Creating Data Representations

The problem with some data representations is that the meaning of the output of one neuron depends on the output of other neurons. This means that each neuron does not represent (detect) a certain feature, but groups of neurons do. In general, such functions are much more difficult to learn.

Such networks usually need more hidden neurons and longer training, and their ability to generalize is weaker than for the one-neuron-per-feature-value networks.

Creating Data Representations

On the other hand, sets of orthogonal vectors (such as 100, 010, 001) representing individual features require more neurons and connections but can be processed by the network more easily. This becomes clear when we consider that a neuron’s net input signal is computed as the inner product of the input and weight vectors. The geometric interpretation of these vectors shows that orthogonal vectors are especially easy to discriminate for a single neuron.

Creating Data Representations

Another way of representing n-ary data in a neural network is using one neuron per feature, but scaling the (analog) value to indicate the degree to which a feature is present.

Good examples:
- the brightness of a pixel in an input image
- the distance between a robot and an obstacle

Poor examples:
- the letter (1 – 26) of a word
- the type (1 – 6) of a chess piece

Creating Data Representations

This can be explained as follows:
The way NNs work (both biological and artificial ones) is that each neuron represents the presence/absence of a particular feature. Activations 0 and 1 indicate absence or presence of that feature, respectively, and in analog networks, intermediate values indicate the extent to which a feature is present.

Consequently, a small change in one input value leads to only a small change in the network’s activation pattern.

Creating Data Representations

Therefore, it is appropriate to represent a non-binary feature by a single analog input value only if this value is scaled, i.e., it represents the degree to which a feature is present.

This is the case for the brightness of a pixel or the output of a distance sensor (feature = obstacle proximity).

It is not the case for letters or chess pieces.

For example, assigning values to individual letters (a = 0, b = 0.04, c = 0.08, ..., z = 1) implies that a and b are in some way more similar to each other than are a and z.

Obviously, in most contexts, this is not a reasonable assumption.

Bag-of-Features Method

- Find key points in training images, such as corners (FAST).
- Compute a descriptor (feature vector) for each of them (e.g., BRIEF).
- Detect clusters in the descriptor space spanned by all resulting descriptors (use mean shift, k-means, etc.).
- Choose one representative descriptor for each cluster.
- For each training image, determine the minimum distance of any key point or pixel window to each of the representatives.
- The vector of minimum distances is then taken as the feature vector for that training image.
- Train and test a classifier with these feature vectors to turn it into an object or scene recognizer.
Convolutional Neural Networks (CNNs)

Convolutional networks perform convolution to extract local features instead of having complete connectivity between layers. They also employ pooling layers to make classification more robust towards slight changes in feature locations.