

Machine Intelligence 1 Tutorial

03 Multilayer Perceptrons and the Backpropagation algorithm

Youssef Kashef

Fachgebiet Neuronale Informationsverarbeitung (NI)

WS 19/20

Components of Training Neural Networks

- MLPs are universal function approximators
- Ingredients for function fitting
- Cost functions
 - Cost functions for Regression

MLPs are universal function approximators

MLPs are universal function approximators. This means that, provided some assumptions are satisfied, they are capable of finding a function with which to map observations \underline{x} to a their corresponding label \underline{y}_T .

The universal approximation theorem by Funahashi (1989)^a

^aFunahashi (1989) On the approximate realization of continuous mappings by neural networks. Neur Netw, 2:183–192

Hornik et al. (1989) Multilayer Feedforward Networks are Universal Approximators. Neur Netw, 2:359–366.

Let $y_T(\underline{x})$ be a continuous, real valued function over a compact interval K and

$$y(\underline{x}; \underline{w}) = \sum_{i=1}^M w_i^{21} f\left(\sum_{j=1}^N w_{ij}^{10} x_j - \theta_i\right)$$

be a three-layered MLP with a non-constant, bounded, monotonously increasing and continuous function $f : \mathbb{R} \rightarrow \mathbb{R}$.

MLPs are universal function approximators

MLPs are universal function approximators. This means that, provided some assumptions are satisfied, they are capable of finding a function with which to map observations \underline{x} to a their corresponding label \underline{y}_T .

The universal approximation theorem by Funahashi (1989)^a

^aFunahashi (1989) On the approximate realization of continuous mappings by neural networks. Neur Netw, 2:183–192

Hornik et al. (1989) Multilayer Feedforward Networks are Universal Approximators. Neur Netw, 2:359–366.

Let $y_T(\underline{x})$ be a continuous, real valued function over a compact interval K and

$$y(\underline{x}; \underline{w}) = \sum_{i=1}^M w_i^{21} f\left(\sum_{j=1}^N w_{ij}^{10} x_j - \theta_i\right)$$

be a three-layered MLP with a non-constant, bounded, monotonously increasing and continuous function $f : \mathbb{R} \rightarrow \mathbb{R}$.

Ingredients for function fitting

Fit an MLP to desired function $y_T(\underline{\mathbf{x}})$ requires the following ingredients:

- ① A cost function with the objective to minimize it: $e(y_T, \underline{\mathbf{x}})$.
- ② A performance measure. Specifically, the generalization error E^G which is defined as:

$$E^G := \langle e \rangle_{y_T, \underline{\mathbf{x}}} = \iint d\underline{\mathbf{x}} dy_T P_{(y_T, \underline{\mathbf{x}})} e_{(y_T, \underline{\mathbf{x}})}$$

- ③ A parameterized model: MLP, connectionist neuron, ...
- ④ An optimization method or learning algorithm for finding the set of parameters in our model that will minimize the cost function. This can be done analytically (depending on the conditions) or through an iterative learning algorithm (e.g. gradient-based learning)

Cost functions

A cost function $e(y_T, \underline{\mathbf{x}})$ (or more explicitly $e(y_T, y(\underline{\mathbf{x}}; \underline{\mathbf{w}}))$) quantifies the discrepancy between the model's prediction for a specific observation and the label it is assigned to.

Selecting a cost function accounts for

- ① the type of problem (i.e. regression vs. classification) and
- ② how the model is penalized for different types of mistakes it can make such as: tolerance for certain error range (e.g. small errors are tolerable), large errors are penalized less.

The choice of cost function has a direct effect on how the model will learn. For example, we will later see in gradient-based learning how the error function can modulate how fast or slow a model learns.

Cost functions

$$\underbrace{\underline{\mathbf{x}} \in \mathbb{R}^N}_{\text{feature vector}} \longrightarrow \underbrace{y \in \mathbb{R}}_{\text{attribute}}$$

y_T : true value of attribute

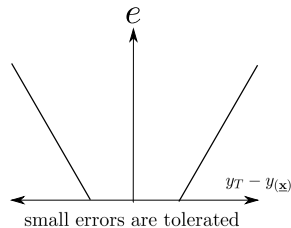
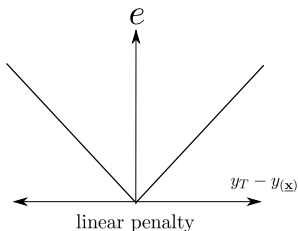
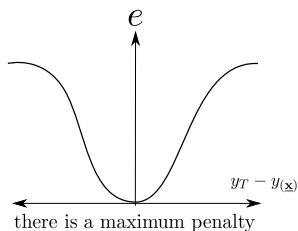
$y(\underline{\mathbf{x}})$: predicted value of attribute (e.g. by MLP)

Cost functions

$$\underbrace{\underline{\mathbf{x}} \in \mathbb{R}^N}_{\text{feature vector}} \longrightarrow \underbrace{y \in \mathbb{R}}_{\text{attribute}}$$

y_T : true value of attribute
 $y(\underline{\mathbf{x}})$: predicted value of attribute (e.g. by MLP)

individual cost $e(y_T, \underline{\mathbf{x}})$



several choices \Rightarrow predictor will depend on error measure!