Introduction

Table of Contents

[Cost of Misclassifications 4](#_Toc138881574)

[Multiple Features 4](#_Toc138881575)

[Complex Models 5](#_Toc138881576)

[Statistical and Syntactical Pattern Recognition 7](#_Toc138881577)

[Problems in Pattern Recognition 7](#_Toc138881578)

[Learning and Adoption 9](#_Toc138881579)

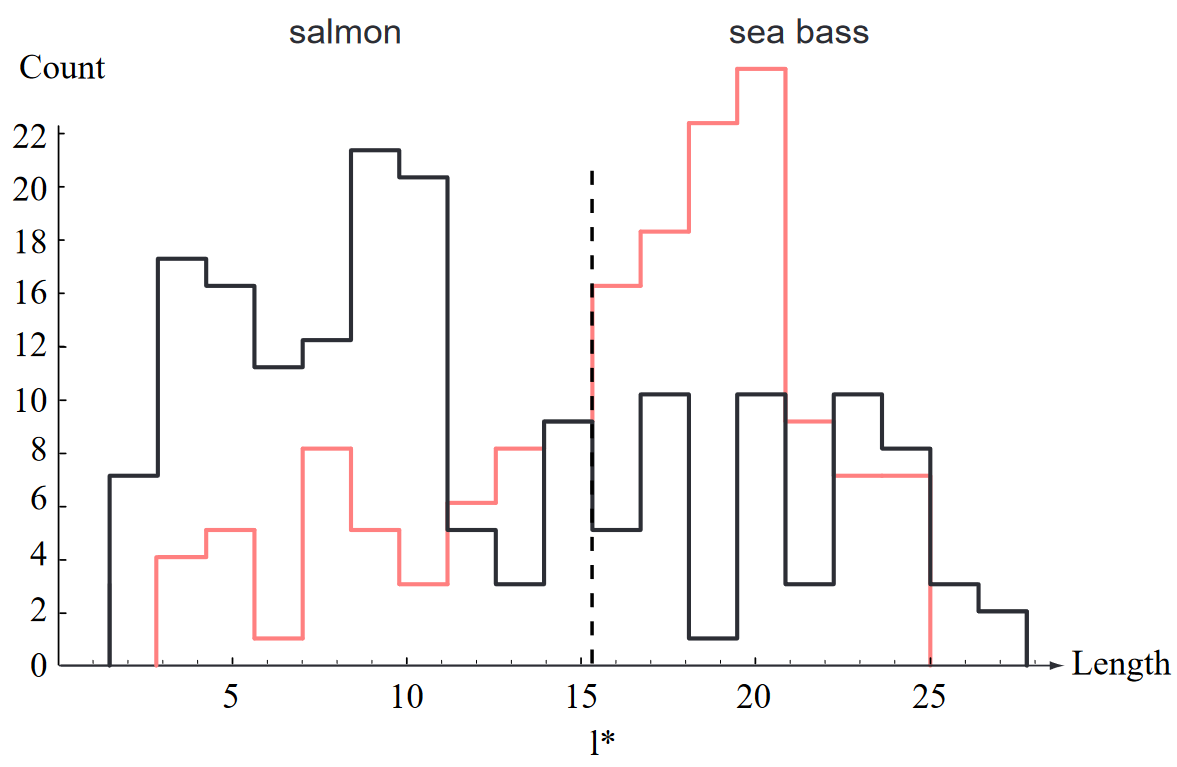
[Components of a Pattern Recognition System 11](#_Toc138881580)

**Pattern Recognition** is the process of taking raw data as input, performing some set of actions, and finding **hidden secrets** that achieve a specific goal. This is essentially a vague description of literally every machine learning model ever made.

To understand how a pattern recognition system works, consider an example. Suppose we have a system that separates two types of fish, salmons and seabass, from a conveyor belt using an optical sensor to identify the type. To be certain about which fish is of which type, we need to identify **features**. Some physical features which will be visible include the length, width, weight, number of fins, position of the mouth and so on.

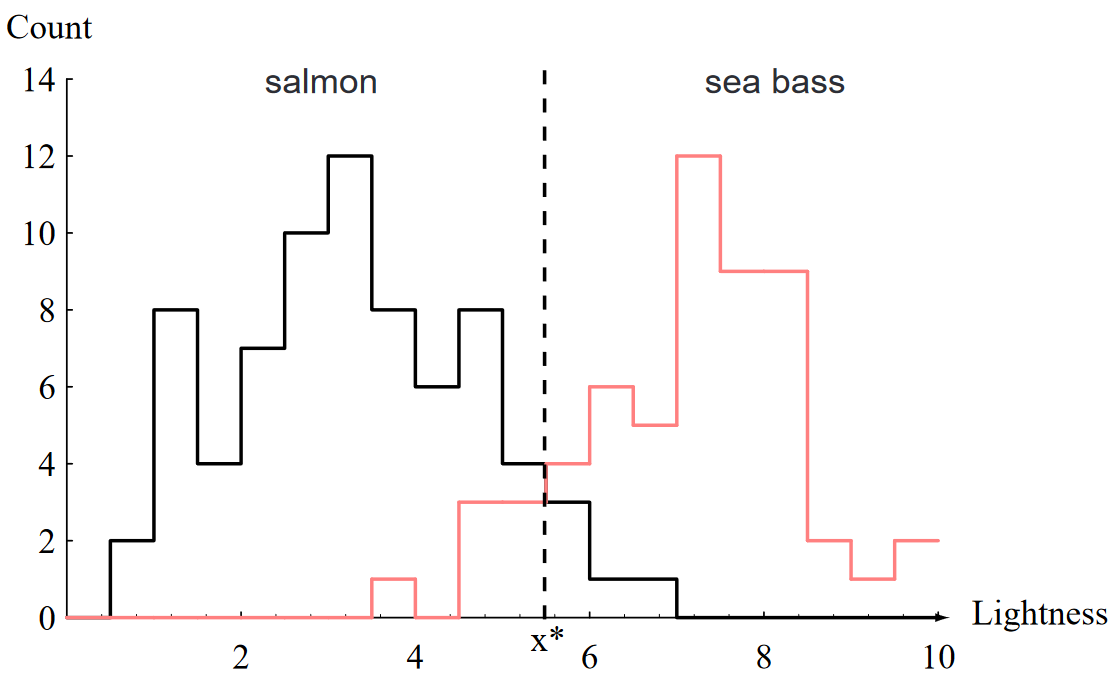
Once we sense the input (which is the equivalent of taking a picture in this case), we also might need to perform some **preprocessing** to remove noise and **segmentation** to separate the fish from the background before we can finally extract the features. These features will be used by the **classifier**. All the features are not equally important. We need to find the **model** that does the best job of identifying the fish using the features. A model is a **mathematical descriptor** being used to make predictions.

Suppose we choose **length** as a feature. The graph below shows the distribution of lengths for samples of the two types of fish.



To be able to separate the fish based on their length, we need to specify a **decision boundary** (). Unfortunately, as can be seen in the graph above, there is no value of length at which we can place the decision boundary to get a clean division. There is significant overlap between the lengths of both types of fish.

Let’s try another feature: weight (lightness). The distribution for this feature is as follows:



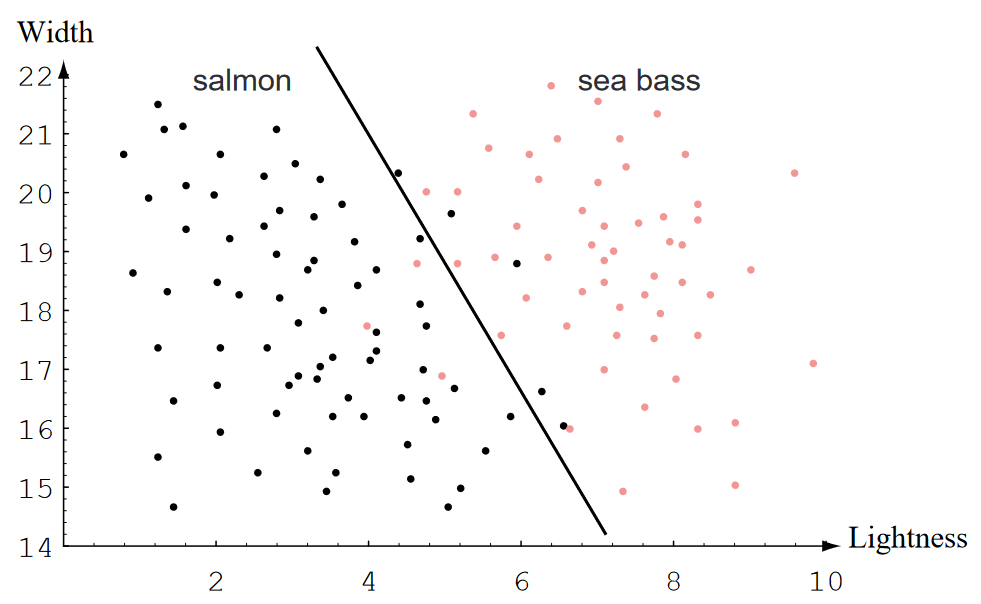
This is a better situation than the last one, since there is less overlap. This means that weight is a better feature choice in this case. Still, we are unable to correctly classify all of the fish.

## Cost of Misclassifications

It is important to understand once concept here: the **cost of misclassifications**. The reason we need to avoid misclassifications to the best of our abilities is because there are sometimes huge costs associated with it. In addition, there are cases where it is better to make one type of misclassification than the other. In this example, salmon is a significantly more expensive fish than seabass, so if we misclassify more of the salmon (by moving to the left), we will be losing money. On the other hand, if we misclassify more of the seabass (by moving to the right), customers will end up with seabass even though they paid for salmon, which is bad for our reputation. These two factors are part of the ‘cost’ of misclassifications. Which of the two is more worthwhile to avoid is a business decision. Since it is not possible to guarantee 0 misclassifications, we sometimes need to take into consideration external factors such as these to determine how to tune our model and reduce the ‘cost’ of misclassifications.

## Multiple Features

Since neither of the features we chose is doing that well, we can try another approach: using **multiple features**. If we want to visualize this graphically and draw a decision boundary, for the two features width and weight, we have a 2D graph. Each fish is represented by a single point, , in the 2D space.

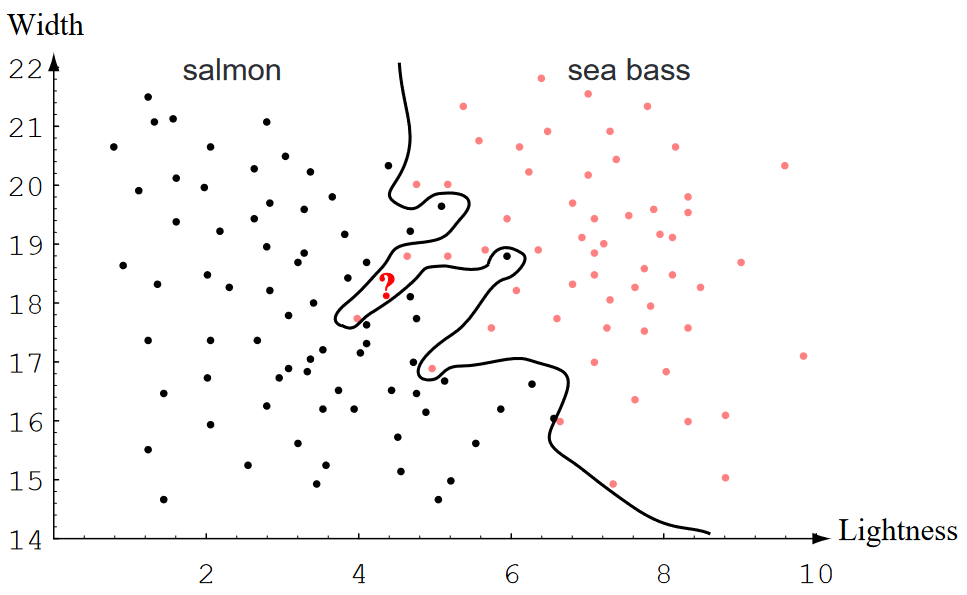


The combination of the two features allows us to draw a decision boundary that results in far fewer misclassifications.

If we decide to use multiple features, we still need to be careful about which features we are using. Features that do not contribute to solving the problem, or ones which are a combination of other features, should be avoided. They provide no additional benefit but increase the **computation cost**.

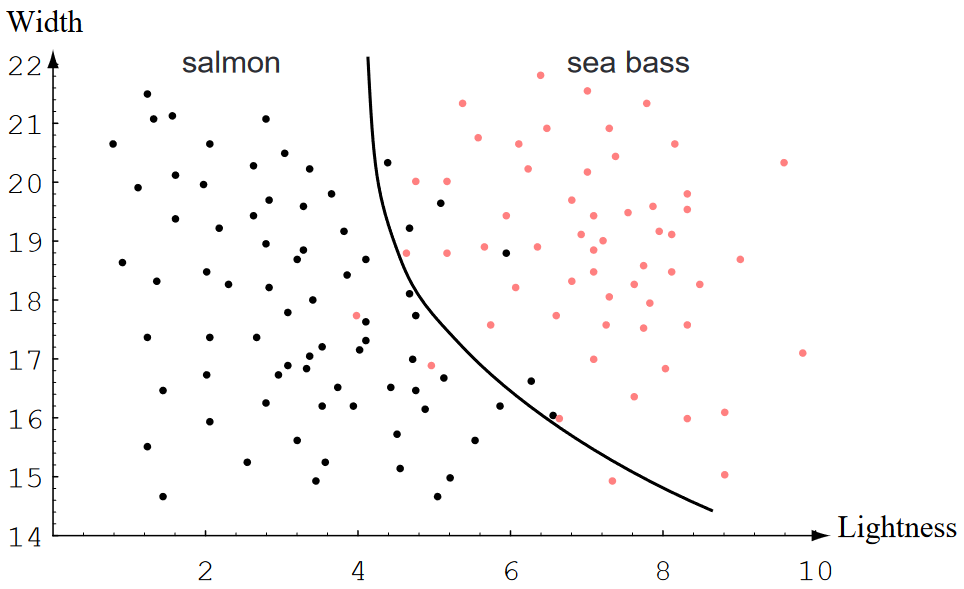
## Complex Models

If we choose to use a **non-linear decision boundary**, we can actually create a boundary that fits the data perfectly and results in no misclassifications.



However, models that do this are said to **overfit** the training data. The point of finding the decision boundary is not to achieve 100% accuracy during training, but to find a boundary that does well during the prediction stage. A model that is overfitting is paying too much attention to noise and fails to **generalize** to real life data. For example, the point marked with a ‘?’ on the graph above is most likely a salmon, but the model thinks it is a seabass.

Instead of trying to find an exact fit, we should try to find a decision boundary that does generally well. A non-linear decision boundary which has better results than the linear one we saw above could look like this:



## Statistical and Syntactical Pattern Recognition

The field of pattern recognition can be broadly divided into two categories, Statistical Pattern Recognition and Syntactical Pattern Recognition.

**Statistical Pattern Recognition** focuses on the statistical properties of patterns, generally expressed as probability densities. Neural networks fall into this category.

**Syntactical Pattern Recognition**, also called Structural Pattern Recognition, has models which consist of crisp logical rules and grammars describing the decisions, e.g., classifying sentences as grammatically correct or not.

## Problems in Pattern Recognition

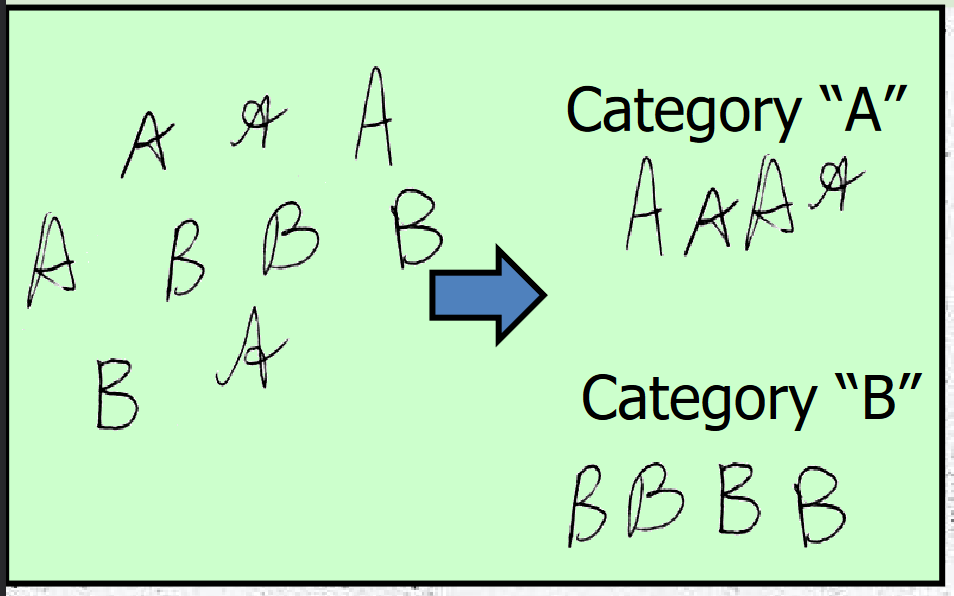
There are a variety of issues we will have to deal with when working in pattern recognition. These include:

* Feature Selection
  + Features vary from one domain to another and must be selected individually for each domain.
  + It is difficult to figure out which features are the most promising.
  + The number of features to use must be decided.
  + Is it possible to automate feature selection? (Yes.)
* Noise
  + Lighting issues in images such as shadows
  + Shaky and blurry images
  + Accidental and incorrect data read by sensors
* Intra-Class Variability, e.g., the same alphabet written in block letters and in cursive.
* Inter-Class Variability, e.g., separate characters that look the same, like ‘2’ and ‘Z’.
* Overfitting
* Model Selection
  + How do we decide when to reject a model and try for a better one?
* Prior Knowledge
  + Having prior knowledge about the domain can help with feature selection, e.g., knowing that two types of fish have significantly different weights provides an indication that weight would be a good feature to use.
* Missing Features
  + What if a sample has essential features missing (e.g., unable to determine the length of a fish due to occlusion)?
* Mereology
  + When reading the word ‘beats’ we combine the alphabets and read it as one, not as ‘be’, ‘beat’, ‘eat’, ‘eats’ or ‘at’, even though those are valid words as well. In a large body of text, how do we figure out how to group the words? More generally, how do we decide on the best categorization of subsets of data.
* Segmentation – We might need to segment out the region of interest (ROI) from the total input.
* Context
  + Using information from the surrounds can help with the task at hand.
  + In a long series of salmon fish, it is very likely that the next fish is also a salmon.
  + The word ‘jeetyet’ was misheard due to noise and could refer to one of multiple phrases. However, knowing the conversation was taking place at a food court clues us in to reading it as ‘Did you eat yet?’.
* Invariances
  + Good features are invariant to scale, rotation, translation, size, perhaps the rate of a pattern (the same sentence said fast and slow).
  + Good features should not be susceptible to deformation, occlusion or illumination changes.
* Evidence Pooling
  + The opinion of two experts is more important than the opinions of ten random people. This is the concept used behind ‘super classifiers’.
* Costs and Risks
  + The cost of misclassification should be taken into account and the model fine-tuned accordingly.
* Computational Complexity

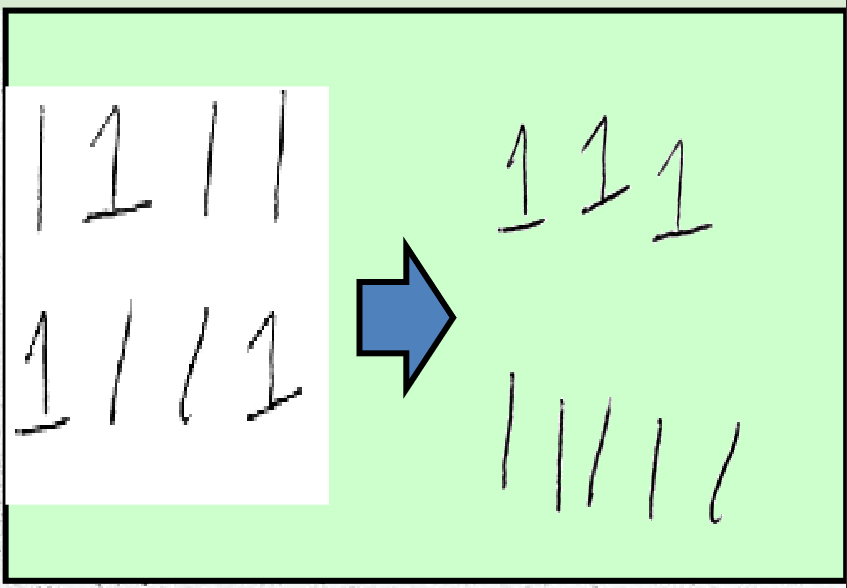
## Learning and Adoption

Any method that incorporates **information from training samples** in the design of the classifier employs **learning**. There are mainly three types of learning, supervised, unsupervised and reinforcement.

In **supervised learning** the category for each sample is provided during the training stage and the model attempts to learn the pattern in the samples that result in them being in a particular category. Supervised learning is used for things like classification problems.



In **unsupervised learning** the model attempts to form clusters of data based on patterns it identifies in the samples. There are no provided categories for the data.



In **reinforcement learning**, the model is only told if it correctly achieved the outcome or not. It is not provided with any information about how close it got. For just two classes, this becomes similar to supervised learning.

## Components of a Pattern Recognition System



The diagram above shows a complete overview of the different parts of a pattern recognition system that have been discussed here.