Backpropogation

Shubham Gupta

January 6, 2020

1 Introduction

- It is the process of finding the effect of the inputs on the outputs. Specifically, how does a small change in the input affect the output from the network?

2 Computing gradients

- Organize the nodes in topological fashion i.e graph will go from input nodes to output nodes.
- Evaluate the nodes in order to find the final backprop solution.
- **Advantage**: Removes the need to store duplicate computations.

3 Automatic Differentiation

- Don't do the math by hand. Let computers run backprop.
- Each node knows how to calculate it's output and gradiesnt wrt its inputs.
- In TF/Pytorch, write forward and backprop formula on your own. But if written, does everything else and runs backprop.

4 Numeric Gradient

- Check gradients using numeric gradients.
- Check value for input and output when you change the value of h by a very small amount. This is the same as rise over run used to compute derivatives.

5 TLDR

- **Backprop**: [downstream gradient] = [upstream gradient] x [local gradient]

6 Important concepts

6.1 Regularization

- Prevers overfitting when we have many features.
- Very important for DL models because of the number of parameters.

6.2 Vectorization

- Needed to increase speed of operations
- Orders of magnitude faster on CPU as well as GPU.
- Matrices are the way to go.

6.3 Non linearities

- Logistic
- Tanh
- Hard Tanh: -1, x and +1 only.Linear function in between.
- ReLU: max(z, 0)
- Leaky ReLU: y = 0.01x

6.4 Parameter intialization

- Small random values as intial values.
- Start biases at 0 mostly
- Xavier intialization

$$Var(W_i) = \frac{2}{n_{in} + n_{out}} \tag{1}$$

6.5 Optimizers

- Plain SGD is most cases but need to hand tune learning rate
- Better to use adaptive oprimizers that scale parameters based on the accumulated gradient.
  - Adagrad
  - RMSProp
  - Adam
  - SparseAdam

6.6 Learning Rates

- Use constant learning rate i.e $1e^{-4}$
- Reduce the gradient by half every k epochs i.e $lr = lr_0 e^{-kt}$ for epoch t.
- Cyclic learning rates as implemented in fast.ai