October 13, 2020

Introduction

- Lack of blackboard for math compounded with remote presentation - feel free to ask questions. Ambitious agenda so I might not be able to get to all questions but will try.
- The group is comfortable with linear algebra, multivariate calculus, gradients and optimization in multiple dimensions etc. Very little linear algebra in the presentation but you will need it.
- What will not be covered - refer to the slide.
- Objective - you should feel like can start playing with python code now.

Using vectors

- This is basic matrix multiplication with vectors. Not very interesting.
- Move on to gradients. Point out that there is nothing special, you just work with elements and try to step back to matrices and see if your notations make sense. Quite possible that you will have totally different notation from others and it will still work! But then you will need to explain it better.
- Element wise - nothing special. We will come across sigmoid again.

Auto-diff

Small models and tens of thousands of variables. You will very quickly reach 100s of thousands to million. BERT base has 110 million and BERT large has 340 million parameters. GPT-3 has a whopping 175 billion parameters.

- Analytical approach is impossible for all practical purposes.
- Finite difference will be too slow with too many numerical issues.
- Auto-diff works because it is something in-between analytical and numeric. You break your computation into layers. For each layer, you implement the function that the layer computes. Since the layer is small, you can actually also compute the analytical derivative. You use the analytical expression to compute the value of gradient. There are still problems with it is still better.
- Reverse mode exploits chain rule in differentiation. It starts computation from the end and reaches all the starting points (parameters) in a single pass (excluding the forward pass).

Before we can use auto-diff **??**, we represent the computation as a graph. This is a pretty standard idea in computer science. We can use this to even implement symbolic differentiation (contact me if you wish to build an engine just for fun).

It is important that you follow the graph carefully.

- Just follow Node 1 and Node 2 to Node 7
- Point out that each node can be implemented independent of others. We can compute the function as well as the derivative (gradients) of inputs W.R.T to output.
- On the analytical slide, point out computation for z. **Do not spend time on other computations**
- Show the example of multiplication node and gradients. Point out that the code is for illustrative purposes only and was written solely for this presentation.

Fit

- For sotchastic gradient descent - we select a batch, compute the gradient of parameters with respect to loss, update the parameters, then move on to

References

[1] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," 2018.

[2] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.

[3] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015.

[4] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *ICLR Workshop*, 2017.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016.

[6] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 3104–3112, Curran Associates, Inc., 2014.