Thesis notepad of ideas and weekly log of papers read etc etc

1 Adam Ideas

The Adam algorithm is applied in the following way.
For each iteration $t \in \mathbb{N}$, let $g^t := \nabla_w L(w)$. The update step is given by

$$\begin{aligned} m^{t+1} &= \beta_1 m^t + (1-\beta_1) g^t \\ v^{t+1} &= \beta_2 v^t + (1-\beta_2) g^t \odot g^t \\ \hat{m}^{t+1} &= \frac{1}{1-\beta_1^{t+1}} m^{t+1} \quad , \quad \hat{v}^{t+1} = \frac{1}{1-\beta_2^{t+1}} v^{t+1} \\ w^{t+1} &= w^t - \eta \frac{\hat{m}^{t+1}}{\sqrt{\hat{v}^{t+1}} + \varepsilon} \end{aligned}$$

LR scheduler ideas

It is sometimes the case that a learning rate scheduler $(\eta_j)_{j\in\mathbb{N}}$ is used in conjunction with Adam. One could also encode a decreasing (resp. increasing) step size by changing the ε parameter with time.
Namely, one has update steps of the form

$$w^{t+1} = w^t - \eta \frac{\hat{m}^{t+1}}{\sqrt{\hat{v}^{t+1}} + \varepsilon_t} \quad \text{instead of} \quad w^{t+1} = w^t - \eta_t \frac{\hat{m}^{t+1}}{\sqrt{\hat{v}^{t+1}} + \varepsilon}$$

probably stupid

Easiest way to alter the existing adam method in torch is just to find the equivalent learning rate schedule which corresponds to a given ε schedule.
More explicitely, given some sequence $(\varepsilon_t)_{t\geq 0}$ and $\eta > 0$, what sequence $(\eta_t)_{t\geq 0}$ is such that

$$w^{t+1} = w^t - \eta \frac{\hat{m}^{t+1}}{\sqrt{\hat{v}^{t+1}} + \varepsilon_t} \quad \text{is equialent to} \quad w^{t+1} = w^t - \eta_t \frac{\hat{m}^{t+1}}{\sqrt{\hat{v}^{t+1}} + \varepsilon_0}$$

Solving would give:

$$\eta_t = \frac{(\eta - \eta_t)\sqrt{\mathrm{EMA}_{\beta_2}(g_t^2)} + \eta\varepsilon_0}{\varepsilon_t}$$

Ok definitely stupid: can't know lr ahead of time. Will just implement directly in torch.

Coupling of momentum and RMS

The momentum and RMS boil down to keeping track of a moving average (EMT_β) parametrized by $\beta \in \mathbb{R}_{\geq 0}$.
While regular adam is given by:

$$w^{t+1} = w^t - \eta \frac{\text{EMA}_{\beta_1}(g_t)}{\sqrt{\text{EMA}_{\beta_2}(g_t^2)} + \varepsilon}$$

one could also consider the double nesting formula

$$w^{t+1} = w^t - \eta \text{EMA}_{\beta_1}\left(\frac{g_t}{\sqrt{EMA_{\beta_2}(g_t^2)} + \varepsilon}\right)$$

or in two steps

$$v^t = \text{bias correct}(\beta_2 v^{t-1} + (1-\beta_2) g^t \odot g^t)$$
$$K^t = \text{bias correct}(\beta_1 K^{t-1} + (1-\beta_1)\frac{g_t}{\sqrt{v^t} + \varepsilon})$$

When doing this, the bias correction term becomes less clear cut. Of course the second moment estimate v^t is the same ($1-\beta_1^t$ if you believe $E[g_t]$ is similar on initialization) . But after the nested average we have a sum of the form

$$K^t = (1-\beta_1)\sum_{j=0}^{t-1} \beta_1^{t-i-1} \frac{g_j}{\sqrt{v_j} + \varepsilon}$$

Can we make the same kind of assumption for initial values of t? Namely that $E[\frac{g_j}{\sqrt{v_j}+\varepsilon}]$ is "close enough" for all small j so that we can reduce the ugly expression above to a geometric sum. If so, great the term is the same. If not, any other clever tricks?

to also consider

- The Frederic Kunstner paper discusses Zipf's law and looking at loss of less common tokens. Shows adam learns the less frequent tokens better than SGD even with no batches
- the Bernstein paper shows how (without first moment ema) most optimizers (e.g Adam, SGD, Prodigy) are simply steepest descent wrt to some norm and how different parameters should have different operator norms applied depending on shape and action. Adam chooses infinity norm and kind of so does SignSGD
- the paper from 2020 "geometry of sign GD" shows that maybe sign methods can sometimes be better. In particular when Hessian is centred near diagonal and when the max eigenvalues are much bigger than average eigenvalues. Also SGD and Adam have some similaraties and are equivalent as params in adam $\to 0$. Also ℓ_∞ smoothness is weaker than seperable smoothness and is a sufficient assumption.

- Maybe some discussion on Music Foundation models?
- implementation pytorch-optimizer forked repo. Good idea?

Week of 25th November

- Read the Optimization for deep learning: Theory and algorithms document
- implemented the ε-scheduler to optimizer (torch optimizer prototypes forked from online repo)
- read the confidential paper sent via slack.
- read Dynamics of SGD with Stochastic Polyak Stepsizes: Truly Adaptive Variants and Convergence to Exact Solution
- Began benchmarking Adam variants on some toy datasets in collab: the cluster contract still awaits!
- Looked at Nikkolo's PlainLM repo repo
- tried reading up a bit on best practices of hyperparameter sweeps etc. Some basic questions

Week of 2 December

- Read A Survey on Efficient Training of Transformers
  - one proposition is SOAP without accumulation for efficiency
  - AdaDiag basically uses an SVD of the gradient to precondition the step size which still doing ADAM type things. Might be an interesting thing to try and nest with the NestedMA optimizer.
- IMPROVING ADAPTIVE MOMENT OPTIMIZATION VIA PRECONDITIONER DIAGONALIZATION
- investigating the modula package (Scalable Optimization in the Modular Norm)
- Looked over the AdamMini paper
- looked over Why transformers need Adam: A Hessian Perspective

some thoughts

- would be interesting to investigate bias correction vs taking some steps with SGD and then initializating adaptive optimizer from there
- would like to try the bernstein norm and explictely assign a different optimizer paradigm depending on the type of layer as discussed in the paper "Scalable Optimization in the Modular Norm".
- What are some reasonable ways for the epsilon scheduler to update. Intuitively a milestones = [] seems useful but maybe a functional relationship could be better? generally ε should increase to have something of a similar affect to the learning rate shrinking but to which extent? is there some theory to be built?
- evalute on vision transformers and small language models, RNNs, GNNs etc. to start to see if adaptive optimizers are more a data or model choice or both...
- Adam usually updates the moving averages per batch but it might be interesting to update more strongly for recent batches since if we use mini batching very quickly we lose info within a single epoch even though it might be nice to retain it

Week of 9 December

- The Impact of the Mini-batch Size on the Variance of Gradients in Stochastic Gradient Descent

ideas

- Work with some adaptive batch size; in the paper AdaBatch: Adaptive Batch Sizes for Training Deep Neural Networks, they implement for vision models. They also call it "adaptive" but really just make a batch size schedule and then make sure the ratio of the learning rate to the batch size remains constant. Also not clear whether they look at the adaptive adam step size or just the fixed learning rate (get the impression it's the latter). Don't see many other papers on this topic and the takeaway of the paper was that performace is similar but since the batch size increases, the later epochs are quicker and therefore training time is reduced.
- This backPACK package let's us compute the variance of batches (or could just do as in The Impact of the Mini-batch Size on the Variance of Gradients in Stochastic Gradient Descent). Idea to monitor variance as proxy for batch update size
- Which Algorithmic Choices Matter at Which Batch Sizes? Insights From a Noisy Quadratic Model
- in optimizer class, detect layer type and find better optimization mechanism for transformer architecture and conv layer. Use shampoo for linear layers.

- get partial hessian of some of the diagonal blocks (the hessian perspective on adam paper shows the structure is basically block diagonal in later epochs). Block diagonal hessian free method kind of ignores the curvature entirely