Title: Feedforward neural network

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

PageID: 1706332

Categories: Category:Neural network architectures

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

Feedforward refers to recognition-inference architecture of neural networks. Artificial neural network architectures are based on inputs multiplied by weights to obtain outputs (inputs-to-output): feedforward. [2] Recurrent neural networks , or neural networks with loops allow information from later processing stages to feed back to earlier stages for sequence processing. [3] However, at every stage of inference a feedforward multiplication remains the core, essential for backpropagation [4] [5] [6] [7] [8] or backpropagation through time. Thus neural networks cannot contain feedback like negative feedback or positive feedback where the outputs feed back to the very same inputs and modify them, because this forms an infinite loop which is not possible to rewind in time to generate an error signal through backpropagation. This issue and nomenclature appear to be a point of confusion between some computer scientists and scientists in other fields studying brain networks. [9]

Mathematical foundations

Activation function

The two historically common activation functions are both sigmoids , and are described by

$$y(v_{i})=\tanh(v_{i})~~\text{and}~~y(v_{i})=(1+e^{-v_{i}})^{-1}.$$

The first is a hyperbolic tangent that ranges from -1 to 1, while the other is the logistic function , which is similar in shape but ranges from 0 to 1. Here y_{i} is the output of the i -th node (neuron) and v_{i} is the weighted sum of the input connections. Alternative activation functions have been proposed, including the rectifier and softplus functions. More specialized activation functions include radial basis functions (used in radial basis networks , another class of supervised neural network models).

In recent developments of deep learning the rectified linear unit (ReLU) is more frequently used as one of the possible ways to overcome the numerical problems related to the sigmoids.

Learning

Learning occurs by changing connection weights after each piece of data is processed, based on the amount of error in the output compared to the expected result. This is an example of supervised learning , and is carried out through backpropagation.

We can represent the degree of error in an output node j in the n -th data point (training example) by $e_{j}(n)=d_{j}(n)-y_{j}(n)$, where $d_{j}(n)$ is the desired target value for n -th data point at node j , and $y_{j}(n)$ is the value produced at node j when the n -th data point is given as an input.

The node weights can then be adjusted based on corrections that minimize the error in the entire output for the n -th data point, given by

$${\mathcal {E}}(n)={\frac {1}{2}}\sum _{{\text{output node }}j}e_{j}^{2}(n).$$

Using gradient descent , the change in each weight w_{ij} is

$$\Delta w_{ji}(n)=-\eta {\frac {\partial {\mathcal {E}}(n)}{\partial v_{j}(n)}}y_{i}(n)$$

where $y_{i}(n)$ is the output of the previous neuron i , and η is the learning rate , which is selected to ensure that the weights quickly converge to a response, without oscillations. In the previous expression, ${\frac {\partial {\mathcal {E}}(n)}{\partial v_{j}(n)}}$ denotes the partial derivative of the error ${\mathcal {E}}(n)$ according to the weighted sum $v_{j}(n)$ of the input connections of neuron i .

The derivative to be calculated depends on the induced local field v_{j} , which itself varies. It is easy to prove that for an output node this derivative can be simplified to

$$-{\frac {\partial {\mathcal {E}}(n)}{\partial v_{j}(n)}}=e_{j}(n)\phi ^{\prime }(v_{j}(n))$$

where $\phi ^{\prime }$ is the derivative of the activation function described above, which itself does not vary. The analysis is more difficult for the change in weights to a hidden node, but it can be shown that the relevant derivative is

$$-{\frac {\partial {\mathcal {E}}(n)}{\partial v_{j}(n)}}=\phi ^{\prime }(v_{j}(n))\sum _{k}-{\frac {\partial {\mathcal {E}}(n)}{\partial v_{k}(n)}}w_{kj}(n).$$

This depends on the change in weights of the k th nodes, which represent the output layer. So to change the hidden layer weights, the output layer weights change according to the derivative of the activation function, and so this algorithm represents a backpropagation of the activation function. [10]

History

Timeline

Circa 1800, Legendre (1805) and Gauss (1795) created the simplest feedforward network which consists of a single weight layer with linear activation functions. It was trained by the least squares method for minimising mean squared error , also known as linear regression . Legendre and Gauss used it for the prediction of planetary movement from training data. [11] [12] [13] [14] [15]

In 1943, Warren McCulloch and Walter Pitts proposed the binary artificial neuron as a logical model of biological neural networks. [16]

In 1958, Frank Rosenblatt proposed the multilayered perceptron model, consisting of an input layer, a hidden layer with randomized weights that did not learn, and an output layer with learnable connections. [17] R. D. Joseph (1960) [18] mentions an even earlier perceptron-like device: [13] "Farley and Clark of MIT Lincoln Laboratory actually preceded Rosenblatt in the development of a perceptron-like device." However, "they dropped the subject."

In 1960, Joseph [18] also discussed multilayer perceptrons with an adaptive hidden layer. Rosenblatt (1962) [19] : section 16 cited and adopted these ideas, also crediting work by H. D. Block and B. W. Knight. Unfortunately, these early efforts did not lead to a working learning algorithm for hidden units, i.e., deep learning .

In 1965, Alexey Grigorevich Ivakhnenko and Valentin Lapa published Group Method of Data Handling , the first working deep learning algorithm, a method to train arbitrarily deep neural networks. [20] [21] It is based on layer by layer training through regression analysis. Superfluous hidden units are pruned using a separate validation set. Since the activation functions of the nodes are Kolmogorov-Gabor polynomials, these were also the first deep networks with multiplicative units or "gates." [13] It was used to train an eight-layer neural net in 1971.

In 1967, Shun'ichi Amari reported [22] the first multilayered neural network trained by stochastic gradient descent , which was able to classify non-linearily separable pattern classes. Amari's student Saito conducted the computer experiments, using a five-layered feedforward network with two learning layers. [13]

In 1970, Seppo Linnainmaa published the modern form of backpropagation in his master thesis (1970). [23] [24] [13] G.M. Ostrovski et al. republished it in 1971. [25] [26] Paul Werbos applied backpropagation to neural networks in 1982 [7] [27] (his 1974 PhD thesis, reprinted in a 1994 book, [28] did not yet describe the algorithm [26]). In 1986, David E. Rumelhart et al. popularised backpropagation but did not cite the original work. [29] [8]

In 2003, interest in backpropagation networks returned due to the successes of deep learning being applied to language modelling by Yoshua Bengio with co-authors. [30]

Linear regression

Perceptron

If using a threshold, i.e. a linear activation function, the resulting linear threshold unit is called a perceptron . (Often the term is used to denote just one of these units.) Multiple parallel non-linear units are able to approximate any continuous function from a compact interval of the real numbers into the interval [–1,1] despite the limited computational power of single unit with a linear threshold function. [31]

Perceptrons can be trained by a simple learning algorithm that is usually called the delta rule . It calculates the errors between calculated output and sample output data, and uses this to create an adjustment to the weights, thus implementing a form of gradient descent .

Multilayer perceptron

A multilayer perceptron (MLP) is a misnomer for a modern feedforward artificial neural network, consisting of fully connected neurons (hence the synonym sometimes used of fully connected network (FCN)), often with a nonlinear kind of activation function, organized in at least three layers, notable for being able to distinguish data that is not linearly separable . [32]

Other feedforward networks

Examples of other feedforward networks include convolutional neural networks and radial basis function networks , which use a different activation function.

See also

Feed forward (control)

Hopfield network

Rprop

References

External links

Feedforward neural networks tutorial

Feedforward Neural Network: Example

Feedforward Neural Networks: An Introduction

v

t

e

Differentiable programming

Information geometry

Statistical manifold

Automatic differentiation

Neuromorphic computing

Pattern recognition

Ricci calculus

Computational learning theory

Inductive bias

IPU

TPU

VPU

Memristor

SpiNNaker

TensorFlow

PyTorch

Keras

scikit-learn

Theano

JAX

Flux.jl

MindSpore

Portals Computer programming Technology

Computer programming

Technology