Chapter 4

Multi-Agent System Design

In the design of a Multi-Agent system, three main stages have been identified. Each of these stages makes use of different components to work with the entity in transit. In our system, the input is a dataset, and the output is a multi-agent system trained on the dataset. Figure 4.1 shows the structure of the system. The three stages in our system are **Preprocess, Training and PostProcess**

4.1. The Preprocess Stage

The input data may be defined in different formats. It may be a binary string, an image, a table of data or any other structure. However, the input data needs to be in a specific format for it to be used in our system. The dataset Ξ must be composed of a sequence of samples X, where each sample contains a list of features and a label: $X_i = \{x_1, x_2, ..., x_n\}, \forall X_i \exists y_i$. Hence, it is necessary for the dataset to already have the desired structure when entering the system, and that responsibility is delegated to the dataset itself.

The dataset structure is then the container of all the data, and is responsible for **a)**providing each sample $X \in \Xi_{train}$ and $X' \in \Xi_{test}$ with all its features and label(s)

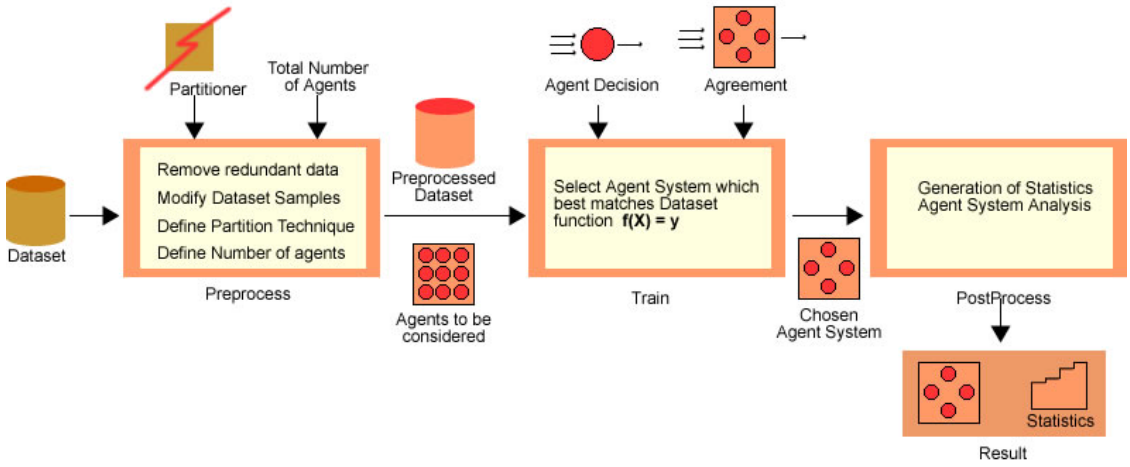

Figure 4.1: Diagram showing the different stages and components in the overall training process of a multi-agent system

and **b)** providing information on the dataset itself such as size, number of labels per sample and number of features.

The dataset is the input into the system. Even when it is expected for it to have the correct format, many times it is necessary to modify it to improve the performance of the learning machine. For instance, there are cases where certain features are irrelevant as they have the same value for every sample in the dataset and hence no distinction is possible by using them. Another example is when two features have identical values for every sample in the dataset. In this case, it is useless to have the second sample as it provides no further information from the first one.

However, the preprocess stage may go further from eliminating features. It is possible to use a data analysis technique like PCA [17] to reduce the number of features. It is done by analyzing which are the dimensions of most variation and discarding those with the least (Figure 4.2). This is specially useful on datasets where the amount of features is too big. Another simplification consists on combining a set of features into a single one. For instance, consider the case of an image. A possible consideration is taking each

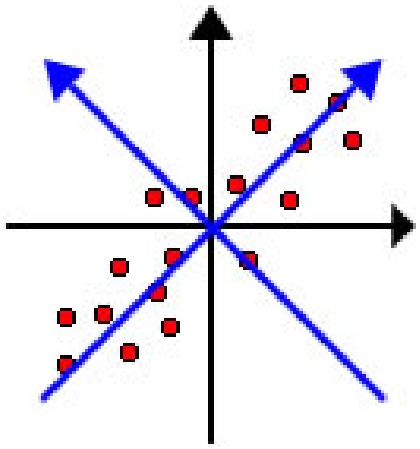

Figure 4.2: In principal component analysis, the dimensions of greatest variation are detected so that data may be represented against those dimensions. Such representation is usually more useful when comparing samples.

pixel as a feature, but this approach might not result useful, as the number of features may be too large. However, if instead of taking a pixel we consider a $m \times n$ window in the image, and take the average pixel value in that window, we would be reducing the number of features to $\frac{|X|}{mn}$. In some cases, such reductions are useful as we make the dataset less sensitive to small variations. However, it is sometimes this sensibility which lets us achieve better training results. (figure 4.3)

Another approach, instead of reducing the number of features, attempts to increase them by using the combinations of different features as single features. This approach stresses the importance of the simultaneous appearance of some features as information to be used during training. For instance, again in the case of an image, the fact that one pixel has minimum value and the contiguous one maximum may be indicative of a certain shape (figure 4.4).

We have then identified the following approaches for Dataset preprocess:

- Mapping the dataset to another representation, as in the case of PCA. Another possibilities are using a Kernel function (A mathematical which takes the value of

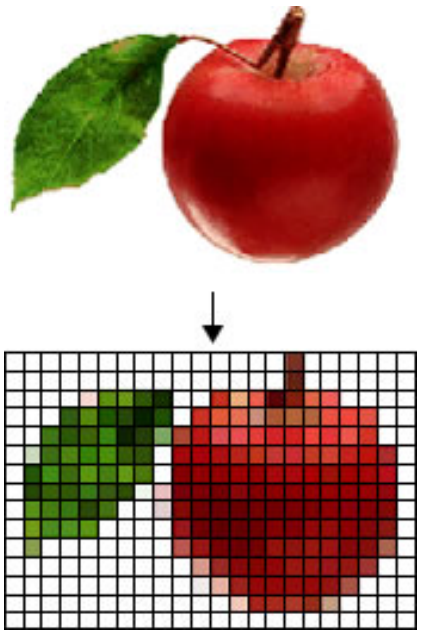

Figure 4.3: Feature reduction in an image. Given a small window, the average is taken of all the pixels inside that window to create a feature.

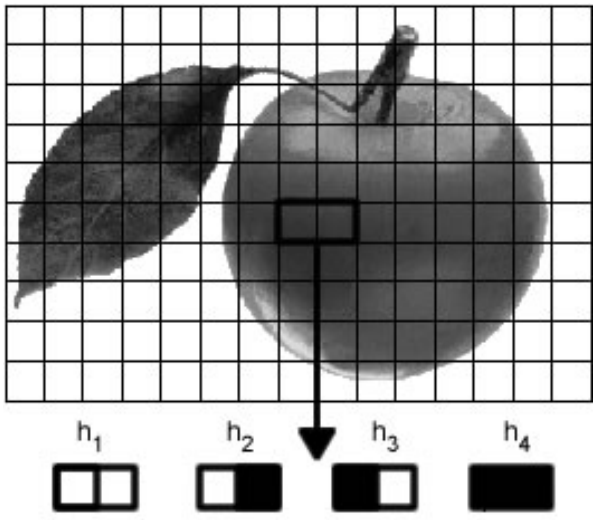

Figure 4.4: Combination of values in features may be considered as features themselves. In the image, 2 possible values for a pixel are considered: light and dark. Given 2 pixels each possible case is considered as a feature.

a feature as an argument. For instance $k(x_i) = x_i^2$) to change the value of a feature or using another transformation method as the fourier transformation.

- Reducing the number of features either by removing unnecessary features or converging the number of features into a single one.
- Incrementing the number of features by considering the simultaneous presence of certain features as a single feature.

The Preprocess stage will output a new dataset ready for the learning process.

4.2. The Learning Stage

Once the dataset is prepared, the learning stage takes place. In the learning stage an agent system is built to best match the function f, such that $f(X_i) = y_i$. In chapters 2 and 3 we analyzed some of the methods used for training. These methods, part of an area of machine learning called ensemble learning, allow us to build multi-agent systems able to overcome common learning problems such as the representation problem or bias. However, several decisions must be made when designing the learning stage of an ensemble algorithm, such as what data does each agent have access to, how does a single agent make a decision and how does the agent system agree on a response. These problems are known as the **partition, agent decision and agreement** problems.

4.2.1. Dataset partition

Dataset partition refers to how the data should be divided among the different agents. Intuitively, it indicates what each agent gets to see. Partition is directly related to how many agents (or learners) will be considered as candidates for the Agent System and whether 2 or more agents will have access to the same information.

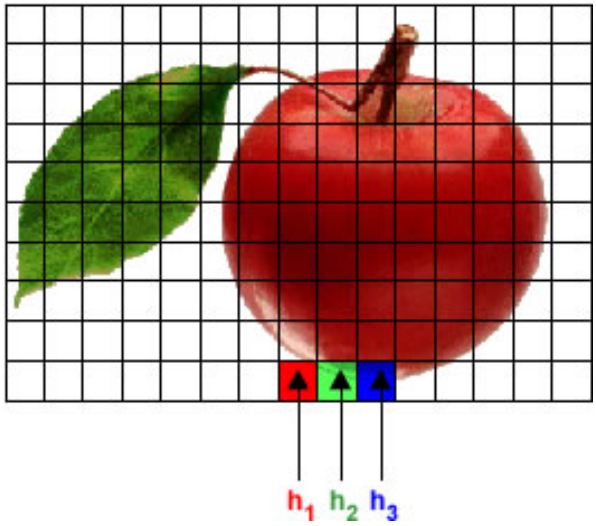

Figure 4.5: Hard partition. No pair of agents has access to the same features.

In general, to basic types of partition may recognized. The first type, **Hard Partition**, is a type of partition where there is no pair of agents h_i, h_j such that both agents have access to a feature x_k in a sample X (Figure 4.5). In hard partition, we assume that there is no relation between the decision of one agent and another, as there is no data shared among the two of them. The relevance in this fact is that in the training process, only if the features available to each agent are similar will they have similar contributions. They will however be different in any other case. This is in many cases desirable in ensemble methods, as the overlap between the decisions of two different agents over the whole Dataset will be smaller, and such diversity will broaden the function space H achievable by the ensemble.

The second type of partition is **Soft Partition**. In soft partition, there exists at least one pair of agents h_i, h_j such that both agents have access to a feature x_k in a sample X (Figure 4.6). In soft partition, two agents may share information, however this does not mean that both agents will make the same decision. The example of taking two pixels in an image is again an useful example in this case. Given three pixels,

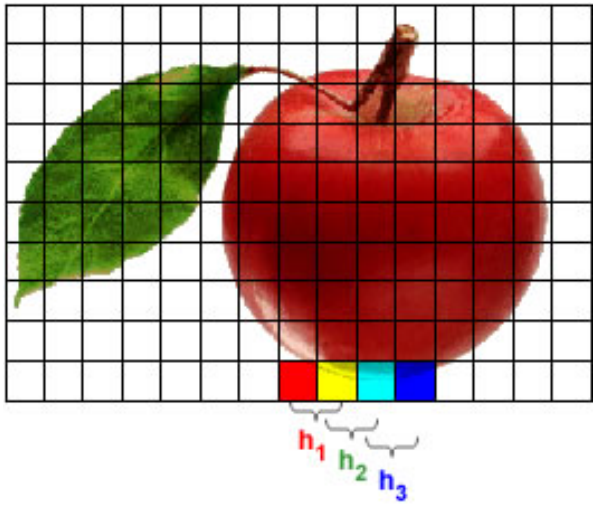

Figure 4.6: In soft partition, agents may share their access to features. For instance, as both h_1(red) and h_2(green) have access to the same feature, the feature is colored as yellow (red + green)

p_1, p_2, p_3, agent h_i may have access to p_1, p_2 while agent h_{i+1} may have access to p_2, p_3. The prediction that each agent gives on an image X which contains such pixels may be different based on the influence of p_1 and p_3 respectively, but it is possible as well that the influence that p_2 produces in such agents is strong enough to overcome p_1 and p_3 and produce the same prediction. As we will analyze in the following section, how each agent makes a decision is crucial in defining such influence, however, what is clear is that the decision of h_i and h_{i+1} will be closer over the complete dataset than it would be if both agents had no data in common. In ensemble systems, such behavior is useful when it is desirable to increase the strength of predictions as it will be more common to have agents which agree with agent h_i on its predictions.

There exist other approaches in partition which attempt to optimize partition and learning simultaneously [26]. However, in our research we only considered these basic forms of partition as such require modifications of the defined agents over the learning stage.

4.2.2. Agent decision

How a single agent makes a decision is an important factor in the learning process. Given an agent h, it will receive a sample $X_i = \{x_1, x_2, ..., x_n\}$ and will produce a response $h(X_i)$. In classification, the response is generally $h(X_i) \times \{-1, 1\}$, however, in prediction the response is generally a real number $h(X_i) \times \Re$. The process that an agent follows for making a decision is shown in the next algorithm:

$AgentDecision(h_i, X)$

1 $X' \leftarrow ExtractFeatures(X, i)$

2 $prediction \leftarrow CalculatePrediction(h, X')$

3 **return** *prediction*

In the algorithm, **ExtractFeatures** performs the partition process. Based on the id of the agent, it extracts the partition of the features $X' \subseteq X$, based on the partition procedure (Soft or Hard Partition). Once having the corresponding partition, the agent calculates its prediction. The most common method for calculating a prediction if the simple average over X', such that

$$p(X) = \frac{1}{k} \sum_{i=1}^{k} X'_k, \ X' \ is \ a \ partition \ of \ X, X' \subseteq X \tag{4.1}$$

A second approach is using the maximum or minimum in X. Finally, another approach involves the use of kernel functions or data transformations.

The previous methods generally output a double value, such that for prediction $h(X) = p(X)$, however in the case of classification it is necessary to output a label. Among methods to map $\Re \rightarrow \{-1, 1\}$ the most common is **Decision Stumps** [8]. In decision stumps, a threshold τ is defined such that

$$h(X) = \begin{cases} 1 & \text{if p(X)} > \tau \\ -1, & \text{otherwise} \end{cases} \tag{4.2}$$

4.2.3. Multi-Agent agreement

Once the agent system has been created, it is necessary to calculate what the decision of the complete Agent System will be, based on the decision of the individual agents. The process in which the output of each of the agents is used in calculating an output for the whole system is called **Agreement**. There exist several agreement methods, some of them linked to an algorithm (as in the case of boosting, where the weighted sum of the agents'predictions is the ensemble's output). The most common agreement function is the simple average, such that

$$E(X) = \frac{1}{|E|} \sum_{i=1}^{|E|} h_i(X), \;\; E = \{h_1, h_2, ...h_{|E|}\} \tag{4.3}$$

Another common method is the weighted sum. This method, used in Adaboost, associates a weight α to the prediction of each agent, such that:

$$E(X) = \sum_{i=1}^{|E|} \alpha_i h_i(X), \;\; E = \{h_1, h_2, ...h_{|E|}\} \tag{4.4}$$

As in the case of classification, we need to map the result from the Ensemble to a label. In the case of binary classification, we again will use decision stumps, but in this case generally $\tau = 0$

$$H(X) = \begin{cases} 1 & \text{if } E(X) \geq 0 \\ -1, & otherwise \end{cases} \tag{4.5}$$

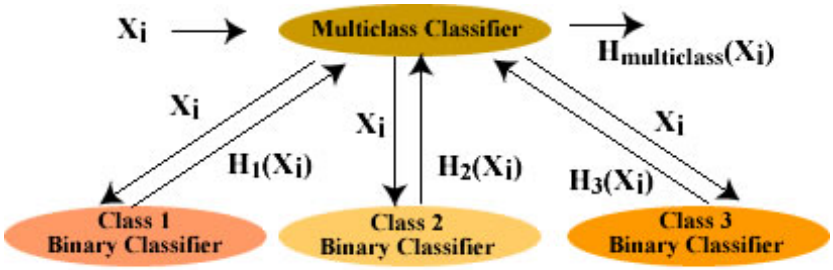

Figure 4.7: In multiclass classification, the class assigned to a sample is based on the result of a set of binary classifiers, one for each class.

An special case of agreement takes place when the problem is a multi-class classification (Figure 4.7). In this case, we assume we have a set of binary classifiers for k different classes. In that case, the classifier with the highest prediction will indicate the label. For that, we define $H_{multiclass}$ as the multiclass classifier and E_i the binary classifier for class i. The agreement function will then be

$$H_{multiclass}(X) = i,\ where E_i(X) > E_j(X) \forall i \neq j\ i, j \in [1, k] \tag{4.6}$$

4.3. The Postprocess Stage

The Postprocess stage takes place once the learning process is finished, and consists on an analysis of the Agent System and its prediction both over training and unseen test area. The most important statistics to be calculated in the postprocess stage are the **TrainingError** and the **GeneralizationError**. We define the training error as

$$err_{train}(H) = \frac{1}{n} \sum_{i=1}^{n} [|H(X_i) \neq y_i|], X_i \in \Xi_{train} \tag{4.7}$$

We define the generalization error as:

$$err(H) = \frac{1}{n}\sum_{i=1}^{n}[|H(X_i) \neq y_i|], X_i \in \Xi_{test} \tag{4.8}$$

The previous 2 statistics provide information regarding what is the proportion of incorrect predictions in the complete dataset. In the case of prediction, instead of requiring the exact same value from $H(X_i)$ and y_i, generally a small error margin m is defined, such that $\|H(X_i) - y_i\| < m$.

There exist as well several statistics dependent on the type of problem. In binary classification, **precision**, **recall**, **Balance Error Rate (BER)** and **Area under curve (AUC)** are useful for measuring the performance of the classifier, while in multiclass classification a **Confusion Matrix** is useful to identify how a certain class is being classified compared to other classes.

In the case of binary classification, precision is defined as the fraction of predicted positives ($H(X_i) = 1$) which are true positives ($y_i = 1$) [13].

$$precision(H) = \frac{1}{\sum_{i=1}^{n}[|H(X_i) = 1|]}\sum_{i=1}^{n}[|H(X_i) = y_i \&\& H(X_i) = 1|], X_i \in \Xi_{test} \tag{4.9}$$

Recall is a complementary measure from precision. It indicates the fraction of predicted positives out all true positives.

$$recall(H) = \frac{1}{\sum_{i=1}^{n}[|y_i = 1|]}\sum_{i=1}^{n}[|H(X_i) = y_i \&\& H(X_i) = 1|], X_i \in \Xi_{test} \tag{4.10}$$

BER is defined as the average of the errors on each of the classes:

$$ber(H) = \frac{1}{2}(\frac{\sum_{i=1}^{n}[|H(X_i) \neq y_i \&\& H(X_i) = 1|]}{\sum_{i=1}^{n}[|H(X_i) = -1|]} + \frac{\sum_{i=1}^{n}[|H(X_i) \neq y_i \&\& H(X_i) = -1|]}{\sum_{i=1}^{n}[|H(X_i) = 1|}]) \tag{4.11}$$

Finally, AUC is defined as $1 - ber(h)$ and it indicates the area obtained by plotting the proportion of true negatives against the proportion of true positives.

In the case of multi-class classification, the confusion matrix indicates, given a certain class, how many times that class was correctly classified and with which classes it was confused. The confusion matrix M is a $k \times k$ matrix, where k is the total number of classes, and for every sample X_i, position $M(y_i, H_{multiclass}(X_i))$ is increased.

In the case of prediction, useful statistics are the prediction average error $\mu_{err(X)} = \frac{1}{n}\sum_{i=1}^{n} \|H(X_i) - y_i\| \in \Xi_{test}$ as well as the standard variation of the average Error $\sigma_{err(X)} = (\frac{1}{n}\sum_{i=1}^{n}(\|H(X_i) - y_i\| - \mu_{err(X)})^2)^{\frac{1}{2}}$.