Title: Weight initialization

URL: https://en.wikipedia.org/wiki/Weight_initialization

PageID: 78053249

Categories: Category:Artificial neural networks, Category:Deep learning, Category:Machine learning

---

Supervised learning

Unsupervised learning

Semi-supervised learning

Self-supervised learning

Reinforcement learning

Meta-learning

Online learning

Batch learning

Curriculum learning

Rule-based learning

Neuro-symbolic AI

Neuromorphic engineering

Quantum machine learning

Classification

Generative modeling

Regression

Clustering

Dimensionality reduction

Density estimation

Anomaly detection

Data cleaning

AutoML

Association rules

Semantic analysis

Structured prediction

Feature engineering

Feature learning

Learning to rank

Grammar induction

Ontology learning

Multimodal learning

Apprenticeship learning

Decision trees

Ensembles Bagging Boosting Random forest

Bagging

Boosting

Random forest

k -NN

Linear regression

Naive Bayes

Artificial neural networks

Logistic regression

Perceptron

Relevance vector machine (RVM)

Support vector machine (SVM)

BIRCH

CURE

Hierarchical

k -means

Fuzzy

Expectation–maximization (EM)

DBSCAN

OPTICS

Mean shift

Factor analysis

CCA

ICA

LDA

NMF

PCA

PGD

t-SNE

SDL

Graphical models Bayes net Conditional random field Hidden Markov

Bayes net

Conditional random field

Hidden Markov

RANSAC

k -NN

Local outlier factor

Isolation forest

Autoencoder

Deep learning

Feedforward neural network

Recurrent neural network LSTM GRU ESN reservoir computing

LSTM

GRU

ESN

reservoir computing

Boltzmann machine Restricted

Restricted

GAN

Diffusion model

SOM

Convolutional neural network U-Net LeNet AlexNet DeepDream

U-Net

LeNet

AlexNet

DeepDream

Neural field Neural radiance field Physics-informed neural networks

Neural radiance field

Physics-informed neural networks

Transformer Vision

Vision

Mamba

Spiking neural network

Memtransistor

Electrochemical RAM (ECRAM)

Q-learning

Policy gradient

SARSA

Temporal difference (TD)

Multi-agent Self-play

Self-play

Active learning

Crowdsourcing

Human-in-the-loop

Mechanistic interpretability

RLHF

Coefficient of determination

Confusion matrix

Learning curve

ROC curve

Kernel machines

Bias–variance tradeoff

Computational learning theory

Empirical risk minimization

Occam learning

PAC learning

Statistical learning

VC theory

Topological deep learning

AAAI

ECML PKDD

NeurIPS

ICML

ICLR

IJCAI

ML

JMLR

Glossary of artificial intelligence

List of datasets for machine-learning research List of datasets in computer vision and image processing

List of datasets in computer vision and image processing

Outline of machine learning

v

t

e

In deep learning , weight initialization or parameter initialization describes the initial step in creating a neural network . A neural network contains trainable parameters that are modified during training: weight initialization is the pre-training step of assigning initial values to these parameters.

The choice of weight initialization method affects the speed of convergence, the scale of neural activation within the network, the scale of gradient signals during backpropagation , and the quality of the final model. Proper initialization is necessary for avoiding issues such as vanishing and exploding gradients and activation function saturation .

Note that even though this article is titled "weight initialization", both weights and biases are used in a neural network as trainable parameters, so this article describes how both of these are initialized.

Similarly, trainable parameters in convolutional neural networks (CNNs) are called kernels and biases, and this article also describes these.

Constant initialization

We discuss the main methods of initialization in the context of a multilayer perceptron (MLP). Specific strategies for initializing other network architectures are discussed in later sections.

For an MLP, there are only two kinds of trainable parameters, called weights and biases. Each layer l contains a weight matrix $W^{(l)}\in \mathbb{R}^{n_{l-1}\times n_{l}}$ and a bias vector $b^{(l)}\in \mathbb{R}^{n_{l}}$, where n_{l} is the number of neurons in that layer. A weight initialization method is an algorithm for setting the initial values for $W^{(l)},b^{(l)}$ for each layer l.

The simplest form is zero initialization : $W^{(l)}=0$, $b^{(l)}=0$ Zero initialization is usually used for initializing biases, but it is not used for initializing weights, as it leads to symmetry in the network, causing all neurons to learn the same features.

In this page, we assume $b=0$ unless otherwise stated.

Recurrent neural networks typically use activation functions with bounded range, such as sigmoid and tanh, since unbounded activation may cause exploding values. (Le, Jaitly, Hinton, 2015) [1] suggested initializing weights in the recurrent parts of the network to identity and zero bias, similar to the idea of residual connections and LSTM with no forget gate.

In most cases, the biases are initialized to zero, though some situations can use a nonzero initialization. For example, in multiplicative units, such as the forget gate of LSTM , the bias can be initialized to 1 to allow good gradient signal through the gate. [2] For neurons with ReLU activation, one can initialize the bias to a small positive value like 0.1, so that the gradient is likely nonzero at initialization, avoiding the dying ReLU problem. [3] : 305 [4]

Random initialization

Random initialization means sampling the weights from a normal distribution or a uniform distribution , usually independently .

LeCun initialization

LeCun initialization , popularized in (LeCun et al., 1998), [5] is designed to preserve the variance of neural activations during the forward pass.

It samples each entry in $W^{(l)}$ independently from a distribution with mean 0 and variance $1/n_{l-1}$. For example, if the distribution is a continuous uniform distribution , then the distribution is $\mathcal{U}(\pm \sqrt{3/n_{l-1}})$.

Glorot initialization

Glorot initialization (or Xavier initialization) was proposed by Xavier Glorot and Yoshua Bengio . [6] It was designed as a compromise between two goals: to preserve activation variance during the forward pass and to preserve gradient variance during the backward pass.

For uniform initialization, it samples each entry in $W^{(l)}$ independently and identically from $\mathcal{U}(\pm \sqrt{6/(n_{l+1}+n_{l-1})})$. In the context, n_{l-1} is also called the " fan-in ", and n_{l+1} the " fan-out ". When the fan-in and fan-out are equal, then Glorot initialization is the same as LeCun initialization.

He initialization

As Glorot initialization performs poorly for ReLU activation, [7] He initialization (or Kaiming initialization) was proposed by Kaiming He et al. [8] for networks with ReLU activation. It samples each entry in $W^{(l)}$ from $\mathcal{N}(0,\sqrt{2/n_{l-1}})$.

Orthogonal initialization

(Saxe et al. 2013) [9] proposed orthogonal initialization : initializing weight matrices as uniformly random (according to the Haar measure) semi-orthogonal matrices , multiplied by a factor that depends on the activation function of the layer. It was designed so that if one initializes a deep linear network this way, then its training time until convergence is independent of depth. [10]

Sampling a uniformly random semi-orthogonal matrix can be done by initializing X {\displaystyle X} by IID sampling its entries from a standard normal distribution, then calculate (X X ■) – 1 / 2 X {\displaystyle \left(XX^{\top }\right)^{-1/2}X} or its transpose, depending on whether X {\displaystyle X} is tall or wide. [11]

For CNN kernels with odd widths and heights, orthogonal initialization is done this way: initialize the central point by a semi-orthogonal matrix, and fill the other entries with zero. As an illustration, a kernel K {\displaystyle K} of shape 3 × 3 × c × c ′ {\displaystyle 3\times 3\times c\times c'} is initialized by filling K [2 , 2 , : , :] {\displaystyle K[2,2,:,:]} with the entries of a random semi-orthogonal matrix of shape c × c ′ {\displaystyle c\times c'} , and the other entries with zero. (Balduzzi et al., 2017) [12] used it with stride 1 and zero-padding. This is sometimes called the Orthogonal Delta initialization . [11] [13]

Related to this approach, unitary initialization proposes to parameterize the weight matrices to be unitary matrices , with the result that at initialization they are random unitary matrices (and throughout training, they remain unitary). This is found to improve long-sequence modelling in LSTM. [14] [15]

Orthogonal initialization has been generalized to layer-sequential unit-variance (LSUV) initialization . It is a data-dependent initialization method, and can be used in convolutional neural networks . It first initializes weights of each convolution or fully connected layer with orthonormal matrices. Then, proceeding from the first to the last layer, it runs a forward pass on a random minibatch, and divides the layer's weights by the standard deviation of its output, so that its output has variance approximately 1. [16] [17]

Fixup initialization

In 2015, the introduction of residual connections allowed very deep neural networks to be trained, much deeper than the ~20 layers of the previous state of the art (such as the VGG-19). Residual connections gave rise to their own weight initialization problems and strategies. These are sometimes called "normalization-free" methods, since using residual connection could stabilize the training of a deep neural network so much that normalizations become unnecessary.

Fixup initialization is designed specifically for networks with residual connections and without batch normalization, as follows: [18]

Initialize the classification layer and the last layer of each residual branch to 0.

Initialize every other layer using a standard method (such as He initialization), and scale only the weight layers inside residual branches by L – 1 2 m – 2 {\displaystyle L^{-{\frac {1}{2m-2}}}} .

Add a scalar multiplier (initialized at 1) in every branch and a scalar bias (initialized at 0) before each convolution, linear, and element-wise activation layer.

Similarly, T-Fixup initialization is designed for Transformers without layer normalization . [19] : 9

Others

Instead of initializing all weights with random values on the order of O (1 / n) {\displaystyle O(1/{\sqrt {n}})} , sparse initialization initialized only a small subset of the weights with larger random values, and the other weights zero, so that the total variance is still on the order of O (1) {\displaystyle O(1)} . [20]

Random walk initialization was designed for MLP so that during backpropagation, the L2 norm of gradient at each layer performs an unbiased random walk as one moves from the last layer to the first. [21]

Looks linear initialization was designed to allow the neural network to behave like a deep linear network at initialization, since W R e L U (x) − W R e L U (− x) = W x {\displaystyle W\;\mathrm {ReLU} (x)-W\;\mathrm {ReLU} (-x)=Wx} . It initializes a matrix W {\displaystyle W} of shape R n 2 × m {\displaystyle \mathbb {R} ^{{\frac {n}{2}}\times m}} by any method, such as orthogonal initialization, then let the R n × m {\displaystyle \mathbb {R} ^{n\times m}} weight matrix to be the concatenation of W , − W {\displaystyle W,-W} . [22]

Miscellaneous

For hyperbolic tangent activation function, a particular scaling is sometimes used: 1.7159 tanh ■ (2 x / 3) {\displaystyle 1.7159\tanh(2x/3)} . This was sometimes called " LeCun 's tanh ". It was designed so that it maps the interval [− 1 , + 1] {\displaystyle [-1,+1]} to itself, thus ensuring that the overall gain is around 1 in "normal operating conditions", and that | f ″ (x) | {\displaystyle |f''(x)|} is at maximum when x = − 1 , + 1 {\displaystyle x=-1,+1} , which improves convergence at the end of training. [23] [5]

In self-normalizing neural networks , the SELU activation function S E L U (x) = λ { x if x > 0 α e x − α if x ≤ 0 {\displaystyle \mathrm {SELU} (x)=\lambda {\begin{cases}x&{\text{if }}x>0\\\alpha e^{x}-\alpha &{\text{if }}x\leq 0\end{cases}}} with parameters λ ≈ 1.0507 , α ≈ 1.6733 {\displaystyle \lambda \approx 1.0507,\alpha \approx 1.6733} makes it such that the mean and variance of the output of each layer has (0 , 1) {\displaystyle (0,1)} as an attracting fixed-point. This makes initialization less important, though they recommend initializing weights randomly with variance 1 / n l − 1 {\displaystyle 1/n_{l-1}} . [24]

History

Random weight initialization was used since Frank Rosenblatt 's perceptrons . An early work that described weight initialization specifically was (LeCun et al., 1998). [5]

Before the 2010s era of deep learning, it was common to initialize models by "generative pre-training" using an unsupervised learning algorithm that is not backpropagation, as it was difficult to directly train deep neural networks by backpropagation. [25] [26] For example, a deep belief network was trained by using contrastive divergence layer by layer, starting from the bottom. [27]

(Martens, 2010) [20] proposed Hessian-free Optimization, a quasi-Newton method to directly train deep networks. The work generated considerable excitement that initializing networks without pre-training phase was possible. [28] However, a 2013 paper demonstrated that with well-chosen hyperparameters, momentum gradient descent with weight initialization was sufficient for training neural networks, without needing either quasi-Newton method or generative pre-training, a combination that is still in use as of 2024. [29]

Since then, the impact of initialization on tuning the variance has become less important, with methods developed to automatically tune variance, like batch normalization tuning the variance of the forward pass, [30] and momentum-based optimizers tuning the variance of the backward pass. [31]

There is a tension between using careful weight initialization to decrease the need for normalization, and using normalization to decrease the need for careful weight initialization, with each approach having its tradeoffs. For example, batch normalization causes training examples in the minibatch to become dependent, an undesirable trait, while weight initialization is architecture-dependent. [32]

See also

Backpropagation

Normalization (machine learning)

Gradient descent

Vanishing gradient problem

References

Further reading

Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016). "8.4 Parameter Initialization Strategies" . Deep learning . Adaptive computation and machine learning. Cambridge, Mass: The MIT press. ISBN 978-0-262-03561-3 .

Narkhede, Meenal V.; Bartakke, Prashant P.; Sutaone, Mukul S. (June 28, 2021). "A review on weight initialization strategies for neural networks". Artificial Intelligence Review . 55 (1). Springer Science and Business Media LLC: 291– 322. doi : 10.1007/s10462-021-10033-z . ISSN 0269-2821 .

v

t

e

History timeline

timeline

Companies

Projects

Parameter Hyperparameter

Hyperparameter

Loss functions

Regression Bias–variance tradeoff Double descent Overfitting

Bias–variance tradeoff

Double descent

Overfitting

Clustering

Gradient descent SGD Quasi-Newton method Conjugate gradient method

SGD

Quasi-Newton method

Conjugate gradient method

Backpropagation

Attention

Convolution

Normalization Batchnorm

Batchnorm

Activation Softmax Sigmoid Rectifier

Softmax

Sigmoid

Rectifier

Gating

Weight initialization

Regularization

Datasets Augmentation

Augmentation

Prompt engineering

Reinforcement learning Q-learning SARSA Imitation Policy gradient

Q-learning

SARSA

Imitation

Policy gradient

Diffusion

Latent diffusion model

Autoregression

Adversary

RAG

Uncanny valley

RLHF

Self-supervised learning

Reflection

Recursive self-improvement

Hallucination

Word embedding

Vibe coding

Machine learning In-context learning

In-context learning

Artificial neural network Deep learning

Deep learning

Language model Large language model NMT

Large language model

NMT

Reasoning language model

Model Context Protocol

Intelligent agent

Artificial human companion

Humanity's Last Exam

Artificial general intelligence (AGI)

AlexNet

WaveNet

Human image synthesis

HWR

OCR

Computer vision

Speech synthesis 15.ai ElevenLabs

15.ai

ElevenLabs

Speech recognition Whisper

Whisper

Facial recognition

AlphaFold

Text-to-image models Aurora DALL-E Firefly Flux Ideogram Imagen Midjourney Recraft Stable Diffusion

Aurora

DALL-E

Firefly

Flux

Ideogram

Imagen

Midjourney

Recraft

Stable Diffusion

Text-to-video models Dream Machine Runway Gen Hailuo AI Kling Sora Veo

Dream Machine

Runway Gen

Hailuo AI

Kling

Sora

Veo

Music generation Riffusion Suno AI Udio

Riffusion

Suno AI

Udio

Word2vec

Seq2seq

GloVe

BERT

T5

Llama

Chinchilla AI

PaLM

GPT 1 2 3 J ChatGPT 4 4o o1 o3 4.5 4.1 o4-mini 5

1

2

3

J

ChatGPT

4

4o

o1

o3

4.5

4.1

o4-mini

5

Claude

Gemini Gemini (language model) Gemma

Gemini (language model)

Gemma

Grok

LaMDA

BLOOM

DBRX

Project Debater

IBM Watson

IBM Watsonx

Granite

PanGu-Σ

DeepSeek

Qwen

AlphaGo

AlphaZero

OpenAI Five

Self-driving car

MuZero

Action selection AutoGPT

AutoGPT

Robot control

Alan Turing

Warren Sturgis McCulloch
Walter Pitts
John von Neumann
Claude Shannon
Shun'ichi Amari
Kunihiko Fukushima
Takeo Kanade
Marvin Minsky
John McCarthy
Nathaniel Rochester
Allen Newell
Cliff Shaw
Herbert A. Simon
Oliver Selfridge
Frank Rosenblatt
Bernard Widrow
Joseph Weizenbaum
Seymour Papert
Seppo Linnainmaa
Paul Werbos
Geoffrey Hinton
John Hopfield
Jürgen Schmidhuber
Yann LeCun
Yoshua Bengio
Lotfi A. Zadeh
Stephen Grossberg
Alex Graves
James Goodnight
Andrew Ng
Fei-Fei Li
Alex Krizhevsky
Ilya Sutskever
Oriol Vinyals
Quoc V. Le
Ian Goodfellow
Demis Hassabis
David Silver

Andrej Karpathy

Ashish Vaswani

Noam Shazeer

Aidan Gomez

John Schulman

Mustafa Suleyman

Jan Leike

Daniel Kokotajlo

François Chollet

Neural Turing machine

Differentiable neural computer

Transformer Vision transformer (ViT)

Vision transformer (ViT)

Recurrent neural network (RNN)

Long short-term memory (LSTM)

Gated recurrent unit (GRU)

Echo state network

Multilayer perceptron (MLP)

Convolutional neural network (CNN)

Residual neural network (RNN)

Highway network

Mamba

Autoencoder

Variational autoencoder (VAE)

Generative adversarial network (GAN)

Graph neural network (GNN)

Category