MLP Calculator

Training Parameter Recommendations

Multilayer Perceptron – Hyperparameter Guide

English

Wolfgang Boehme

wolfgang.boehme@economic-modeling.de

www.economic-modeling.de

25 October 2026

The documentation on this website was developed with the support of the AI Claude (Anthropic).

Table of Contents

1 Fundamental Concepts **2**

2 Currency Data (4 Inputs → 1 Output) **2**

3 Logic Gates: XOR, AND, OR **2**

4 Circle Classification (200 Samples) **3**

5 Simple Regression – Sine Curve (50 Samples) **3**

6 Further Typical Use Cases **4**
6.1 Iris Flower Classification (3 Classes) 4
6.2 Time Series Regression (e.g. Temperature, Stock Prices) 4
6.3 House Price Prediction (Multiple Features) 5
6.4 Spam Detection / Binary Text Classification 5

7 General Rules of Thumb **6**

Fundamental Concepts

Activation Functions at a Glance:

- **ReLU** (*Rectified Linear Unit*): $f(x) = \max(0, x)$ – fast, stable, standard for hidden layers in regression
- **Sigmoid**: $f(x) = \frac{1}{1+e^{-x}}$ – output in $[0, 1]$, suitable for binary classification
- **Tanh**: $f(x) = \tanh(x)$ – output in $[-1, 1]$, good for symmetric data and time series
- **Linear**: $f(x) = x$ – mandatory for continuous regression in the output layer

Loss Functions:

- **MSE** (*Mean Squared Error*): Standard for regression, penalises large deviations heavily
- **BCE** (*Binary Cross-Entropy*): Standard for binary classification (output with sigmoid)

Currency Data (4 Inputs → 1 Output)

Use case: USD/UZS, CNY/UZS, GBP/UZS, CHF/UZS as inputs – EUR/UZS as output. 251 samples, continuous values, narrow value ranges.

Data description: Daily exchange rates of four currencies against the Uzbek som (UZS) over approximately one year. The values lie within narrow bands (e.g. USD: 12 394–12 624, CNY: 1 716–1 749) and show strong mutual correlation, as exchange rates are driven by shared market mechanisms.

Goal of the MLP: The network should estimate the EUR/UZS rate from four known exchange rates – a nonlinear interpolation between correlated financial time series. Pure *regression problem*. Practical use: when the EUR rate is not directly available, it can be approximately reconstructed from the other rates.

Parameter	Recommendation
Architecture	4 → 8–16 → 8–16 → 1 (2 hidden layers)
Hidden activation	ReLU
Output activation	**Linear** (mandatory for continuous output)
Loss function	MSE
Learning rate	0.001 – 0.005
Epochs	2 000 – 5 000
Batch size	16 – 32
Normalisation	**always active** (values in range 12 000–16 000)
Train/test split	80 / 20

Note: The valid input range must be respected in manual prediction mode. The calculator displays this range automatically after training. Extrapolation outside the training data leads to unreliable results.

Logic Gates: XOR, AND, OR

Classic teaching examples for binary classification with 2 inputs and 4 samples.

Data description: Truth tables of Boolean logic. Two binary inputs (0 or 1) and one binary output. AND outputs 1 only when both inputs are 1; OR outputs 1 when at least one input is 1; XOR outputs 1 when exactly one input is 1. The dataset covers all four possible input combinations.

Goal of the MLP: The network should learn the logical function. XOR is particularly significant: it is *not linearly separable* and therefore cannot be solved by a network without a hidden layer. The example clearly demonstrates why multi-layer networks are necessary – a historically important result in AI research.

Parameter	Recommendation
Architecture	$2 \to 4\text{–}8 \to 1$
Hidden activation	Sigmoid
Output activation	Sigmoid
Loss function	BCE
Learning rate	0.1
Epochs	5 000 – 10 000 (XOR needs more)
Batch size	1 – 4 (all samples)
Normalisation	off (data already 0/1)

Note on XOR: XOR is not linearly separable – at least 1 hidden layer with ≥ 4 neurons is mandatory. AND and OR are linearly separable and could theoretically be solved without a hidden layer.

Circle Classification (200 Samples)

200 random points in $[-1, 1]^2$. *Output 1 if inside the circle with radius* $\sqrt{0.5}$, *otherwise 0.*

Data description: 200 randomly generated points in the square $[-1, 1]^2$. A point belongs to class 1 (inside) if $x^2 + y^2 < 0.5$, otherwise to class 0 (outside). The decision boundary is a circle – a typically nonlinear structure that linear classifiers cannot learn.

Goal of the MLP: The network should learn a nonlinear decision boundary in two-dimensional space. This is a *binary classification problem* and shows how MLPs can approximate arbitrarily shaped decision surfaces. Well suited for experimenting with network depth and neuron count.

Parameter	Recommendation
Architecture	$2 \to 8\text{–}16 \to 8 \to 1$
Hidden activation	ReLU
Output activation	Sigmoid
Loss function	BCE
Learning rate	0.01
Epochs	2 000 – 3 000
Batch size	16 – 32
Normalisation	off (data in $[-1, 1]$)
Train/test split	80 / 20

Simple Regression – Sine Curve (50 Samples)

$y = 0.5 + 0.4 \cdot \sin(6x) + \varepsilon$, *with* $x \in [0, 1]$ *and noise* ε.

Data description: 50 uniformly distributed points along a damped sine curve with superimposed random noise. The input x runs from 0 to 1, the output y varies between approx. 0.1 and 0.9. The curve contains several periods and tests the network's ability to capture periodic structures.

Goal of the MLP: The network should approximate a continuous nonlinear function from noisy samples – a classic *regression problem*. Illustrates the difference between underfitting and overfitting: too few neurons cannot represent the curve; too many memorise the noise.

Parameter	Recommendation
Architecture	$1 \rightarrow 8$–$16 \rightarrow 8 \rightarrow 1$
Hidden activation	Tanh (symmetric, good for sine function)
Output activation	Linear
Loss function	MSE
Learning rate	0.01
Epochs	3 000 – 5 000
Batch size	8 – 16
Normalisation	off (values in $[0, 1]$)

Further Typical Use Cases

Iris Flower Classification (3 Classes)

4 botanical features → 3 classes (Setosa, Versicolor, Virginica). 150 samples.

Data description: 150 iris flowers with 4 features each: sepal length/width and petal length/width (in cm). Three classes of 50 samples each. Setosa is well separable from the other two; Versicolor and Virginica partially overlap.

Goal of the MLP: Multi-class classification based on botanical features. As the calculator does not support a softmax output, each class is modelled as its own output neuron (one-hot). Classic benchmark dataset for comparing classification algorithms.

Parameter	Recommendation
Architecture	$4 \rightarrow 8$–$16 \rightarrow 3$
Hidden activation	ReLU
Output activation	Sigmoid per neuron (Softmax not available)
Loss function	BCE (per output neuron)
Learning rate	0.01
Epochs	2 000 – 4 000
Batch size	16 – 32
Normalisation	active

Time Series Regression (e.g. Temperature, Stock Prices)

Multiple time-lagged values as input → next value as output.

Data description: 366 daily temperatures with seasonally dependent behaviour (sinusoidal annual cycle ± noise). Inputs are the temperatures from the previous 3 days; output is the temperature of the following day. Values lie in the range approx. −3 to +23 °C.

Goal of the MLP: Prediction of the next value in a time series from past observations – so-called *autoregression*. Shows how an MLP can exploit short-term dependencies in time series without explicit memory. Transferable to stock prices, energy consumption or sensor data.

Parameter	Recommendation
Architecture	$n \to 16$–$32 \to 16 \to 1$
Hidden activation	Tanh or ReLU
Output activation	Linear
Loss function	MSE
Learning rate	0.001 – 0.005
Epochs	3 000 – 8 000
Batch size	16 – 64
Normalisation	**always active**

House Price Prediction (Multiple Features)

Area, rooms, location, year of construction, etc. → price. 500 samples, 4 inputs.

Data description: 500 synthetic property records with features: floor area (40–200 m^2), number of rooms (1–6), year of construction (1950–2023) and distance to city centre (1–30 km). The price was generated as a weighted linear combination of these factors with noise (realistic: 50 000–650 000 EUR).

Goal of the MLP: Estimation of a continuous property value from multiple heterogeneous features – a typical *regression problem* with practical relevance. Demonstrates handling of differently scaled inputs (hence normalisation is mandatory) and the importance of train/test separation for assessing generalisation.

Parameter	Recommendation
Architecture	$n \to 32$–$64 \to 16$–$32 \to 1$
Hidden activation	ReLU
Output activation	Linear
Loss function	MSE
Learning rate	0.001
Epochs	3 000 – 10 000
Batch size	32 – 64
Normalisation	**always active**
Train/test split	80 / 20

Spam Detection / Binary Text Classification

Feature vectors from text (e.g. word frequencies) → spam (1) / not spam (0). 400 samples, 5 inputs.

Data description: 400 synthetic email representations with 5 normalised features: frequency of the words *free*, *win*, *click*, number of links and frequency of exclamation marks. Spam emails (200 samples) show clearly higher values in all features than normal emails (200 samples). Balanced class distribution (50/50).

Goal of the MLP: Binary classification for automatic spam detection – a *classification problem* with direct practical value. Shows how an MLP can learn a reliable decision rule from simple text features. Transferable to other binary classification tasks such as fraud detection or medical diagnosis.

Parameter	Recommendation
Architecture	$n \to 32$–$64 \to 16 \to 1$
Hidden activation	ReLU
Output activation	Sigmoid
Loss function	BCE
Learning rate	0.001 – 0.01
Epochs	2 000 – 5 000
Batch size	32 – 64
Normalisation	active (normalise TF-IDF values)
Train/test split	80 / 20

General Rules of Thumb

Situation	Recommendation
Loss diverges (NaN)	Halve the learning rate, check normalisation
Loss stagnates early	Slightly increase learning rate, add more neurons
Loss falls but test loss rises	Overfitting – reduce neurons or obtain more data
Output always near 0 or 1	Wrong output activation (Linear instead of Sigmoid?)
Very slow convergence	Check Xavier initialisation (already active in calculator)
Small dataset (<100)	Small batch size (4–8), more epochs

Xavier Initialisation (already implemented in Calculator v19): The starting weights are scaled to $\pm\sqrt{1/n_{\text{inputs}}}$. This prevents exploding or vanishing gradients at the start of training and ensures stable, reproducible convergence – independent of randomness.