# EE-559 – Deep learning

# 5.5. Parameter initialization


François Fleuret


https://fleuret.org/ee559/

Dec 23, 2019

# Vanishing gradient

Consider the gradient estimation for a standard MLP:

**Forward pass**

$$\forall n,\ x^{(0)} = x, \quad \forall l = 1, \dots, L, \begin{cases} s^{(l)} = w^{(l)} x^{(l-1)} + b^{(l)} \\ x^{(l)} = \sigma\left(s^{(l)}\right) \end{cases}$$

**Backward pass**

$$\begin{cases} \left[\frac{\partial \ell}{\partial x^{(L)}}\right] = \nabla_1 \ell\left(x^{(L)}\right) \\ \text{if } l < L, \left[\frac{\partial \ell}{\partial x^{(l)}}\right] = \left(w^{(l+1)}\right)^T \left[\frac{\partial \ell}{\partial s^{(l+1)}}\right] \end{cases} \qquad \left[\frac{\partial \ell}{\partial s^{(l)}}\right] = \left[\frac{\partial \ell}{\partial x^{(l)}}\right] \odot \sigma'\left(s^{(l)}\right)$$

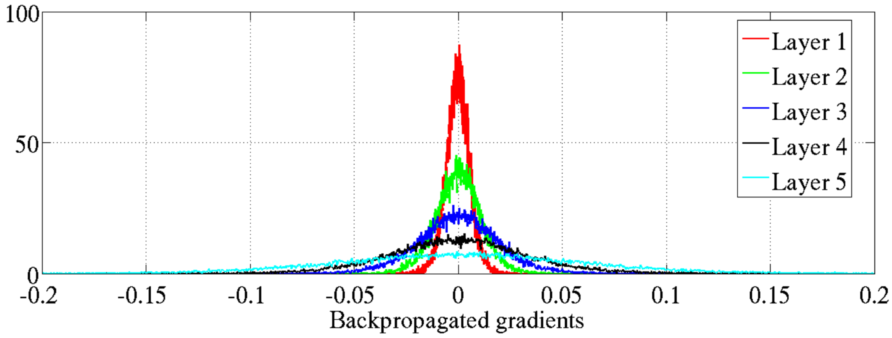
$$\left[\!\left[\frac{\partial \ell}{\partial w^{(l)}}\right]\!\right] = \left[\frac{\partial \ell}{\partial s^{(l)}}\right] \left(x^{(l-1)}\right)^T \qquad \left[\frac{\partial \ell}{\partial b^{(l)}}\right] = \left[\frac{\partial \ell}{\partial s^{(l)}}\right].$$



We have

$$\left[\frac{\partial \ell}{\partial x^{(l)}}\right] = \left(w^{(l+1)}\right)^T \left(\sigma'\left(s^{(l)}\right) \odot \left[\frac{\partial \ell}{\partial x^{(l+1)}}\right]\right).$$

**so the gradient "vanishes" exponentially with the depth if the $w$s are ill-conditioned or the activations are in the saturating domain of $\sigma$.**




(Glorot and Bengio, 2010)

# Weight initialization



The analysis for the weight initialization relies on controlling

$$\mathbb{V}\left(\frac{\partial \ell}{\partial w_{i,j}^{(l)}}\right) \text{ and } \mathbb{V}\left(\frac{\partial \ell}{\partial b_{i}^{(l)}}\right)$$

where the parameters and inputs are randomized, so that **weights evolve at the same rate across layers during training,** and no layer reaches a saturation behavior before others.

We will use that, if $A$ and $B$ are independent

$$\mathbb{V}(AB) = \mathbb{V}(A)\,\mathbb{V}(B) + \mathbb{V}(A)\,\mathbb{E}(B)^2 + \mathbb{V}(B)\,\mathbb{E}(A)^2\,.$$

Notation in the coming slides will drop indexes when variances are identical for all activations or parameters in a layer.



In a standard layer

$$x_i^{(l)} = \sigma\left(\sum_{j=1}^{N_{l-1}} w_{i,j}^{(l)} x_j^{(l-1)} + b_i^{(l)}\right)$$

where $N_l$ is the number of units in layer $l$, and $\sigma$ is the activation function.

Assuming $\sigma'(0) = 1$, and we are in the linear regime

$$x_i^{(l)} \simeq \sum_{j=1}^{N_{l-1}} w_{i,j}^{(l)} x_j^{(l-1)} + b_i^{(l)}.$$

From which, if both the $w^{(l)}$s and $x^{(l-1)}$s are centered, and biases set to zero:

$$\begin{aligned}\mathbb{V}\left(x_i^{(l)}\right) &\simeq \mathbb{V}\left(\sum_{j=1}^{N_{l-1}} w_{i,j}^{(l)} x_j^{(l-1)}\right)\\ &= \sum_{j=1}^{N_{l-1}} \mathbb{V}\left(w_{i,j}^{(l)}\right)\mathbb{V}\left(x_j^{(l-1)}\right)\end{aligned}$$

and the $x^{(l)}s$ are centered.

So if the $w_{i,j}^{(l)}$ are sampled i.i.d in each layer, then

$$\begin{aligned}\mathbb{V}\left(x^{(l)}\right) &\simeq \sum_{j=1}^{N_{l-1}} \mathbb{V}\left(w^{(l)}\right) \mathbb{V}\left(x^{(l-1)}\right) \\ &= N_{l-1} \mathbb{V}\left(w^{(l)}\right) \mathbb{V}\left(x^{(l-1)}\right).\end{aligned}$$

So **we have for the variance of the activations:**

$$\mathbb{V}\left(x^{(l)}\right) \simeq \mathbb{V}\left(x^{(0)}\right) \prod_{q=1}^{l} N_{q-1} \mathbb{V}\left(w^{(q)}\right).$$



This leads to a first type of initialization

$$\mathbb{V}\left(w^{(l)}\right) = \frac{1}{N_{l-1}}.$$

In `torch/nn/modules/linear.py`

```
def reset_parameters(self):
    stdv = 1. / math.sqrt(self.weight.size(1))
    self.weight.data.uniform_(-stdv, stdv)
    if self.bias is not None:
        self.bias.data.uniform_(-stdv, stdv)
```

We can look at the variance of the gradient w.r.t. the activations. Since

$$\frac{\partial \ell}{\partial x_i^{(l)}} \simeq \sum_{h=1}^{N_{l+1}} \frac{\partial \ell}{\partial x_h^{(l+1)}} w_{h,i}^{(l+1)}$$

we get

$$\mathbb{V}\left(\frac{\partial \ell}{\partial x^{(l)}}\right) \simeq N_{l+1} \mathbb{V}\left(\frac{\partial \ell}{\partial x^{(l+1)}}\right) \mathbb{V}\left(w^{(l+1)}\right).$$

So **we have for the variance of the gradient w.r.t. the activations:**

$$\mathbb{V}\left(\frac{\partial \ell}{\partial x^{(l)}}\right) \simeq \mathbb{V}\left(\frac{\partial \ell}{\partial x^{(L)}}\right) \prod_{q=l+1}^{L} N_q \mathbb{V}\left(w^{(q)}\right).$$



Since

$$x_i^{(l)} \simeq \sum_{j=1}^{N_{l-1}} w_{i,j}^{(l)} x_j^{(l-1)} + b_i^{(l)}$$

we have

$$\frac{\partial \ell}{\partial w_{i,j}^{(l)}} \simeq \frac{\partial \ell}{\partial x_i^{(l)}} x_j^{(l-1)}$$

and **we get the variance of the gradient w.r.t. the weights**

$$\begin{aligned}
\mathbb{V}\left(\frac{\partial \ell}{\partial w^{(l)}}\right) &\simeq \mathbb{V}\left(\frac{\partial \ell}{\partial x^{(l)}}\right) \mathbb{V}\left(x^{(l-1)}\right) \\
&= \mathbb{V}\left(\frac{\partial \ell}{\partial x^{(L)}}\right) \left(\prod_{q=l+1}^{L} N_q \mathbb{V}\left(w^{(q)}\right)\right) \mathbb{V}\left(x^{(0)}\right) \left(\prod_{q=1}^{l} N_{q-1} \mathbb{V}\left(w^{(q)}\right)\right) \\
&= \frac{N_0}{N_l} \left(\prod_{q=1}^{L} N_q \mathbb{V}\left(w^{(q)}\right)\right) \mathbb{V}\left(x^{(0)}\right) \mathbb{V}\left(\frac{\partial \ell}{\partial x^{(L)}}\right).
\end{aligned}$$

Similarly, since

$$x_i^{(l)} \simeq \sum_{j=1}^{N_{l-1}} w_{i,j}^{(l)} x_j^{(l-1)} + b_i^{(l)}$$

we have

$$\frac{\partial \ell}{\partial b_i^{(l)}} \simeq \frac{\partial \ell}{\partial x_i^{(l)}}$$

so **we get the variance of the gradient w.r.t. the biases**

$$\mathbb{V}\left(\frac{\partial \ell}{\partial b^{(l)}}\right) \simeq \mathbb{V}\left(\frac{\partial \ell}{\partial x^{(l)}}\right).$$



So finally, the variance of the gradient w.r.t. the weights is the same in all layers.

To control the variance of activations, we need

$$\mathbb{V}\left(w^{(l)}\right) = \frac{1}{N_{l-1}},$$

and to control the variance of the gradient w.r.t. activations, and through it the variance of the gradient w.r.t. the biases

$$\mathbb{V}\left(w^{(l)}\right) = \frac{1}{N_l}.$$

From which we get as a compromise the "Xavier initialization"

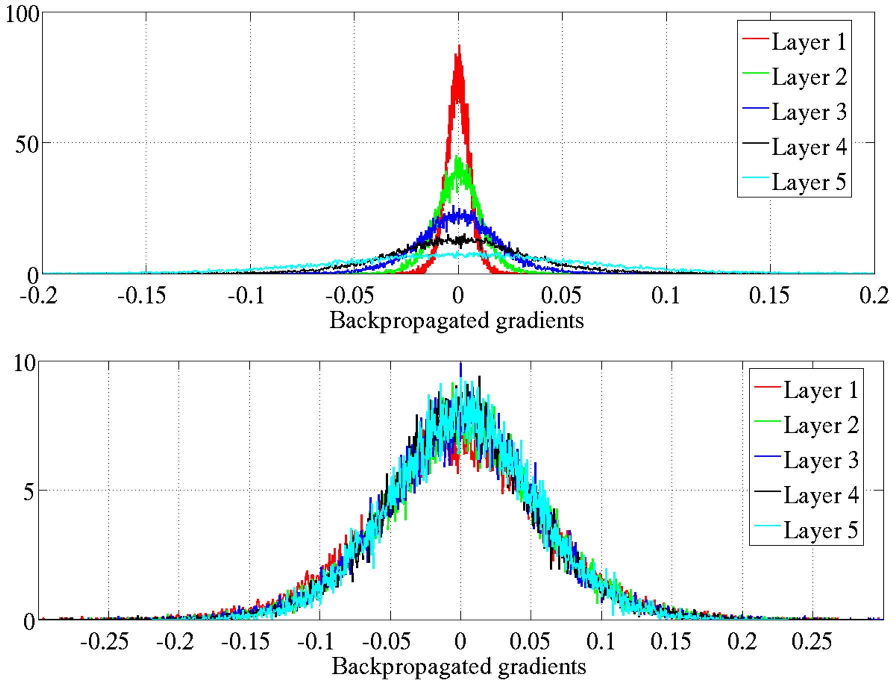
$$\mathbb{V}\left(w^{(l)}\right) = \frac{1}{\frac{N_{l-1}+N_l}{2}} = \frac{2}{N_{l-1}+N_l}.$$

(Glorot and Bengio, 2010)

In torch/nn/init.py

```
def xavier_normal_(tensor, gain = 1):
    fan_in, fan_out = _calculate_fan_in_and_fan_out(tensor)
    std = gain * math.sqrt(2.0 / (fan_in + fan_out))
    with torch.no_grad():
        return tensor.normal_(0, std)
```






(Glorot and Bengio, 2010)

The weights can also be scaled to account for the activation functions.

Remember that if $A$ and $B$ are independent, we have

$$
\begin{aligned}
\mathbb{V}(AB) &= \mathbb{V}(A)\,\mathbb{V}(B) + \mathbb{V}(A)\,\mathbb{E}(B)^2 + \mathbb{V}(B)\,\mathbb{E}(A)^2 \\
&= \mathbb{V}(A)\,\mathbb{E}\left(B^2\right) + \mathbb{V}(B)\,\mathbb{E}(A)^2 .
\end{aligned}
$$



For the forward pass, if

$$
\begin{aligned}
s_i^{(l)} &= \sum_{j=1}^{N_{l-1}} w_{i,j}^{(l)} \sigma\left(s_j^{(l-1)}\right) + b_i^{(l)} \\
x_i^{(l)} &= \sigma\left(s_i^{(l)}\right),
\end{aligned}
$$

and $\mathbb{E}\left(w^{(l)}\right) = 0$, $s^{(l-1)}$ is symmetric, and $\sigma$ is ReLU, we have

$$
\begin{aligned}
\mathbb{V}\left(s_i^{(l)}\right) &= N_{l-1}\mathbb{V}\left(w^{(l)}\sigma\left(s^{(l-1)}\right)\right) \\
&= N_{l-1}\mathbb{V}\left(w^{(l)}\right)\mathbb{E}\left(\sigma\left(s^{(l-1)}\right)^2\right) \\
&= N_{l-1}\mathbb{V}\left(w^{(l)}\right)\frac{1}{2}\mathbb{E}\left(\left(s^{(l-1)}\right)^2\right) \\
&= \frac{1}{2}N_{l-1}\mathbb{V}\left(w^{(l)}\right)\mathbb{V}\left(s^{(l-1)}\right).
\end{aligned}
$$

For the backward

$$
\begin{aligned}
\mathbb{V}\left(\frac{\partial \ell}{\partial x_i^{(l)}}\right) &= \sum_{h=1}^{N_{l+1}} \mathbb{V}\Big(\underbrace{\sigma'\left(s_h^{(l+1)}\right)}_{0/1} \underbrace{\frac{\partial \ell}{\partial x_h^{(l+1)}} w_{h,i}^{(l+1)}}_{\mathbb{E}(.)=0,\ \text{symmetric}}\Big) \\
&= \sum_{h=1}^{N_{l+1}} \mathbb{E}\left(\sigma'\left(s_h^{(l+1)}\right)\left(\frac{\partial \ell}{\partial x_h^{(l+1)}} w_{h,i}^{(l+1)}\right)^2\right) \\
&= \sum_{h=1}^{N_{l+1}} \frac{1}{2} \mathbb{E}\left(\left(\frac{\partial \ell}{\partial x_h^{(l+1)}} w_{h,i}^{(l+1)}\right)^2\right) \\
&= \frac{1}{2} \sum_{h=1}^{N_{l+1}} \mathbb{V}\left(\frac{\partial \ell}{\partial x_h^{(l+1)}}\right) \mathbb{V}\left(w_{h,i}^{(l+1)}\right).
\end{aligned}
$$



So ReLU impacts the forward and backward pass as if the weights had half their variances, which motivates multiplying them by a corrective gain of $\sqrt{2}$.

(He et al., 2015)

The same type of reasoning can be applied to other activation functions.

In `torch/nn/init.py`

```
def calculate_gain(nonlinearity, param=None):

    linear_fns = ['linear', 'conv1d', 'conv2d', 'conv3d',
                  'conv_transpose1d', 'conv_transpose2d', 'conv_transpose3d']
    if nonlinearity in linear_fns or nonlinearity == 'sigmoid':
        return 1
    elif nonlinearity == 'tanh':
        return 5.0 / 3
    elif nonlinearity == 'relu':
        return math.sqrt(2.0)
    /.../
```

# Data normalization



The analysis for the weight initialization relies on keeping the activation variance constant.

For this to be true, not only the variance has to remained unchanged through layers, but it has to be correct for the input too.

$$\mathbb{V}\left(x^{(0)}\right) = 1.$$

This can be done in several ways. Under the assumption that all the input components share the same statistics, we can do

```
mu, std = train_input.mean(), train_input.std()
train_input.sub_(mu).div_(std)
test_input.sub_(mu).div_(std)
```

Thanks to the magic of broadcasting we can normalize component-wise with

```
mu, std = train_input.mean(0), train_input.std(0)
train_input.sub_(mu).div_(std)
test_input.sub_(mu).div_(std)
```

To go one step further, some techniques initialize the weights explicitly so that the empirical moments of the activations are as desired.

As such, they take into account the statistics of the network activation induced by the statistics of the data.



## References

X. Glorot and Y. Bengio. **Understanding the difficulty of training deep feedforward neural networks**. In International Conference on Artificial Intelligence and Statistics (AISTATS), 2010.

K. He, X. Zhang, S. Ren, and J. Sun. **Delving deep into rectifiers: Surpassing human-level performance on imagenet classification**. CoRR, abs/1502.01852, 2015.