**CMPE 480 INTRODUCTION TO ARTIFICIAL INTELLIGENCE**

**FINAL ANSWERS**

1. The features are the words in the documents. A very simple ANN is the following, where the inputs are the words (features) and the output is the index of the class.

word1

word2 class

...

wordn

*An example*: In the training set, we have the following five documents and each document contains a few words as shown below. We have two categories, music and sport.

D1 = “a music document” (class: music)

D2 = “football and basketball” (class: sport)

D3 = “music is very good” (class: music)

D4 = “is playing violin” (class: music)

D5 = “football player” (class: sport)

We can show the training data as follows. Features are boolean features such that 0 denotes that the word does not occur in the document and 1 indicates that the word occurs in the document. The output is also boolean; 0 indicates the music class and 1 indicates the sport class.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | a | and | basketball | document | football | good | is | music | player | playing | very | violin | Class |
| D1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| D2 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| D3 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| D4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| D5 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

There are 12 features (i.e. inputs to the neuron). Suppose that all the initial weights are 0.10. That is, wi = 0.10 for 1≤i≤12. Suppose that the threshold t is 0.50 and the learning factor d is 0.05.

Iteration 1:

D1 : < t. So assigned to music class. Correct.

D2 : < t. So assigned to music class. Incorrect. Weights are updated as {0.10, 0.15, 0.15, 0.10, 0.15, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10}

D3 : < t. So assigned to music class. Correct.

D4 : < t. So assigned to music class. Correct.

D5 : < t. So assigned to music class. Incorrect. Weights are updated as {0.10, 0.15, 0.15, 0.10, 0.20, 0.10, 0.10, 0.10, 0.15, 0.10, 0.10, 0.10}

Iteration 2:

D1 : < t. So assigned to music class. Correct.

D2 : ≥ t. So assigned to sport class. Correct.

D3 : < t. So assigned to music class. Correct.

D4 : < t. So assigned to music class. Correct.

D5 : < t. So assigned to music class. Incorrect. Weights are updated as {0.10, 0.15, 0.15, 0.10, 0.25, 0.10, 0.10, 0.10, 0.20, 0.10, 0.10, 0.10}

The process continues in this way. Note that after the second iteration, the weight of the word “football” was increased to 0.25, indicating that it mostly occurs in the documents of “sport” class. (As the value of increases, the document becomes more close to the “sport” class, since the “sport” class is denoted by the output value 1.) Also, the weights of “and”, “basketball”, “player” were increased, which occur in “sport” documents.

1. The classical domain-independent attributes which are important in knowledge representation are *isa* and *instance* attributes. These attributes automatically support property inheritance They are represented in different ways in different formalisms.

We can give the following example for semantic network representation:

instance isa

In logic, we can represent as follows:

instance (Ali, Pitcher)

isa (Pitcher, Baseball Player)

or

Pitcher (Ali)

∀x Pitcher (x) 🡪 Baseball Player (x)

1. An inverse attribute of an attribute enables us to interpret a relation in both ways. In logical representation, this is done automatically. For instance,

team (Ali, GS)

can be interpreted in both directions: Team of Ali is GS or GS is the team for Ali.

In semantic networks, we can use two different attributes as shown below:

team

team-member

1. Like classes and values, attributes may also form a hierarchy. The example below shows an hierarchical relation (isa relation) between two attributes:

isa

Such relations may be important during reasoning. For example, given a request such as “Display the physical sizes of ...” in a system, we can understand that “height” is a “physical size” by making use of the above relationship and display the height data.

1. A single-valued attribute may take a single value at a time, whereas a multi-valued attribute may take more than one value. In the case of a single-valued attribute, it is not allowed to assert more than one value at a time in the knowledge base. For example, height of a person may be a single-valued attribute. When another height value is tried to be inserted for a particular person in the KB, a contradiction signal can be given by the system.
2. Low-level primitives are used to represent the data in a more detailed level, whereas high-level primitives are used to represent in a more abstract way. For instance, the sentence “John spotted Sue” can be represented in high-level as

spot (agent(John), object(Sue))

or in low-level as

see (agent(John), object(Sue), timespan(briefly)).

In the first case, we can answer questions like “Who spotted Sue?” easily, but we cannot answer questions like “Did John see Sue?” without knowing that “spotting” implies “seeing”.

The basic advantage of low-level primitives is that we can represent similar but different things (e.g. “see”, “spot”, “look”, etc.) in the same way (using “see” only). This means that we use few number of primitives (operators), thus few inference rules. This makes reasoning in a system much easier. Remember the conceptual dependency notation. It assumes that every action can be represented with just 10-15 primitives (ptrans, attend, etc.).

The disadvantage of low-level primitives is that it uses more storage. For instance, in the representation of the data “John spotted Sue” and “Sue spotted Mary” with the above “see” primitive, “timespan(briefly)” argument is stored twice. Another disadvantage is that it is more difficult to convert from natural language to the representation, since more detail should be extracted from the data.

1. The scripts can be indexed by the significant words they contain. For instance, in the “restaurant” script, we can index it with the words like “steak”, “order”, “bill”. Then, when these words occur in the given text, we can access that script (although the word “restaurant” may not appear in the text).

But, there is a problem that a word may point to more than one script. For instance, the word “steak” may indicate both “restaurant” and “supermarket” scripts. In this case, we can use the important words in the text and take the intersection of the structures they point to. For instance, if the word “steak” indicates “restaurant” and “supermarket” scripts and the word “bill” indicates “restaurant” and “shopping” scripts, the intersection is the “restaurant” script and we access it.

Another problem is that the intersection may be empty. For instance, in the text “John rode his bicycle to Steak and Ale ... He paid the bill ...”, the intersection of the words “steak”, “bill” and “bicycle” will be empty, since the word “bicycle” is not related to restaurants. In this case, we can give an order to the words and begin from the most important word. If we first use “steak” and then “bill”, the “restaurant” script will be identified, without the need to consider the word “bicycle”.