# Pattern Classification

The principle behind pattern classification algorithms is to differentiate different classes of objects by viewing them as having different *models* (mathematical descriptions - typically of sets of combinations of [processed] input signals). [1, p. 2]

A typical pattern recognition system can be partitioned into six components as shown in Figure 1.

|  |
| --- |
|  |
| ***Figure 1.*** *Six components/steps of a typical pattern recognition system.* |

**Sensing:** The input is usually a transducer (i.e. a camera, microphone array, etc.). The transducers themselves will have their own characteristics and limitations such as bandwidth, resolution, sensitivity, distortion, signal-to-noise ratio, latency, etc. [1, p. 9] Though the quality of sensors may have an impact on the performance of the pattern classification system, and thus the limitations of the sensor modules being used in this prototype should be considered and taken into account during the design and testing of the system, the scope of this project is *not* inclusive of sensor and sensor interface design.

**Pre-processing:** The input signals may have to be rectified to reduce impact of irrelevant variations, increase the signal-to-noise ratio, and wrap input signals to a “standard template” (normalization).

**Segmentation and Grouping:** The main purpose of segmentation is to hone in on the features of interest (or ultimately, the object of interest) that would be useful in the classification process. This in itself is one of the major problems in pattern recognition; how can you segment the input signals (i.e. image of many fishes on a conveyor belt) before they’ve been categorized? Also, how can you categorize the same input signals before they’ve been segmented? This requires the system to ‘know’ when there is just a background or “no category” object present. [1, p. 10]

**Feature Extraction:** The purpose of this component is to yield a representation of the input signals such that the task of the Classification component seems trivial. Feature extraction may be considered a part and parcel of the classification system – the distinction being practical rather than theoretical.

Ideally, features should be measurements which be very similar for objects in the same category, and very different for objects in different categories – leading to the idea of seeking ‘distinguishing features’. Features must also be invariant to irrelevant transformations of the input i.e. translation, rotation, and scale. Occlusion (parts of an object becoming hidden), projective distortion (with distance, angle, etc.), and complex transformations of objects (causing deformation of measurements even though the object remains the same – i.e. when fingers are clicking, the hand is still a hand) need to be considered when selecting features – hence it is very important to know the exact domain of the pattern classification system at this stage.

**Classification:** This components uses the feature vector provided (from previous component) to assign objects to categories/classes. Generally, this involves determining the probability for each of the possible categories. The problems associated with classification usually depend on the variability in the feature values for objects in the same category in relation to objects of different categories. [1, p. 12]

The classification component should also consider the fact that there may be occasions where, due to occlusion or another reason, not all of the features associated with a particular category may be present. To assume the value of the missing feature is zero or an average of the values previously seen does not (provably) yield an optimal result – how a classifier should be trained or used in this situation is one of the concerns of classifier design.

**Post-processing:** The post-processor decides on a recommended action dependant on the result from the classifier. The component should take into account the typical error rate of the classifier (percentage of patterns assigned to the wrong category); this could potentially act as a negative feedback mechanism for the entire pattern classification system in order to improve feature patterns for classification.

It is to be noted however, that in the current design of the cognition system, the decision making process is outsourced to a separate faculty, the decision-making faculty. However, as the decision-making faculty relies very much on the classifications deduced by the visuospatial faculty, it is important to consider that the boundaries of the post-processing component should not be limited to just the visuospatial faculty alone as the entire cognition system can be considered a holistic and highly coupled system (or at least it should be due to the level of interdependence of the faculties – see figure x). Therefore, the post-processing stage should take into account the cost (or risk) of each particular action that results from the classification.

## Multiple Classifiers?

It may be possible that having multiple pattern classification systems yields a more accurate result. The most common approach would be a pattern in parallel, with a final “super”classifier that produces the *final* classification using all *previous* classifications. The inherent problem with this (and often a subject discussion) is the matter of establishing how to determine when the minority opinion is correct. [1, p. 13] This topic is discussed in greater detail in Roli’s lecture notes titled ‘Mini Tutorial on Multiple Classifier Systems’. [2]

## The Design Process

|  |
| --- |
|  |
| ***Figure 2.*** *Steps involved in the design of a pattern recognition system.* |

**Data Collection: Because** pattern recognition and classification systems need to be trained and tested with a set of training data large enough to assure good performance and representation (of objects).

**Feature Choice:** This is a crucial design step and depends largely on prior knowledge of the domain, and access to (fully representative) example data.

**Model Choice:** Designing the mathematical model that would ultimately allow for the designation of objects into their appropriate categories. How feedback and supervision will be used in the mathematical model (if at all), would be ascertained.

We are interested exclusively in the implementation of a *multi-class classification model*, as we must ultimately allow for the capability of this sub-faculty to create as many classifications of objects as necessary; this would allow for greater versatility in the way that the entire autonomous navigation system could be trained.Effectively, multi-class classification models may comprise of one or many neural networks, and the most common approach to tackle the issue of modelling such classification system is to decompose the problem into multiple two-class classification problems. [3]

**Training:** The process by which the system uses data to determine the classifier (or “pattern class”) must be determined. There are three *types* of learning paradigms by which a pattern classification system may be *‘learn’* classification [1, pp. 15-17]

1. **Supervised Learning** – a *teacher* provides the category labels *or* cost for each pattern in the training set.
2. **Unsupervised Learning.** This method uses no explicit guidance from a “teacher”. Hence, the system forms clusters or “natural groupings” of the input patterns alone.
3. **Reinforcement Learning.** This method of learning essentially involves a *teacher* reinforcing correct outputs, and discouraging incorrect outputs. I.e. the only input from the *teacher* is whether or not the outcome of the classifier is right or wrong.

**Evaluation:** An error rate must be obtained for the classification system in order to evaluate its performance and identity the need for improvement in its components. [1, p. 15]

**A note on the consideration of Computational Complexity**

It is important that the labelling time and storage requirements are considered when implementing the pattern classification system – with respect to the entire framework, including robot hardware, user PC and *where exactly the pattern classification code will be executed, how the data will be stored, how data will would be transferred (wirelessly) and whether it is feasible to consider custom logic to enhance overall performance (with respect to time).*

# Supervised Learning Methods

## Bayesian Decision Theory and parameter Estimation

Considered a fundamental statistical approach to pattern classification.

Involves quantifying trade-offs (between various classification decisions) using probability and cost (of decisions).

### Procedure

The Bayesian Decision Theory can be utilized for the purpose of classification in the following manner [1, pp. 21-64] [4]:

1. Define a discrete number of categories, for example:
2. Assume a *priori probability* i.e. the likelihood of the next “catch” belonging to a particular category in terms of a probability. This is expressed as a *mass probability function*, P(•), which is in essence a probability value for a discrete variable (in this case ), where each value of the variable has its own probability value (i.e. each value of discrete variable will have a probability value describing it’s prior/likelihood of occurrence).   
    **Rules:**

, and

1. Rarely are decisions made using the *priori probability* alone.Instead, we supplement the classifier with other variables (measurements and indicators to *help* with the classification problem). These are expressed as *probability density functions*, in the form p(•), which essentially indicates probability for a continuous variable – the function essentially returns the *probability* of a defined *range* (i.e. some *unit distance*). This can be expressed as:  
     
    or   
     
   **Rules:**  
     
   , and  
     
      
     
   (i.e. the probability density function is normalized so that the area under each curve is exactly 1)  
     
   Ultimately, if x is the continuous variable and the class in question, the *class-conditional probability density* function can be expressed thus: – “the density function *given* the state of nature ”.
2. Bayes formula is used to deduce the *mass probability* of the object in question belonging to a particular category, given a particular *feature of interest* **x** (the measurement or indication to support the *priori probability*):

So…

### Generalization of the Bayes Formula

Bayes Formula can be expressed (very loosely) in English in order to describe its generic operation:

***Evidence*** is essentially the *scale factor* that, in the equation, assures us that . Essentially a constant if all feature variables are known:

(visio illustration here)

### Error & Decision Making

For , the error for any value of **x** is deduced in the following way:

If we go by the convention and define as a rule, that is selected so long as that , else is selected, then, the same equation becomes: .

The overall error in classification (termed *average probability of error*) is given by:

*Where denotes the probability of the and value occurring simultaneously.*

If we eliminate “evidence”, i.e. , from the equation – we ultimately end up with a more concise expression of (exactly the same) decision rule:

Decide if , else decide in favour of .

#### Risk Minimization

A general decision rule describes every possible action that can be taken for every possible observation of the continuous variable **x**. The risk associated with each action is denoted. Hence, the overall risk of the classifier is given by:

Therefore, minimizing for every **x**, we can reduce the overall risk of the system. This can be achieved by the computation of the following equation:

Where is the loss function that indicates how costly each action is, and ultimately allows the conversion of a probability into a decision.

*The minimum overall risk is called the Bayes risk –indicative of best possible performance of the classifier, and is denoted R\*.*

#### Insight into Bayesian Decision Making

If for any given value of **x**, we set , then the ultimate decision hinges entirely on the *priori probabilities*.

If, instead, we set – i.e. both states of nature are *equally* probable, *then* the decision hinges entirely on *likelihoods*.

In general *both o t*hese factors are important, and the purpose of Bayes decision rule is to combine both in order to achieve *minimum* probability of error.

### Application in AVINSOR Prototype 1

The proposed use of the Bayes classification algorithm is illustrated in figure x.

#### Input streams - VARx

In the AVINSoR framework, the Bayes classification algorithm is used to classify the values of input variables (i.e. the different streams of input data) amongst a number of differentcategories, to create a number of *different* ‘perceptions’ that will ultimately aid the navigation system in decision-making. [[1]](#footnote-1) All input variables (whether from sensors, or other sources) are referred to as *input streams*.

#### The Bayes Classifier Module

AVINSoR Client is designed to allow the user to instantiate an (ideally) unlimited number of *Bayes Classifier Modules*– each to be associated with a particular ‘perception’. Hence the term *Bayes Classifier Module* should be considered synonymous to ‘perception’ beyond this point.

The client allows the user to define:

1. The different *input streams* from the system to be associated with the Bayes Classifier Module.
2. The different *categories* that the Bayes Classifier Module should classify the input values (of the associated input streams) amongst.

##### Bayes Classifier Object

For each input stream associated with a Bayes Classifier Module in the *client* application - a Bayes Classifier object is instantiated in *MATLAB®*. A MATLAB Bayes Classifier object functions to generate and maintain all probability, risk, and action matrices that associate the particular input variable stream with the defined list of categories.

Through the client, the user can associate any potential value that could be seen at an *input variable stream* with the *maximum likelihood* *of occurrence of a particular category*. This processes is referred to as *“creating an association” (*between the *input value* and a *category*,for a particular *‘perception’).*  This is achieved through the Bayes Classifier object which represents the input stream of the ‘perception’ in MATLAB - essentially a Gaussian curve should be created at the index of the category of the *likelihoods* matrix, which peaks at the *associated* value.

Note: for each Bayesian Classifier, each category also has a *prior probability* associated with it, the default value of the *priory* (upon instantiation) is the reciprocal of the number of categories (i.e. all are equivalent, hence decision is made *entirely* on the basis of the *likelihoods* matrix). The priories can be changed by the user if a category is deemed to have a higher or lower likelihood of occurrence regardless of the actual input value seen. This will be achieved through the client GUI – which will provide mathematical abstraction and some automation to guide the process.

#### Result

The decision logic of a Bayes Classifier Module essentially returns the *mode* (most frequently occurring) *action* *category* (*viz.* decision) of the Bayes Classifiers - *if* there is an *obvious* mode (i.e. the frequency of occurrence of a particular action is *exclusively* greater than all other actions). On the case of there being no obvious mode, the decision logic returns the action category of the Bayes Classifier whose action is associated with the *minimal risk* value. In the case of their being more than one Bayes Classifier with the same *minimal* level of risk, the *action category* of the *first to be iterated over* will be returned.



# Object-Orientated Design



Figure x. UML Class diagram of the classes that will be used to realize the Bayes Classifier Module in AVINSoR Client.

## Class Descriptions

**ClassVariable** is the class that represents a variable associated with classification – only a single instantiation should be required per input stream *in* the system. All classification variables of interest to a Bayes Classifier Module (object) are passed through as reference via the **ModuleVariables** parameter.

**ModuleVariables** and **ModuleCategories** are essentially enumerable collections that should inherit from .NET Framework’s iEnumerable class.

**ClassCategory** is the class the represents a classification category to be used by an instantiation of a Bayes Classifier Module. Note that unlike ClassVariable, ClassCategory is exclusive to a particular instantiation of a BayesClassifierModule, hence its complementary collection type ModuleCategoriesdoes not have to be instantiated outside of the classifier module and there is no need for it to be passed as a parameter. Instead, the collection of categories is initialized upon the instantiation of the classifier and can be accessed publicly to add/remove/modify a particular category.

**MatlabInterface** is a static class that initializes the global connection to the MATLAB® COM Automation Server and provides a reference to it. The class will also feature methods that may be useful when transferring data between the client application and MATLAB® server.

**BayesClassifierModule** is the class that represents a module like that depicted in figure x. Though its properties and behaviours are self-explanatory through the UML alone

Behaviours

BayesClassifierModule(…)

Represents the Bayes Classifier Module.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| server | MLApp | Reference to the global MATLAB COM Automation Server. |
| moduleVariables | ModuleVariables | A reference to all the variables (input data streams) to be associated with this particular Bayes Classifier Module. |

**Returns** a BayesClassiferModule object (class constructor).

Associate(…)

Associates a particular (specified) value of an input stream with the maximum likelihood of belonging to the specified classification category.

If the referenced variable or category does not exist in the BayesClassifierModule’s collections, an exception should be thrown with a notification (for the GUI).

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| variable | ClassVariable | Object reference to the variable. |
| category | ClassCategory | Object reference to the category. |
| value | int | Value of *variable* to associate with *category*. |

SetPriory(…)

Adjusts the prior probability of the referenced category with respect to the specified variable.

If the referenced variable or category does not exist in the BayesClassifierModule’s collections, an exception should be thrown with a notification (for the GUI).

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| variable | ClassVariable | Object reference to the variable. |
| category | ClassCategory | Object reference to the category. |
| priory | double | The prior probability. |

GetDecision(…)

Gets the ultimate classification decision made by the Bayes Classifier Module (via *decision logic*) using values currently seen (or last sampled) in the input stream variables associated with the module.

**Parameters:** *none.*

**Returns** a ClassCategory object reference to the resultant category.

GetDecision(…)

Gets the ultimate classification decision made by the Bayes Classifier Module (via *decision logic*).

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| values | int[] | An integer array of the values to use with respect to the input variables. The integer at index 0 should be complimentary to the variable at index 0 of the module’s variable collection.  Note that this method could be implemented using a custom collection class of (for example) ‘VariableValue’ objects with the purpose of associating an integer value with a referenced input stream variable, hence eradicating ambiguity in indexes. |

**Returns** a ClassCategory object reference to the resultant category.

# Appendix

## Appendix A: Probability Notation

The notation used when describing most probability theorems including Bayes’ Theorem [4]:

probability that event A occurs.

probability that event B occurs.

probability that event A does not occurs.

probability that event A occurs given that event B has already occurred.

probability that event B occurs given that even A has already occurred.

probability that event B occurs given that event A has not occurred already.

# References

|  |  |
| --- | --- |
| [1] | R. O. Duda, P. E. Hart and D. G. Stork, Pattern Classification, Second Edition, New York: John Wiley & Sons, Inc., 2001. |
| [2] | F. Roli, “School on the Analysis of Patterns: Mini Tutorial Three Hours on Multiple Classifier Systems,” 2009. [Online]. Available: http://www.analysis-of-patterns.net/files/MCS-Part1.pdf. [Accessed 1 January 2015]. |
| [3] | G. Ou and Y. L. Murphey, “Multi-class pattern classification using neural networks,” *Pattern Recognition,* vol. 40, no. 1, pp. 4-18, 2007. |
| [4] | P. R. Zanibbi, “Bayesian Decision Theory,” 2009. [Online]. Available: http://www.cs.rit.edu/~rlaz/prec20092/slides/BayesianDecisionTheory.pdf. [Accessed 26 February 2015]. |
| [5] | V. J. Easton and J. H. McColl, “STEPS Statistics Glossary v1.1,” The University of Glasgow: STatistical Education through Problem Solving, 2004. [Online]. Available: http://www.stats.gla.ac.uk/steps/glossary/probability.html. [Accessed 26 February 2015]. |

1. Example of a *‘perception’* may be: distance, reflectivity, loudness, etc. [↑](#footnote-ref-1)