5. Weight Initialization

Dong-Gyu, Lee

Dept. of Statistics, KU

2020, Mar 10

Contents

Today's Goal

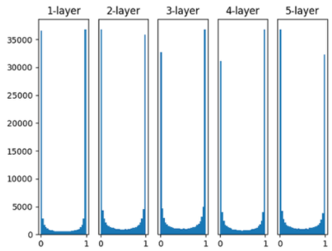
- In this time, we will focus on the various initializing methods.
- And we will look at the characteristics of each method through a formula and a picture.

Importance of initialization

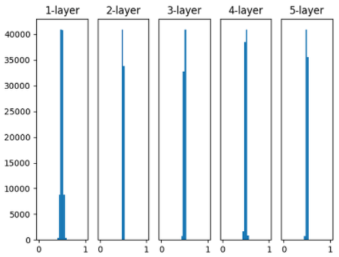
- We obtain the appropriate W (often called the parameter matrix) from training a model in MLP, CNN, RNN, etc.
- Because of the large number of parameters, we can't set all of them individually.
- So many people start thinking about various methods how to simply initialize the parameters.

Importance of initialization

- If all of the initial values are 0, there is no learning of the parameters.
- Plus, if the same initial value is used, the correct improvement is not made because the input values are learned by the same ratio.
- Also, even if all of the initial values follow the proper distribution, you may not get the desired result as shown in the figures below.



(a) N(0,1)



(b) N(0, 0.0001)

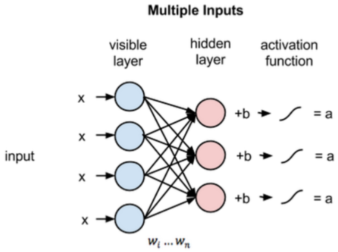
Figure 1: Initial Values Following Normal Distribution

Importance of initialization

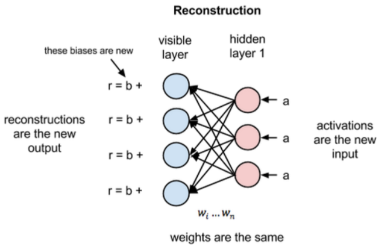
- The result of Figure 1 is the distribution of $Y = \sigma(XW + b)$. after each hidden layer.
- Each notation follows:
  1. Y is an output.
  2. $\sigma(\cdot)$ is a sigmoid function.
  3. X is an input.
  4. W is a parameter matrix which is initialized.
  5. b is a bias term which is initialized with 0.
- Since the intercept term is an added value, it can be initialized to 0.
- The (a) of Figure 1 has eventually gradient vanishing because of being distributed many 0 and 1 values.
- The (b) of Figure 1 does not happen gradient vanishing, but loses the advantage of using nodes a lot.

1. RBM & DBM

- Restricted Boltzmann Machine(RBM)[3] is a generative stochastic artificial neural network proposed by Professor Hinton in 2006.
- It is mainly used to determine the initial value and works like unsupervised learning.
- As it is complicated, RBM is not used well these days.



(a) Forward



(b) Backward

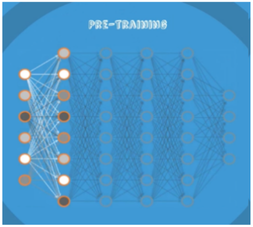
Figure 2: RBM

1. RBM & DBM

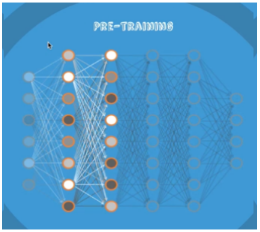
- "Restricted" means no connection between nodes in each layer.
- Also, calculate W by training until the difference between the input and the result of Figure 2 (a) and (b) is small.
- RBM slightly is distinguished from the Autoencoder in using different bias term in Figure 2 (a) from in Figure 2 (b).
- And last, if you stack multiple RBM, you get Deep Boltzmann Machine(DBM).
- On a lighter note, Deep Belief Network(DBN) is made with the initial values of DBM.

1. RBM & DBM

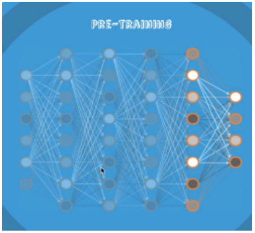
- The DBM can be seen in the figure below.



(a) Keep



(b) Going



(c) On

Figure 3: DBM

1. RBM & DBM

- Finally, you can use the W, parameter matrix, obtained in this way as the initial value for the training model.
- This initial values assignment is also called 'Fine Tuning'.

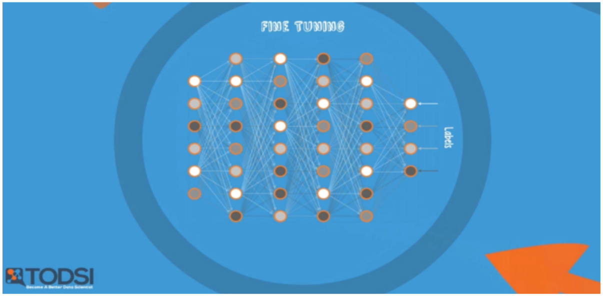


Figure 4: Fine Tuning

2. Simple Uniform Initialization

- Very simply, you can give the initial values to follow the uniform distribution.

$$W\text{'s elements} \sim U(-0.5, 0.5)$$

- Absolutely, it is not used nowadays.
- Note that W is a parameter matrix.

3. LeCun Initialization

- Also known as the founder of the LeNet and the father of CNN, Yann LeCun suggests giving the following initial values:
- LeCun method[5] have not be used well recently since ReLU came out.

1. LeCun Normal Initialization

$$W\text{'s elements} \sim N(0, \sigma^2) \quad , \quad \sigma = \sqrt{\frac{1}{n_{in}}}$$

2. LeCun Uniform Initialization

$$W\text{'s elements} \sim U(-a, a) \quad , \quad a = \sqrt{\frac{3}{n_{in}}}$$

- Note that n_{in} is the number of previous layer nodes.

3. LeCun Initialization

Pf)

- Let n=number of input nodes , x=input, Y=output, w=weight. (w, x, Y : R.V. & independent each other)
- And, let's not consider the activation function. Then,

$$Y = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

- Thus, variance of Y is:

$$\begin{aligned} Var[Y] &= Var[\sum_{i=1}^{n} w_i x_i] \\ &= n\left[E[x_i]^2 Var[w_i] + E[w_i]^2 Var[x_i] + Var[w_i] Var[x_i]\right] \end{aligned}$$

3. LeCun Initialization

Pf)

- Let the mean of x and w be 0. Then,

$$Var[Y] = nVar[w_i]Var[x_i]$$

- Therefore, in order to maintain the variance of x and Y, the variance of w_i must be $\frac{1}{n}$.
- In the end, the appropriate initialization values for normal and uniform distributions are set.

4. Xavier(Glorot) Initialization

- It is the initialization method that Xavier Glorot first proposed in 2010[1].
- It is still widely used as an initialization method unless otherwise specified.

1. Xavier Normal Initialization

$$W\text{'s elements} \sim N(0, \sigma^2) \quad , \quad \sigma = \sqrt{\frac{1}{(n_{in} + n_{out})/2}}$$

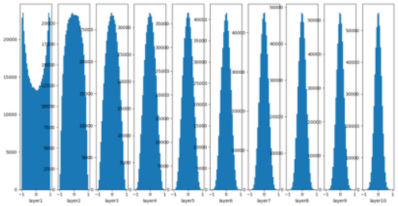
2. Xavier Uniform Initialization

$$W\text{'s elements} \sim U(-a, a) \quad , \quad a = \sqrt{\frac{3}{(n_{in} + n_{out})/2}}$$

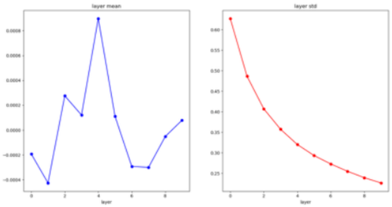
- n_{in} : the number of previous layer nodes.
- n_{out} : the number of next layer nodes.

4. Xavier(Glorot) Initialization

- Xavier(or Glorot) initialization is effective when using tanh or sigmoid as an activation function.



(a) Distribution Trend

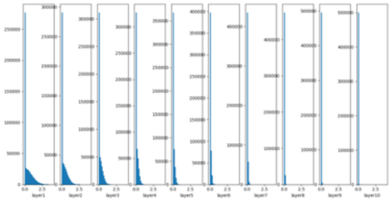


(b) Mean & Std.

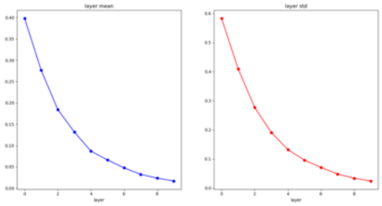
Figure 5: Xavier with Sigmoid

4. Xavier(Glorot) Initialization

- However, it is not effective when using ReLU as an activation function.
- So we use the He initialization, which will be explained in the next slide.



(a) Distribution Trend



(b) Mean & Std.

Figure 6: Xavier with ReLU

5. He Initialization

- It is the initialization method that Kaiming He first proposed in 2015[2].
- He initialization is used a lot in the CNN models with ReLU.

1. He Normal Initialization

$$W\text{'s elements} \sim N(0, \sigma^2) \quad , \quad \sigma = \sqrt{\frac{2}{n_{in}}}$$

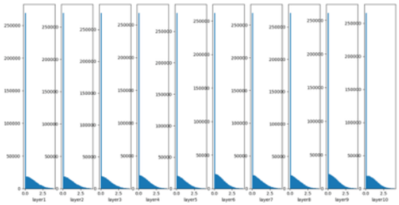
2. He Uniform Initialization

$$W\text{'s elements} \sim U(-a, a) \quad , \quad a = \sqrt{\frac{2 \times 3}{n_{in}}}$$

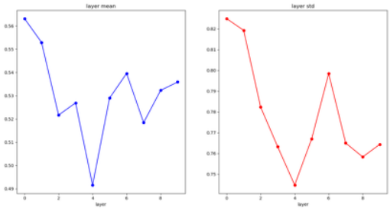
- n_{in} : the number of previous layer nodes.
- n_{out} : the number of next layer nodes.

5. He Initialization

- The difference from Xavier initialization is that the output node is not taken into account and the variance of the initial value is doubled.
- The reason for doubling the variance comes from a simple statistical calculation that takes into account the form of $Y = max(0, x)$.



(a) Distribution Trend



(b) Mean & Std.

Figure 7: He with ReLU

5. He Initialization

Showing)

- Let x=output before applying activation function, Y=final output. ($Y = max(0, x)$, i.e. $Y = x$+ReLU)
- Also, x, Y : R.V. & independent each other.
- And let's look at the relationship between Y's variance and x's variance.
- Thus, variance of Y is:

$$\begin{aligned} V[Y] &= V[X \cdot I(X > 0)] \\ &= V[X] + V[X \cdot I(X \leq 0)] - 2Cov[X, X \cdot I(X \leq 0)] \\ &= V[X] - V[X \cdot I(X \leq 0)] \end{aligned}$$

5. He Initialization

Showing)

- That is,

$$V[X \cdot I(X > 0)] = V[X] - V[X \cdot I(X \leq 0)]$$

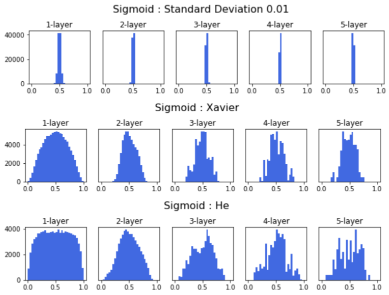
- Assuming symmetry for $x = 0$,

$$V[Y] = \frac{1}{2} V[X]$$

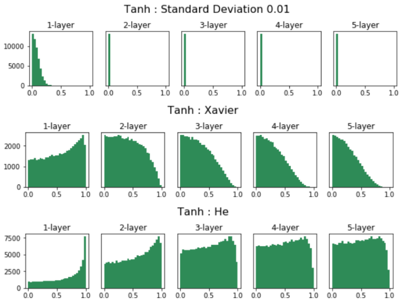
- In combination with what we got in slide 13, we multiply the variance of the initial value by 2 to make the output distribution safe.

Interim Summary

- The summary is as follows :



(a) Sigmoid



(b) Tanh

Figure 8: Various Initialization Methods with Sigmoid & Tanh

Interim Summary

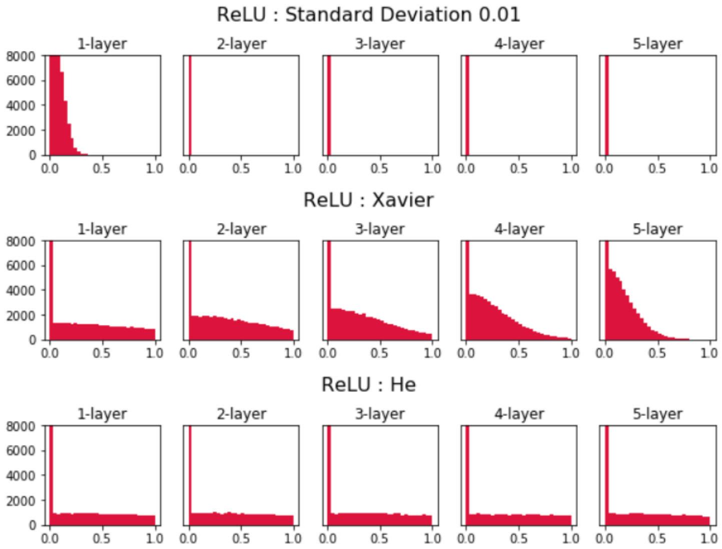
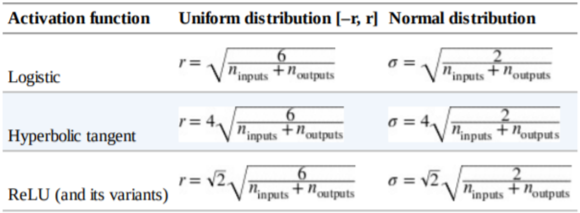


Figure 9: Various Initialization Methods with ReLU

Interim Summary

- Finally, in the end, the initialization is performed in the following distribution.



Activation function	Uniform distribution [–r, r]	Normal distribution
Logistic	$r = \sqrt{\frac{6}{n_{\text{inputs}} + n_{\text{outputs}}}}$	$\sigma = \sqrt{\frac{2}{n_{\text{inputs}} + n_{\text{outputs}}}}$
Hyperbolic tangent	$r = 4\sqrt{\frac{6}{n_{\text{inputs}} + n_{\text{outputs}}}}$	$\sigma = 4\sqrt{\frac{2}{n_{\text{inputs}} + n_{\text{outputs}}}}$
ReLU (and its variants)	$r = \sqrt{2}\sqrt{\frac{6}{n_{\text{inputs}} + n_{\text{outputs}}}}$	$\sigma = \sqrt{2}\sqrt{\frac{2}{n_{\text{inputs}} + n_{\text{outputs}}}}$

Figure 10: Various Initialization Methods

- To learn more, see 'Aurélien Géron(2017), Hands-On Machine Learning with Scikit-Learn and TensorFlow'.

6. Other Initialization Methods

- The three initialization methods I will introduce are mainly used in RNN.

  1. Orthogonal[6] : This is how you initialize using the Singular Value Decomposition(SVD). If $W = U\Lambda V^T$ then W is randomly generated from the standard normal distribution, and U calculated through the SVD is used as the initial value. Especially, it works well on RNN.
  2. Le et al.[4] : This is the initialization method used with ReLU. By giving $\boldsymbol{W} = \boldsymbol{I}$ and $\boldsymbol{b} = \boldsymbol{0}$ to their initial values, they start off ordinarily at first learning.
  3. Talathi et al.[7] : This is the initialization method used with ReLU. It is explained specifically in the next slide.

6. Other Initialization Methods

- Talathi et al., hypothesize that an initialization where one eigenvalue is equal to 1 and the rest are less than 1 is better.
- Talathi et al. initialization is as follows:
  1. Sample a matrix $\mathbf{A} \in R^{N \times N}$ whose values are drawn from $N(0,1)$ and N is the number of units in the RNN.
  2. Compute $\mathbf{B} = \frac{1}{N}\mathbf{A}\mathbf{A}^{\mathbf{T}}$ and let λ_{max} be the the largest eigenvalue of the matrix $\mathbf{B} + \mathbf{I}$.
  3. Initialize $\mathbf{W} = \frac{1}{\lambda_{max}}\mathbf{B} + \mathbf{I}$.
- Empirically this is better than initializing $\mathbf{W} = \mathbf{I}$.

Conclusion

- Initialization methods are still being studied as a major concern.
- Batch Normalization(BN) has the effect of making the initialization method less important.
- For the initial value distribution, there is no clear usage criteria for normal and uniform distribution.
- Nevertheless, we use nomral initialization rather than uniform, and in the case of CNN, we almost always use a combination of 'He initialization + ReLU'.

Next

- Next time, we'll deal with the followings:
  1. Type of loss function.
  2. More complex CNN models.
  3. About the GAN.
- Of course, Python code learning proceeds at the same time.

Reference I

[1] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[3] Geoffrey E Hinton. A practical guide to training restricted boltzmann machines. In *Neural networks: Tricks of the trade*, pages 599–619. Springer, 2012.

[4] Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

Reference II

[5] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[6] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[7] Sachin S Talathi and Aniket Vartak. Improving performance of recurrent neural network with relu nonlinearity. *arXiv preprint arXiv:1511.03771*, 2015.