Machine Learning Reading Group

Pattern Classification

Richard O. Duda, Peter E. Hart, David G. Stork

Chapter 1 - Introduction

University “Transilvania”
Faculty of Mathematics and Informatics
Brasov
31.Oct.2018

Kerestély Árpád
k_arpi2004@yahoo.co.uk

What is pattern classification

- A range of problems we (humans) solve with ease and machines can’t do so easily
    - recognize a face
    - understand spoken words
    - read handwritten characters
    - identify our car keys in our pocket by feel
    - decide whether an apple is ripe by its smell
- the act of taking in raw data and taking an action based on the “category” of the pattern

Is pattern classification necessary? Why?

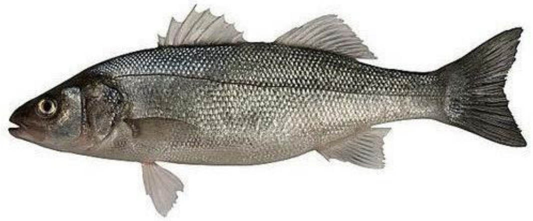
- Machines could do task such as:
  - automated speech recognition
  - fingerprint identification
  - optical character recognition
  - DNA sequence identification
  - etc.
- By solving many of these problems, we gain deeper understanding and appreciation for pattern recognition systems in the natural world - most particularly in humans.

Aim of this chapter

- Provide a high level overview of:
  - what is a pattern classification problem by providing a concrete example
  - what problems arise and should be considered when doing pattern classification
  - what different categories of machine learning algorithms exist

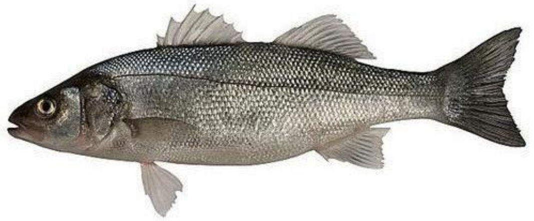
Fish classification – the concrete example

- Consider the following imaginary and somewhat fanciful example:
- Suppose that a fish packing plant wants to automate the process of sorting incoming fish on a conveyor belt according to species. As a pilot project it is decided to try to separate sea bass from salmon using optical sensing.
- We set up a camera, take some sample images and begin to note some physical differences between the two types offish — length, lightness, width, number and shape of fins, position of the mouth, and so on - these suggest ***features*** to explore for use in our classifier.

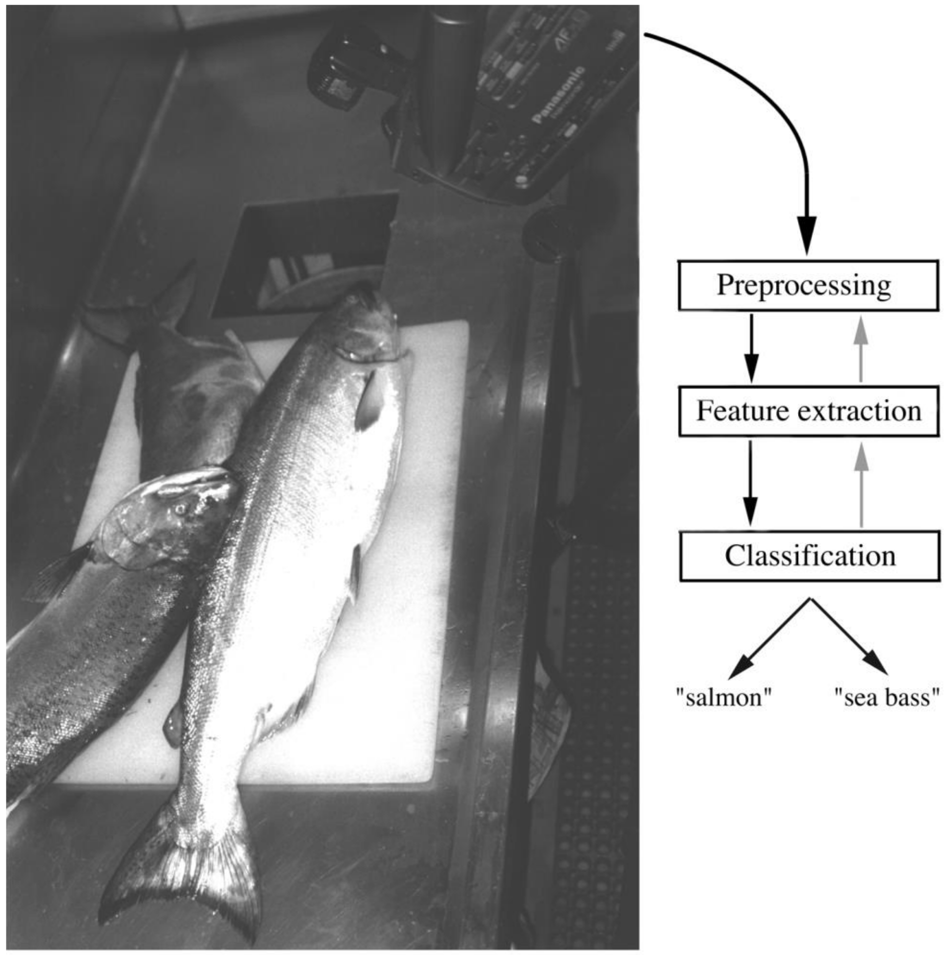


Fish classification – the concrete example

- Given that there truly are differences between the population of sea bass and that model of salmon, we view them as having different ***models*** — different descriptions, which are typically mathematical in form.
- First the camera captures an image of the fish. Next, the camera's signals are ***preprocessed*** to simplify subsequent operations without loosing relevant processing information.
- We might use a *segmentation* operation in which the images of different fish are somehow isolated from one another and from the background.



Fish classification – the concrete example



- The information from a single fish is then sent to a ***feature extractor***, whose purpose is to reduce the data by measuring certain “features” or “properties.”
- These features (or, more precisely, the values of these features) are then passed to a ***classifier*** that evaluates the evidence presented and makes a final decision as to the species.

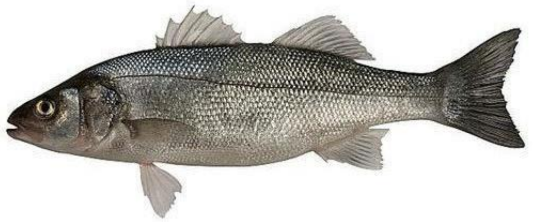
Goal of a classifier

- The overarching goal and approach in pattern classification is to:
  - hypothesize the class of these models,
  - process the sensed data to eliminate noise (not due to the models)
  - for any sensed pattern choose the model that corresponds best.

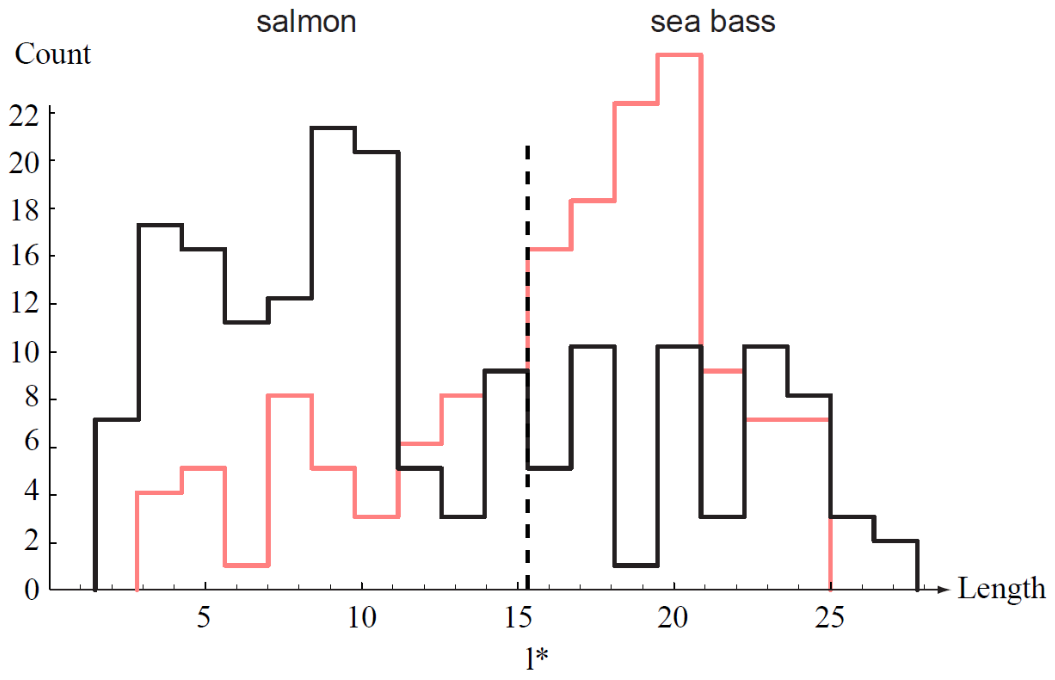
- Any techniques that further this aim should be in the conceptual toolbox of the designer of pattern recognition systems.

Designing a feature extractor and a classifier

- Suppose somebody at the fish plant tells us that a sea bass is generally longer than a salmon (***domain knowledge***).
- These, then, give us our tentative *models* for the fish: sea bass have some typical length, and this is greater than that for salmon.
- Then length becomes an obvious *feature*, and we might attempt to classify the fish merely by seeing whether or not the length *L* of a fish exceeds some critical value *L*.*

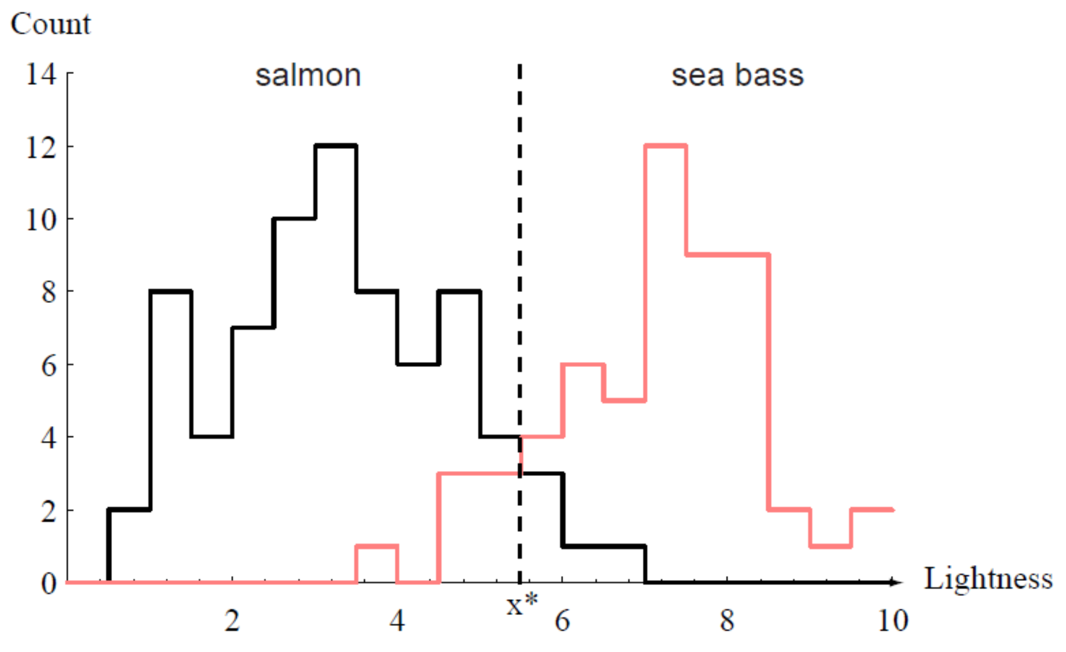


Designing a feature extractor and a classifier



- Suppose that we do this, and obtain the histograms shown on the left.
- The disappointing histograms bear out the statement that sea bass are somewhat longer than salmon, on average, but it is clear that this single criterion is quite poor; no matter how we choose $L*$, we cannot reliably separate sea bass from salmon by length alone.

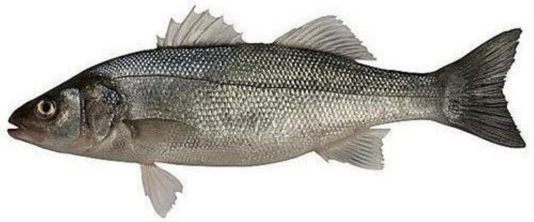
Designing a feature extractor and a classifier



- The same is true when we take the average lightness of the fish scales
- No single threshold value *x** (decision boundary) will serve to unambiguously discriminate between the two categories
- using lightness alone, we will have some errors, although we see a clearer separation

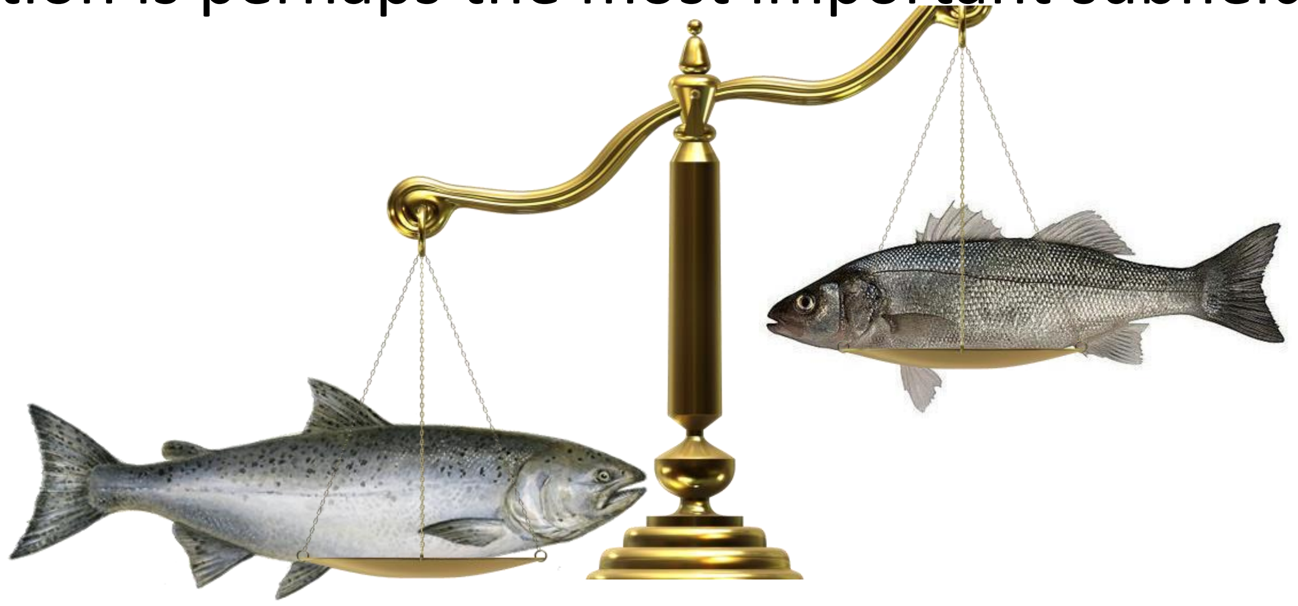
What is the cost of mistakes?

- So far we have tacitly assumed that the consequences of our actions are equally costly: deciding the fish was a sea bass when in fact it was a salmon was just as undesirable as the converse. Such a symmetry in the ***cost*** is often, but not invariably the case.
- As a fish packing company we may know that our customers easily accept occasional pieces of tasty salmon in their cans labeled “sea bass,” but they object vigorously if a piece of sea bass appears in their cans labeled “salmon.” If we want to stay in business, we should adjust our decision boundary to avoid antagonizing our customers, even if it means that more salmon makes its way into the cans of sea bass. In this case, then, we should move our decision boundary *x** to smaller values of lightness, thereby reducing the number of sea bass that are classified as salmon



What is the cost of mistakes?

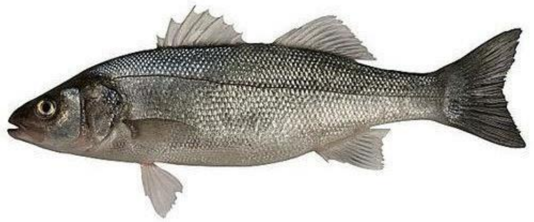
- Such considerations suggest that there is an overall single cost associated with our **decision**, and our true task is to make a **decision rule** (i.e., set a decision boundary) so as to minimize such a **cost**.
- This is the central task of ***decision theory*** of which pattern classification is perhaps the most important subfield.



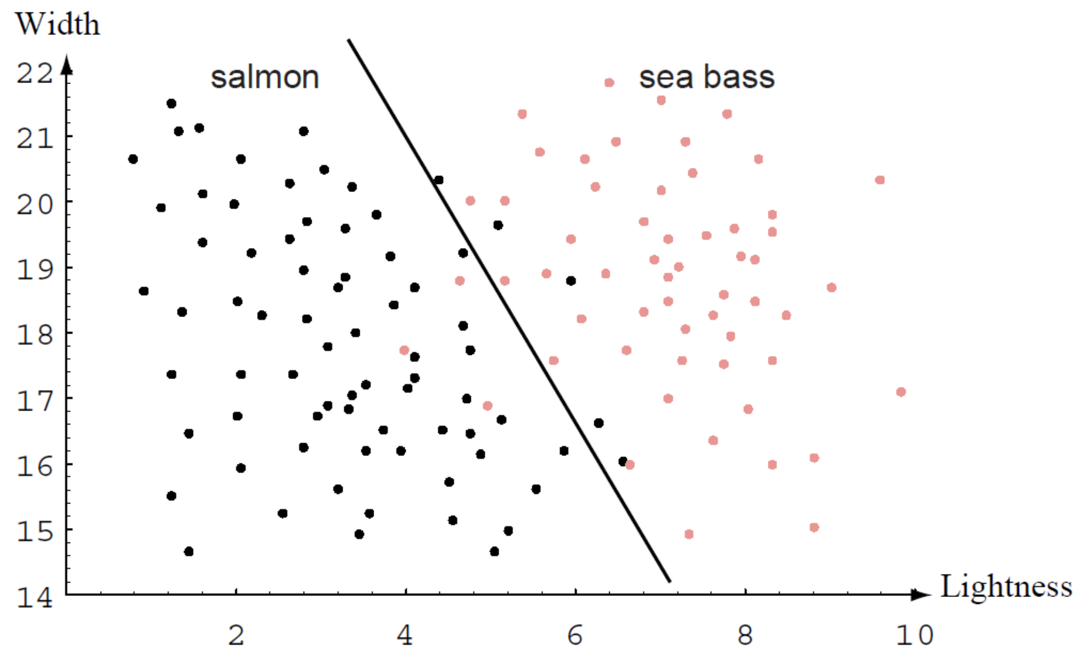
Designing a feature extractor and a classifier

- To improve recognition, then, we must resort to the use of *more* than one feature at a time.
- We keep the lightness feature as it appeared to be a good feature, but also include the width of the fish as the second feature.
- We realize that the feature extractor has thus reduced the image of each fish to a point or *feature vector* **x** in a two-dimensional *feature space, where:*

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$



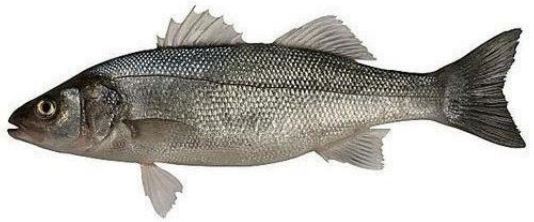
Designing a feature extractor and a classifier



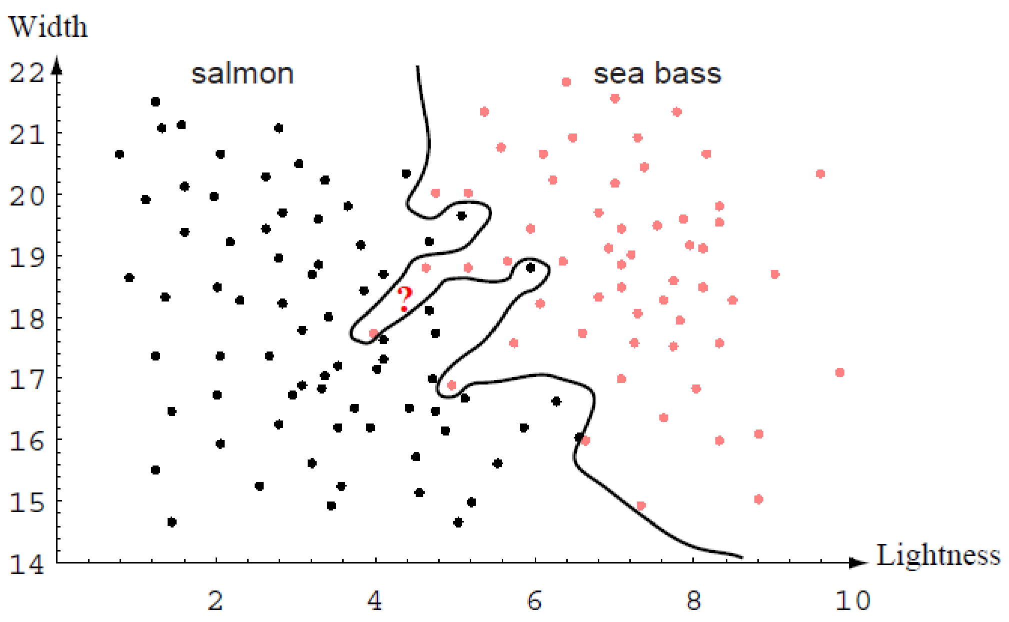
- Our problem now is to partition the feature space (2D) into two regions, where for all patterns in one region we will call the fish a sea bass, and all points in the other we call it a salmon.
- This plot suggests the following rule for separating the fish: Classify the fish as sea bass if its feature vector falls above the ***decision boundary*** shown, and as salmon otherwise.

More features, better classifier?

- This rule appears to do a good job of separating our samples and suggests that boundary perhaps incorporating yet more features would be desirable.
- But, how do we know beforehand which of these features will work best? Some features might be redundant: for instance if the eye color of all fish correlated perfectly with width, then classification performance need not be improved if we also include eye color as a feature.
- Even if the difficulty or computational cost in attaining more features is of no concern, might we ever have *too many* features?

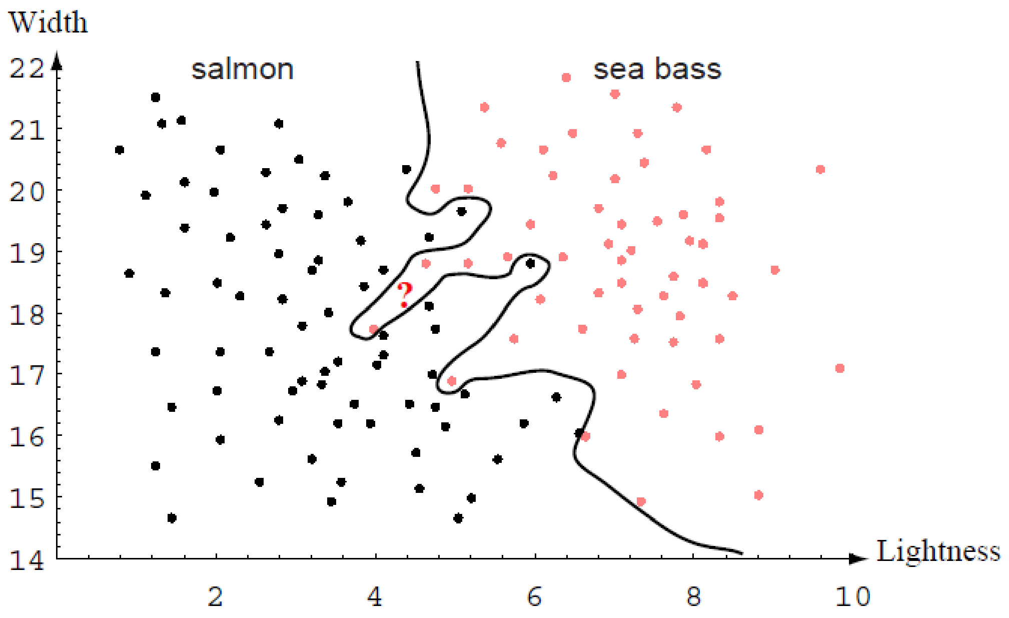


Designing a feature extractor and a classifier



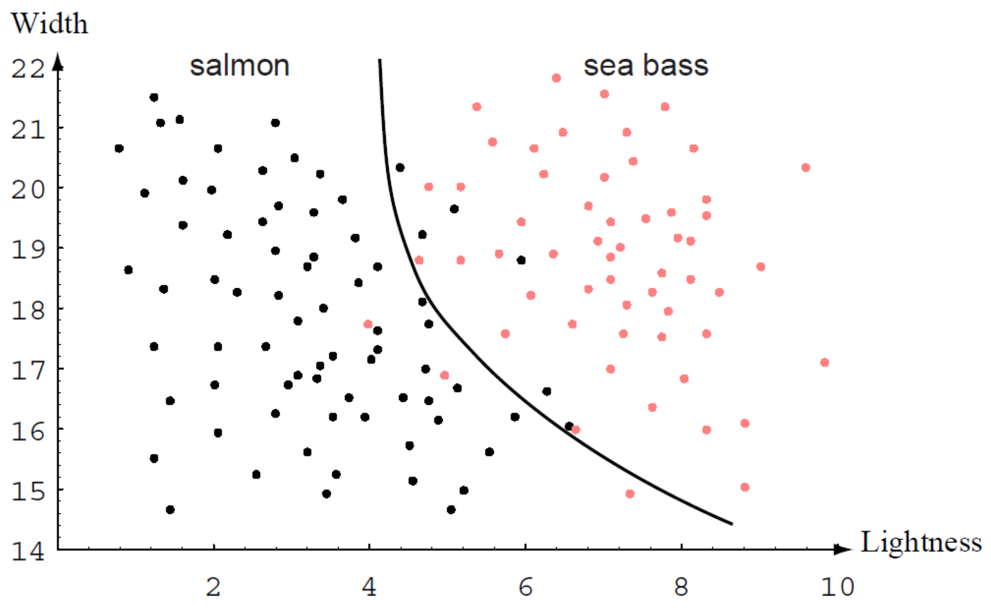
- If our models were extremely complicated, our classifier would have a decision boundary more complex than the simple straight line. In that case all the training patterns would be separated perfectly.
- With such a “solution,” though, our satisfaction would be premature because the central aim of designing a classifier is to suggest actions when presented with *novel* patterns, i.e. fish not yet seen. This is the issue of ***generalization***.
- It is unlikely that the complex decision boundary would provide good generalization, since it seems to be “tuned” to the particular training samples.

Designing a feature extractor and a classifier



- Naturally, one approach would be to get more training samples for obtaining a better estimate of the true underlying characteristics, for instance the probability distributions of the categories. In most pattern recognition problems, however, the amount of such data we can obtain easily is often quite limited. Even with a vast amount of training data in a continuous feature space though, if we followed the approach from the figure on the left, our classifier would give a horrendously complicated decision boundary — one that would be unlikely to do well on novel patterns.

Designing a feature extractor and a classifier



- We might seek to "simplify" the recognizer, motivated by a belief that the underlying models will not require a decision boundary that is so complex.
- We might be satisfied with the slightly poorer performance on the training samples if it means that our classifier will have better performance on novel patterns.
- But if designing a very complex recognizer is unlikely to give good generalization, precisely how should we quantify and favor simpler classifiers?

An all-knowing classifier?

- This makes it quite clear that our decisions are fundamentally task or cost specific, and that creating a ***single general purpose*** artificial pattern recognition device — i.e., one capable of acting accurately based on a wide variety of tasks — is a profoundly difficult challenge.
- This, too, should give us added appreciation of the ability of humans to switch rapidly and fluidly between pattern recognition tasks.

Statistical Pattern Recognition

- Since classification is, at base, the task of recovering the model that generated the patterns, different classification techniques are useful depending on the type of candidate models themselves.
- In ***statistical pattern recognition*** we focus on the statistical properties of the patterns (generally expressed in probability densities).
- Occasionally it is claimed that *neural* pattern recognition (or neural network pattern classification) should be considered its own discipline, but despite its somewhat different intellectual pedigree, we will consider it a close descendant of statistical patter recognition, for reasons that will become clear later in the book.

The Sub-problems of Pattern Classification

- Feature Extraction:
  - an ideal feature extractor would yield a representation that makes the job often classifier trivial
  - conversely, an omnipotent classifier would not need the help of a sophisticated feature extractor
  - the task of feature extraction is much more problem and domain dependent than is classification proper, and thus requires knowledge of the domain
  - a good feature extractor for sorting fish would surely be of little use for identifying fingerprints, or classifying photomicrographs of blood cells

The Sub-problems of Pattern Classification

- Noise
  - The lighting of the fish may vary, there could be shadows cast by neighboring equipment, the conveyor belt might shake — all reducing the reliability of the feature values actually measured.
  - We define *noise* very general terms: any property of the sensed pattern due not to the true underlying model but instead to randomness in the world or the sensors.
- Overfitting
  - In our fish classification problem, we were, implicitly, using a more complex model of sea bass and of salmon. That is, we were adjusting the complexity of our classifier. While an overly complex model may allow perfect classification of the training samples, it is unlikely to give good classification of novel patterns — a situation known as *overfitting*.

The Sub-problems of Pattern Classification

- Model Selection
  - We might have been unsatisfied with the performance of our fish classifier and thus jumped to an entirely different class of model, for instance one based on some function of the number and position of the fins, the color of the eyes, the weight, shape of the mouth, and so on.
  - How do we know when a hypothesized model differs significantly from the true model underlying our patterns, and thus a new model is needed? In short, how are we to know to reject a class of models and try another one? Are we as designers reduced to random and tedious trial and error in model selection, never really knowing whether we can expect improved performance? Or might there be principled methods for knowing when to jettison one class of models and invoke another? Can we automate the process?

The Sub-problems of Pattern Classification

- Prior Knowledge
  - How to incorporate such information?
- Missing Features
  - How to handle?
- Mereology
  - This is the problem of *subsets and supersets* — formally part of mereology, the study of part/whole relationships.
  - It appears as though the best classifiers try to incorporate as much of the input into the categorization as “makes sense,” but not too much. How can this be done?
- Segmentation
  - In image or word processing.

The Sub-problems of Pattern Classification

- Context
  - input-dependent information other than from the target pattern itself
  - For instance, it might be known for our fish packing plant that if we are getting a sequence of salmon, that it is highly likely that the next fish will be a salmon (since it probably comes from a boat that just returned from a fishing area rich in salmon). Thus, if after a long series of salmon our recognizer detects an ambiguous pattern (i.e., one very close to the nominal decision boundary), it may nevertheless be best to categorize it too as a salmon.
- Invariances
  - In our fish example, the absolute position on the conveyor belt is irrelevant to the category and thus our representation should also be insensitive to absolute position of the fish. Here we seek a representation that is invariant to the transformation of *translation.*
- Evidence Pooling
  - We might imagine that we could do better if we had several component *classifiers*. If these categorizers agree on a particular pattern, there is no difficulty. But suppose they disagree. How should a “super” classifier *pool the evidence* from the component recognizers to achieve the best decision?
- Costs and Risks
  - Conceptually, the simplest such risk is the classification error: what percentage of new patterns are called the wrong category.
- Computational Complexity
  - Some pattern recognition problems can be solved using algorithms that are highly impractical, thus the computational complexity of different algorithms is of importance, especially for practical applications.

Learning and Adaptation

- Supervised Learning
  - In supervised learning, a teacher provides a category label or cost for each pattern in a training set, and we seek to reduce the sum ofthe costs for these patterns.
- Unsupervised Learning
  - In *unsupervised learning* or *clustering* there is no explicit teacher, and the system forms clusters or “natural groupings” of the input patterns.
- Reinforcement Learning
  - In *reinforcement learning* or *learning with a critic*, no desired category signal is given; critic instead, the only teaching feedback is that the tentative category is right or wrong. This is analogous to a critic who merely states that something is right or wrong, but does not say specifically *how* it is wrong.