The Cascade-Correlation Learning Architecture – summary

Matěj Nikl

August 6, 2016

The Cascade-Correlation Architecture (CCA) is a architecture as well as a supervised learning algorithm for artificial neural networks (ANNs). Conventional way of training ANNs is to *somehow* choose a fixed network topology and then train it using back-propagation algorithm. CCA eliminates the need of choosing a fixed network topology and provides a way of training this kind of a dynamic network.

1 Principles of growing

The network starts with no hidden units, only output units are present. Then, one hidden unit is added at a time, receiving all network inputs and all outputs of the previously added hidden units, each time adding more complexity to the network, possibly creating more and more complex feature detectors.

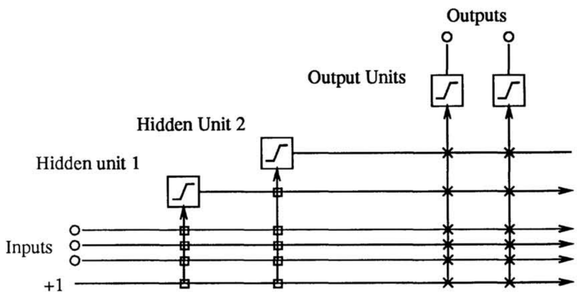

Figure 1: The CCA after two hidden units have been added. The vertical lines sum all incoming activations. Boxed connections are frozen, X connections are trained repeatedly.

2 Training

The process of training can be divided into two parts, which will be described in the following subsections, however one constraint holds through both of them: once a new hidden unit is added, its input weights are forever frozen.

2.1 Network output training

Having everything except output units' weights frozen translates into optimization of a single layer network, which was one of the authors' goals (to avoid slow back-propagation). The following steps take place:

1. optimize the single layer network, until a convergence is achieved
2. evaluate the test error and if it is low enough, stop
3. otherwise, proceed to subsection 2.2

2.2 New unit training

If network's test error is not low enough, we need to add a new hidden unit. To do so, the following steps take place:

1. create a pool of *candidate* units, having their input weights randomly initialized, each receiving all available inputs
2. maximize S, the sum of covariances between the candidate unit's value and the residual output error of all output units, for each candidate unit, by adjusting their input weights, until convergence

$$S = \sum_{o \in O} \left| \sum_{p \in P} (V_p - \overline{V})(E_{p,o} - \overline{E_o}) \right|$$

   - O is a set of output units
   - P is a set of training patterns (training data)
   - V_p is candidate unit's value for pattern p
   - $E_{p,o}$ is the residual output error at output unit o for pattern p
   - averages are computed across P

3. choose the candidate unit with the largest S and add it to the network, proceed to 2.1

3 Principles of modularization

The only sense of modularization in the CCA can be seen in viewing individual hidden units as modules, since they are being added throughout the learning process. However, the network as a whole is not modular in the full sense. Hidden units cannot be detached from a network and reattached to a different one. A hidden unit can only be used in conjunction with the hidden units it has been trained with and nowhere else.

Similar design A similar partially modular design can be seen in Progressive Neural Networks, where new columns (full-blown NNs) are being added with lateral connections to all of the previously trained columns, showing the same aspects of *(non)modularity* as CCAs.