Multilayer Perceptrons and the Backpropagation algorithm

Contents

1 Components of Training Neural Networks **2**
1.1 MLPs are universal function approximators 2
1.2 Ingredients for function fitting 2
1.3 Cost functions 2
1.3.1 Cost functions for Regression 3

1 Components of Training Neural Networks

1.1 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 $\underline{\mathbf{x}}$ to a their corresponding label $\underline{\mathbf{y}}_T$.

The universal approximation theorem by Funahashi (1989)[1]

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

$$y(\underline{\mathbf{x}}; \underline{\mathbf{w}}) = \sum_{i=1}^{M} \mathrm{w}_i^{21} f\Big(\sum_{j=1}^{N} \mathrm{w}_{ij}^{10} \mathrm{x}_j - \theta_i \Big) \tag{1.1}$$

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

Then there exists a set of parameters $M, N \in \mathbb{N}$ and $\mathrm{w}_i^{21}, \mathrm{w}_{ij}^{10}, \theta_i \in \mathbb{R}$ such that for every $\varepsilon > 0$:

$$\max_{\underline{\mathbf{x}} \in K} \Big| \, y(\underline{\mathbf{x}}; \underline{\mathbf{w}}) - y_T(\underline{\mathbf{x}}) \, \Big| \leq \varepsilon \tag{1.2}$$

1.2 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:
   $$E^G \quad := \quad \langle \, e \, \rangle_{y_T, \underline{\mathbf{x}}} \quad = \quad \iint d\underline{\mathbf{x}} \, dy_T \; P_{(y_T, \underline{\mathbf{x}})} \, e_{(y_T, \underline{\mathbf{x}})} \tag{1.3}$$
3. A parameterized model: MLP, connecitonist 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)

1.3 Cost functions

A cost function $e(y_T, \underline{\mathbf{x}})$ (or more explicitly $e(y_T, y(\underline{\mathbf{x}}; \underline{\mathbf{w}}))$ quanifties 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.

[1] Funahashi (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.

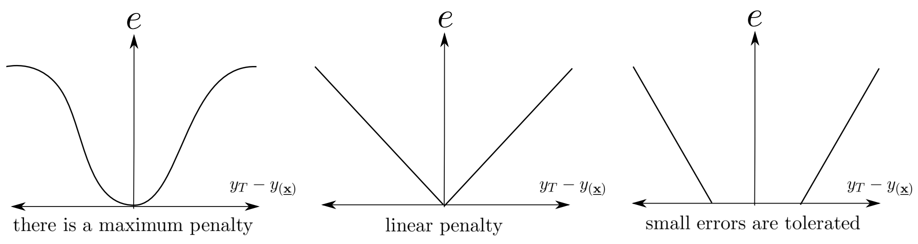
The choice of cost function has a direct effect on how the model will learn. For example, we will later see in gradient-based learning how the error function can modulate how fast or slow a model learns.

1.3.1 Cost functions for Regression

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

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!