Title: Softmax function

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

PageID: 6152185

Categories: Category:Artificial neural networks, Category:Computational neuroscience, Category:Exponentials, Category:Functions and mappings, Category:Logistic regression

-----

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

The softmax function, also known as softargmax [1] : 184 or normalized exponential function , [2] : 198 converts a tuple of K real numbers into a probability distribution over K possible outcomes. It is a generalization of the logistic function to multiple dimensions, and is used in multinomial logistic regression . The softmax function is often used as the last activation function of a neural network to normalize the output of a network to a probability distribution over predicted output classes.

Definition

The softmax function takes as input a tuple z of K real numbers, and normalizes it into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers. That is, prior to applying softmax, some tuple components could be negative, or greater than one; and might not sum to 1; but after applying softmax, each component will be in the interval (0 , 1) {\displaystyle (0,1)} , and the components will add up to 1, so that they can be interpreted as probabilities. Furthermore, the larger input components will correspond to larger probabilities.

Formally, the standard (unit) softmax function σ : R K → (0 , 1) K {\displaystyle \sigma \colon \mathbb {R} ^{K}\to (0,1)^{K}} , where ■ K > 1 {\displaystyle K>1} ■ , takes a tuple z = (z 1 , … , z K) ∈ R K {\displaystyle \mathbf {z} =(z_{1},\dotsc ,z_{K})\in \mathbb {R} ^{K}} and computes each component of vector σ (z) ∈ (0 , 1) K {\displaystyle \sigma (\mathbf {z})\in (0,1)^{K}} with

σ (z) i = e z i ∑ j = 1 K e z j . {\displaystyle \sigma (\mathbf {z})_{i}={\frac {e^{z_{i}}}{\sum _{j=1}^{K}e^{z_{j}}}}\,.}

In words, the softmax applies the standard exponential function to each element z i {\displaystyle z_{i}} of the input tuple z {\displaystyle \mathbf {z} } (consisting of K {\displaystyle K} real numbers), and normalizes these values by dividing by the sum of all these exponentials. The normalization ensures that the sum of the components of the output vector σ (z) {\displaystyle \sigma (\mathbf {z})} is 1. The term "softmax" derives from the amplifying effects of the exponential on any maxima in the input tuple. For example, the standard softmax of (1 , 2 , 8) {\displaystyle (1,2,8)} is approximately (0.001 , 0.002 , 0.997) {\displaystyle (0.001,0.002,0.997)} , which amounts to assigning almost all of the total unit weight in the result to the position of the tuple's maximal element (of 8).

In general, instead of e a different base b > 0 can be used. As above, if b > 1 then larger input components will result in larger output probabilities, and increasing the value of b will create probability distributions that are more concentrated around the positions of the largest input values. Conversely, if 0 < b < 1 then smaller input components will result in larger output probabilities, and decreasing the value of b will create probability distributions that are more concentrated around the positions of the smallest input values. Writing b = e β {\displaystyle b=e^{\beta }} or b = e − β {\displaystyle b=e^{-\beta }} [a] (for real β) [b] yields the expressions: [c]

σ (z) i = e β z i ∑ j = 1 K e β z j or σ (z) i = e − β z i ∑ j = 1 K e − β z j for i = 1 , … , K . {\displaystyle \sigma (\mathbf {z})_{i}={\frac {e^{\beta z_{i}}}{\sum _{j=1}^{K}e^{\beta z_{j}}}}{\text{ or }}\sigma (\mathbf {z})_{i}={\frac {e^{-\beta z_{i}}}{\sum _{j=1}^{K}e^{-\beta z_{j}}}}{\text{ for }}i=1,\dotsc ,K.}

A value proportional to the reciprocal of β is sometimes referred to as the temperature : β = 1 / k T {\textstyle \beta =1/kT} , where k is typically 1 or the Boltzmann constant and T is the temperature. A higher temperature results in a more uniform output distribution (i.e. with higher entropy ; it is "more random"), while a lower temperature results in a sharper output distribution, with one value dominating.

In some fields, the base is fixed, corresponding to a fixed scale, [d] while in others the parameter β (or T) is varied.

Interpretations

Smooth arg max

The Softmax function is a smooth approximation to the arg max function: the function whose value is the index of a tuple's largest element. The name "softmax" may be misleading. Softmax is not a smooth maximum (that is, a smooth approximation to the maximum function). The term "softmax" is also used for the closely related LogSumExp function, which is a smooth maximum. For this reason, some prefer the more accurate term "softargmax", though the term "softmax" is conventional in machine learning. [3] [4] This section uses the term "softargmax" for clarity.

Formally, instead of considering the arg max as a function with categorical output 1 , … , n {\displaystyle 1,\dots ,n} (corresponding to the index), consider the arg max function with one-hot representation of the output (assuming there is a unique maximum arg): a r g m a x ■ (z 1 , … , z n) = (y 1 , … , y n) = (0 , … , 0 , 1 , 0 , … , 0) , {\displaystyle \operatorname {arg\,max} (z_{1},\,\dots ,\,z_{n})=(y_{1},\,\dots ,\,y_{n})=(0,\,\dots ,\,0,\,1,\,0,\,\dots ,\,0),} where the output coordinate y i = 1 {\displaystyle y_{i}=1} if and only if i {\displaystyle i} is the arg max of (z 1 , … , z n) {\displaystyle (z_{1},\dots ,z_{n})} , meaning z i {\displaystyle z_{i}} is the unique maximum value of (z 1 , … , z n

) {\displaystyle (z_{1},\,\dots ,\,z_{n})} . For example, in this encoding a r g m a x ■ (1 , 5 , 10) = (0 , 0 , 1) , {\displaystyle \operatorname {arg\,max} (1,5,10)=(0,0,1),} since the third argument is the maximum.

This can be generalized to multiple arg max values (multiple equal z i {\displaystyle z_{i}} being the maximum) by dividing the 1 between all max args; formally 1/k where k is the number of arguments assuming the maximum. For example, a r g m a x ■ (1 , 5 , 5) = (0 , 1 / 2 , 1 / 2) , {\displaystyle \operatorname {arg\,max} (1,\,5,\,5)=(0,\,1/2,\,1/2),} since the second and third argument are both the maximum. In case all arguments are equal, this is simply a r g m a x ■ (z , … , z) = (1 / n , … , 1 / n) . {\displaystyle \operatorname {arg\,max} (z,\dots ,z)=(1/n,\dots ,1/n).} Points z with multiple arg max values are singular points (or singularities, and form the singular set) – these are the points where arg max is discontinuous (with a jump discontinuity) – while points with a single arg max are known as non-singular or regular points.

With the last expression given in the introduction, softargmax is now a smooth approximation of arg max: as ■ β → ∞ {\displaystyle \beta \to \infty } ■ , softargmax converges to arg max. There are various notions of convergence of a function; softargmax converges to arg max pointwise , meaning for each fixed input z as ■ β → ∞ {\displaystyle \beta \to \infty } ■ , σ β (z) → a r g m a x ■ (z) . {\displaystyle \sigma _{\beta }(\mathbf {z})\to \operatorname {arg\,max} (\mathbf {z}).} However, softargmax does not converge uniformly to arg max, meaning intuitively that different points converge at different rates, and may converge arbitrarily slowly. In fact, softargmax is continuous, but arg max is not continuous at the singular set where two coordinates are equal, while the uniform limit of continuous functions is continuous. The reason it fails to converge uniformly is that for inputs where two coordinates are almost equal (and one is the maximum), the arg max is the index of one or the other, so a small change in input yields a large change in output. For example, σ β (1 , 1.0001) → (0 , 1) , {\displaystyle \sigma _{\beta }(1,\,1.0001)\to (0,1),} but σ β (1 , 0.9999) → (1 , 0) , {\displaystyle \sigma _{\beta }(1,\,0.9999)\to (1,\,0),} and σ β (1 , 1) = 1 / 2 {\displaystyle \sigma _{\beta }(1,\,1)=1/2} for all inputs: the closer the points are to the singular set (x , x) {\displaystyle (x,x)} , the slower they converge. However, softargmax does converge compactly on the non-singular set.

Conversely, as ■ β → − ∞ {\displaystyle \beta \to -\infty } ■ , softargmax converges to arg min in the same way, where here the singular set is points with two arg min values. In the language of tropical analysis , the softmax is a deformation or "quantization" of arg max and arg min, corresponding to using the log semiring instead of the max-plus semiring (respectively min-plus semiring), and recovering the arg max or arg min by taking the limit is called "tropicalization" or "dequantization".

It is also the case that, for any fixed β , if one input ■ z i {\displaystyle z_{i}} ■ is much larger than the others relative to the temperature, T = 1 / β {\displaystyle T=1/\beta } , the output is approximately the arg max. For example, a difference of 10 is large relative to a temperature of 1: σ (0 , 10) := σ 1 (0 , 10) = (1 / (1 + e 10) , e 10 / (1 + e 10)) ≈ (0.00005 , 0.99995) {\displaystyle \sigma (0,\,10):=\sigma _{1}(0,\,10)=\left(1/\left(1+e^{10}\right),\,e^{10}/\left(1+e^{10}\right)\right)\approx (0.00005,\,0.99995)} However, if the difference is small relative to the temperature, the value is not close to the arg max. For example, a difference of 10 is small relative to a temperature of 100: σ 1 / 100 (0 , 10) = (1 / (1 + e 1 / 10) , e 1 / 10 / (1 + e 1 / 10)) ≈ (0.475 , 0.525) . {\displaystyle \sigma _{1/100}(0,\,10)=\left(1/\left(1+e^{1/10}\right),\,e^{1/10}/\left(1+e^{1/10}\right)\right)\approx (0.475,\,0.525).} As ■ β → ∞ {\displaystyle \beta \to \infty } ■ , temperature goes to zero, T = 1 / β → 0 {\displaystyle T=1/\beta \to 0} , so eventually all differences become large (relative to a shrinking temperature), which gives another interpretation for the limit behavior.

Statistical mechanics

In statistical mechanics , the softargmax function is known as the Boltzmann distribution (or Gibbs distribution): [5] : 7 the index set 1 , … , k {\displaystyle {1,\,\dots ,\,k}} are the microstates of the system; the inputs z i {\displaystyle z_{i}} are the energies of that state; the denominator is known as the partition function , often denoted by Z ; and the factor β is called the coldness (or thermodynamic beta , or inverse temperature).

Applications

The softmax function is used in various multiclass classification methods, such as multinomial logistic regression (also known as softmax regression), [2] : 206–209 [6] multiclass linear discriminant analysis , naive Bayes classifiers , and artificial neural networks . [7] Specifically, in multinomial logistic regression and linear discriminant analysis, the input to the function is the result of K distinct linear functions , and the predicted probability for the j th class given a sample tuple x and a weighting vector w is:

P (y = j ■ x) = e x T w j ∑ k = 1 K e x T w k {\displaystyle P(y=j\mid \mathbf {x})={\frac {e^{\mathbf {x} ^{\mathsf {T}}\mathbf {w} _{j}}}{\sum _{k=1}^{K}e^{\mathbf {x} ^{\mathsf {T}}\mathbf {w} _{k}}}}}

This can be seen as the composition of K linear functions x ■ x T w 1 , … , x ■ x T w K {\displaystyle \mathbf {x} \mapsto \mathbf {x} ^{\mathsf {T}}\mathbf {w} _{1},\ldots ,\mathbf {x} \mapsto \mathbf {x} ^{\mathsf {T}}\mathbf {w} _{K}} and the softmax function (where x T w {\displaystyle \mathbf {x} ^{\mathsf {T}}\mathbf {w} } denotes the inner product of x {\displaystyle \mathbf {x} } and w {\displaystyle \mathbf {w} }). The operation is equivalent to applying a linear operator defined by w {\displaystyle \mathbf {w} } to tuples x {\displaystyle \mathbf {x} } , thus transforming the original, probably highly-dimensional, input to vectors in a K -dimensional space R K {\displaystyle \mathbb {R} ^{K}} .

Neural networks

The standard softmax function is often used in the final layer of a neural network-based classifier. Such networks are commonly trained under a log loss (or cross-entropy) regime, giving a non-linear variant of multinomial logistic regression.

Since the function maps a tuple and a specific index i {\displaystyle i} to a real value, the derivative needs to take the index into account:

∂ ∂ q k σ (q , i) = σ (q , i) (δ i k − σ (q , k)) . {\displaystyle {\frac {\partial }{\partial q_{k}}}\sigma ({\textbf {q}},i)=\sigma ({\textbf {q}},i)(\delta _{ik}-\sigma ({\textbf {q}},k)).}

This expression is symmetrical in the indexes i , k {\displaystyle i,k} and thus may also be expressed as

∂ ∂ q k σ (q , i) = σ (q , k) (δ i k − σ (q , i)) . {\displaystyle {\frac {\partial }{\partial q_{k}}}\sigma ({\textbf {q}},i)=\sigma ({\textbf {q}},k)(\delta _{ik}-\sigma ({\textbf {q}},i)).}

Here, the Kronecker delta is used for simplicity (cf. the derivative of a sigmoid function , being expressed via the function itself).

To ensure stable numerical computations subtracting the maximum value from the input tuple is common. This approach, while not altering the output or the derivative theoretically, enhances stability by directly controlling the maximum exponent value computed.

If the function is scaled with the parameter β {\displaystyle \beta } , then these expressions must be multiplied by β {\displaystyle \beta } .

See multinomial logit for a probability model which uses the softmax activation function.

Reinforcement learning

In the field of reinforcement learning , a softmax function can be used to convert values into action probabilities. The function commonly used is: [8] P t (a) = exp ■ (q t (a) / τ) ∑ i = 1 n exp ■ (q t (i) / τ) , {\displaystyle P_{t}(a)={\frac {\exp(q_{t}(a)/\tau)}{\sum _{i=1}^{n}\exp(q_{t}(i)/\tau)}}{\text{,}}}

where the action value q t (a) {\displaystyle q_{t}(a)} corresponds to the expected reward of following action a and τ {\displaystyle \tau } is called a temperature parameter (in allusion to statistical mechanics). For high temperatures (τ → ∞ {\displaystyle \tau \to \infty }), all actions have nearly the same probability and the lower the temperature, the more expected rewards affect the probability. For a low temperature (τ → 0 + {\displaystyle \tau \to 0^{+}}), the probability of the action with the highest expected reward tends to 1.

Computational complexity and remedies

In neural network applications, the number K of possible outcomes is often large, e.g. in case of neural language models that predict the most likely outcome out of a vocabulary which might contain millions of possible words. [9] This can make the calculations for the softmax layer (i.e. the matrix multiplications to determine the z i {\displaystyle z_{i}} , followed by the application of the softmax function itself) computationally expensive. [9] [10] What's more, the gradient descent backpropagation method for training such a neural network involves calculating the softmax for every training example, and the number of training examples can also become large. The computational effort for the softmax became a major limiting factor in the development of larger neural language models, motivating various remedies to reduce training times. [9] [10]

Approaches that reorganize the softmax layer for more efficient calculation include the hierarchical softmax and the differentiated softmax . [9] The hierarchical softmax (introduced by Morin and Bengio in 2005) uses a binary tree structure where the outcomes (vocabulary words) are the leaves and the intermediate nodes are suitably selected "classes" of outcomes, forming latent variables . [10] [11] The desired probability (softmax value) of a leaf (outcome) can then be calculated as the product of the probabilities of all nodes on the path from the root to that leaf. [10] Ideally, when the tree is balanced, this would reduce the computational complexity from O (K) {\displaystyle O(K)} to O (log 2 ■ K) {\displaystyle O(\log _{2}K)} . [11] In practice, results depend on choosing a good strategy for clustering the outcomes into classes. [10] [11] A Huffman tree was used for this in Google's word2vec models (introduced in 2013) to achieve scalability. [9]

A second kind of remedies is based on approximating the softmax (during training) with modified loss functions that avoid the calculation of the full normalization factor. [9] These include methods that restrict the normalization sum to a sample of outcomes (e.g. Importance Sampling, Target Sampling). [9] [10]

Numerical algorithms

The standard softmax is numerically unstable because of large exponentiations. The safe softmax method calculates instead σ (z) i = e β (z i − m) ∑ j = 1 K e β (z j − m) {\displaystyle \sigma (\mathbf {z})_{i}={\frac {e^{\beta (z_{i}-m)}}{\sum _{j=1}^{K}e^{\beta (z_{j}-m)}}}} where m = max i z i {\displaystyle m=\max _{i}z_{i}} is the largest factor involved. Subtracting by it guarantees that the exponentiations result in at most 1.

The attention mechanism in Transformers takes three arguments: a "query vector" q {\displaystyle q} , a list of "key vectors" k 1 , … , k N {\displaystyle k_{1},\dots ,k_{N}} , and a list of "value vectors" v 1 , … , v N {\displaystyle v_{1},\dots ,v_{N}} , and outputs a softmax-weighted sum over value vectors: o = ∑ i = 1 N e q T k i − m ∑ j = 1 N e q T k j − m v i {\displaystyle o=\sum _{i=1}^{N}{\frac {e^{q^{T}k_{i}-m}}{\sum _{j=1}^{N}e^{q^{T}k_{j}-m}}}v_{i}} The standard softmax method involves several loops over the inputs, which would be bottlenecked by memory bandwidth . The FlashAttention method is a communication-avoiding algorithm that fuses these operations into a single loop, increasing the arithmetic intensity . It is an online algorithm that computes the following quantities: [12] [13] z i = q T k i m i = max (z 1 , … , z i) = max (m i − 1 , z i) l i = e z 1 − m i + ■ + e z i − m i = e m i − 1 − m i l i − 1 + e z i − m i o i = e z 1 − m i v 1 + ■ + e z i − m i v i = e m i − 1 − m i o i − 1 + e z i − m i v i {\displaystyle {\begin{aligned}z_{i}&=q^{T}k_{i}&\\m_{i}&=\max(z_{1},\dots ,z_{i})&=&\max(m_{i-1},z_{i})\\l_{i}&=e^{z_{1}-m_{i}}+\dots +e^{z_{i}-m_{i}}&=&e;^{m_{i-1}-m_{i}}l_{i-1}+e^{z_{i}-m_{i}}\\o_{i}&=e^{z_{1}-m_{i}}v_{1}+\dots +e^{z_{i}-m_{i}}v_{i}&=&e;^{m_{i-1}-m_{i}}o_{i-1}+e^{z_{i}-m_{i}}v_{i}\end{aligned}}} and returns o N / l N {\displaystyle o_{N}/l_{N}} . In practice, FlashAttention operates over multiple queries and keys per loop iteration, in a similar way as blocked matrix multiplication . If backpropagation is needed, then the output vectors and the intermediate arrays [m 1 , … , m N] , [l 1 , … , l N] {\displaystyle [m_{1},\dots ,m_{N}],[l_{1},\dots ,l_{N}]} are cached, and during the backward pass, attention matrices are rematerialized from these, making it a form of gradient checkpointing.

Mathematical properties

Geometrically the softmax function maps the Euclidean space R K {\displaystyle \mathbb {R} ^{K}} to the boundary of the standard (K − 1) {\displaystyle (K-1)} -simplex , cutting the dimension by one (the range is a (K − 1) {\displaystyle (K-1)} -dimensional simplex in K {\displaystyle K} -dimensional space), due to the linear constraint that all output sum to 1 meaning it lies on a hyperplane .

Along the main diagonal (x , x , … , x) , {\displaystyle (x,\,x,\,\dots ,\,x),} softmax is just the uniform distribution on outputs, (1 / n , … , 1 / n) {\displaystyle (1/n,\dots ,1/n)} : equal scores yield equal probabilities.

More generally, softmax is invariant under translation by the same value in each coordinate: adding c = (c , … , c) {\displaystyle \mathbf {c} =(c,\,\dots ,\,c)} to the inputs z {\displaystyle \mathbf {z} } yields σ (z + c) = σ (z) {\displaystyle \sigma (\mathbf {z} +\mathbf {c})=\sigma (\mathbf {z})} , because it multiplies each exponent by the same factor, e c {\displaystyle e^{c}} (because e z i + c = e z i · e c {\displaystyle e^{z_{i}+c}=e^{z_{i}}\cdot e^{c}}), so the ratios do not change: σ (z + c) j = e z j + c ∑ k = 1 K e z k + c = e z j · e c ∑ k = 1 K e z k · e c = σ (z) j . {\displaystyle \sigma (\mathbf {z} +\mathbf {c})_{j}={\frac {e^{z_{j}+c}}{\sum _{k=1}^{K}e^{z_{k}+c}}}={\frac {e^{z_{j}}\cdot e^{c}}{\sum _{k=1}^{K}e^{z_{k}}\cdot e^{c}}}=\sigma (\mathbf {z})_{j}.}

Geometrically, softmax is constant along diagonals: this is the dimension that is eliminated, and corresponds to the softmax output being independent of a translation in the input scores (a choice of 0 score). One can normalize input scores by assuming that the sum is zero (subtract the average: c {\displaystyle \mathbf {c} } where c = 1 n ∑ z i {\textstyle c={\frac {1}{n}}\sum z_{i}}), and then the softmax takes the hyperplane of points that sum to zero, ∑ z i = 0 {\textstyle \sum z_{i}=0} , to the open simplex of positive values that sum to 1 ∑ σ (z) i = 1 {\textstyle \sum \sigma (\mathbf {z})_{i}=1} , analogously to how the exponent takes 0 to 1, e 0 = 1 {\displaystyle e^{0}=1} and is positive.

By contrast, softmax is not invariant under scaling. For instance, σ ((0 , 1)) = (1 / (1 + e) , e / (1 + e)) {\displaystyle \sigma {\bigl (}(0,\,1){\bigr)}={\bigl (}1/(1+e),\,e/(1+e){\bigr)}} but σ ((0 , 2)) = (1 / (1 + e 2) , e 2 / (1 + e 2)) . {\displaystyle \sigma {\bigl (}(0,2){\bigr)}={\bigl (}1/\left(1+e^{2}\right),\,e^{2}/\left(1+e^{2}\right){\bigr)}.}

The standard logistic function is the special case for a 1-dimensional axis in 2-dimensional space, say the x -axis in the (x, y) plane. One variable is fixed at 0 (say z 2 = 0 {\displaystyle z_{2}=0}), so e 0 = 1 {\displaystyle e^{0}=1} , and the other variable can vary, denote it z 1 = x {\displaystyle z_{1}=x} , so e z 1 / ∑ k = 1 2 e z k = e x / (e x + 1) , {\textstyle e^{z_{1}}/\sum _{k=1}^{2}e^{z_{k}}=e^{x}/\left(e^{x}+1\right),} the standard logistic function, and e z 2 / ∑ k = 1 2 e z k = 1 / (e x + 1) , {\textstyle e^{z_{2}}/\sum _{k=1}^{2}e^{z_{k}}=1/\left(e^{x}+1\right),} its complement (meaning they add up to 1). The 1-dimensional input could alternatively be expressed as the line (x / 2 , − x / 2) {\displaystyle (x/2,\,-x/2)} , with outputs e x / 2 / (e x / 2 + e − x / 2) = e x / (e x + 1) {\displaystyle e^{x/2}/\left(e^{x/2}+e^{-x/2}\right)=e^{x}/\left(e^{x}+1\right)} and e − x / 2 / (e x / 2 + e − x / 2) = 1 / (e x + 1) . {\displaystyle e^{-x/2}/\left(e^{x/2}+e^{-x/2}\right)=1/\left(e^{x}+1\right).}

Gradients

The softmax function is also the gradient of the LogSumExp function: ∂ ∂ z i LSE ■ (z) = exp ■ z i ∑ j = 1 K exp ■ z j = σ (z) i , for i = 1 , … , K , z = (z 1 , … , z K) ∈ R K , {\displaystyle {\frac {\partial }{\partial z_{i}}}\operatorname {LSE} (\mathbf {z})={\frac {\exp z_{i}}{\sum _{j=1}^{K}\exp z_{j}}}=\sigma (\mathbf {z})_{i},\quad {\text{ for }}i=1,\dotsc ,K,\quad \mathbf {z} =(z_{1},\,\dotsc ,\,z_{K})\in \mathbb {R} ^{K},} where the LogSumExp function is defined as LSE ■ (z 1 , … , z n) = log ■ (exp ■ (z 1) + ■ + exp ■ (z n)) {\displaystyle \operatorname {LSE} (z_{1},\,\dots ,\,z_{n})=\log \left(\exp(z_{1})+\cdots +\exp(z_{n})\right)} .

The gradient of softmax is thus ∂ z j σ i = σ i (δ i j − σ j) {\displaystyle \partial _{z_{j}}\sigma _{i}=\sigma _{i}(\delta _{ij}-\sigma _{j})} .

History

The softmax function was used in statistical mechanics as the Boltzmann distribution in the foundational paper Boltzmann (1868) , [14] formalized and popularized in the influential textbook Gibbs (1902) . [15]

The use of the softmax in decision theory is credited to R. Duncan Luce , [16] : 1 who used the axiom of independence of irrelevant alternatives in rational choice theory to deduce the softmax in Luce's choice axiom for relative preferences. [citation needed]

In machine learning, the term "softmax" is credited to John S. Bridle in two 1989 conference papers, Bridle (1990a) : [16] : 1 and Bridle (1990b) : [3]

We are concerned with feed-forward non-linear networks (multi-layer perceptrons, or MLPs) with multiple outputs. We wish to treat the outputs of the network as probabilities of alternatives (e.g. pattern classes), conditioned on the inputs. We look for appropriate output non-linearities and for appropriate criteria for adaptation of the parameters of the network (e.g. weights). We explain two modifications: probability scoring, which is an alternative to squared error minimisation, and a normalised exponential (softmax) multi-input generalisation of the logistic non-linearity. [17] : 227

For any input, the outputs must all be positive and they must sum to unity. ...

Given a set of unconstrained values, ■ V j (x) {\displaystyle V_{j}(x)} ■ , we can ensure both conditions by using a Normalised Exponential transformation: Q j (x) = e V j (x) / ∑ k e V k (x) {\displaystyle Q_{j}(x)=\left.e^{V_{j}(x)}\right/\sum _{k}e^{V_{k}(x)}} This transformation can be considered a multi-input generalisation of the logistic, operating on the whole output layer. It preserves the rank order of its input values, and is a differentiable generalisation of the 'winner-take-all' operation of picking the maximum value. For this reason we like to refer to it as softmax . [18] : 213

Example

With an input of (1, 2, 3, 4, 1, 2, 3) , the softmax is approximately (0.024, 0.064, 0.175, 0.475, 0.024, 0.064, 0.175) . The output has most of its weight where the "4" was in the original input. This is what the function is normally used for: to highlight the largest values and suppress values which are significantly below the maximum value. But note: a change of temperature changes the output. When the temperature is multiplied by 10, the inputs are effectively (0.1, 0.2, 0.3, 0.4, 0.1, 0.2, 0.3) and the softmax is approximately (0.125, 0.138, 0.153, 0.169, 0.125, 0.138, 0.153) . This shows that high temperatures de-emphasize the maximum value.

Computation of this example using Python code:

Alternatives

The softmax function generates probability predictions densely distributed over its support . Other functions like sparsemax or α- entmax can be used when sparse probability predictions are desired. [19] Also the Gumbel-softmax reparametrization trick can be used when sampling from a discrete-discrete distribution needs to be mimicked in a differentiable manner.

See also

Softplus

Multinomial logistic regression

Dirichlet distribution – an alternative way to sample categorical distributions

Partition function

Exponential tilting – a generalization of Softmax to more general probability distributions

Notes

References

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