WEIGHTS INITIALIZATION

Edit

I had quite a few people who were very apprehensive about how slightly different scaling of their weights initialization completely altered the training. In this post, we will take a closer look at the sensitive weights initialization process for any neural net. Here's the agenda:

1. What is weights initialization?
2. How we shouldn't initialize it.
3. How we should and why.
4. Numpy and Tensorflow options

WHAT IS IT?

Image a simple two-layer MLP. Each layer has neurons and each neuron is initialized with a weight. These weights are the values that are adjusted via backpropagation (for more information about gradients check out this **post**). But do we have to do special operations on these weights, like how we may choose to normalize (zero mean, unit variance) our input data? It turns out that it is very helpful if we can properly initialize our weights for training efficiently.

THE INCORRECT WAY

First, we will see how not to initialize our weights.

One option is to initialize to zeros. This may seem intuitive since our input is normalized we will need positive and negative weights in order to process them and 0 is a nice balance. Unfortunately, with backpropagation, the updates to the weights via the gradients is dependent on the previous weights as well. And since our weights are 0, we will never be able to update these weights and no learning will occur.

We can also choose to initialize with very small random numbers with maybe even some large deviances.Using random small weights can still result in large errors (especially during the initial epochs) so this will result in large gradients which can blow up during backpropagation. So we add some randomness in our weights so they all are able to update and learn but we need some way to initialize the weights so that we can control the variance of their outputs, which lets us control the gradient flow.

THE PROPER WAY

Now let's take a look at the proper way to initialize our weights. The objective is to have weights that are able to produce outputs that follow a similar distribution across all neurons. This will greatly help convergence during training and we will be able to train faster and effectively. But how do we calibrate the weights so that we can normalize the variance of our outputs?

We want the output of our weights to have unit variance prior to sending to the activation so we start with this:

$$Var(\sum_i^N X_i W_i) = 1$$
$$= \sum_i^N Var(X_i W_i)$$
$$= \sum_i^N E(W_i)^2 Var(X_i) + E(X_i)^2 Var(W_i) + Var(X_i)Var(W_i)$$

Assume inputs X have 0 mean and weights are from normal distribution (mean 0)

$$= \sum_i^N Var(X_i)Var(W_i)$$
$$= n * Var(X) * Var(W)$$
$$n * Var(X) * Var(\alpha W) = 1$$
$$\alpha = \frac{1}{\sqrt{n}} \quad \text{b/c} \quad \alpha^2 Var(W) = Var(\alpha W)$$

So we need to scale our random normal weight initializations by 1/sqrt(n) in order to have unit

- SEARCH -

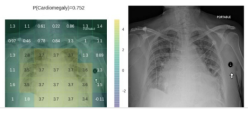
Search …

- FOLLOW ME ON TWITTER -

Tweets by @GokuMohandas

Goku Mohandas
@GokuMohandas

Your model may be performing really well by incorrectly focusing on confounding features (extraneous influencers in the data that aren't accounted for). Check out @johnrzech's post where x-ray stickers unintentionally influenced the classifications: medium.com/@jrzech/what-a…



Embed View on Twitter

- RECENT POSTS -

Update on Embeddings (Spring 2017)

Exploring Sparsity in Recurrent Neural Networks

Question Answering from Unstructured Text by Retrieval and Comprehension

Overcoming Catastrophic Forgetting in Neural Networks

Opening the Black Box of Deep Neural Networks via Information

So we need to scale our random normal weight initializations by 1/sqrt(n) in order to have unit variance. For ReLU, units this becomes sqrt(2/n) since a ReLU unit is zero for non positive inputs, so we will need to double the scale to have unit variance. Take a lot this **paper** for more of Xavier Glorot initialization and this **paper** for the ReLU initializations.

NUMPY AND TENSORFLOW IMPLEMENTATIONS

Just a few examples (many more options for initializations):

Numpy:

```
W1_init = np.random.randn(784, 100).astype(np.float32) * np
b1_init = np.zeros([100]).astype(np.float32)
W2_init = np.random.randn(100, 100).astype(np.float32) * np
b2_init = np.zeros([100]).astype(np.float32)
W3_init = np.random.randn(100, 10).astype(np.float32) * np.
b3_init = np.zeros([10]).astype(np.float32)
W_inits = [W1_init, b1_init, W2_init, b2_init, W3_init, b3_
```

Tensorflow:

```
W = tf.get_variable("W", shape=[784, 100],
           initializer=tf.contrib.layers.xavier_initializer
```

LOOKING AHEAD

Sure we can initialize our weights at the beginning to control the variance of the outputs but what about when we update our weights and they change. Well, since our weights were properly initialized and our inputs were normalized, the updates themselves are calibrated in a sense and backpropagation will not cause major variations in the initialized weights very quickly. Additionally, we have techniques like batchnorm and layernorm that help will controlling the normalization of the outputs continuously throughout training. You can find information and implementations of those techniques **here**.

Posted in: *Uncategorized*

← *Using Fast Weights to Attend to the Recent Past* *Improved Techniques for Training GANs* →

6 THOUGHTS ON "WEIGHTS INITIALIZATION"

NONE December 22, 2016 at 5:40 am EDIT REPLY→

BPTT stands for "Backpropagation through time" which is used to explain backprop through the unrolled time-steps in RNNs… why do you keep using this term everywhere where you mean simple backprop.

★ Liked by you

GOKUMOHANDAS December 22, 2016 at 11:14 am EDIT REPLY→

Ahh! Nice catch, I didn't even realize I was using 'BPTT' here!

★ Like

RAPHEY March 3, 2017 at 10:19 pm EDIT REPLY→

Thanks for posting this!
Is it possible the ReLU standard deviation is meant to be sqrt(2/n) rather than 2 / sqrt(n)? That seems to be what's needed to keep unit variance, and I think that's what the He et al paper says.

★ Liked by you

GOKUMOHANDAS March 3, 2017 at 10:59 pm EDIT REPLY→

Good catch Raphey! All updated now

★ Like

OM March 23, 2018 at 7:49 pm EDIT REPLY→

initializer=tf.contrib.layers.xavier_initializer() when i use this in AWS then its giving me

error if i run this on jupyter notebook then its working fine.so i there any issue of xavier initalizer on AWS

★ Like

RAHUL May 14, 2018 at 3:07 am EDIT REPLY →

Check the version of tensorflow , I think xavier_initializer is present in version >=0.8

★ Like

LEAVE A REPLY

Enter your comment here...