
EECS 182 Deep Neural Networks

Fall 2022 Anant Sahai

Final Project Problem Set

1. Weight Decoupling, Adaptive Methods, and Finally, AdamW (1)

AdamW, by definition, is an weight-decoupled adaptive stochastic optimization method that implements weight decaying effects onto Adam (ADAPtive Moment estimation) through entering in corresponding values for the `weight_decay` argument. But if you take a look at PyTorch's documentation on **Adam**, we also see a `weight_decay` parameter for us to set - which makes one wonder, "setting this parameter for Adam should be equivalent to doing the same with AdamW - right?". In this problem, we will explore how these two methods are not equivalent and how `weight_decay` in Adam is a potential misnomer as ℓ_2 -regularization \neq weight decay in Adam. Along the way, we'll also introduce you to the idea of weight decoupling, fundamentals of adaptive gradient methods, and its relationship with ℓ_2 -regularization and weight decay. Finally, we will convince you that AdamW, a weight decoupled version of Adam, leads to a hyperparameter space that is better behaved and easier to tune.

(a) SGD and Weight Decoupling

Before taking a look into Adam and AdamW, let's look at Stochastic Gradient Descent (SGD) to familiarize the idea of weight decoupling (SDGW) and how for SGD optimization methods, ℓ_2 regularization is equivalent to weight decay - even with momentum.

- (i) Assume that we have a dataset $\{\mathbf{x}_i, y_i\}_{i=1}^n$ with data $\mathbf{x}_i \in \mathbb{R}^d$ and label $y_i \in \mathbb{R}$ and we have access to a model $f_{\theta}(\cdot)$ with parameter θ . Recall that weight decay is defined as weights (a.k.a. parameters) θ decaying exponentially per time-step t as

$$\theta_{t+1} = (1 - \lambda_w)\theta_t - \eta \nabla L_{\theta}(y_i, f_{\theta}(\mathbf{x}_i)) \Big|_{\theta=\theta_t} \quad (1)$$

for some weight decaying factor λ_w , learning rate η , initial weight value θ_0 , and loss function $L_{\theta}(\cdot, \cdot)$ evaluating on a random data point (\mathbf{x}_i, y_i) . Suppose we define a new loss function $L_{\theta}^{reg}(\cdot, \cdot)$ that has an ℓ_2 -regularization term added to our original loss function

$$L_{\theta}^{reg}(y_i, f_{\theta}(\mathbf{x}_i)) = L_{\theta}(y_i, f_{\theta}(\mathbf{x}_i)) + \frac{\lambda_{\ell_2}}{2} \|\theta\|_2^2 \quad (2)$$

for some penalization factor λ_{ℓ_2} . **Show that the update rule of SGD with the $L_{\theta}^{reg}(\cdot, \cdot)$ loss function is equivalent to performing weight decay updates with $L_{\theta}(\cdot, \cdot)$. Derive the relationship between λ_w and λ_{ℓ_2} in terms of the learning rate η and potentially other variables in the update rule.**

NOTE: Recall that the update rule of SGD under a loss function $L_{\theta}(\cdot, \cdot)$ is

$$\theta_{t+1} = \theta_t - \eta \nabla L_{\theta}(y_i, f_{\theta}(\mathbf{x}_i)) \Big|_{\theta=\theta_t} \quad (3)$$

- (ii) We saw that ℓ_2 -regularized loss function operating in SGD can have weight-decaying effect under a constraint where $\lambda_w = g(\lambda_{\ell_2}, \eta, \dots)$ for some function g . Since the weight decaying factor λ_w

is dependent on the learning rate η , we say these two parameters are *coupled*. In other words, for some optimal weight decaying factor λ_w , we would need to scale the learning rate η and ℓ_2 penalization term λ_{ℓ_2} correctly to parameterize in SGD. Therefore, we are constraining our search in hyperparameter space as one is coupled with the other.

Now, we introduce the idea of *weight decoupling* - a technique to decouple the relationship between λ_w and η in our code implementation. The weight decoupled SGD algorithm with weight decay is known as *SGDW*. SGDW allows us to search through both hyperparameter space independently. **Given the SGD algorithm with momentum optimizing an ℓ_2 -regularized loss function $L_{\theta}^{reg}(\cdot, \cdot)$ (Algorithm 1), what modification(s) should be made to turn it into SGDW with momentum for some weight decay factor λ_w ?**

HINT: Refer to (1) and ask yourself if this equation have any dependence on η .

NOTE: Keep in mind that this question additionally incorporates momentum. Therefore, be sure to also decouple hyperparameters that are used in momentum as well.

Algorithm 1 SGD with momentum optimizing ℓ_2 regularized loss function $L_{\theta}^{reg}(\cdot, \cdot)$

- 1: **given** learning rate $\eta \in \mathbb{R}_+$, ℓ_2 regularization factor $\lambda_{\ell_2} \in \mathbb{R}_+$, momentum factor $\beta_1 \in (0, 1]$, and data set $\{\mathbf{x}_i, y_i\}_{i=1}^n$.
 - 2: **initialize** time step $t \leftarrow 0$, parameter vector θ_0 , first moment (momentum) vector $\mathbf{m}_0 \leftarrow \mathbf{0}$
 - 3: **while** stopping criterion not met **do**
 - 4: $t \leftarrow t + 1$
 - 5: $\mathbf{g}_t \leftarrow \nabla L_{\theta}^{reg}(y_i, f_{\theta}(\mathbf{x}_i)) \Big|_{\theta=\theta_{t-1}}$ $\triangleright (\mathbf{x}_i, y_i)$ is the random point
 - 6: $\mathbf{m}_t \leftarrow (1 - \beta_1)\mathbf{m}_{t-1} + \beta_1\mathbf{g}_t$
 - 7: $\theta_t \leftarrow \theta_{t-1} - \eta\mathbf{m}_t$
 - 8: **end while**
 - 9: **return** optimized parameters θ_t
-

(b) **Weight Decay $\neq \ell_2$ -Regularization in Adaptive Gradient Methods (and Hopefully, Adam)**

We'll now try to prove that ℓ_2 -regularization does not give the same effect as weight decay under Adaptive Gradient Methods such as AdaGrad and RMSProp. Unfortunately, Adam cannot fall under this proof entirely due to the algorithm's additional complexity in containing momentum and using the bias-corrected momentum term for weight update. But Adam, containing adaptive methods in its core structure, can get very close to a convincing argument that weight decay $\neq \ell_2$ -regularization in which we'll summarize in the problem statement of Question 2.

- (i) Below is the pseudo-code for the **RMSProp** algorithm with bias-correcting second moment. Such is an adaptive algorithm made as close to Adam as possible while retaining its ability to prove the inequivalence between ℓ_2 -regularization and weight decay.

Adaptive methods follow a general update rule of the form:

$$\theta_t \leftarrow \theta_{t-1} - \eta \mathbf{M}_{t-1} \mathbf{g}_{t-1} \quad (4)$$

where $\mathbf{M}_t \in \mathbb{R}^{m \times m}$ and $\mathbf{g}_t \in \mathbb{R}^m$ are variables that change with time t .

Consider a function $\mathbf{G}_t : \mathbb{R}_{(0,1]} \rightarrow \mathbb{R}^{m \times m}$ that takes the exponential average (factor of $\alpha \in (0, 1]$) of the all previous gradients' outer-products:

$$\mathbf{G}_t(\alpha) = \sum_{\tau=1}^t \alpha^{t-\tau} \mathbf{g}_{\tau} \mathbf{g}_{\tau}^T \quad (5)$$

Algorithm 2 Root Mean Square Propagation (RMSProp) with bias-correcting second moment

```

1: given learning rate  $\eta \in \mathbb{R}_+$ , small value  $\epsilon$ ,  $\theta \in \mathbb{R}^m$ , second moment factor  $\beta_2 \in (0, 1]$ , and data set  $\{\mathbf{x}_i, y_i\}_{i=1}^n$ .
2: initialize time step  $t \leftarrow 0$ , parameter vector  $\theta_0$ , exponential average second moment vector  $\mathbf{v}_0 \leftarrow \mathbf{0}$ 
3: while stopping criterion not met do
4:    $t \leftarrow t + 1$ 
5:    $\mathbf{g}_t \leftarrow \nabla L_\theta(y_i, f_\theta(\mathbf{x}_i)) \Big|_{\theta=\theta_{t-1}}$   $\triangleright (\mathbf{x}_i, y_i)$  is the random point
6:    $\mathbf{v}_t \leftarrow \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) \mathbf{g}_t^2$   $\triangleright \mathbf{g}_t^2$  is component-wise squared of  $\mathbf{g}_t$ 
7:    $\hat{\mathbf{v}}_t \leftarrow \mathbf{v}_t / (1 - \beta_2^t)$   $\triangleright$  Bias-correcting second-moment;  $\beta_2$  is taken to the power of  $t$ 
8:    $\theta_t \leftarrow \theta_{t-1} - \eta [\mathbf{g}_t / (\sqrt{\hat{\mathbf{v}}_t} + \epsilon \mathbf{1}_m)]$   $\triangleright$  Division and  $\sqrt{\hat{\mathbf{v}}_t}$  here are component-wise operations.
9: end while
10: return optimized parameters  $\theta_t$ 

```

For the RMSProp algorithm with bias-correcting second moment, explicitly determine what the matrix \mathbf{M}_t is in terms of the function $\mathbf{G}_t(\alpha)$ and the second moment factor β_2 . For simplicity, assume that $\mathbf{v}_0 = \mathbf{0}$, $\epsilon = 0$, and $\mathbf{g}_t \in \mathbb{R}^m$.

HINT: Recall a couple of component-wise operations can be written as:

- $\mathbf{x}^2 = \mathbf{x} \odot \mathbf{x} = \text{diag}(\mathbf{x}\mathbf{x}^T)$
- $\mathbf{x} \odot \mathbf{y} = \text{diag}(\mathbf{x})\mathbf{y} = \text{diag}(\mathbf{y})\mathbf{x}$
- $\mathbf{x}^p = \mathbf{x} \odot \dots \odot \mathbf{x} = \text{diag}(\text{diag}(\mathbf{x}^p)) = \text{diag}(\text{diag}(\mathbf{x})^p)$

for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ where $\text{diag}(\cdot)$, when having a matrix input, will vectorize all the diagonal elements. For a vector input, will construct a diagonal matrix where each diagonal containing each element of the vector. (Similar to the `numpy.diag` function)

- (ii) Now that we've showed that Algorithm 2 also satisfies equation (4) where \mathbf{g}_t is the gradient of the loss, we can proceed with our proof. **Prove that for $\mathbf{M}_t \neq k\mathbf{I}$ for some constant $k \in \mathbb{R}$, there exists no ℓ_2 -coefficient λ_{ℓ_2} such that running adaptive gradient methods through the loss function $L_\theta^{\text{reg}}(\cdot, \cdot)$ would give the same effect as weight decay.**
- (iii) Now assume \mathbf{M}_t (i.e. known as a preconditioner matrix) is a positive-definite and symmetric matrix across all time. This is can be true as many adaptive gradient methods contains a particular expression containing $\mathbf{G}_t(\alpha)$ for some value of $\alpha \in (0, 1]$ (Can you see why this is can be true? Does this line up to our answer of \mathbf{M}_t from part(b)(i)?). **Given this, show that we can scale-adjust ℓ_2 -regularization in our loss function (dependent on time) like below to have a weight decaying effect:**

$$L_{\theta,t}^{\text{special-reg}}(y_i, f_\theta(\mathbf{x}_i)) = L_\theta(y_i, f_\theta(\mathbf{x}_i)) + \frac{\lambda_{\ell_2}}{2} \|\mathbf{M}_t^{-\frac{1}{2}} \theta\|_2^2 \quad (6)$$

NOTE: Observe that $L_{\theta,t}^{\text{special-reg}}(\cdot, \cdot)$ is a special loss function that changes per iteration. This is unique from all the loss functions we've covered as many optimization problems only look at one loss function to minimize. Although this is something we can implement in code, you should realize that this is a special modification of our natural understanding of loss functions.

2. Coding - Motivation behind AdamW over ℓ_2 -regularized Adam

We have proved that Weight Decay $\neq \ell_2$ -Regularization for adaptive methods, or more specifically, RMSProp with bias-correcting second moment. Now, to bridge this argument to Adam, we just have to extend this idea to accommodate for momentum and bias-correcting momentum in its parameter update.

For momentum, we saw in Part(a)(ii) that adding momentum doesn't affect SGD's relationship between ℓ_2 -reg and weight decay. Therefore, we'll be using this intuition to *argue* that hopefully, the same would apply to RMSProp with bias-correcting second moment. In addition, since we already have a bias-correcting term for our second moment in our modification of RMSProp, we can also *argue* that having bias-correction for momentum won't change the algorithm's relationship between ℓ_2 -regularization and weight decay as well. *Therefore, we can vaguely claim that Adam falls victim to the phenomenon of weight decay \neq ℓ_2 -regularization!* Don't worry, we have practical coding demos in this part to further convince you this is the case.

(a) **Implementing Adam(W) and Experimentation**

Provided below (Algorithm 3) is the pseudocode for Adam without weight decoupling nor ℓ_2 regularization. **Based on this algorithm, implement Adam with ℓ_2 regularization and AdamW in section (i) of `explore_Adam.ipynb`. HINT: The additional changes needed to correctly implement regularized Adam and AdamW are nearly identical to part a(ii).**

Algorithm 3 Adam

```

1: given learning rate  $\eta \in \mathbb{R}_+$ , small value  $\epsilon$ ,  $\theta \in \mathbb{R}^m$ , momentum factor  $\beta_1 \in (0, 1]$ , second moment factor  $\beta_2 \in (0, 1]$ , and data set  $\{\mathbf{x}_i, y_i\}_{i=1}^n$ .
2: initialize time step  $t \leftarrow 0$ , parameter vector  $\theta_0$ , first moment vector  $\mathbf{m}_0 \leftarrow \mathbf{0}$ , exponential average second moment vector  $\mathbf{v}_0 \leftarrow \mathbf{0}$ 
3: while stopping criterion not met do
4:    $t \leftarrow t + 1$ 
5:    $\mathbf{g}_t \leftarrow \nabla L_\theta(y_i, f_\theta(\mathbf{x}_i)) \Big|_{\theta=\theta_{t-1}}$ 
6:    $\mathbf{m}_t \leftarrow \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \mathbf{g}_t$  ▷ Calculating momentum
7:    $\mathbf{v}_t \leftarrow \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) \mathbf{g}_t^2$ 
8:    $\hat{\mathbf{m}}_t \leftarrow \mathbf{m}_t / (1 - \beta_1^t)$  ▷ Bias-correcting momentum
9:    $\hat{\mathbf{v}}_t \leftarrow \mathbf{v}_t / (1 - \beta_2^t)$ 
10:   $\theta_t \leftarrow \theta_{t-1} - \eta [\hat{\mathbf{m}}_t / (\sqrt{\hat{\mathbf{v}}_t} + \epsilon \mathbb{1}_m)]$ 
11: end while
12: return optimized parameters  $\theta_t$ 

```

(b) **Visualizing Hyperparameter Space** In this question, we will see empirical evidence of the decoupling of the learning rate with weight decay in AdamW compared to Adam with ℓ_2 regularization. **Run the cells and report the four heatmaps generated. Then, answer the following questions.**

- i. What similarities and difference do you notice between the accuracy heat maps of the two algorithms? What about similarities and differences between the loss heat maps of the two algorithms?
- ii. Adjust the difficulty of the data in some way, either by making it “harder” or “easier” (see notebook for a more detailed description), and report on differences between heat maps generated for the two algorithms and also differences between heat maps from part a.
- iii. What can you conjecture about the processes of tuning hyperparameters of ℓ_2 regularized Adam and AdamW? Are there differences in the absolute best validation accuracy of the two algorithms? What about differences in the frequency of “good” hyperparameter settings above a certain threshold (say 90%) between the two algorithms?

References

- [1] Loshchilov, Ilya, and Frank Hutter. ‘Decoupled Weight Decay Regularization’. 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. <https://arxiv.org/pdf/1711.05101.pdf>
- [2] *Agarwal, Naman & Bullins, Brian & Chen, Xinyi & Hazan, Elad & Singh, Karan & Zhang, Cyril & Zhang, Yi. (2018). The Case for Full-Matrix Adaptive Regularization. <https://arxiv.org/pdf/1806.02958.pdf>
- [3] *Bushae, Vitaly. (2018). Adam — latest trends in deep learning optimization. <https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c>
- [4] *Jiang, Lili. (2020). A Visual Explanation of Gradient Descent Methods (Momentum, AdaGrad, RMSProp, Adam). <https://towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c>
- [5] *Villarraga, Daniel. (2021). AdaGrad. <https://optimization.cbe.cornell.edu/index.php?title=AdaGrad>
- [6] *Zhang, Jiawei. (2019). Gradient Descent based Optimization Algorithms for Deep Learning Models Training. <https://arxiv.org/pdf/1903.03614.pdf>
- [7] *Zhang, T., Wu, F., Katiyar, A., Weinberger, K., Artzi, Y.. (2020). Revisiting Few-sample BERT Fine-tuning. <https://arxiv.org/pdf/2006.05987.pdf>

*These sources were not directly referenced in the paper, but were crucial in understand the concepts better (Adaptive Methods, Adam, etc.) in order to create the problem set.

Contributors:

- Yuki Ito - Over 25 hours of work: written the outline, problem background, and introduction for all parts (Q1 and Q2). Specifically wrote out all of Q1(a), researched adaptive gradient methods (AdaGrad / RMSProp) extending from the lemmas presented and have proved in our main paper to originally write up all of Q1(b), revised rest of the parts and commentary, and facilitated the overall group project.
- Chase Adams - About 25 hours of work. Created Adam and AdamW implementations and auto-grader for Q2(a), experimented with the optimization algorithms for different datasets, reviewed the homework document, and wrote the commentary document.
- Zili Wang - Over 25 hours of work: contributed to writing all algorithm portions of written questions, wrote original draft of proving 3 lemmas found in paper, implemented synthetic data generation in IPython Notebook, created all data visualizations, wrote Q2(a-c).