Committee Machines

Introduction to Neural Networks : Lecture 15

1. What is a Committee Machine?
2. Types of Committee Machine
3. Ensemble Averaging
4. Boosting
5. Mixtures of Experts

What is a Committee Machine?

We have seen that it is standard practice with neural networks to train many different candidate networks, and then to select and keep the best (for example, on the basis of performance on an independent validation set) while discarding the rest.

There are two obvious disadvantages to this approach

1. All the effort involved in training the discarded networks is wasted.
2. The validation set has a random component, so the network that has the best performance on it will not necessarily have the best performance on the test set.

These drawbacks can be overcome by ***combining*** the networks together to form a ***committee machine***.

The importance of this approach is that it can lead to significant improvements in the performance on new data, with little extra computational effort. In fact, the committee can often do better than the best single constituent network in isolation.

Types of Committee Machine

Committee machines can be conveniently classified into two major categories:

1. Static Structures

The outputs of several constituent (expert) networks are combined by a mechanism that does not involve the input signal, hence the designation ***static***. Examples include

- ***Ensemble averaging***, where the constituent outputs are linearly combined.
- ***Boosting***, where weak learners are combined to give a strong learner.

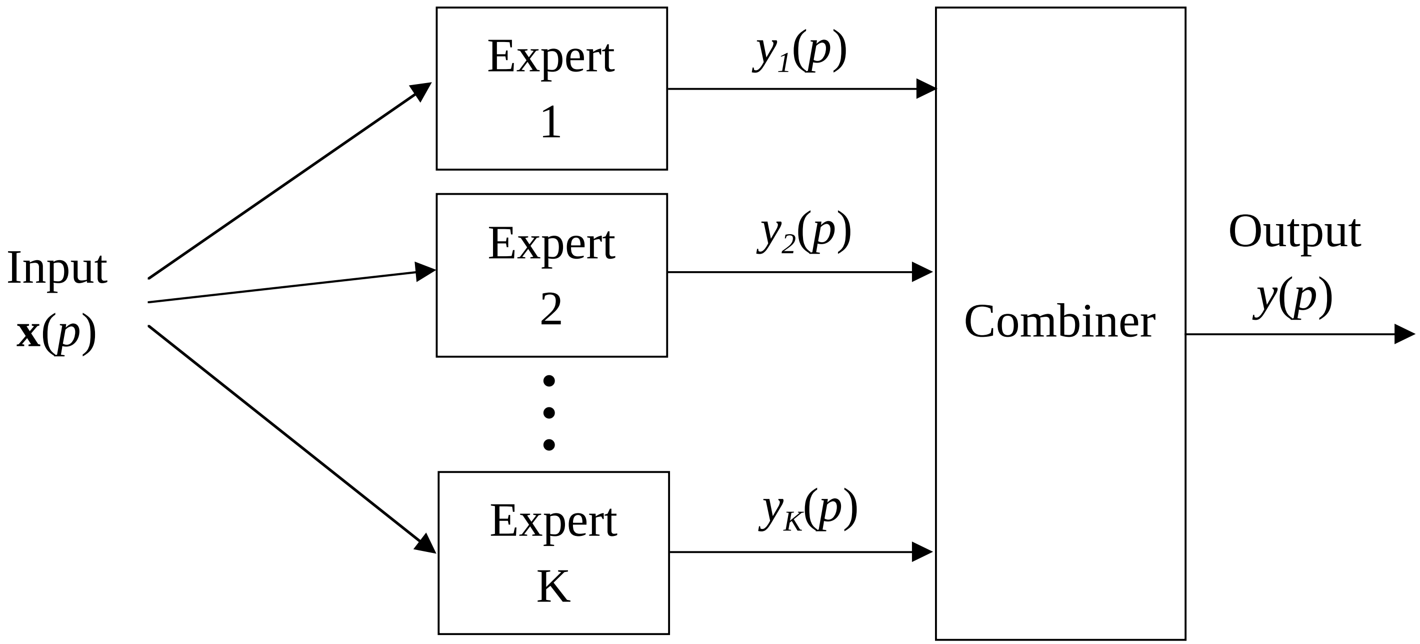
2. Dynamic Structures

The input signal is directly involved in actuating the mechanism that integrates/ combines the constituent outputs, hence the designation ***dynamic***. The main example is

- ***Mixtures of experts***, where the constituent outputs are non-linearly combined by some form of gating system (which may itself be a neural network).

Ensemble Averaging

An ***ensemble average*** consists of a set of trained networks (i.e. experts) which share a common input $\mathbf{x}(p)$ for training pattern p, and whose individual outputs $y_i(p)$ are somehow combined to produce an overall output $y(p)$:



Usually, an output combination by simple linear averaging or voting proves sufficient.

Theory Behind Ensemble Averaging

The motivation behind ensemble averaging is three-fold:

1. If the combination of experts were replaced by a single large network, it would be difficult to train in parallel and hence take longer to train.
2. The corresponding single network would have many times the number of free parameters, and thus over-fitting is more likely to occur.
3. We can expect differently trained networks to converge to different local minima (or different patterns of over-fitting) and so the average performance is likely to be better than that of any individual.

There are many possible ways to make the individuals different, and many possible ways to combine their outputs.

Simply starting identical networks from ***different initial weights*** and averaging their outputs gives good results. A full analysis shows that the ensemble average has the ***same bias*** as the individual networks, but the ***variance is reduced***. Overall, this results in better generalization by the ensemble.

Boosting

Boosting is quite different to ensemble averaging. In a ***boosting machine*** the experts are trained on data sets with entirely different distributions. It is a general method that can be used to improve the performance of ***any*** learning algorithm.

The idea is that a weak learning algorithm, which performs only slightly better than guessing, can be ***boosted*** into a strong learning algorithm by the operation of a committee machine.

In practice, we train a series of networks, each one of which concentrates more strongly on the patterns learned incorrectly by the previous networks. The final output can then be obtained by averaging the constituent outputs.

The boosting algorithm can be applied recursively and the error rate made arbitrarily small. There is still some debate about how well such systems can generalize. Often with noisy data, any outliers can be given undue influence, and consequently the performance of this approach will suffer.

Implementing Boosting

Boosting can be implemented in (at least) three fundamentally different ways:

1. Boosting by filtering

The training patterns are filtered by different versions of the weak learning algorithm. We end up with three networks trained on three equal sized sub-sets of the large (in theory, infinite) full training set. The three sub-sets have different statistics.

2. Boosting by sub-sampling

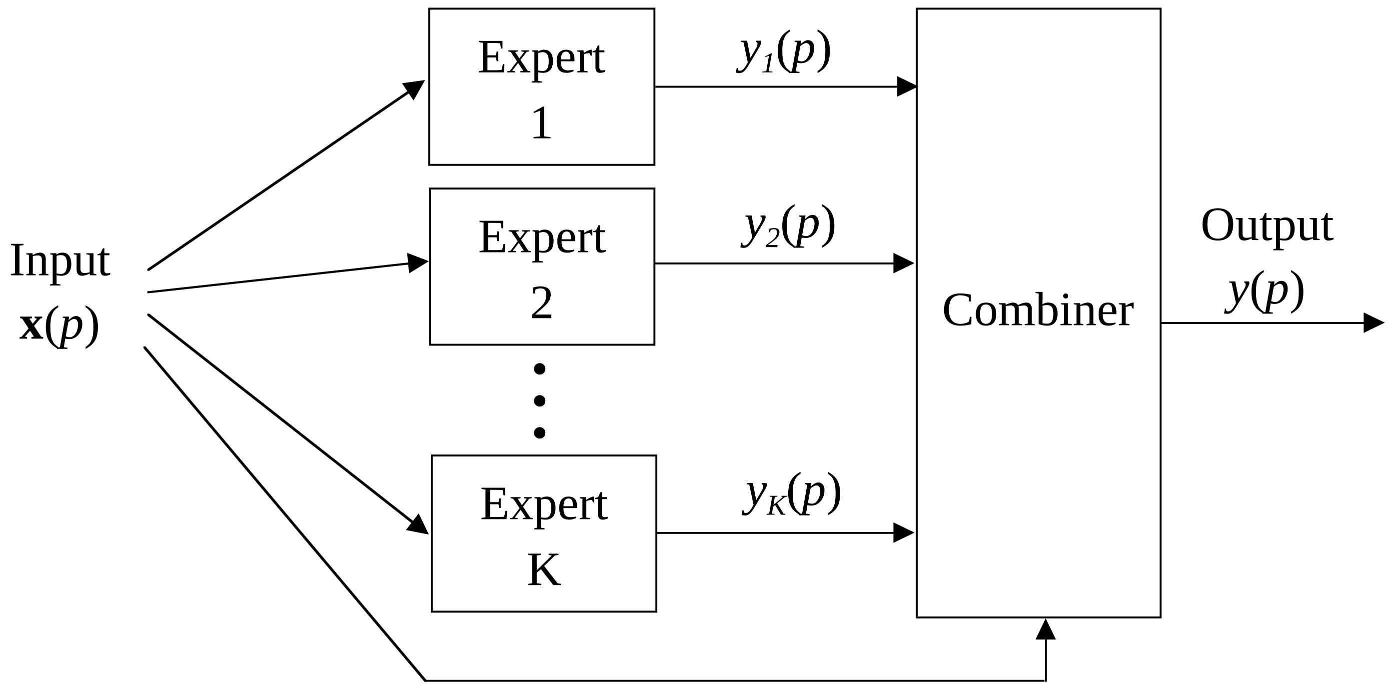
The training set is of fixed size and patterns are re-sampled according to a given probability distribution during training. In effect, the patterns learnt incorrectly get sampled more often in the next in a series of networks. The error is calculated with respect to the fixed training set. ***AdaBoost*** is a well known example of this approach.

3. Boosting by re-weighing

The training set is fixed, but the weak learning algorithm can receive “weighted” patterns. The error is calculated with respect to the weighted examples.

Mixtures of Experts

With ***mixtures of experts***, the principle of ***divide and conquer*** distributes the given learning task between a set of expert networks, and combines the constituent outputs to produce an overall output that is superior to that of any single network acting on its own.



Here the inputs $\mathbf{x}(p)$ influence (or gate) the combination of the constituent outputs $y_i(p)$.

Combining the Experts

Unlike with static committee machines, in a mixture of experts, each expert really can concentrate on one part of the problem and ignore the rest. The constituent outputs can be combined more intelligently so that only the right (i.e. best) experts contribute to the output for any given input pattern.

In the simple ensemble averaging approach we just take the average of the outputs

$$y(p) = \sum_{i=1}^{K} \frac{1}{K} y_i(p)$$

whereas in the mixtures of experts approach the outputs are gated according to the inputs

$$y(p) = \sum_{i=1}^{K} g_i(\mathbf{x}(p)) y_i(p)$$

In this way the system can become truly ***modular*** with separate modules dealing with different types of inputs, or even different tasks.

Generating the Gates

In principle, the gates $g_i(\mathbf{x}(p))$ can be generated in any convenient manner. In practice, it is usual to have them produced as the output of a ***gating network***, a simple single layer network with one output for each expert. The weights a_{ji} for that network can then be learnt at the same time as the weights in the expert networks. It is convenient for the gates themselves to be produced with a ***softmax*** activation function

$$g_i(p) = \exp\left(\sum_j a_{ij} x_j(p)\right) \Bigg/ \sum_{l=1}^{K} \exp\left(\sum_j a_{lj} x_j(p)\right)$$

because this automatically gives the gates two useful properties:

$$0 \le g_i(p) \le 1 \qquad\qquad \sum_{i=1}^{K} g_i(p) = 1$$

We can then talk about weighted averages of the experts, and a gate of zero indicates the expert has no influence on the overall output, while a gate of one means total control.

Training a Mixture of Experts

Once we have all the gates and expert networks defined mathematically, we can easily define an output error function as with any other network. For example

$$E(w_{jkl}, a_{ij}) = \tfrac{1}{2}\sum_p \big(t(p) - y(p)\big)^2 = \tfrac{1}{2}\sum_p \left(t(p) - \sum_j g_j(p) y_j(p) \right)^2$$

in which we have training patterns p, normal neural network weights w_{jkl} for each expert network j, weights a_{ij} for the gating network, and overall output target $t(p)$. We can then use standard gradient descent weight updates to minimise the output error function

$$\Delta w_{mnp} = -\eta \frac{\partial E(w_{jkl}, a_{ij})}{\partial w_{mnp}} \quad , \quad \Delta a_{mn} = -\eta \frac{\partial E(w_{jkl}, a_{ij})}{\partial a_{mn}}$$

Note that we do not need targets for the gates. In the literature you will find numerous variations on this theme: various different error functions, ***Gaussian Mixture Models***, training by the ***Expectation Maximisation*** (EM) algorithm, and so on.

Overview and Reading

1. We began with the basic idea of a committee machine, and the distinction between static and dynamic structures for those machines.
2. We then looked at two types of static committee approaches: ensemble averaging and boosting. In each case the outputs from the expert constituent networks are combined with no reference to the inputs.
3. We ended by looking at a dynamic approach: mixtures of experts. Here the outputs of the experts are combined by gates which learn appropriate dependencies on the inputs. In this approach, the committee can learn to take a fully modular approach to its tasks.

Reading

1. Haykin: Sections 7.1, 7.2, 7.3, 7.4, 7.5, 7.6
2. Bishop: Sections 9.6, 9.7