2$, and how this would affect stability vs. responsiveness.

$$t=1: \text{With } m_0=v_0=0, \text{ bias-corrected } \tilde{m}_1 = \frac{(1-\beta_1)g_t}{1-\beta_1} = g_t, \quad \tilde{v}_1 = \frac{(1-\beta_2)g_t^2}{1-\beta_2} = g_t^2,$$

↓
unchanged

$t \geq 2$: smaller β_2 , \tilde{v}_t adapts very quickly to recent magnitudes.

Steps can shrink or grow rapidly with new gradient scale.

This { improves responsiveness to nonstationarity (ft)}

{ but increases noise sensitivity and jitter (ft)}

In contrast, $\beta_2=0.99$ smooths variance more, yielding to steadier but slower adaptation