Xavier and He Normal (He-et-al) Initialization

29. September 2018

Why shouldn't you initialize the weights with zeroes or randomly (without knowing the distribution):

- If the weights in a network start too small, then the signal shrinks as it passes through each layer until it's too tiny to be useful.
- If the weights in a network start too large, then the signal grows as it passes through each layer until it's too massive to be useful.

Types of Initializations:

Xavier/Glorot Initialization

Xavier Initialization initializes the weights in your network by drawing them from a distribution with zero mean and a specific variance,

$$Var(w_i) = \frac{1}{fan\_in}$$

where fan_in is the number of incoming neurons.

It draws samples from a truncated normal distribution centered on 0 with `stddev = sqrt(1 / fan_in)` where `fan_in` is the number of input units in the weight tensor.

Generally used with tanh activation.

Also generally,

$$Var(w_i) = \frac{1}{fan\_in + fan\_out}$$

is used where fan_out is the number of neurons the result is fed to.

He Normal (He-et-al) Initialization

This method of initializing became famous through a paper submitted in 2015 by He-et-al, and is similar to Xavier initialization, with the factor multiplied by two. In this method, the weights are initialized keeping in mind the size of the previous layer which helps in attaining a global minimum of the cost function faster and more efficiently.The weights are still random but differ in range depending on the size of the previous layer of

RECENT POSTS

- Fastai—Image Similarity Search—Pytorch Hooks & Spotify's Annoy
- Dissecting The Role of Return_state and Return_seq Options in LSTM Based Sequence Models
- Flood Forecasting
- Using Deep Learning for Image Analogies
- Computer Vision—A journey from CNN to Mask R-CNN and YOLO -Part 2

neurons. This provides a controlled initialization hence the faster and more efficient gradient descent.

if RELU activation:

$$Y = w_1x_1 + w_2x_2 + \ldots + w_nx_n$$

$$Var(w_i) = \frac{2}{fan\_in}$$

It draws samples from a truncated normal distribution centered on 0 with `stddev = sqrt(2 / fan_in)` where `fan_in` is the number of input units in the weight tensor.

Proof why :

We have an input *X* with n components and a linear neuron with random weights *W*and output *Y*.

$$Y = w_1x_1 + w_2x_2 + \ldots + w_nx_n$$

$$Var(Y) = Var(w_ix_i) = E[x_i]^2Var(w_i) + E[w_i]^2Var(x_i) + Var(w_i)Var(x_i)$$

can be found on Wikipedia

Now lets assume mean =0

$$Var(Y) = Var(w_i)Var(x_i)$$

since

$$\operatorname{Var}\left(\sum_{i=1}^{n} X_i\right) = \sum_{i=1}^{n} \operatorname{Var}(X_i).$$

and if we make a assumption of i.i.d., we get

$$Var(Y) = nVar(w_i)Var(x_i)$$

So we want this Var(Y) =1

$$Var(w_i) = \frac{1}{n} = \frac{1}{fan\_in}$$

In Glorot & Bengio's, If we go through the same steps for the backpropagated signal, we get

$$Var(w_i) = \frac{1}{fan\_out}$$

to keep the variance of the input gradient & the output gradient the same. These two constraints can only be satisfied simultaneously if fan_in=fan_out, so a compromise,

we take the average of the two:

$$Var(w_i) = \frac{2}{fan\_in + fan\_out}$$

In a recent paper by He, Rang, Zhen and Sun they build on Glorot & Bengio and suggest using

$$Var(w_i) = \frac{2}{fan\_in}$$

Implementations:

Numpy Initialization

```
w=np.random.randn(layer_size[l],layer_size[l-1])*np.sqrt(1/layer_size[l
```

```
w=np.random.randn(layer_size[l],layer_size[l-1])*np.sqrt(2/(layer_size[
```

Tensorflow Implementation

```
tf.contrib.layers.xavier_initializer(
    uniform=True,
    seed=None,
    dtype=tf.float32
)
```

This initializer is designed to keep the scale of the gradients roughly the same in all layers. In uniform distribution this ends up being the range: `x = sqrt(6. / (in + out)); [-x, x]` and for normal distribution a standard deviation of `sqrt(2. / (in + out))` is used.

You can use the below to use all types:

```
tf.contrib.layers.variance_scaling_initializer(factor=2.0, mode='FAN_IN
```

- To get Delving Deep into Rectifiers (also know as the "MSRA initialization"), use (Default):
  `factor=2.0 mode='FAN_IN' uniform=False`
- To get Convolutional Architecture for Fast Feature Embedding, use:
  `factor=1.0 mode='FAN_IN' uniform=True`
- To get Understanding the difficulty of training deep feedforward neural networks, use:
  `factor=1.0 mode='FAN_AVG' uniform=True.`
- To get `xavier_initializer` use either:
  `factor=1.0 mode='FAN_AVG' uniform=True`, or
  `factor=1.0 mode='FAN_AVG' uniform=False`.

```
if mode='FAN_IN': # Count only number of input connections.
    n = fan_in
  elif mode='FAN_OUT': # Count only number of output connections.
    n = fan_out
  elif mode='FAN_AVG': # Average number of inputs and output connectior
    n = (fan_in + fan_out)/2.0

    truncated_normal(shape, 0.0, stddev=sqrt(factor / n))
```

Keras Initialization

- `tf.keras.initializers.glorot_normal(seed=None)`

It draws samples from a truncated normal distribution centered on 0 with `stddev = sqrt(2 / (fan_in + fan_out))`where `fan_in` is the number of input units in the weight tensor and `fan_out` is the number of output units in the weight tensor.

- `tf.keras.initializers.glorot_uniform(seed=None)`

It draws samples from a uniform distribution within [-limit, limit] where `limit` is `sqrt(6 / (fan_in + fan_out))`where `fan_in` is the number of input units in the weight tensor and `fan_out` is the number of output units in the weight tensor.

- `tf.keras.initializers.he_normal(seed=None)`

It draws samples from a truncated normal distribution centered on 0 with `stddev = sqrt(2 / fan_in)` where `fan_in`is the number of input units in the weight tensor.

- `tf.keras.initializers.he_uniform(seed=None)`

It draws samples from a uniform distribution within [-limit, limit] where `limit` is `sqrt(6 / fan_in)` where `fan_in` is the number of input units in the weight tensor.

- `tf.keras.initializers.lecun_normal(seed=None)`

It draws samples from a truncated normal distribution centered on 0 with `stddev = sqrt(1 / fan_in)` where `fan_in`is the number of input units in the weight tensor.

- `tf.keras.initializers.lecun_uniform(seed=None)`

It draws samples from a uniform distribution within [-limit, limit] where `limit` is `sqrt(3 / fan_in)` where `fan_in` is the number of input units in the weight tensor.

References:

1. http://andyljones.tumblr.com/post/110998971763/an-explanation-of-xavier-initialization

Thrown in a like if you liked it to keep me motivated.

Source: Deep Learning on Medium

Training Deep Neural Networks | Weight Initialization in Neural Networks: A Journey From the | Weight initialization for CNNs: A Deep Dive into He Initialization

Deep Learning AccessoriesDeep neural networks are key break through in the field of computer vision and speech recognition. For the past decade, deep
11. September 2018
Similar post

Basics to Kaiming
Source: Deep Learning on Medium James DellingerApr 3I'd like to invite you to join me on an exploration through different approaches to initializing layer
3. April 2019
Similar post

Today is the first day of my sabbatical (thanks Asana!), so I tried to learn something useful!I decided to investigate how state-of-the-art Convolutional
6. November 2018
Similar post

« 100 DAYS OF ML — DAY 13 — SKILLS NEEDED TO LEARN AI AT EVERY LEVEL

100 DAYS OF MACHINE LEARNING CODE: MY JOURNEY TO ML »