**Interpolative Associative Memory**

Sometimes it is possible to obtain a training set with orthonormal (that is, normalized and pairwise orthogonal) input vectors.

In that case, our two-layer network with linear neurons can solve its task perfectly and does not even require training.

We call such a network an **interpolative associative memory**.

You may ask: How does it work?

**Interpolative Associative Memory**

With an orthonormal set of exemplar input vectors (and any associated output vectors) we can simply calculate a weight matrix that realizes the desired function and does not need any training procedure.

For exemplars \( (x_1, y_1), (x_2, y_2), \ldots, (x_p, y_p) \) we obtain the following weight matrix \( W \):

\[
W = \sum_{p=1}^{P} y_p x_p^T
\]

Note that an \( N \)-dimensional vector space cannot have a set of more than \( N \) orthonormal vectors!
The network structure looks as follows:

\[
W = \begin{bmatrix}
3 & 0 & -1 & 0 \\
0 & 1 & 0 & 0 \\
-3 & 0 & 1 & 0 \\
-3 & 0 & 0 & 1
\end{bmatrix} + \begin{bmatrix}
1 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 \\
0 & 3 & 0 & 0 \\
0 & 0 & 3 & 0
\end{bmatrix} + \begin{bmatrix}
5 & 0 & 0 & 0 \\
0 & 2 & 0 & 0 \\
0 & 0 & 2 & 0 \\
0 & 0 & 0 & 2
\end{bmatrix} = \begin{bmatrix}
3 & -3 & 5 \\
-3 & 2 & 2 \\
3 & 3 & 8 \\
-3 & 3 & 8
\end{bmatrix}
\]

If you set the weights \(w_{nn}\) to these values, the network will realize the desired function.

This is identical to the following formula:

\[
W_{ij} = \begin{cases}
0 & \text{if } i = j \\
\text{otherwise}
\end{cases}
\]

The Hopfield model is a single-layered recurrent network. Like the associative memory, it is usually initialized with appropriate weights instead of being trained. The network structure looks as follows:

\[X_n \xrightarrow{\text{Hopfield}} X_{n+1} \]

For input-output pairs \((x_1, y_1), (x_2, y_2), \ldots, (x_p, y_p)\), we can initialize the weights in the same way as we did it with the associative memory:

\[
W = \sum_{p=1}^{P} y_p x_p'
\]

This is identical to the following formula:

\[
W_{ij} = \sum_{p=1}^{P} y_p^{(i)} x_p^{(j)}
\]

where \(x_p^{(j)}\) is the \(j\)-th component of vector \(x_p\), and \(y_p^{(i)}\) is the \(i\)-th component of vector \(y_p\).

So if you want to implement a linear function \(R^N \rightarrow R^M\) and can provide exemplars with orthonormal input vectors, then an interpolative associative memory is the best solution. It does not require any training procedure, realizes perfect matching of the exemplars, and performs plausible interpolation for new input vectors. Of course, this interpolation is linear.

The Hopfield model, because its mathematical description is more straightforward.

In the discrete model, the output of each neuron is either 1 or -1.

In its simplest form, the output function is the sign function, which yields 1 for arguments \(\geq 0\) and -1 otherwise.

In the discrete version of the model, each component of an input or output vector can only assume the values 1 or -1.

The output of a neuron \(i\) at time \(t\) is then computed according to the following formula:

\[
o_i(t) = \text{sgn}\left(\sum_{j=1}^{N} w_{ij} o_j(t-1)\right)
\]

This recursion can be performed over and over again. In some network variants, external input is added to the internal, recurrent one.
The Hopfield Network

Usually, the vectors $x_p$ are **not orthonormal**, so it is not guaranteed that whenever we input some pattern $x_p$, the output will be $y_p$, but it will be a pattern **similar** to $y_p$.

Since the Hopfield network is recurrent, its behavior depends on its previous state and in the general case is difficult to predict.

However, what happens if we initialize the weights with a set of patterns so that each pattern is being associated with itself, $(x_1, x_1), (x_2, x_2), \ldots, (x_p, x_p)$?

This initialization is performed according to the following equation:

$$W_p = \sum_{p=1}^{P} x_p^{(i)} x_p^{(j)}$$

You see that the weight matrix is symmetrical, i.e., $w_{ij} = w_{ji}$.

We also demand that $w_{ii} = 0$, in which case the network shows an interesting behavior.

It can be mathematically proven that under these conditions the network will reach a **stable activation state** within a finite number of iterations.

And what does such a stable state look like?

The network associates input patterns with themselves, which means that in each iteration, the activation pattern will be drawn towards one of those patterns.

After converging, the network will most likely present one of the patterns that it was initialized with.

Therefore, Hopfield networks can be used to **restore** incomplete or noisy input patterns.

After providing only one fourth of the “face” image as initial input, the network is able to perfectly reconstruct that image within only two iterations.

Adding noise by changing each pixel with a probability $p = 0.3$ does not impair the network’s performance.

After two steps the image is perfectly reconstructed.
The Hopfield Network

However, for noise created by $p = 0.4$, the network is unable to recognize the original image. Instead, it converges against one of the 19 random patterns.

Problems with the Hopfield model are that:

- it cannot recognize patterns that are shifted in position,
- it can only store a very limited number of different patterns.

Nevertheless, the Hopfield model constitutes an interesting neural approach to identifying partially occluded objects and objects in noisy images. These are among the toughest problems in computer vision.