Artificial Neural Networks/Error-Correction Learning

Contents

Error-Correction Learning

Error-Correction Learning, used with supervised learning, is the technique of comparing the system output to the desired output value, and using that error to direct the training. In the most direct route, the error values can be used to directly adjust the tap weights, using an algorithm such as the backpropagation algorithm. If the system output is *y*, and the desired system output is known to be *d*, the error signal can be defined as:

$$e = d - y$$

Error correction learning algorithms attempt to minimize this error signal at each training iteration. The most popular learning algorithm for use with error-correction learning is the backpropagation algorithm, discussed below.

Gradient Descent

The gradient descent algorithm is not specifically an ANN learning algorithm. It has a large variety of uses in various fields of science, engineering, and mathematics. However, we need to discuss the gradient descent algorithm in order to fully understand the backpropagation algorithm. The gradient descent algorithm is used to minimize an error function *g(y)*, through the manipulation of a weight vector *w*. The cost function should be a linear combination of the weight vector and an input vector *x*. The algorithm is:

$$w_{ij}[n+1] = w_{ij}[n] + \eta g(w_{ij}[n])$$

Here, η is known as the step-size parameter, and affects the rate of convergence of the algorithm. If the step size is too small, the algorithm will take a long time to converge. If the step size is too large the algorithm might oscillate or diverge.

The gradient descent algorithm works by taking the gradient of the weight space to find the path of steepest descent. By following the path of steepest descent at each iteration, we will either find a minimum, or the algorithm could diverge if the weight space is infinitely decreasing. When a minimum is found, there is no guarantee that it is a global minimum, however.

Backpropagation

The **backpropagation** algorithm, in combination with a supervised error-correction learning rule, is one of the most popular and robust tools in the training of artificial neural networks. Back propagation passes error signals backwards through the network during training to update the weights of the network. Because of this dependence on bidirectional data flow during training, backpropagation is not a plausible reproduction of biological learning mechanisms. When talking about backpropagation, it is useful to define the term **interlayer** to be a layer of neurons, and the corresponding input tap weights to that layer. We use a superscript to denote a specific interlayer, and a subscript to denote the specific neuron from within that layer. For instance:

$$\zeta_j^l = \sum_{i=1}^{N^{l-1}} w_{ij}^l x_i^{l-1} \quad (1)$$
$$x_j^l = \sigma(\zeta_j^l) \quad (2)$$

Where x_i^{l-1} are the outputs from the previous interlayer (the inputs to the current interlayer), w_{ij}^l is the tap weight from the *i* input from the previous interlayer to the *j* element of the current interlayer. N^{l-1} is the total number of neurons in the previous interlayer.

The backpropagation algorithm specifies that the tap weights of the network are updated iteratively during training to approach the minimum of the error function. This is done through the following equation:

$$w_{ij}^l[n] = w_{ij}^l[n-1] + \delta w_{ij}^l[n] \quad (3)$$
$$w_{ij}^{l-1}[n] = \eta \delta_j^l x_i^{l-1}[n] + \mu \Delta w_{ij}^l[n-1] \quad (3)$$

The relationship between this algorithm and the gradient descent algorithm should be immediately apparent. Here, η is known as the learning rate, not the step-size, because it affects the speed at which the system learns (converges). The parameter μ is known as the momentum parameter. The momentum parameter forces the search to take into account its movement from the previous iteration. By doing so, the system will tend to avoid local minima or saddle points, and approach the global minimum. We will discuss these terms in greater detail in the next section.

The parameter δ is what makes this algorithm a "back propagation" algorithm. We calculate it as follows:

$$\delta_j^l = \frac{dx_j^l}{dt} \sum_{k=1}^{r} \delta_k^{l+1} w_{kj}^{l+1} \quad (4)$$

The δ function for each layer depends on the δ from the previous layer. For the special case of the output layer (the highest layer), we use this equation instead:

$$\delta_j^l = \frac{dx_j^l}{dt}(x_j^l - y_j) \quad (5)$$

In this way, the signals propagate backwards through the system from the output layer to the input layer. This is why the algorithm is called the backpropagation algorithm.

Log-Sigmoid Backpropagation

If we use log-sigmoid activation functions for our neurons, the derivatives simplify, and our backpropagation algorithm becomes:

$$\delta_j^l = x_j^l(1 - x_j^l)(x_j^l - y_j)$$

For the output layer, and

$$\delta_j^l = x_j^l(1 - x_j^l) \sum_{k=1}^{r} \delta_k^{l+1} w_{kj}^{l+1}$$

for all the hidden inner layers. This property makes the sigmoid function desirable for systems with a limited ability to calculate derivatives.

Learning Rate

The **learning rate** is a common parameter in many of the learning algorithms, and affects the speed at which the ANN arrives at the minimum solution. In backpropagation, the learning rate is analogous to the step-size parameter from the gradient-descent algorithm. If the step-size is too high, the system will either oscillate about the true solution, or it will diverge completely. If the step-size is too low, the system will take a long time to converge on the final solution.

Momentum Parameter

The **momentum parameter** is used to prevent the system from converging to a local minimum or saddle point. A high momentum parameter can also help to increase the speed of convergence of the system. However, setting the momentum parameter too high can create a risk of overshooting the minimum, which can cause the system to become unstable. A momentum coefficient that is too low cannot reliably avoid local minima, and also can slow the training of the system.