Manual: 7.2.2. Feed-Forward Networks

Generally, most neural network methods assume that the data points used to train it are independent measurements. This is a pivotal point in modeling and bears some discussion.

Suppose we have a collection of digital images and we classify them into two groups: Those showing human faces and those showing something else. Neural networks can learn the classification into these two groups if the collection is large enough. The images are unrelated to each other; there is no cause-effect relationship between any two images — at least not one relevant to the task of learning to differentiate a human face from other images.

Suppose, however, that we are classifying winter versus summer images of nature and that our images were of the same location and arranged chronologically with relatively high cadence. Now the images are not independent but rather they have a cause-effect relationship ordered in time. This implies that the function *f(...)* that we were looking for is really quite different. Its accuracy would be a lot better if it did not assess each image on its own merits but rather had a memory of the last few images because often a few images of summer, we are likely to get another image of summer. In this version, we see a dependence upon history that implies a time-oriented memory of the system over a certain time scale that must somehow be determined.

As a consequence of this, we have network models that work well for datasets with independent data points and others that work well for datasets in which the data points are time-dependent. The networks that deal with independent points are called feed-forward networks and form the historical beginning of the field of neural networks as well as the principal methods being used. The networks that deal with time-dependent points are called recurrent networks, which are newer and more complex to apply.

The most popular neural network is called the multi-layer perceptron and takes the form $y = a_N(W_N \cdot a_{N-1}(W_{N-1} \cdot \ldots\ a_1\ (W_1 \cdot x + b_1)\ \ldots + b_{N-1}) + b_N)$ where N is the number of layers, the *W_i* are weight matrices, the *b_i* are bias vectors, the *tanh(...)* are activation functions and x are the inputs as before.

The weight matrices and the bias vectors are the place-holders for the model's parameters. The so-called topology of the network refers to the freedom that we have in choosing the size of the matrices and vectors, and also the number of layers N. The only restriction that we have is the problem-inherent size of the input and output vectors. Once the topology is chosen, then the model has a specific number of parameters that reside in these matrices and vectors.

In training such a network, we must first choose the topology of the network and the nature of the activation functions. After that we must determine the value of the parameters inside the weight matrices and bias vectors. The first step is a matter of human choice and involves considerable experience. After decades of research into this topic, the initial topological choices of a neural network are still effectively a black art. There are many results to guide the practitioners of this art in their rituals but these go far beyond the scope of this book. The second step can be accomplished by standard training algorithms that we mentioned before and also will not treat here.

A single-layer perceptron is thus $y = \tanh(Wx + b)$ and was one of the first neural networks to be investigated. The single-layer perceptron can represent only linearly separable patterns. It is possible to prove that a two-layer perceptron can approximate virtually any function of interest to any degree of accuracy if only the weight matrices and bias vectors in each layer are chosen large enough.