# CS 584
# Data Mining

Classification 1

# Classification: Definition

- Given a collection of records (*training set* )
  - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Find a *model* for class attribute as a function of the values of other attributes.
- Goal: previously unseen records should be assigned a class as accurately as possible.
  - A *test set* is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

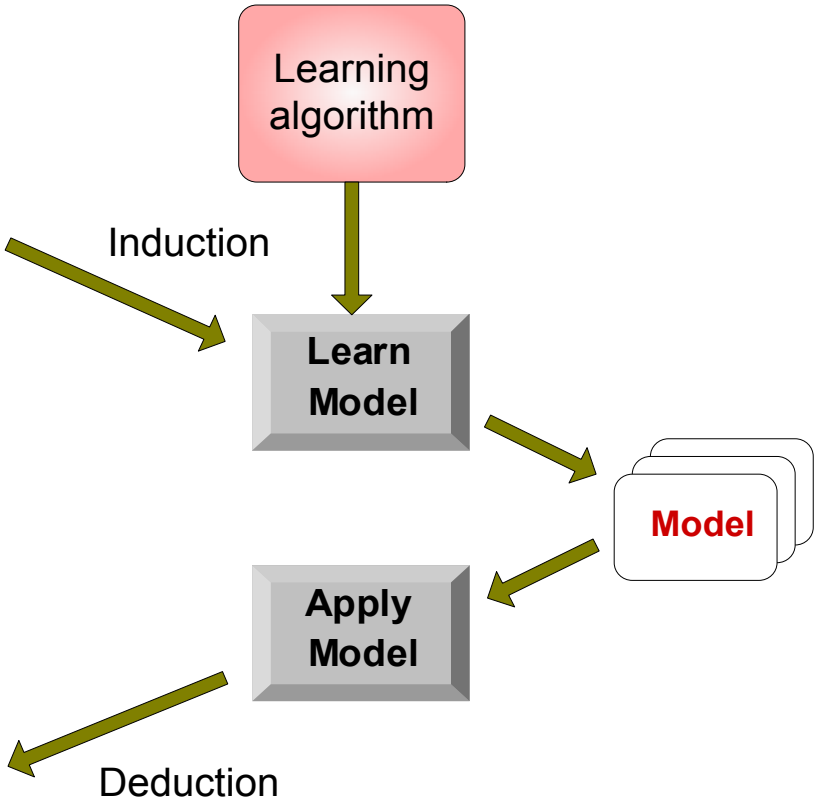
# Illustrating Classification Task

| *Tid* | Attrib1 | Attrib2 | Attrib3 | Class |
|---|---|---|---|---|
| 1 | Yes | Large | 125K | No |
| 2 | No | Medium | 100K | No |
| 3 | No | Small | 70K | No |
| 4 | Yes | Medium | 120K | No |
| 5 | No | Large | 95K | Yes |
| 6 | No | Medium | 60K | No |
| 7 | Yes | Large | 220K | No |
| 8 | No | Small | 85K | Yes |
| 9 | No | Medium | 75K | No |
| 10 | No | Small | 90K | Yes |

Training Set

| *Tid* | Attrib1 | Attrib2 | Attrib3 | Class |
|---|---|---|---|---|
| 11 | No | Small | 55K | ? |
| 12 | Yes | Medium | 80K | ? |
| 13 | Yes | Large | 110K | ? |
| 14 | No | Small | 95K | ? |
| 15 | No | Large | 67K | ? |

Test Set

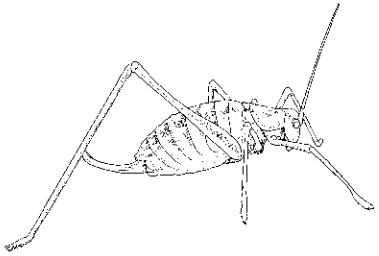


# Examples of Classification Task

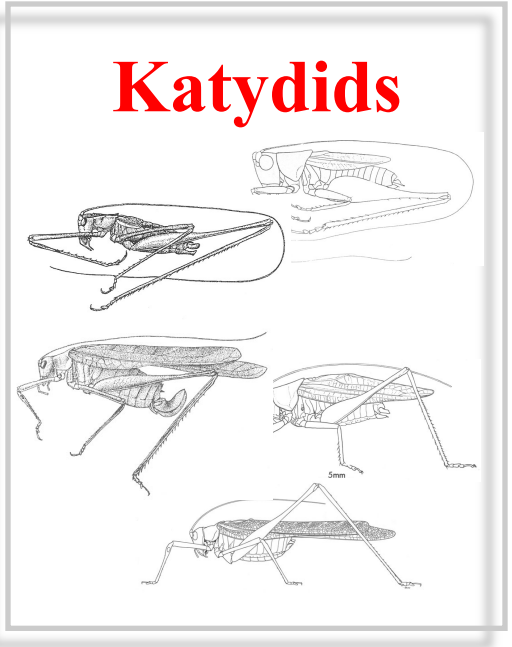
- Predicting tumor cells as benign or malignant
- Classifying credit card transactions as legitimate or fraudulent
- Categorizing news stories as finance, weather, entertainment, sports, etc

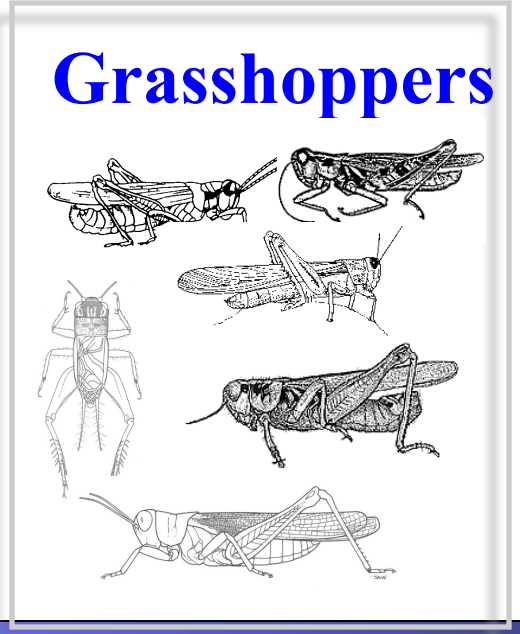
# The Classification Problem
## (informal definition)

Given a collection of annotated data. In this case 5 instances of **Katydids** and five of **Grasshoppers**, decide what type of insect the unlabeled example is.



Katydid or Grasshopper?

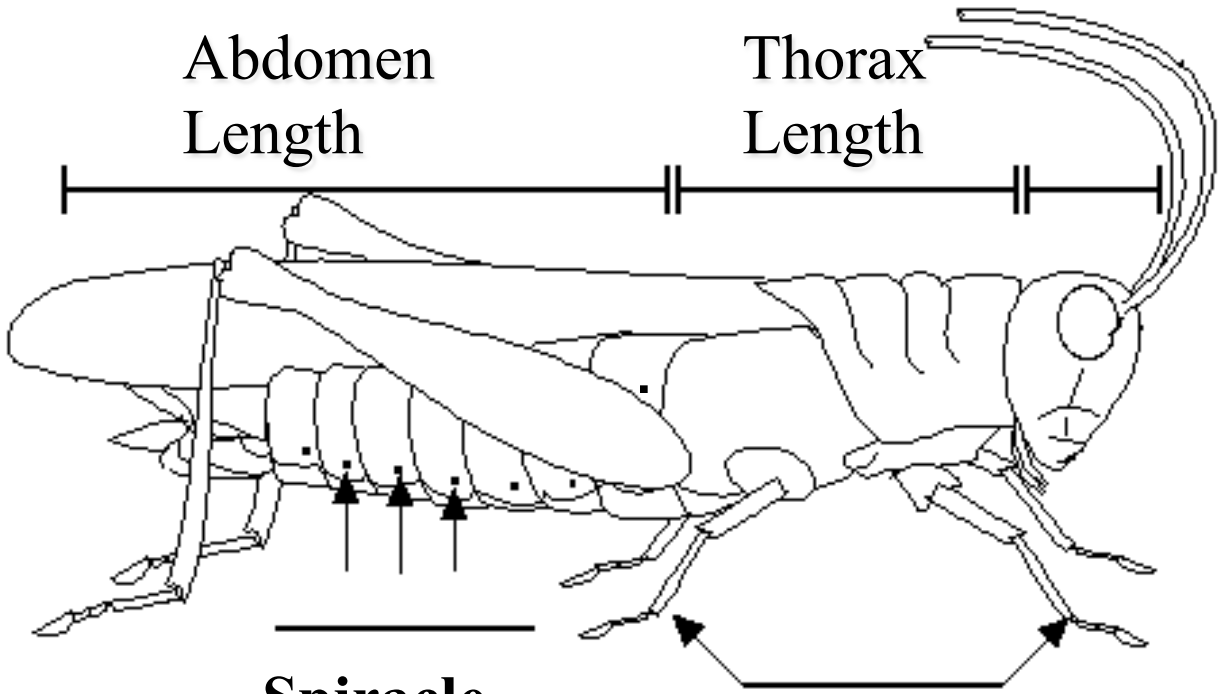


# For any domain of interest, we can measure *features*

Color {Green, Brown, Gray, Other}

Has Wings?



We can store features in a database.

The classification problem can now be expressed as:

Given a training database (My_Collection), predict the class label of a previously unseen instance

My_Collection

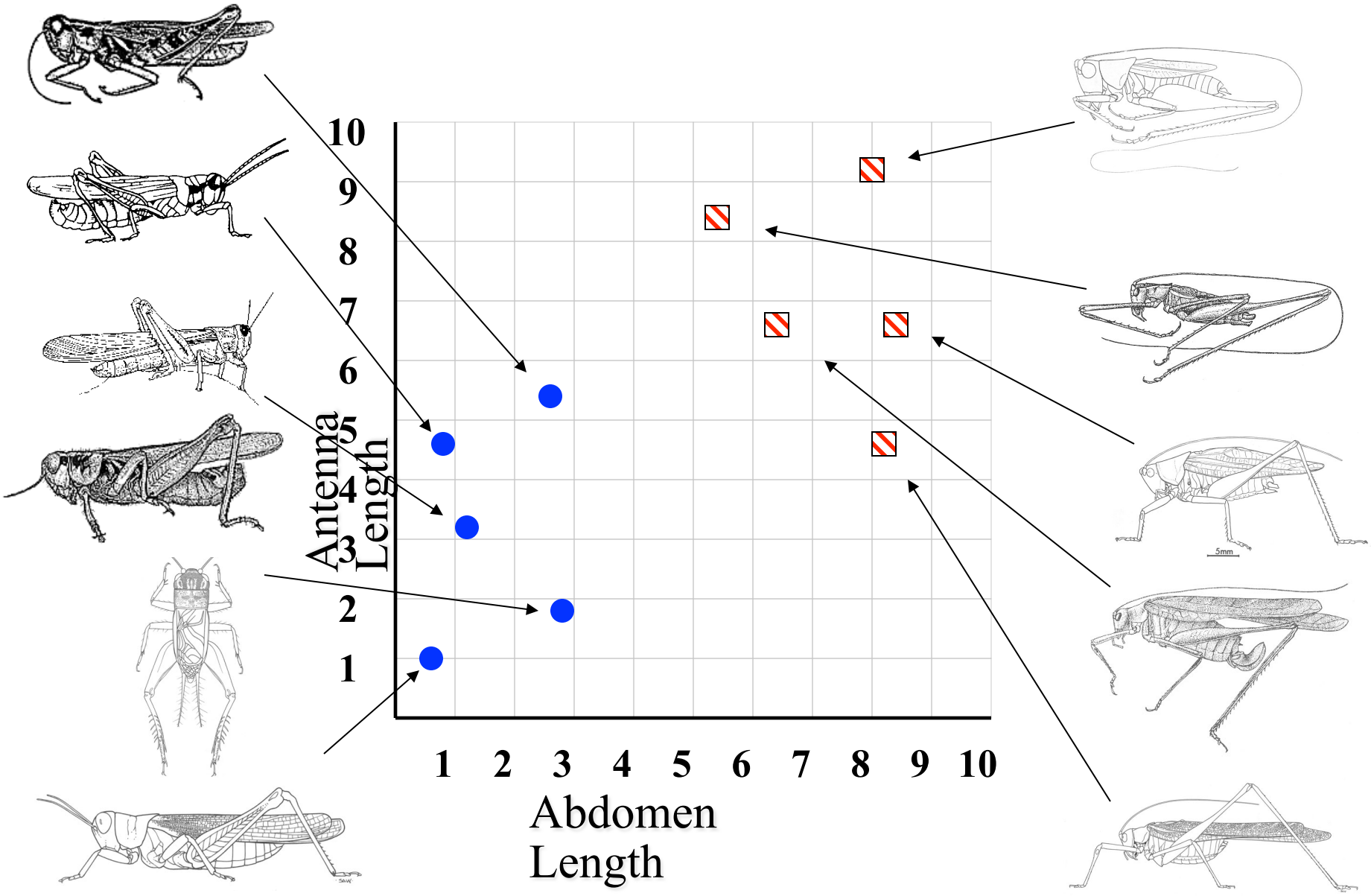
| Insect ID | Abdomen Length | Antennae Length | Insect Class |
|---|---|---|---|
| *1* | 2.7 | 5.5 | **Grasshopper** |
| *2* | 8.0 | 9.1 | **Katydid** |
| *3* | 0.9 | 4.7 | **Grasshopper** |
| *4* | 1.1 | 3.1 | **Grasshopper** |
| *5* | 5.4 | 8.5 | **Katydid** |
| *6* | 2.9 | 1.9 | **Grasshopper** |
| *7* | 6.1 | 6.6 | **Katydid** |
| *8* | 0.5 | 1.0 | **Grasshopper** |
| *9* | 8.3 | 6.6 | **Katydid** |
| *10* | 8.1 | 4.7 | **Katydids** |

previously unseen instance =

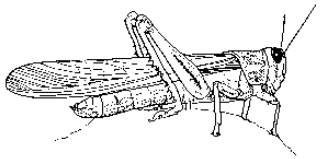
| | | | |
|---|---|---|---|
| *11* | 5.1 | 7.0 | ?????? |

# Grasshoppers

# Katydids

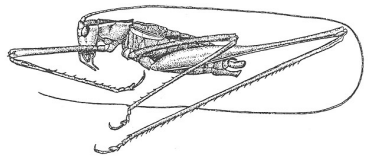


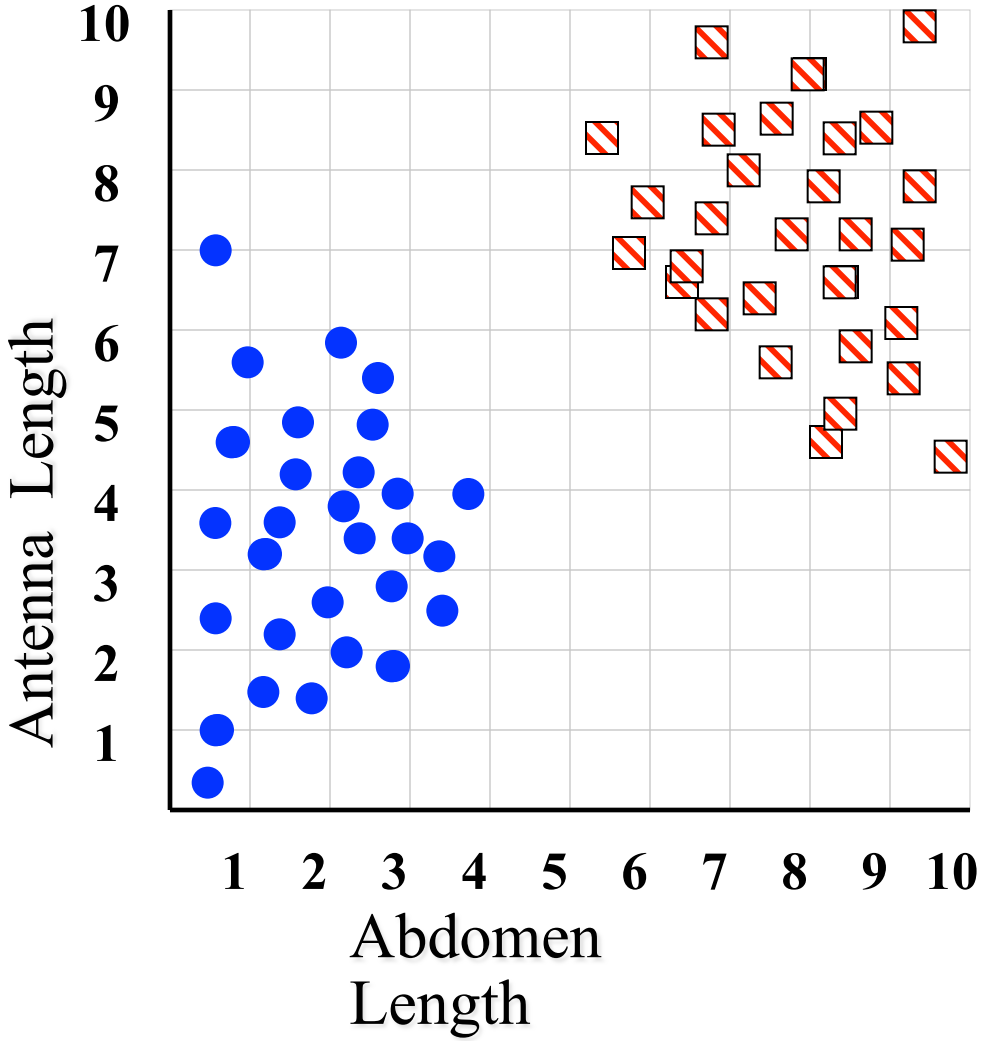
**Grasshoppers**



We will also use this lager dataset as a motivating example…

**Katydids**





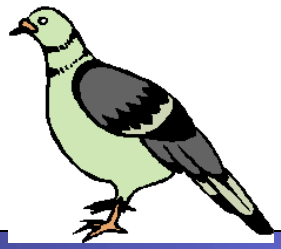

Each of these data objects are called…

- exemplars
- (training) examples
- instances
- tuples

We will return to the previous slide in two minutes. In the meantime, we are going to play a quick game.

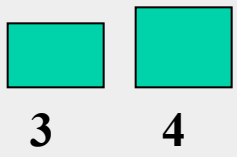
I am going to show you some classification problems which were shown to pigeons!

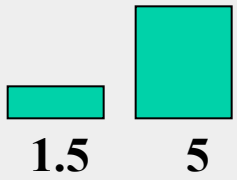
Let us see if you are as smart as a pigeon!

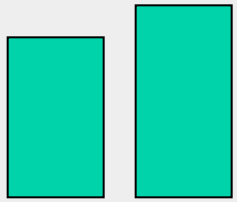


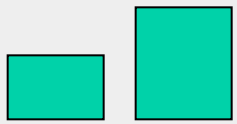
# Pigeon Problem 1

**Examples of class A**

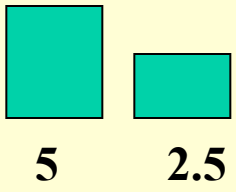


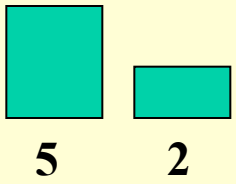



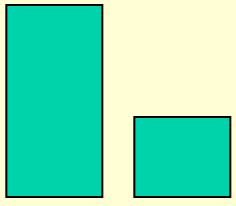



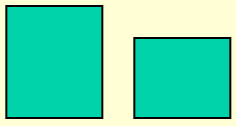




**Examples of class B**



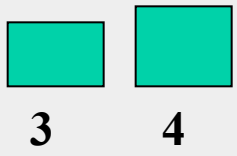



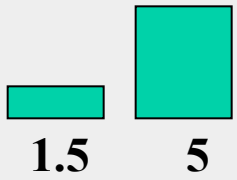



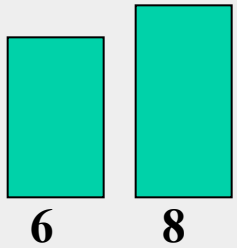



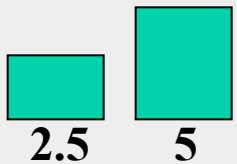
# Pigeon Problem 1

**Examples of class A**

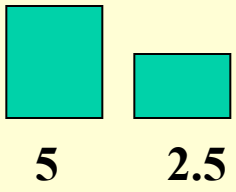


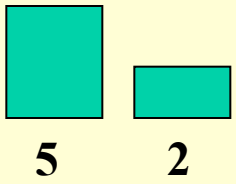



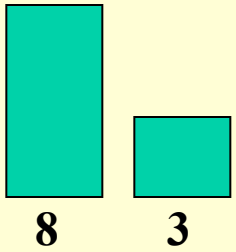



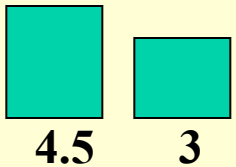



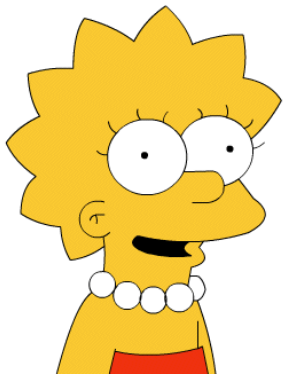

**Examples of class B**



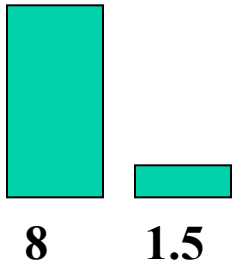



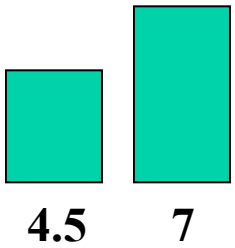



What class is this object?

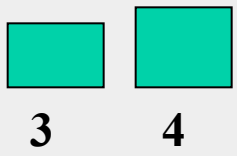


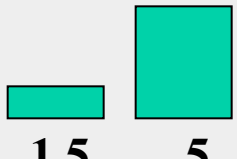

What about this one, A or B?



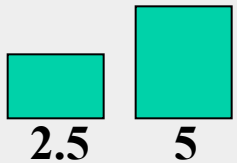
# Pigeon Problem 1

**Examples of class A**

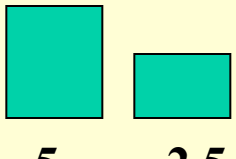


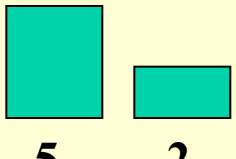



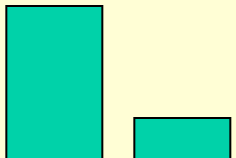


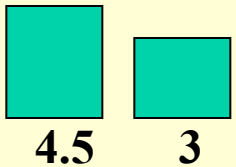



**Examples of class B**

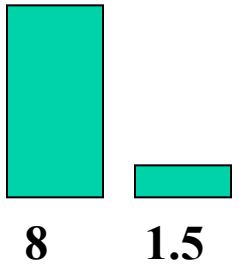


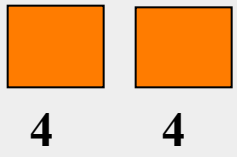

This is a B!

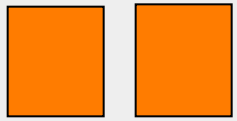


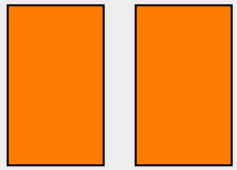

Here is the rule.
If the left bar is smaller than the right bar, it is an A, otherwise it is a B.

# Pigeon Problem 2

**Examples of class A**

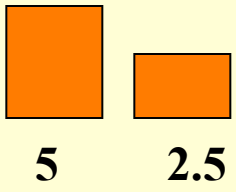


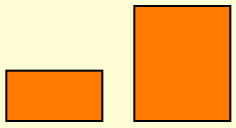



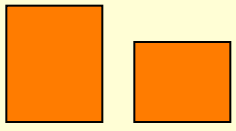



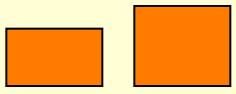


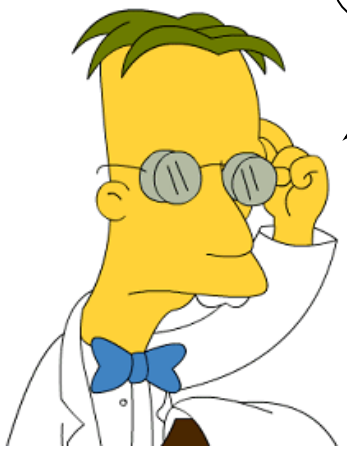
**Examples of class B**

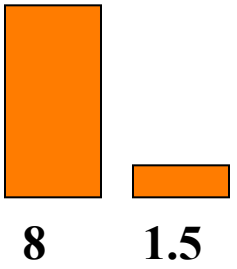


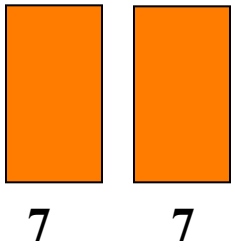



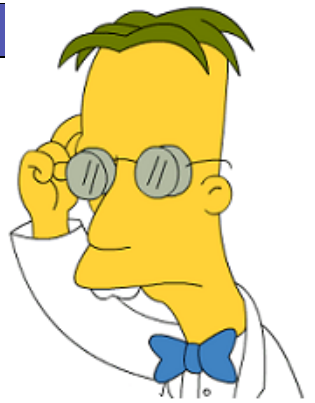



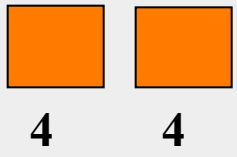


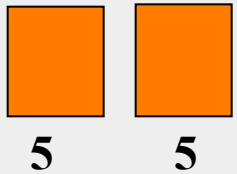


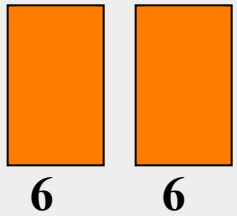
# Pigeon Problem 2



**Examples of class A**

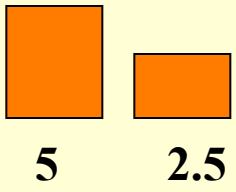


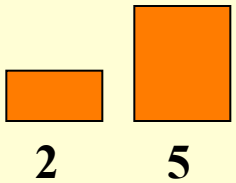



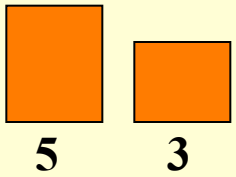



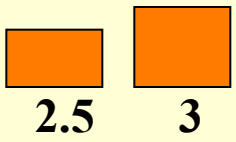


**Examples of class B**



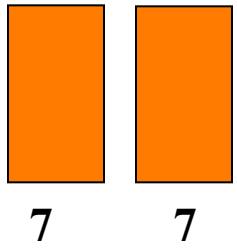



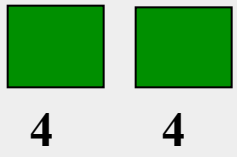

The rule is as follows, if the two bars are equal sizes, it is an A. Otherwise it is a B.

So this one is an A.

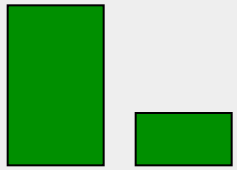


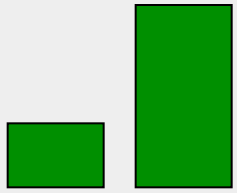
# Pigeon Problem 3

**Examples of class A**

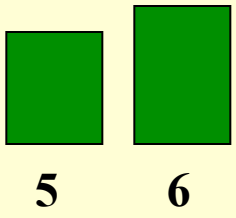


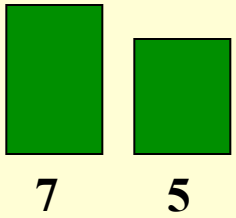


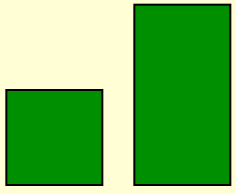


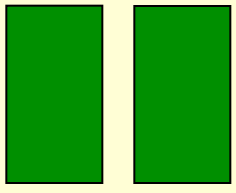



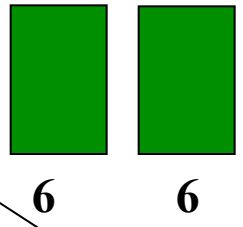

**Examples of class B**



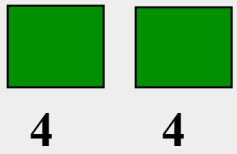



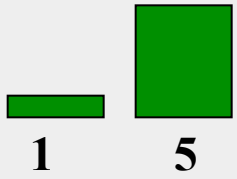



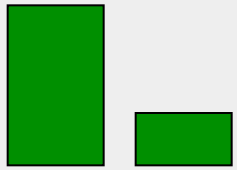

This one is really hard!
What is this, A or B?

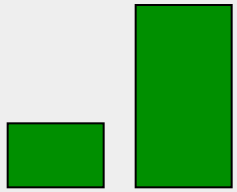
# Pigeon Problem 3

**Examples of class A**

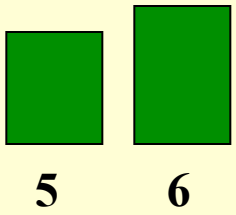


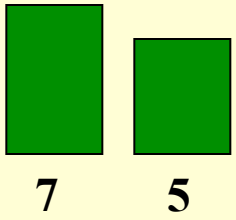



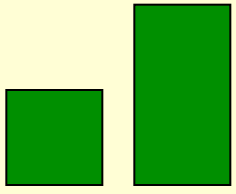



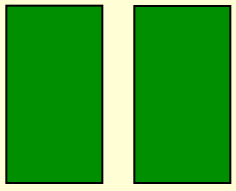




**Examples of class B**

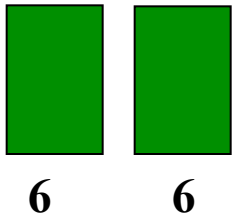


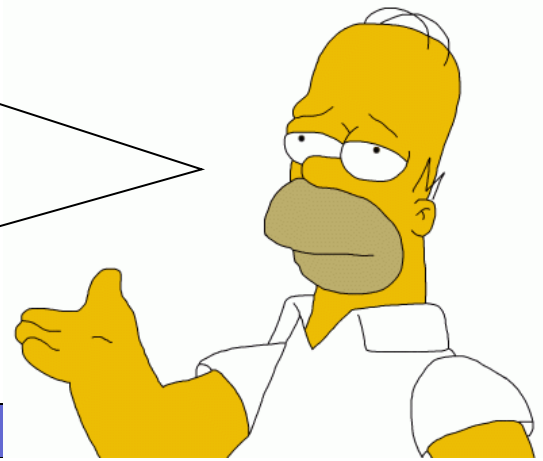

It is a B!




The rule is as follows, if the square of the sum of the two bars is less than or equal to 100, it is an A. Otherwise it is a B.

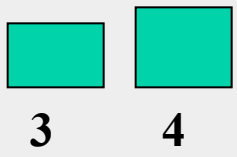
Why did we spend so much time with this game?

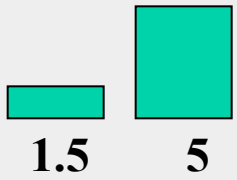
Because we wanted to show that almost all classification problems have a geometric interpretation, check out the next 3 slides…

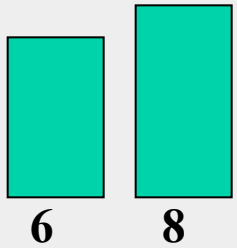


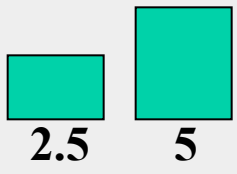
# Pigeon Problem 1

**Examples of class A**

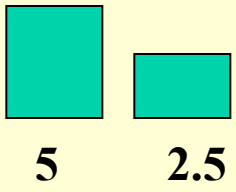


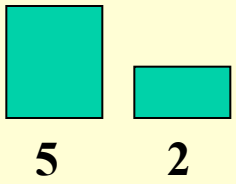



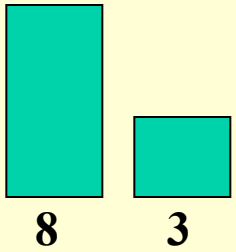



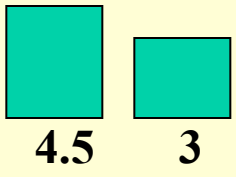



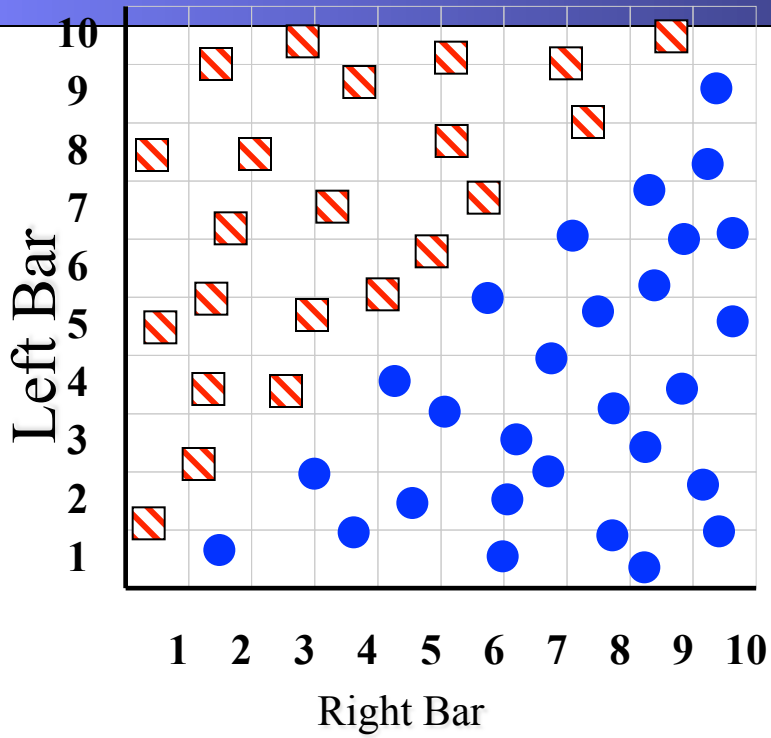

**Examples of class B**



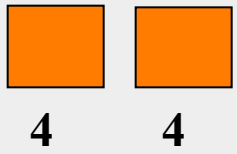



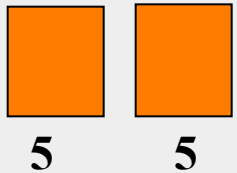



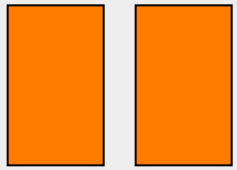

Here is the rule again. If the left bar is smaller than the right bar, it is an A, otherwise it is a B.

# Pigeon Problem 2

**Examples of class A**

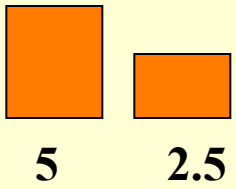


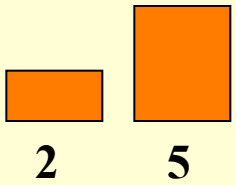



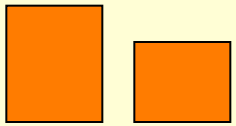



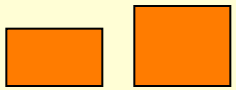


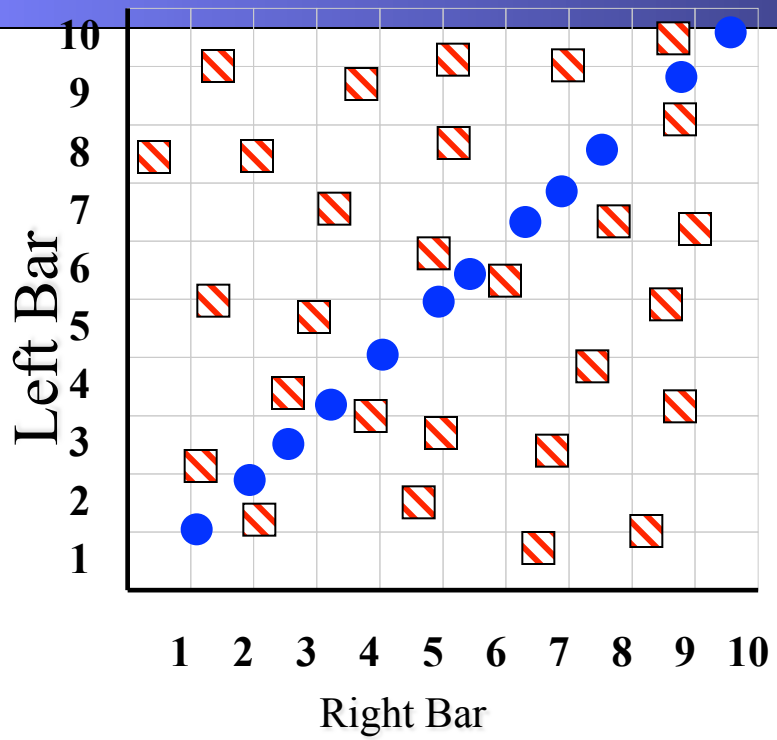
**Examples of class B**



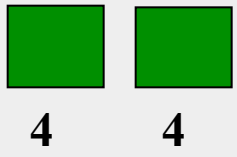



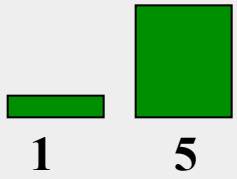



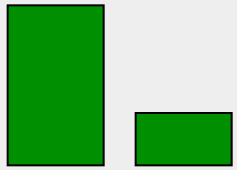

Let me look it up… here it is.. the rule is, if the two bars are equal sizes, it is an A. Otherwise it is a B.

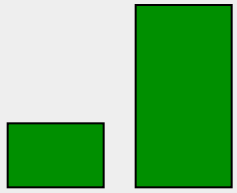
# Pigeon Problem 3

**Examples of class A**

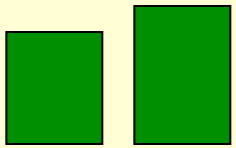


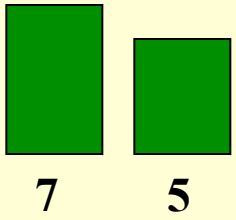



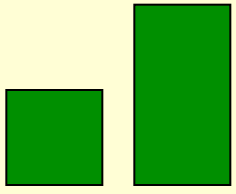



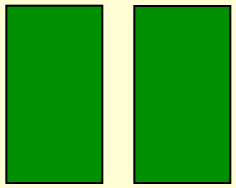



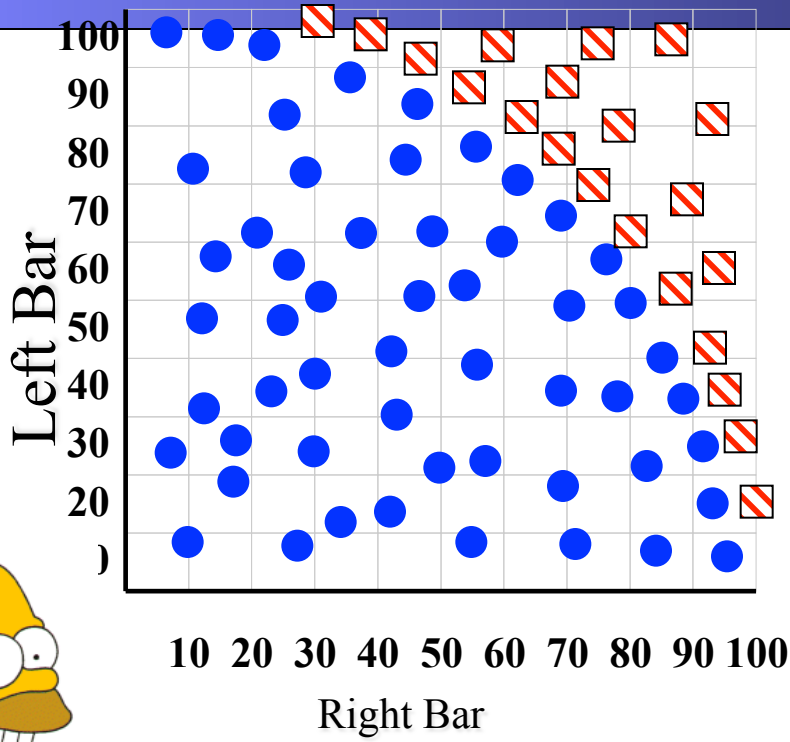

**Examples of class B**



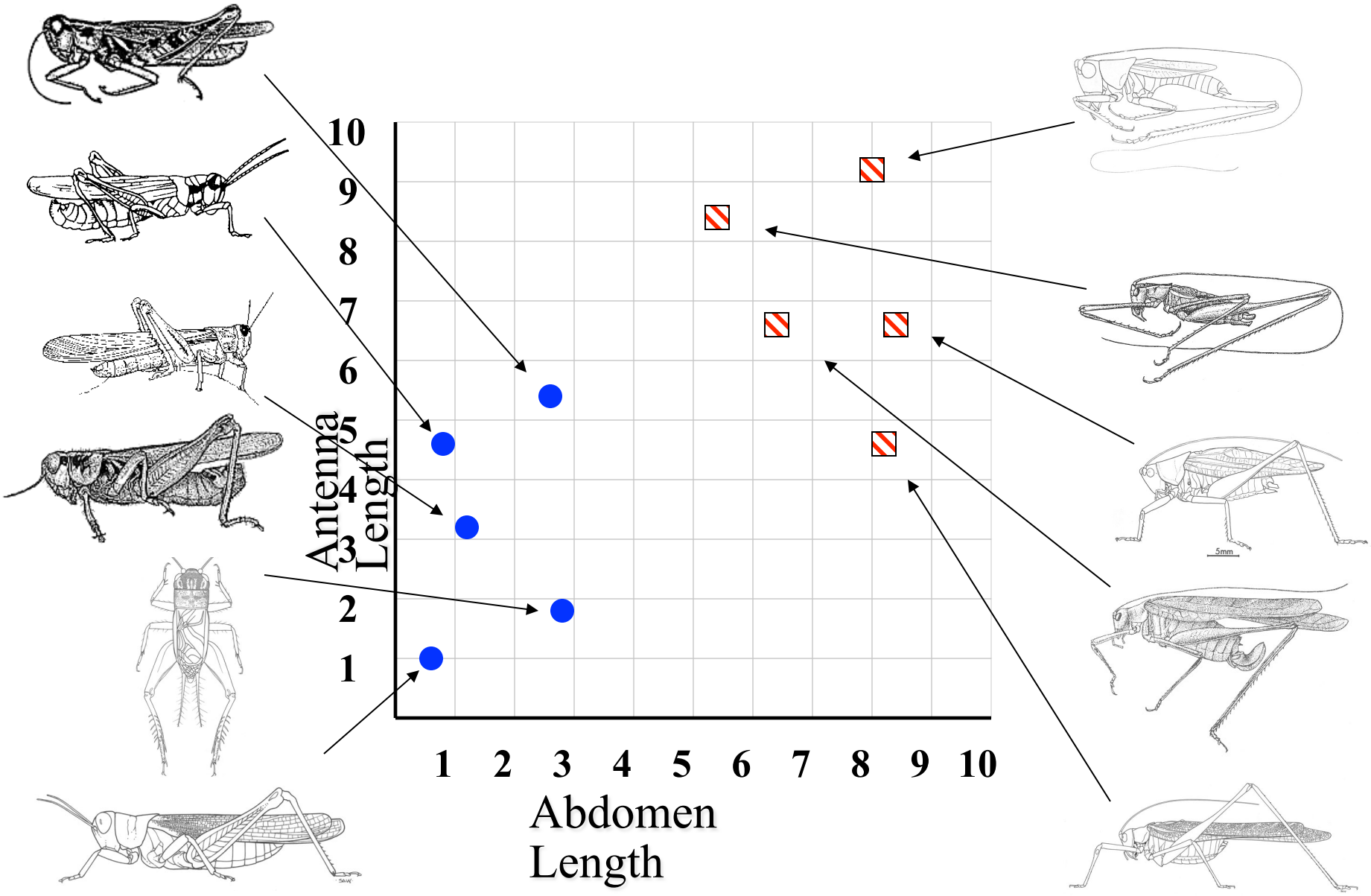



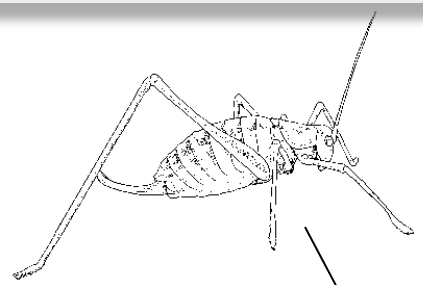

The rule again:
if the square of the sum of the two bars is less than or equal to 100, it is an A. Otherwise it is a B.

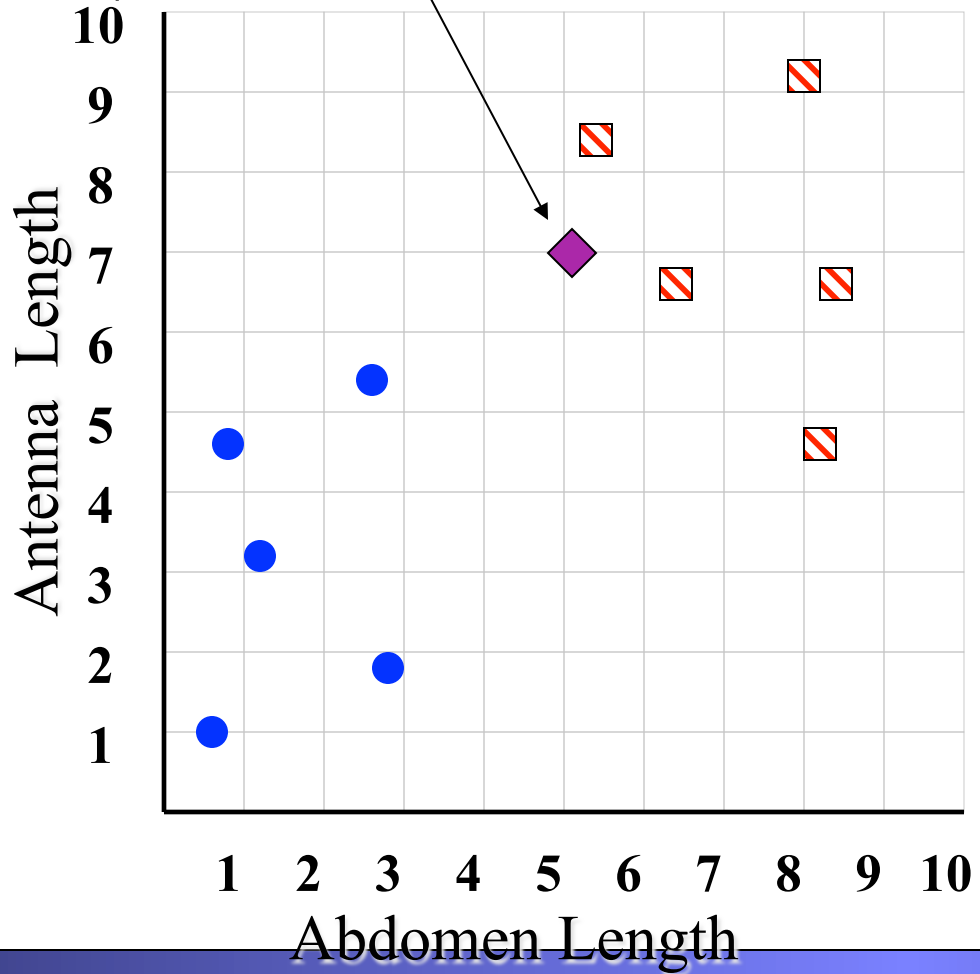
# Grasshoppers

# Katydids



| previously unseen instance = | *11* | **5.1** | **7.0** | **??????** |
|---|---|---|---|---|






We can "project" the **previously unseen instance** into the same space as the database.

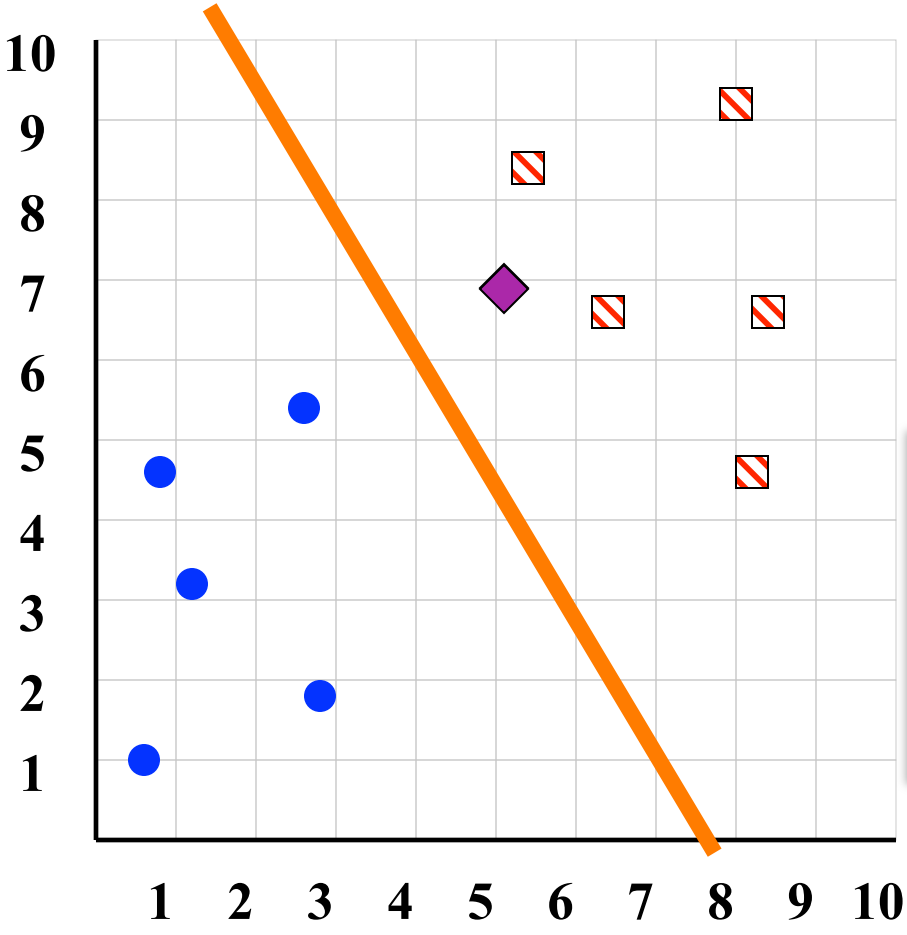
We have now abstracted away the details of our particular problem. It will be much easier to talk about points in space.

▨ **Katydids**

● **Grasshoppers**

# Simple Linear Classifier
## (Linear Discriminant Analysis)

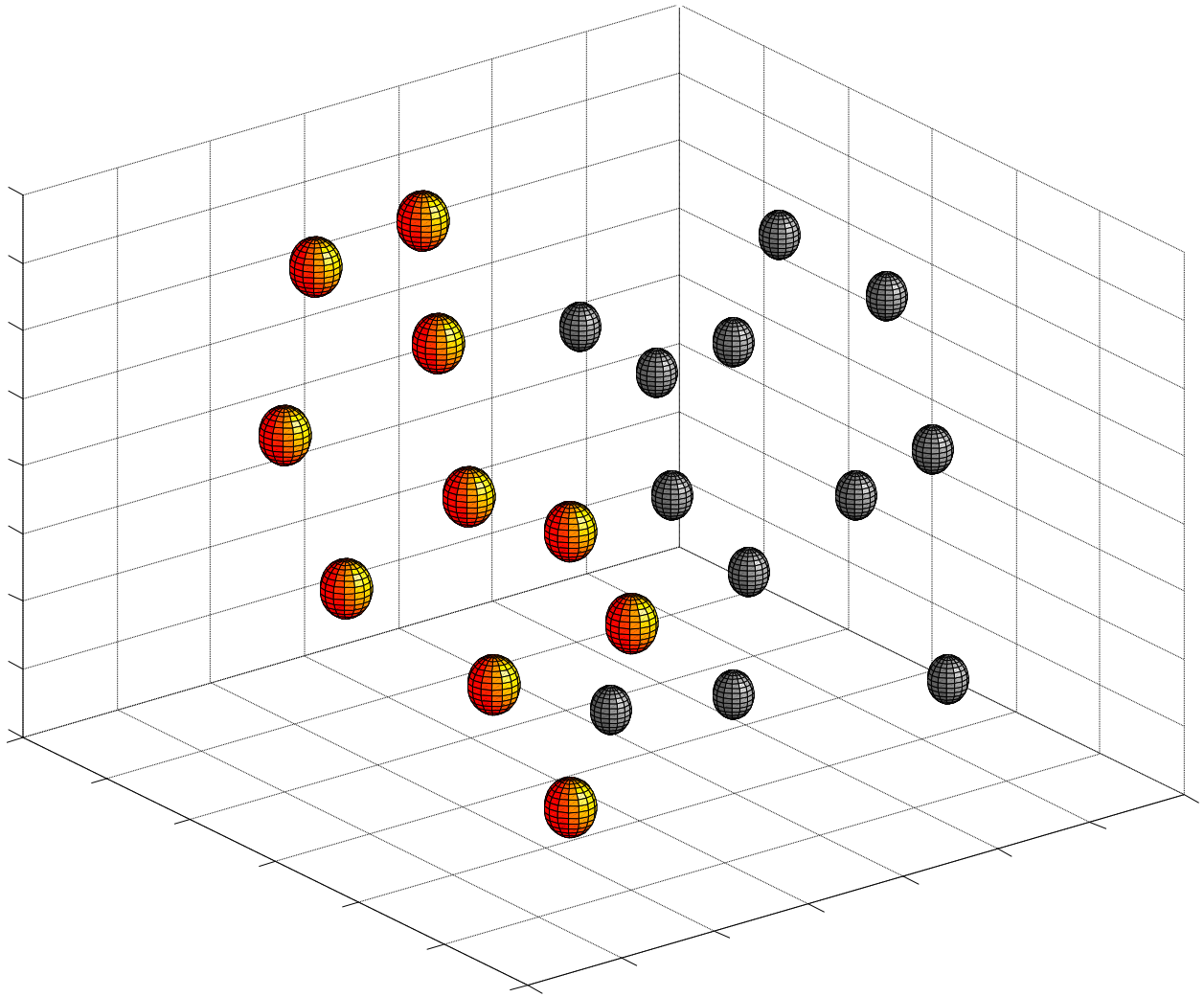
**R.A. Fisher**
**1890-1962**



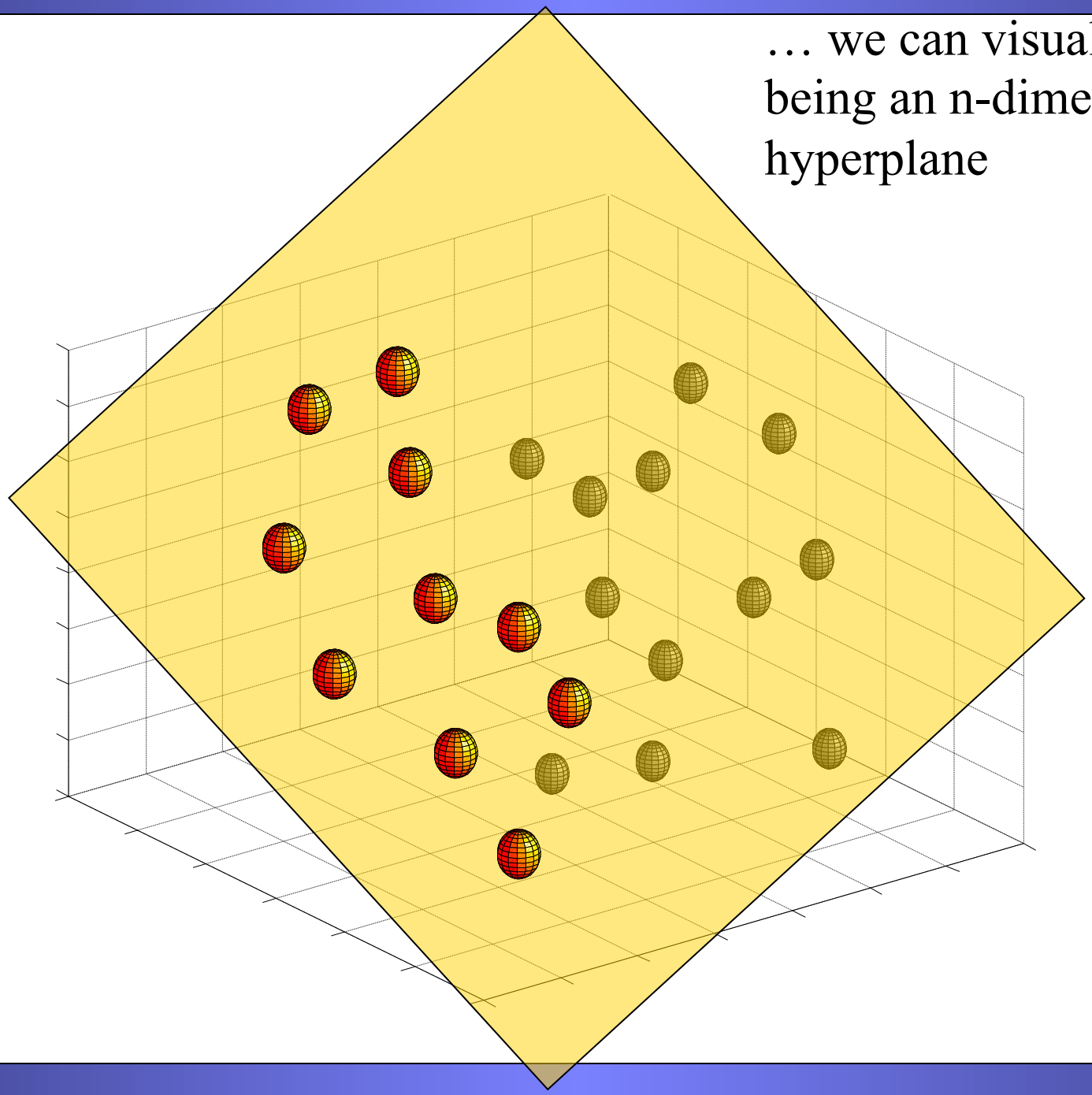

If previously unseen instance above the line
then
    class is Katydid
else
    class is Grasshopper

**Katydids**
**Grasshoppers**

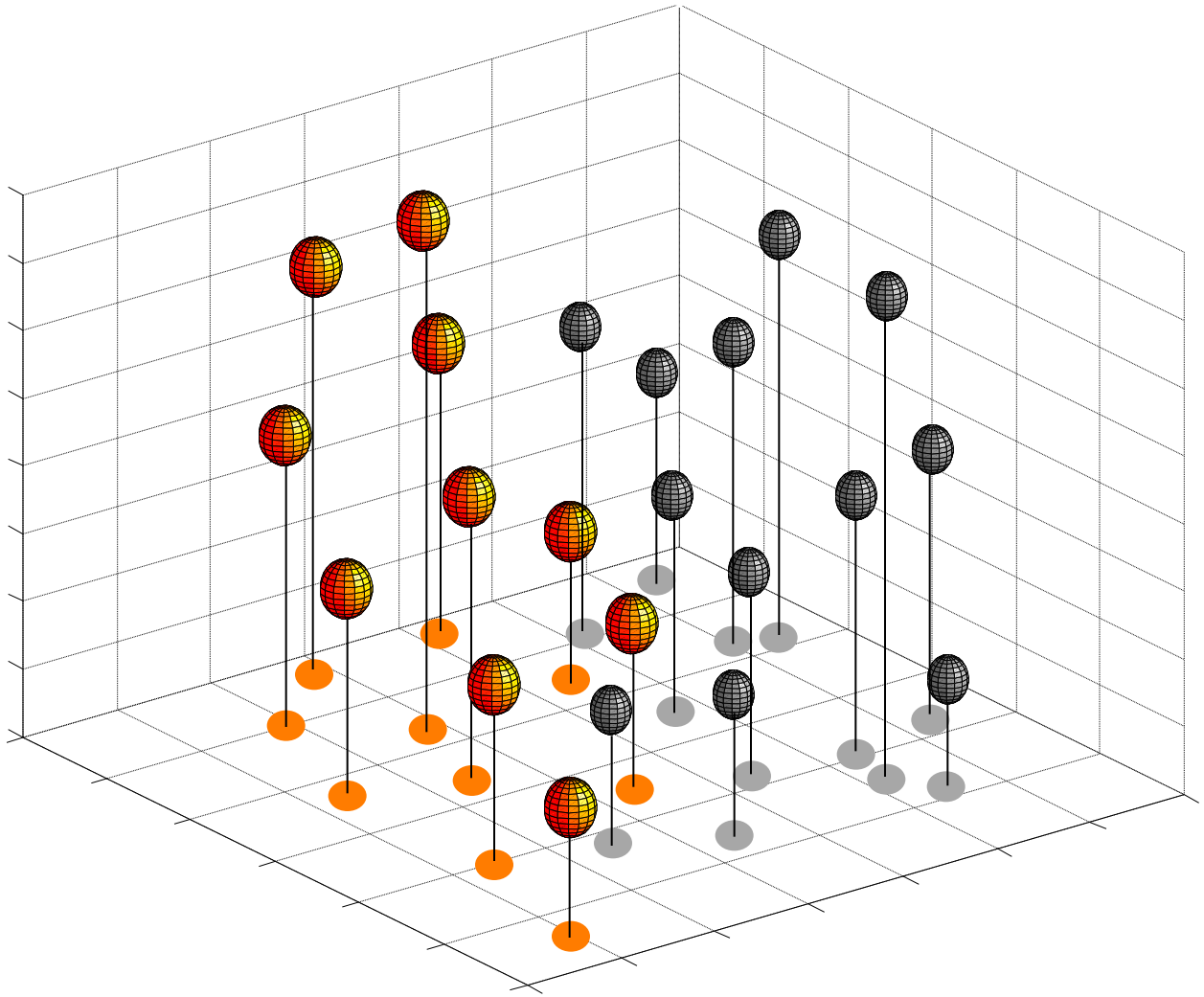
The simple linear classifier is defined for higher dimensional spaces…

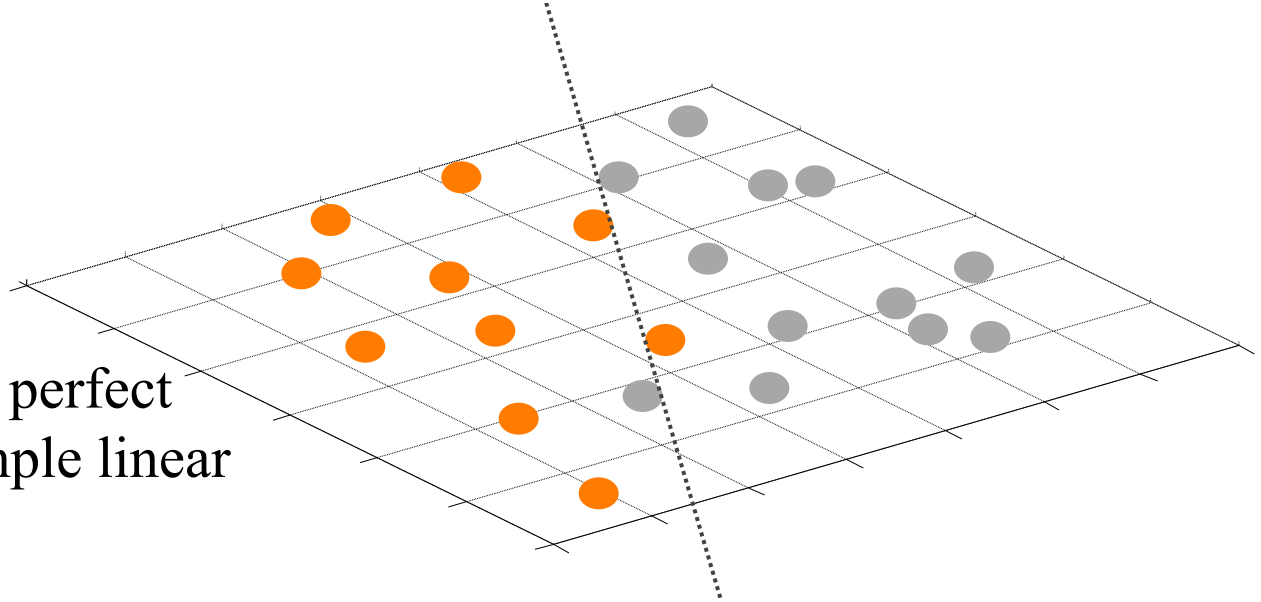


… we can visualize it as being an n-dimensional hyperplane



It is interesting to think about what would happen in this example if we did not have the 3rd dimension…

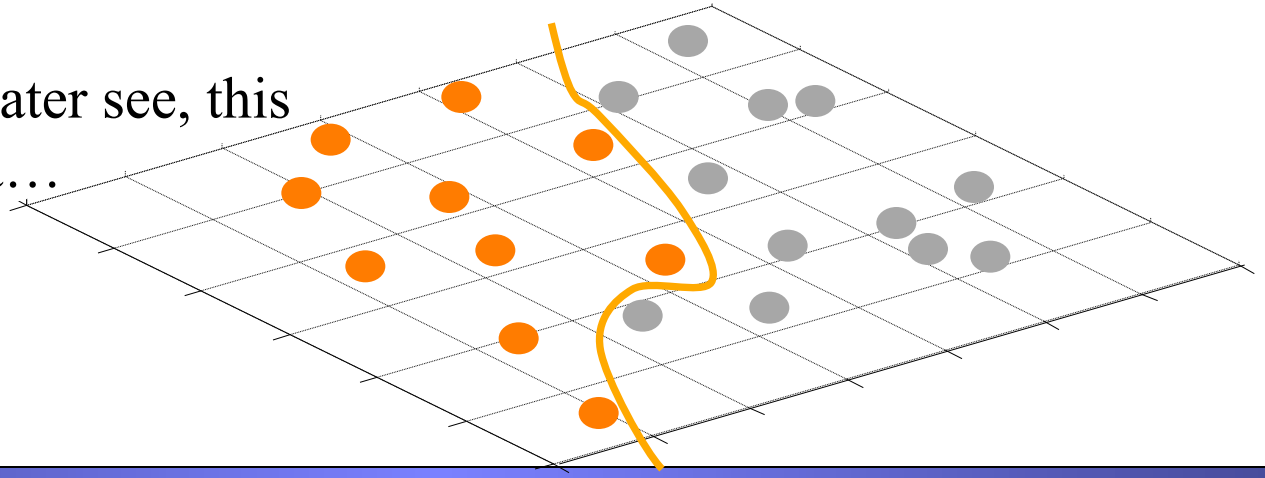




We can no longer get perfect accuracy with the simple linear classifier…

We could try to solve this problem by user a simple *quadratic* classifier or a simple *cubic* classifier..

However, as we will later see, this is probably a bad idea…

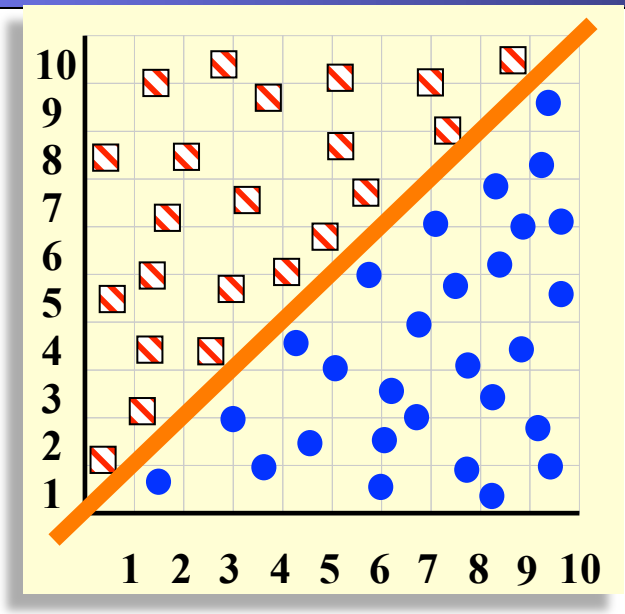


Which of the "Pigeon Problems" can be solved by the Simple Linear Classifier?

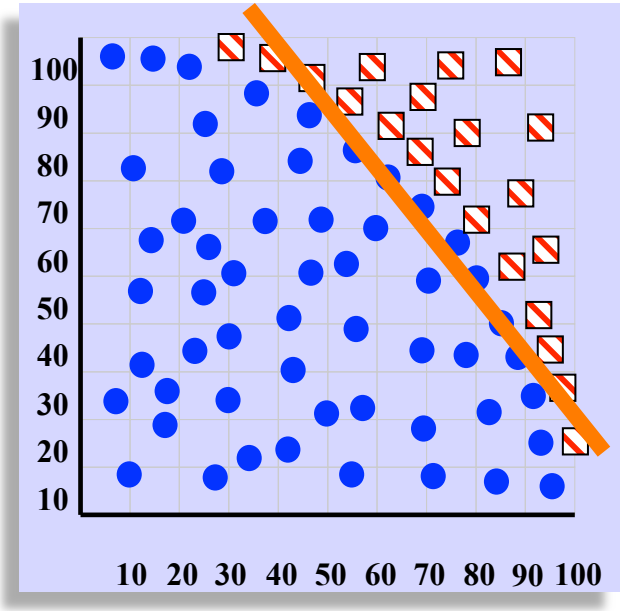
Perfect

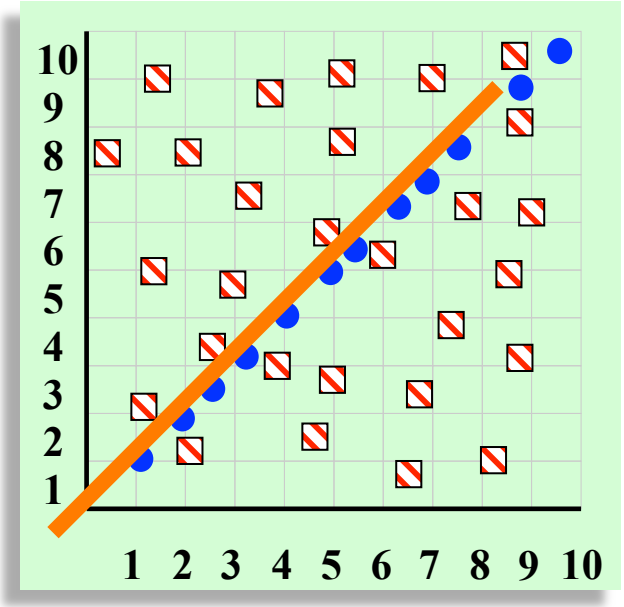
Useless

Pretty Good




**Problems that can be solved by a linear classifier are called linearly separable.**



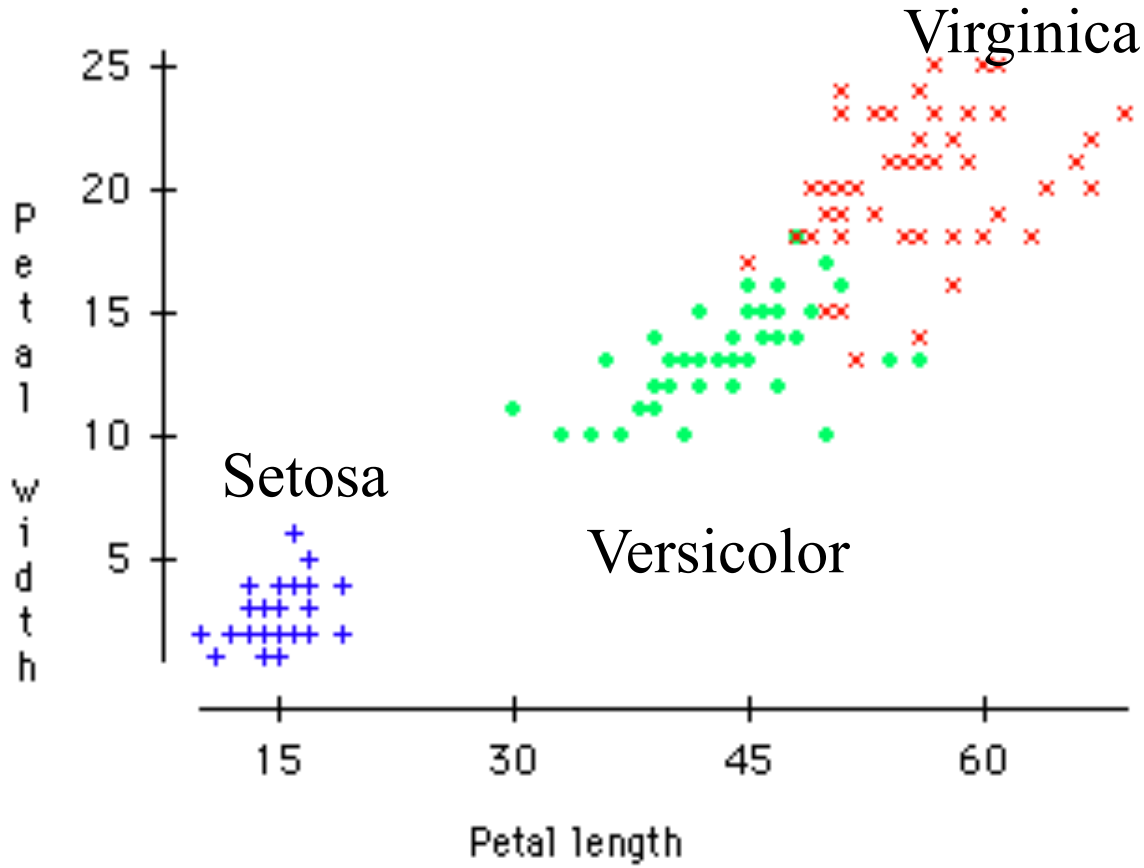
# A Famous Problem

R. A. Fisher's Iris Dataset.

3 classes

50 of each class

The task is to classify Iris plants into one of 3 varieties using the Petal Length and Petal Width.

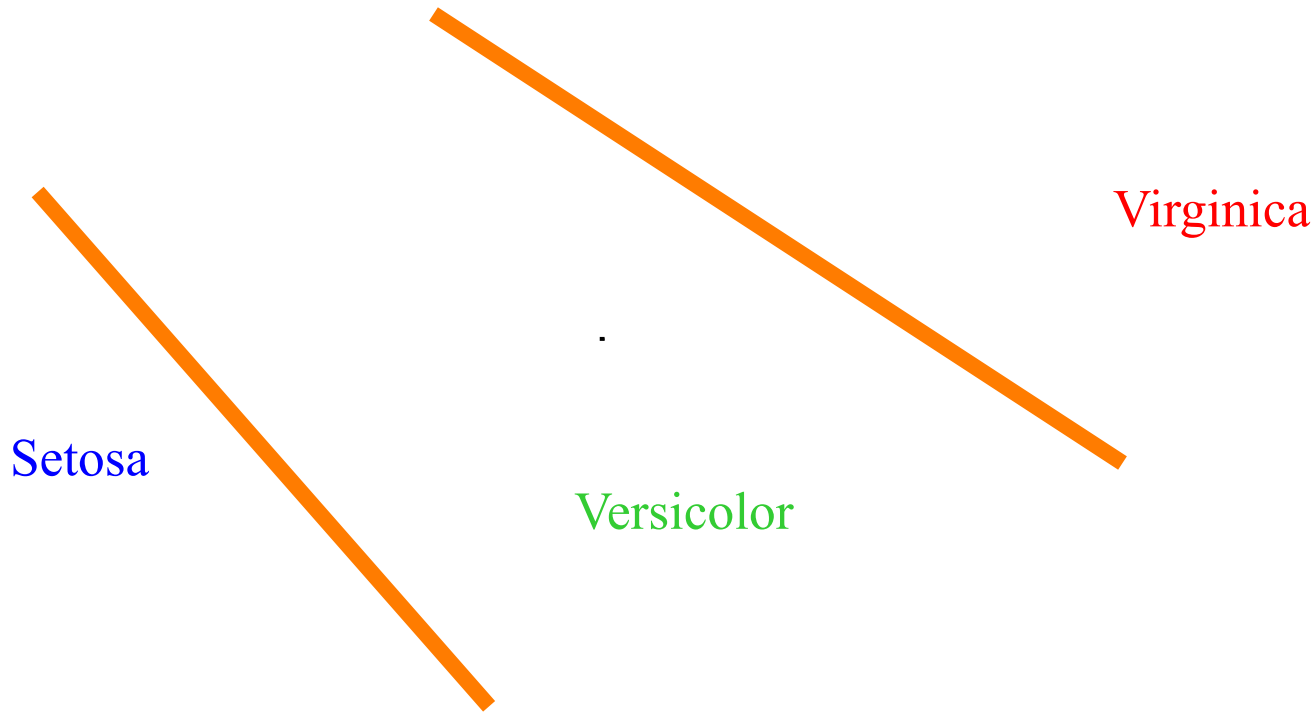



Iris Setosa

Iris Versicolor

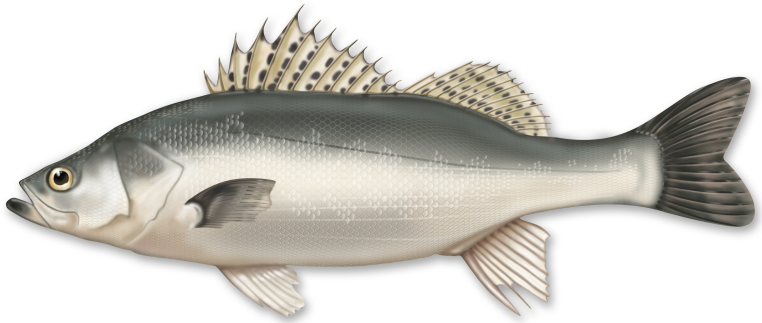
Iris Virginica

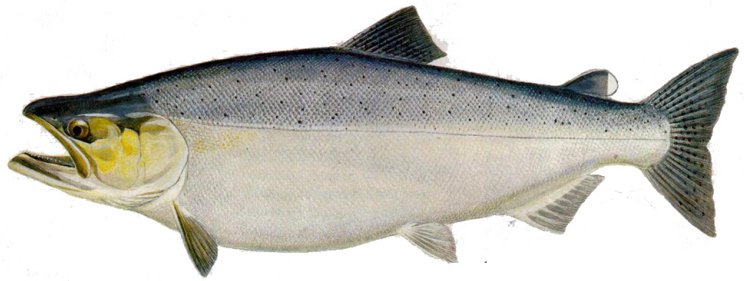
We can generalize the piecewise linear classifier to N classes, by fitting N-1 lines. In this case we first learned the line to (perfectly) discriminate between Setosa and Virginica/Versicolor, then we learned to approximately discriminate between Virginica and Versicolor.




If petal width > 3.272 – (0.325 * petal length) then class = Virginica
Elseif petal width…

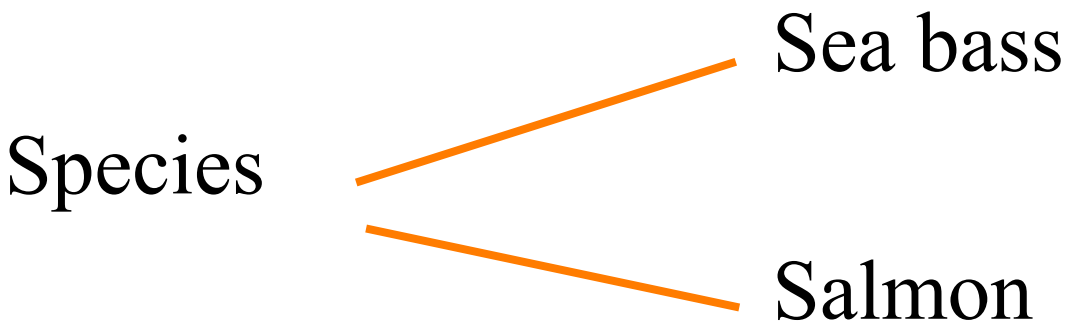
# Case Study: Sea Bass vs Salmon ?





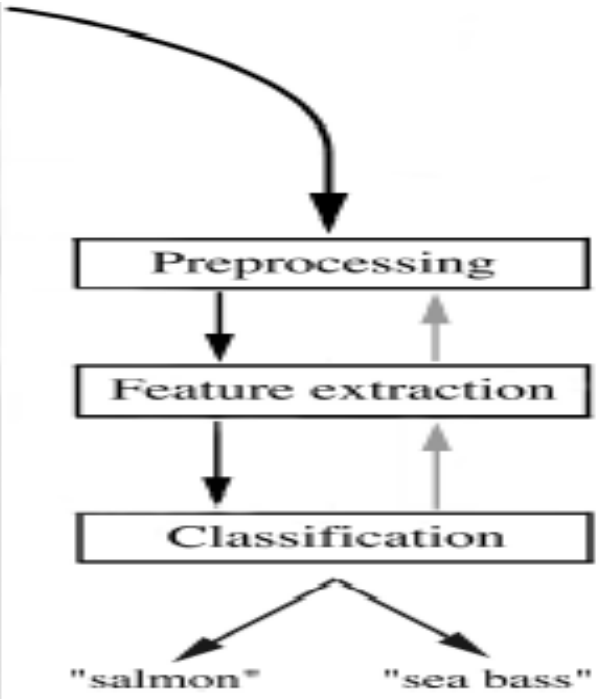
# An Example

- "Sorting incoming Fish on a conveyor according to species using optical sensing"

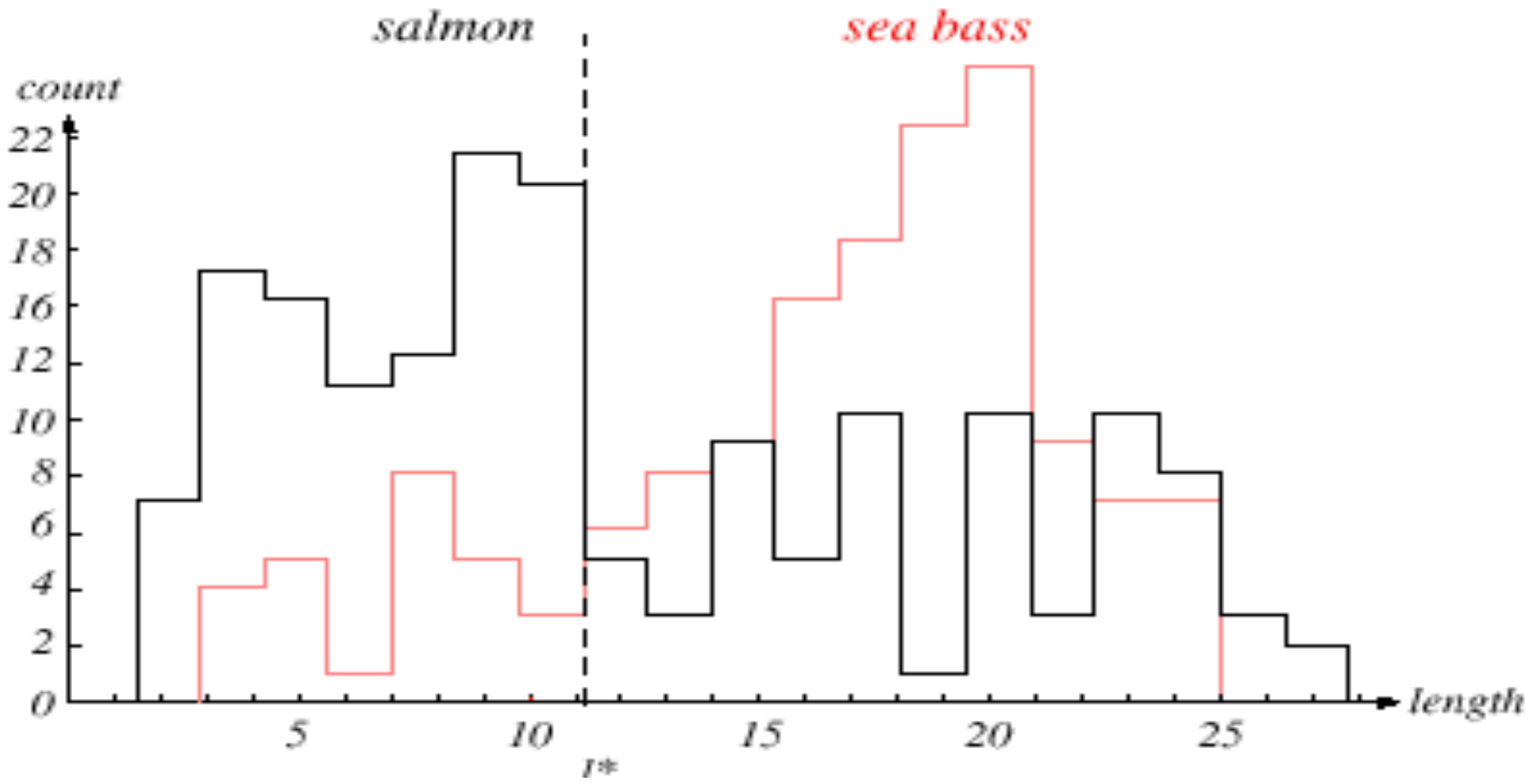


# Problem Analysis

- Set up a camera and take some sample images to extract features
  - Length
  - Lightness
  - Width
  - Number and shape of fins
  - Position of the mouth, etc…

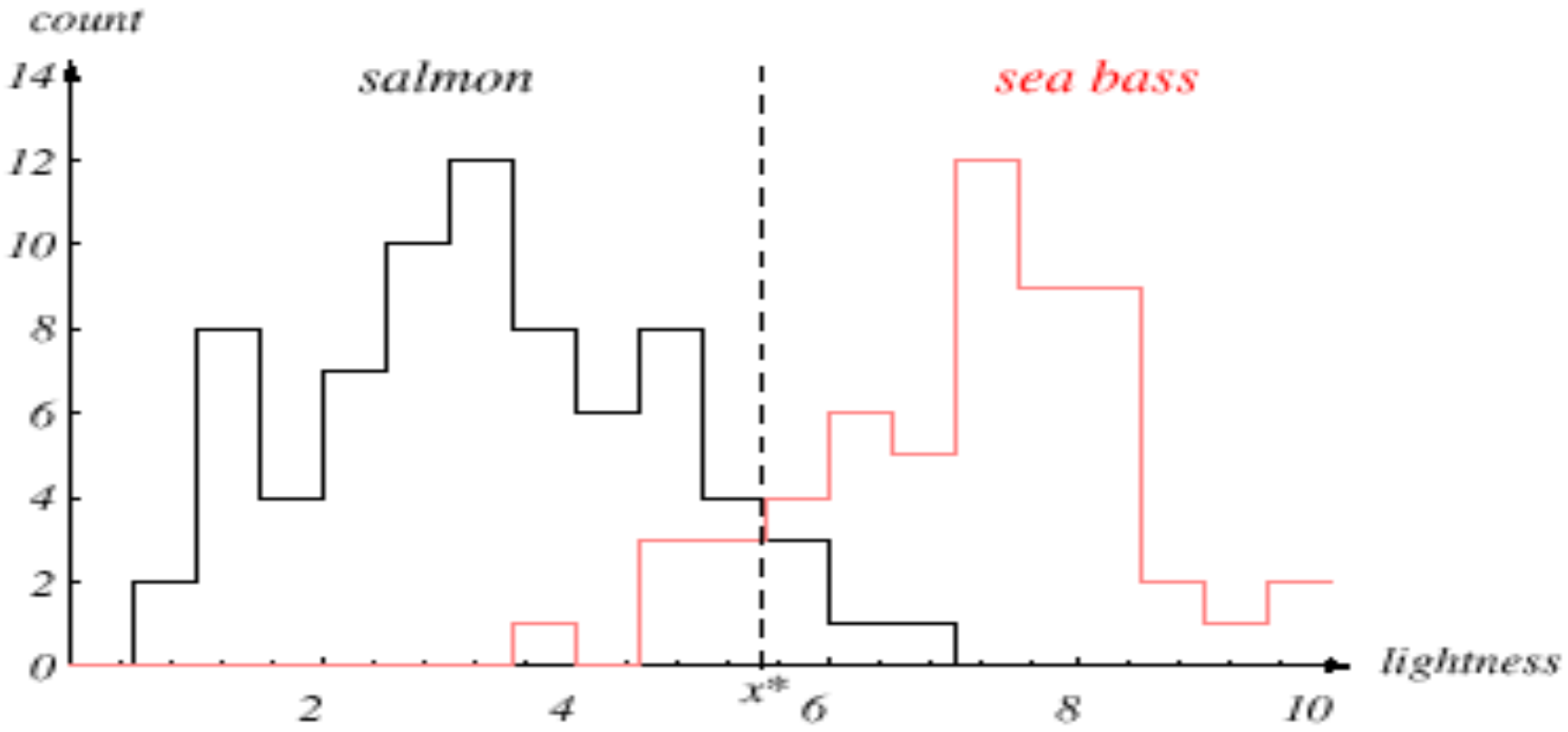

Preprocessing
Feature extraction
Classification
"salmon"
"sea bass"

- Classification
  - Select the length of the fish as a possible feature for discrimination

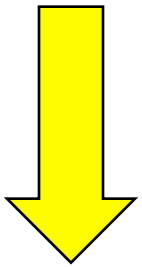

salmon
sea bass
count
22
20
18
16
12
10
8
6
4
2
0
5
10
15
20
25
length
l*

The length is a poor feature alone!

Select the lightness as a possible feature.

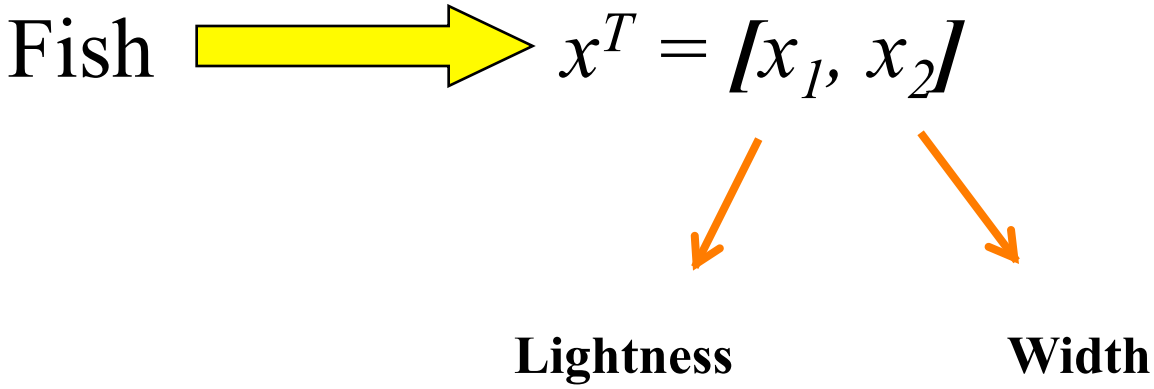

count
14
12
10
8
6
4
2
0
salmon
sea bass
2
4
x*
6
8
10
lightness

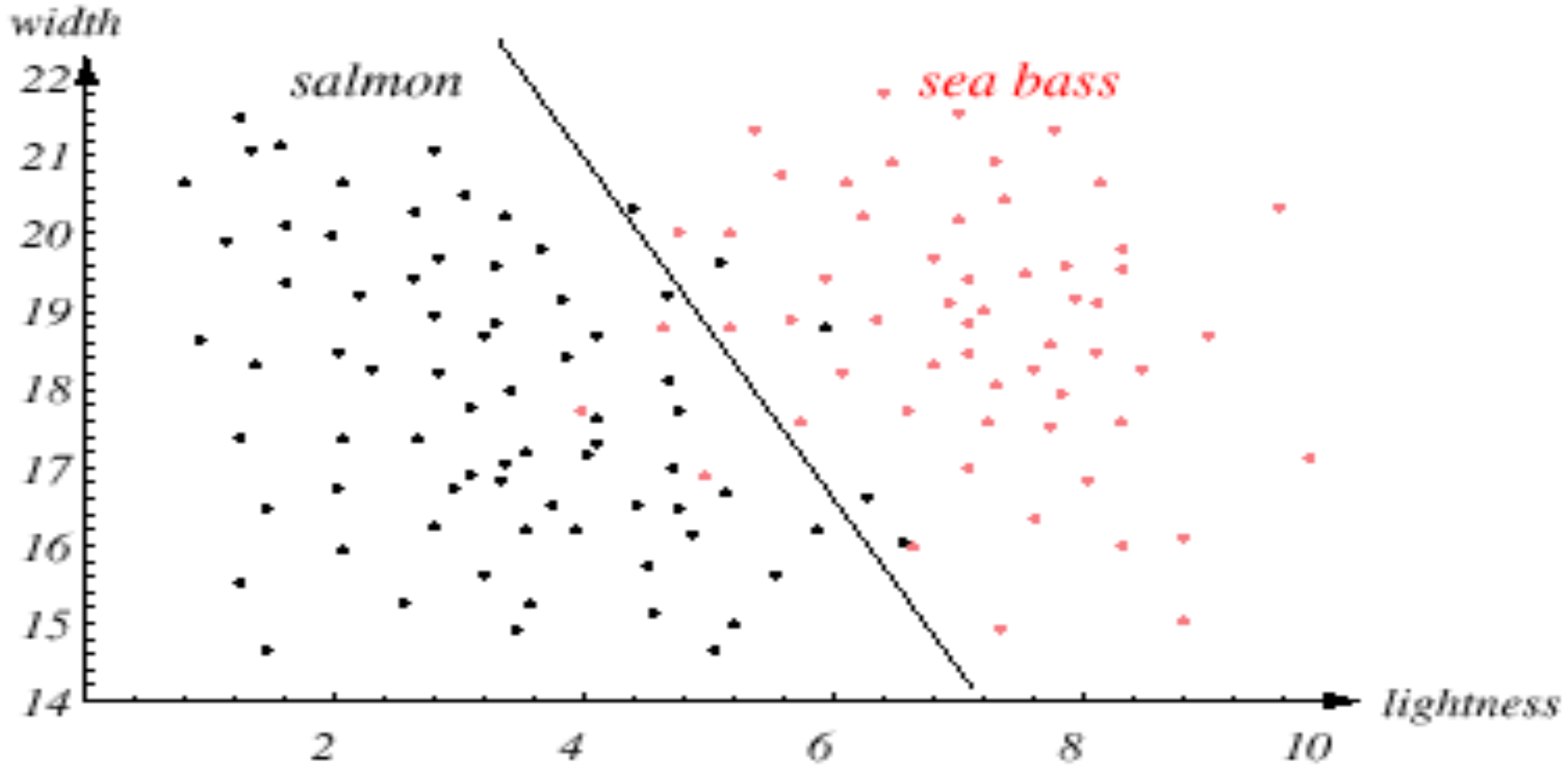
- Threshold decision boundary and cost relationship
  - Move our decision boundary toward smaller values of lightness in order to minimize the cost (reduce the number of sea bass that are classified salmon!)



Task of decision theory

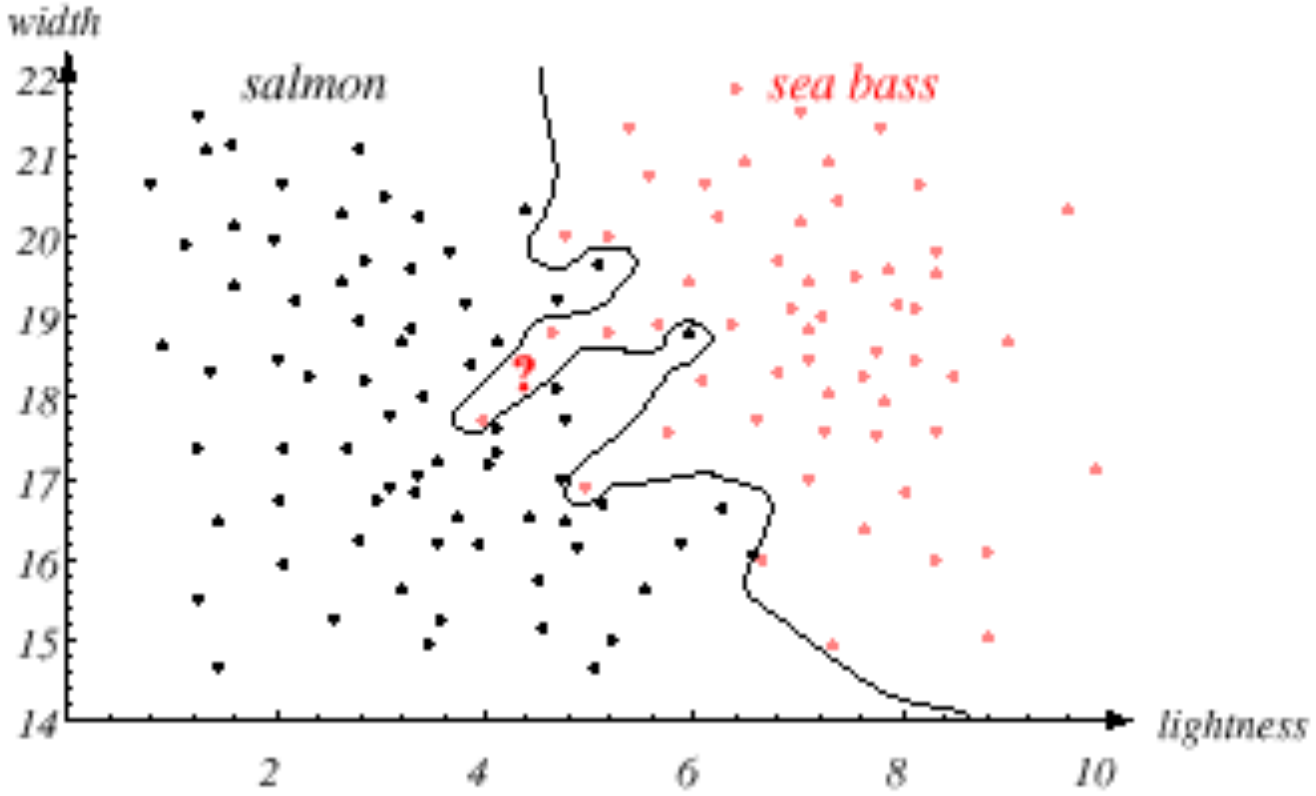
- Adopt the lightness and add the width of the fish



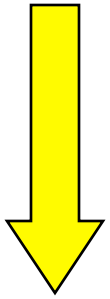

width
22
21
20
19
18
17
16
15
14
salmon
sea bass
2
4
6
8
10
lightness

- We might add other features that are not correlated with the ones we already have. A precaution should be taken not to reduce the performance by adding such “noisy features”

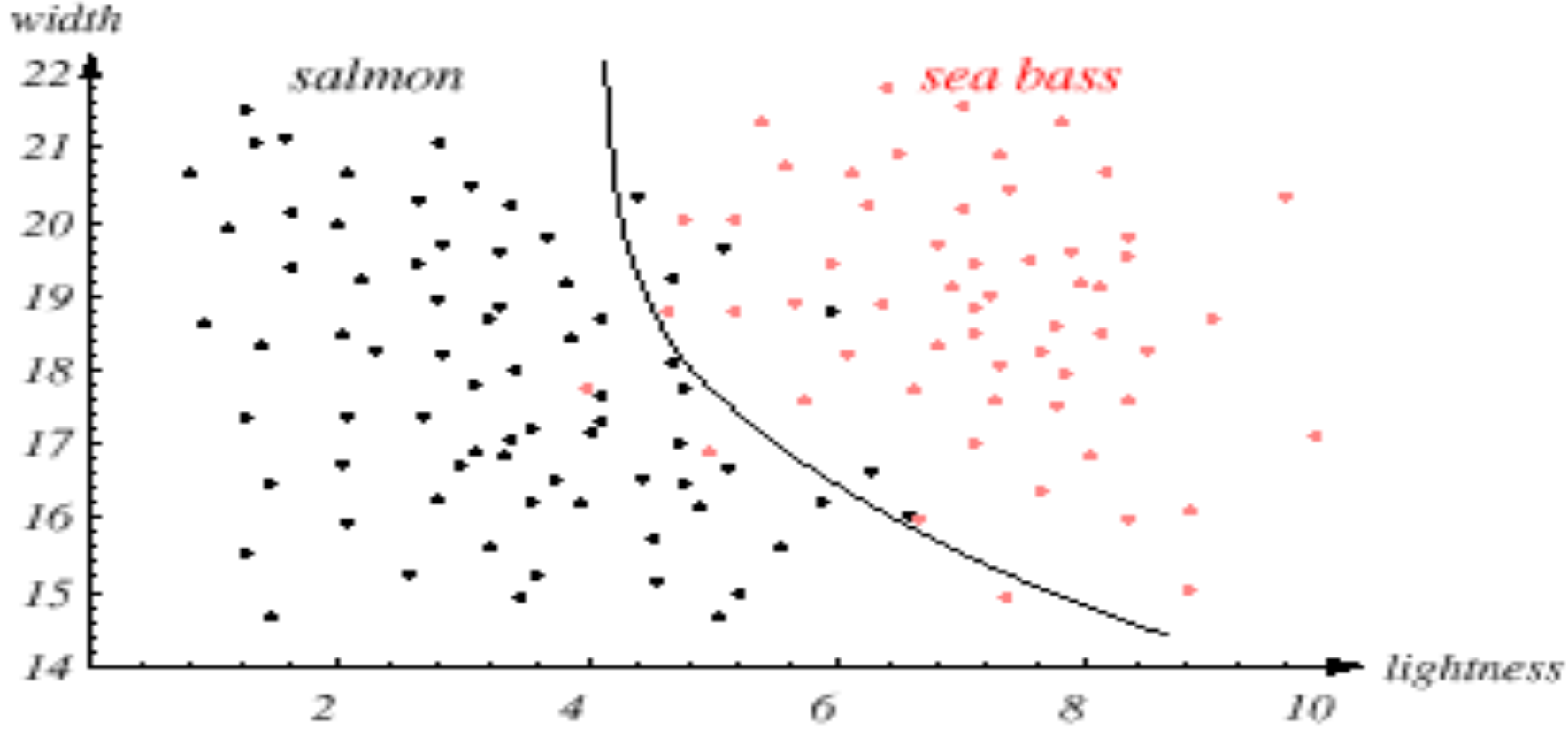
- Ideally, the best decision boundary should be the one which provides an optimal performance such as in the following figure:


width
salmon
sea bass
?
22
21
20
19
18
17
16
15
14
2
4
6
8
10
lightness

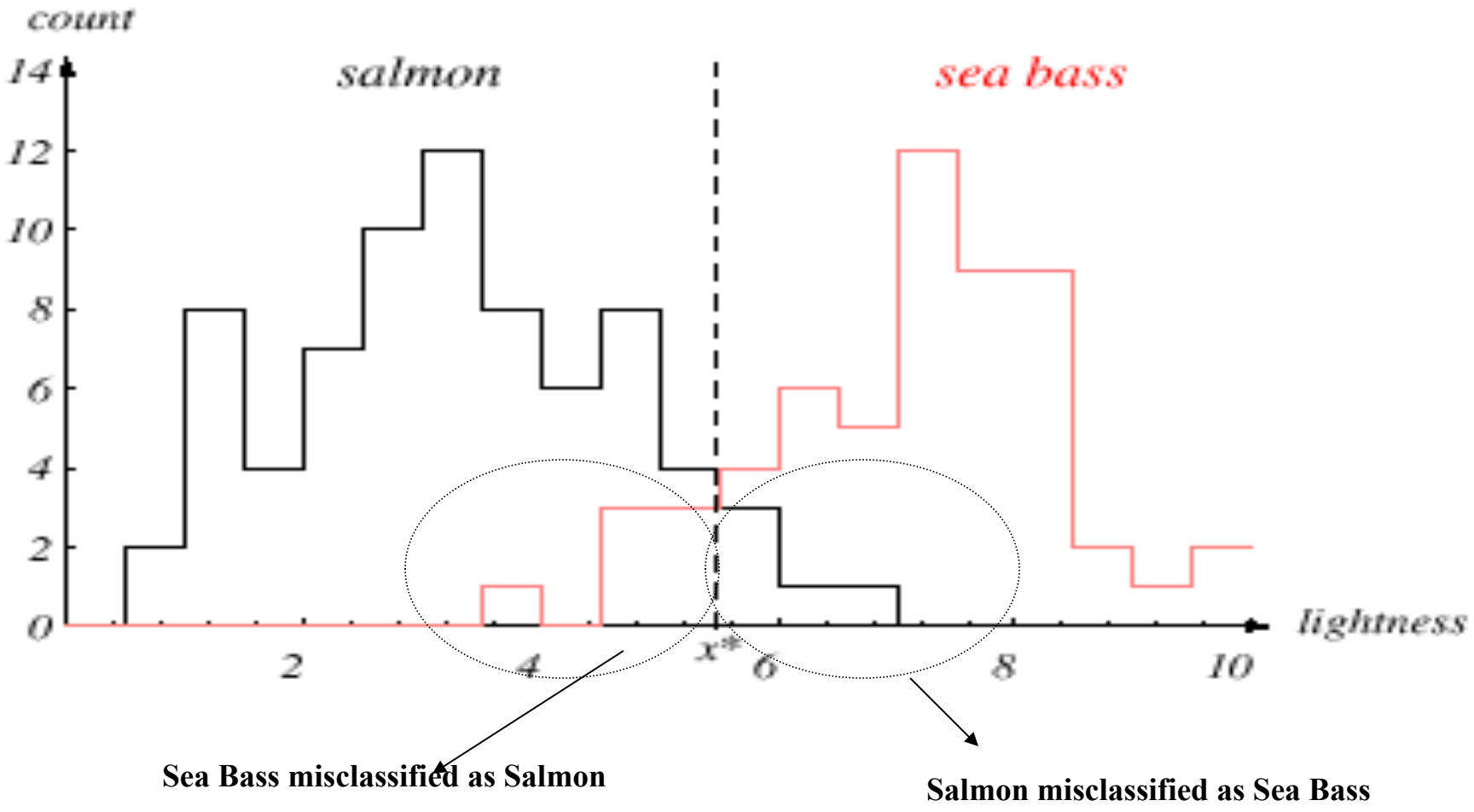
- However, our satisfaction is premature because the central aim of designing a classifier is to correctly classify novel input



Issue of generalization!


width
22
21
20
19
18
17
16
15
14
salmon
sea bass
2
4
6
8
10
lightness

# Misclassifications



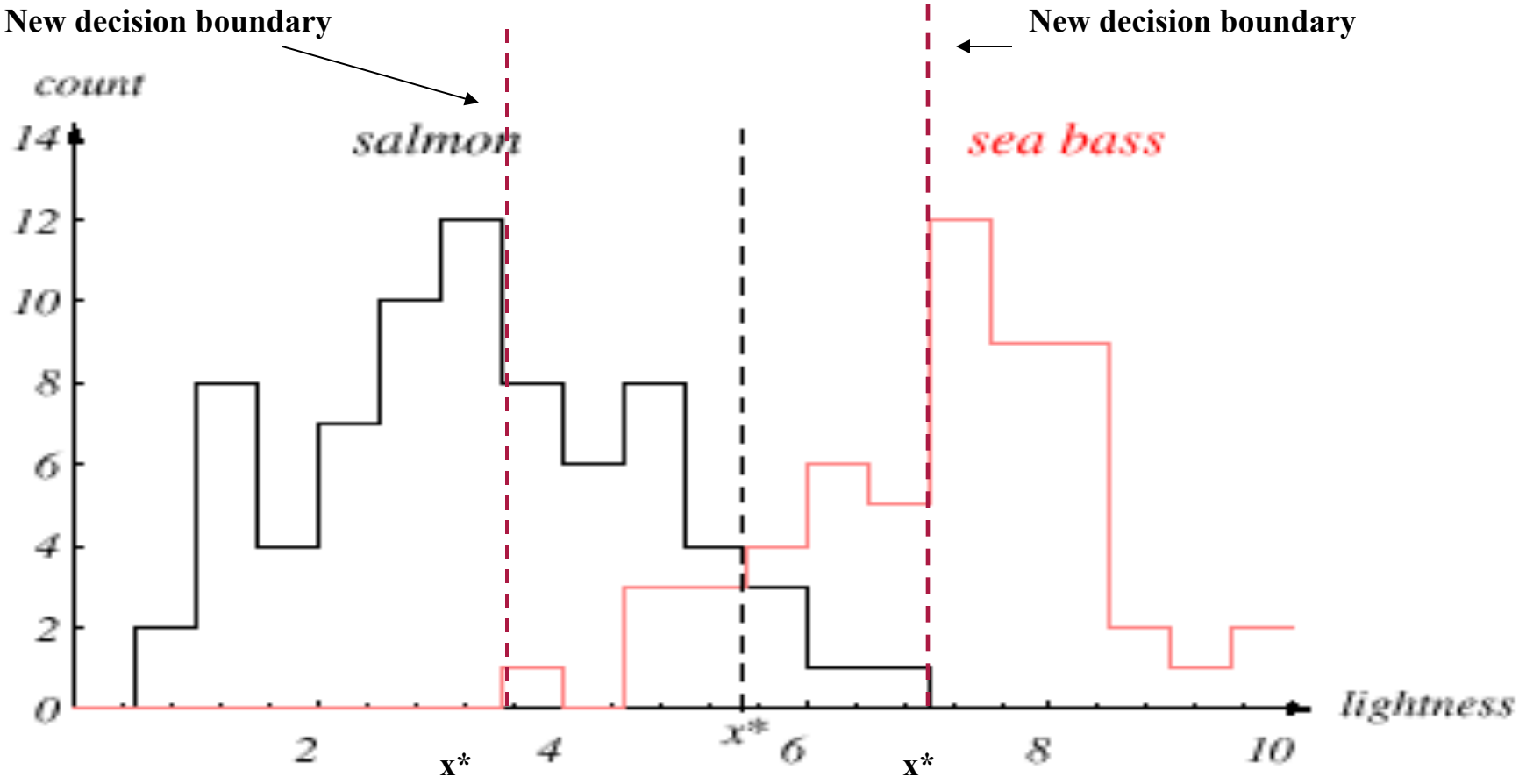
# Cost sensitive classification

- Penalize misclassifications of one class more than the other
- Changes decision boundaries

**What if?**

Salmon is more expensive than Bass?

Bass is more expensive than Salmon?



# Classification Techniques

- Instance-Based Classifiers
- Decision Tree based Methods
- Rule-based Methods
- Neural Networks
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines

# Instance-Based Classifiers

Set of Stored Cases

| Atr1 | ……. | AtrN | Class |
|---|---|---|---|
| | | | A |
| | | | B |
| | | | B |
| | | | C |
| | | | A |
| | | | C |
| | | | B |

- **Store the training records**
- **Use training records to predict the class label of unseen cases**

Unseen Case

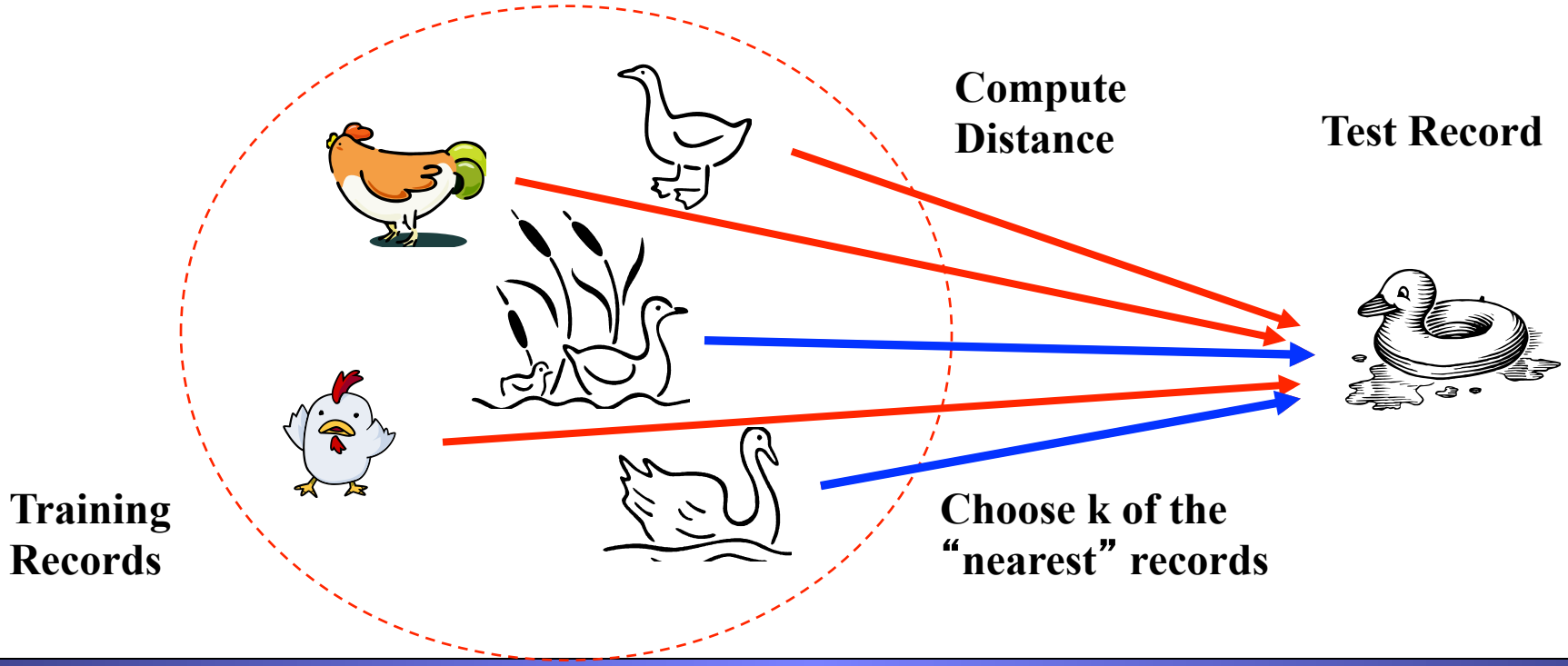
| Atr1 | ……. | AtrN |
|---|---|---|
| | | |

# Instance Based Classifiers

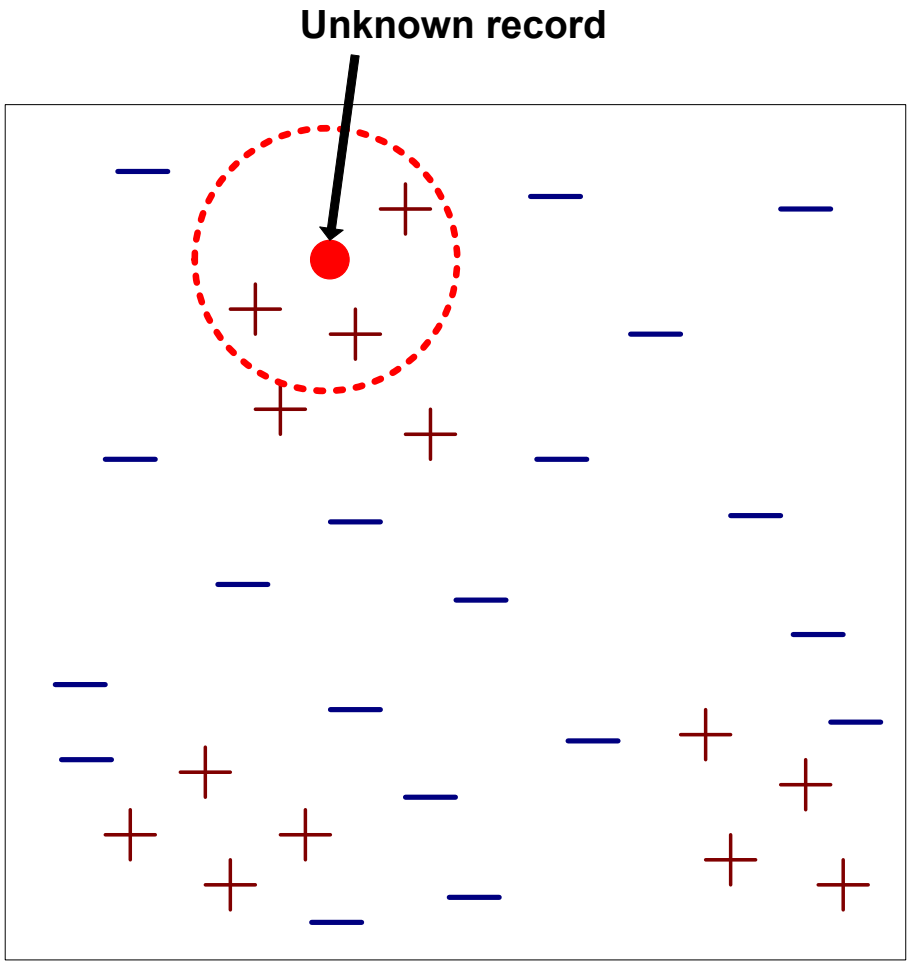
- Examples:
  - Rote-learner
    - Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
  - Nearest neighbor
    - Uses the "closest" points (nearest neighbors) for performing classification

# Nearest Neighbor Classifiers

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck



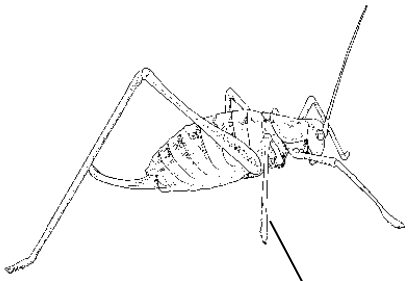
# Nearest-Neighbor Classifiers

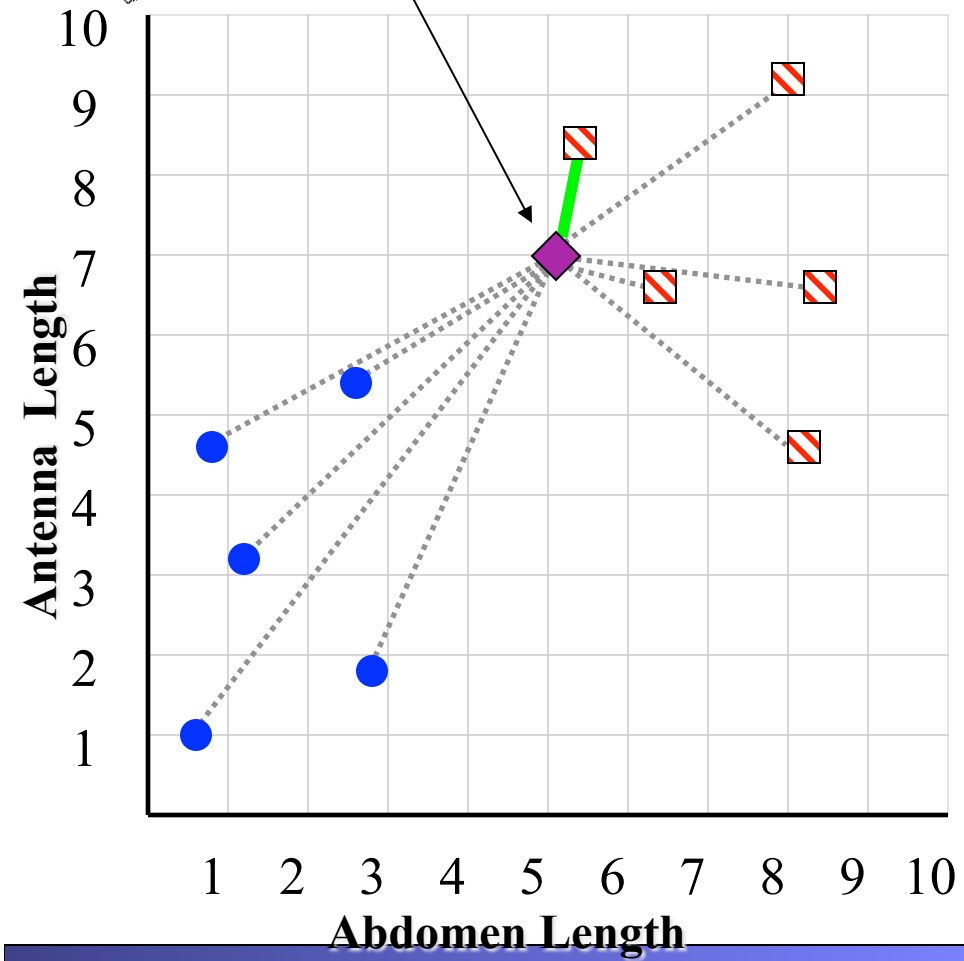



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve

- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

# Nearest Neighbor Classifiers





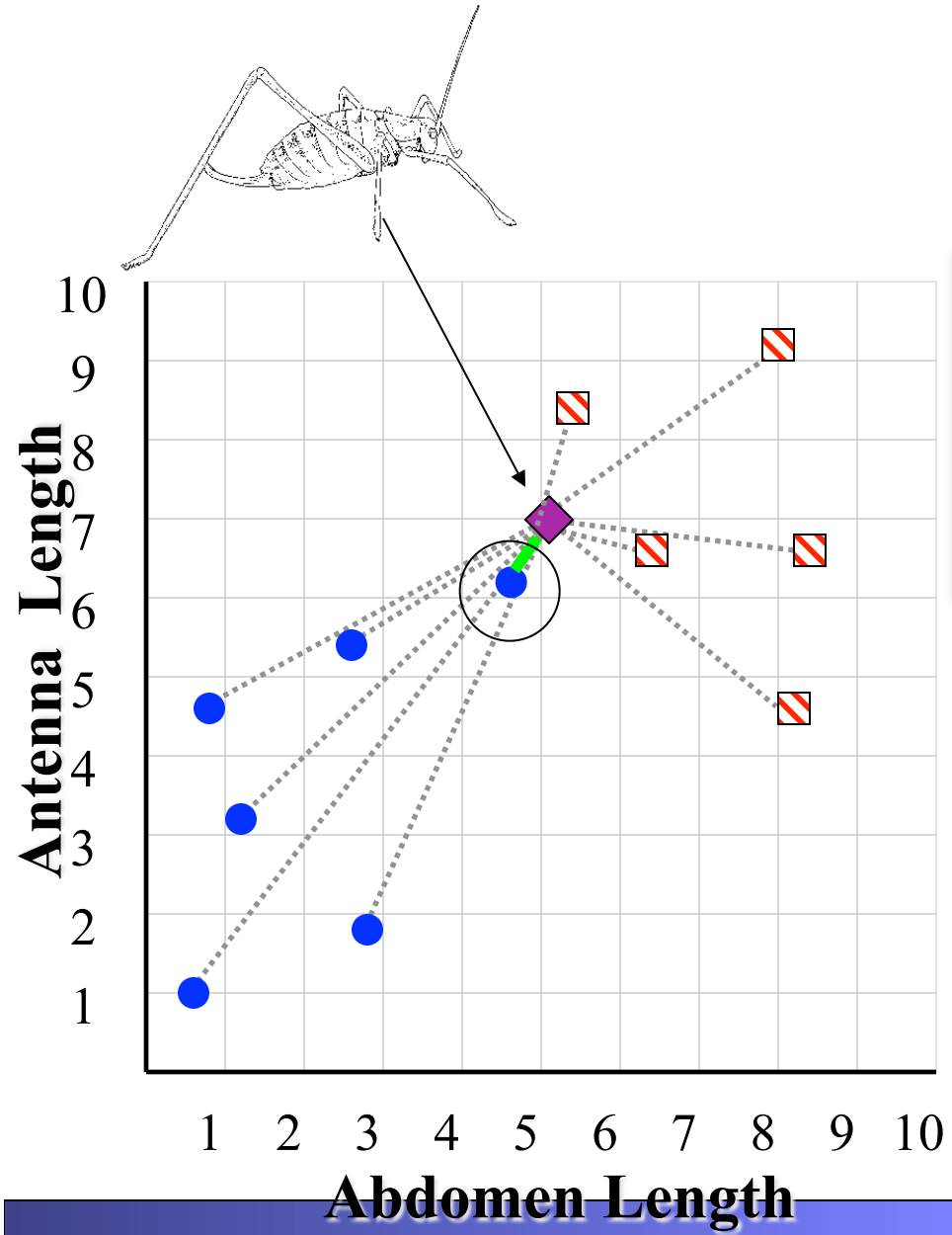

Evelyn Fix
1904-1965

Joe Hodges
1922-2000

**If** the **nearest** instance to the previously unseen instance **is a Katydid**
    class is **Katydid**
**else**
    class is **Grasshopper**

**Katydids**
**Grasshoppers**

# Nearest Neighbor Classifier is sensitive to outliers

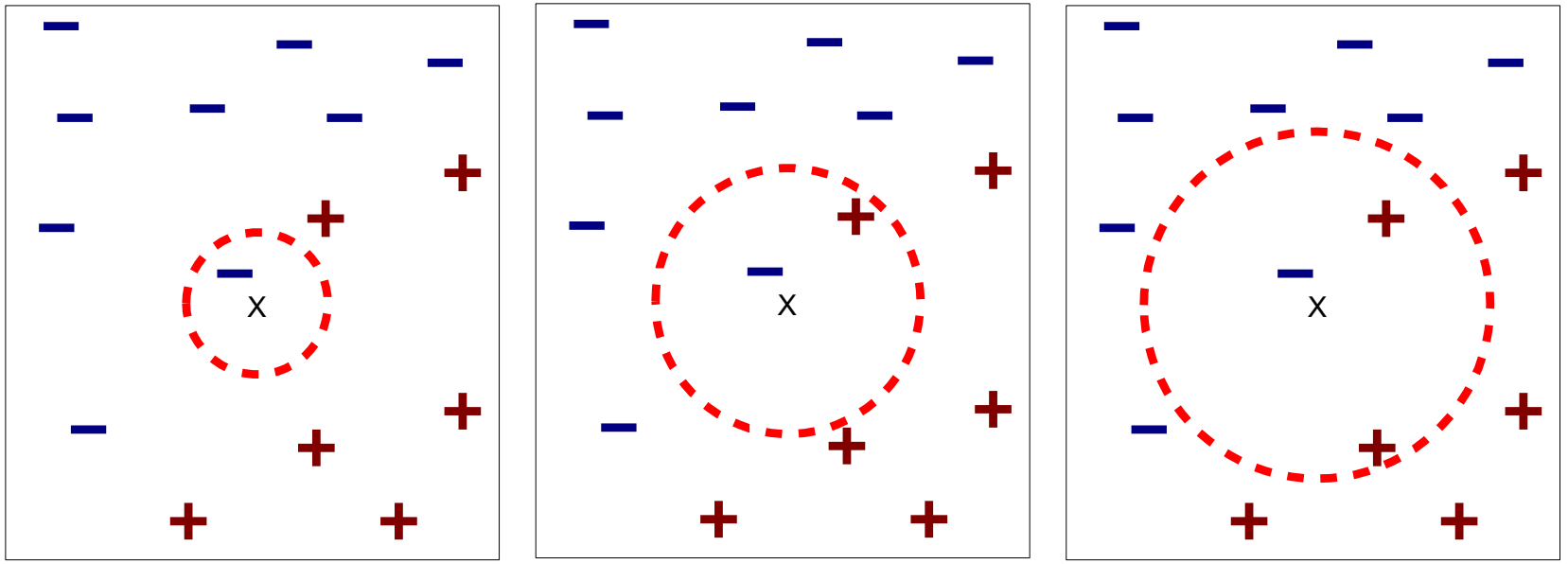



**If** the **nearest** instance to the previously unseen instance **is a Katydid**
class is **Katydid**
**else**
class is **Grasshopper**

**Katydids**
**Grasshoppers**

Solution: Use K nearest neighbors instead, and take majority vote!

# Definition of Nearest Neighbor

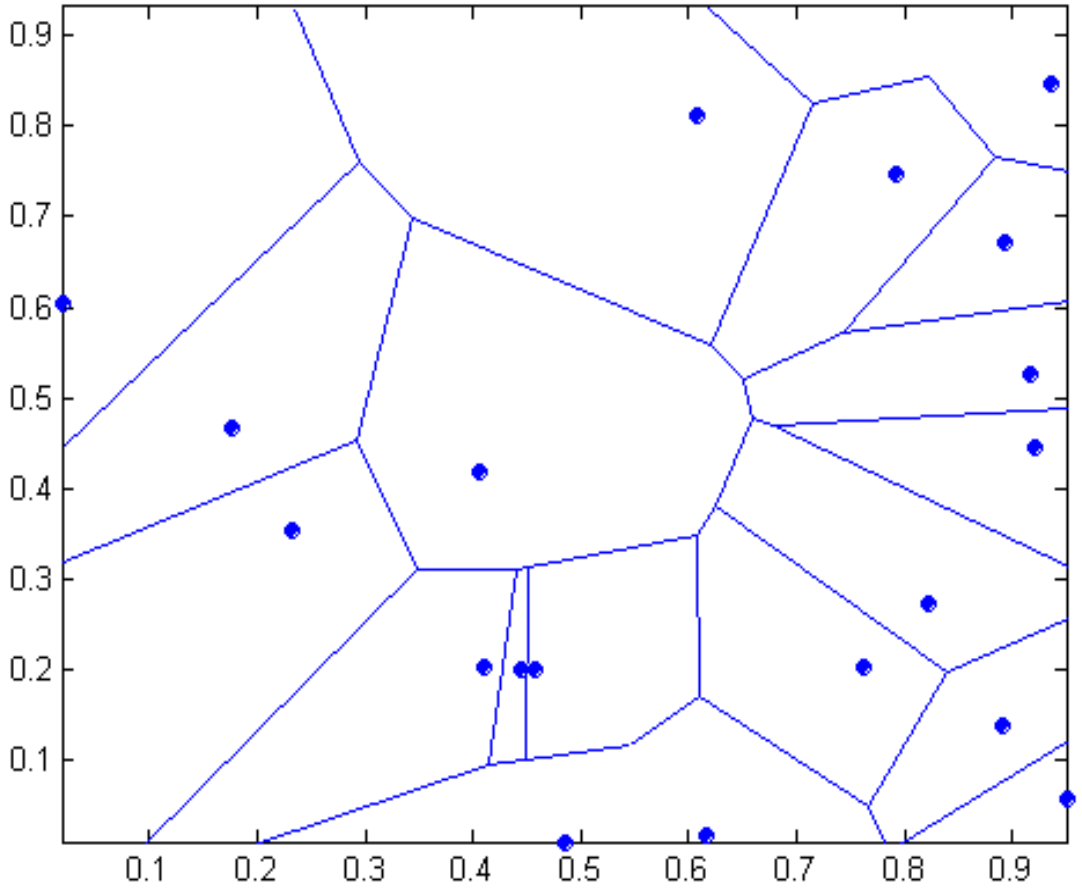



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

# 1-nearest-neighbor

Voronoi Diagram



# Nearest Neighbor Classification

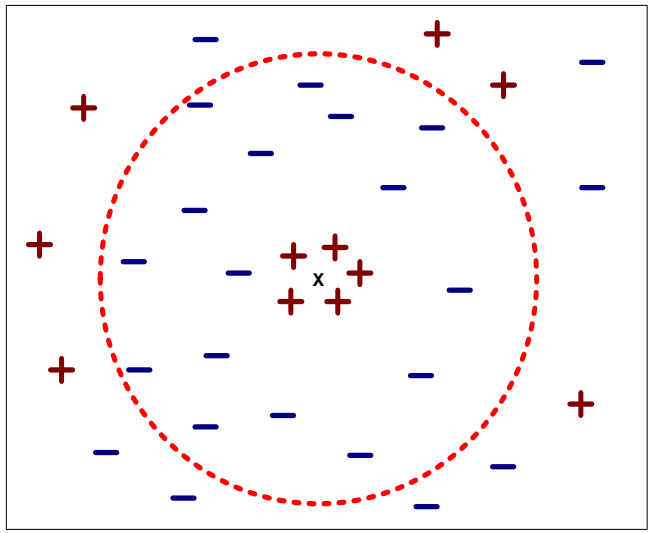
- Compute distance between two points:
  - Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor, $w = 1/d^2$

# Nearest Neighbor Classification…

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes
  - What if we have a tie?

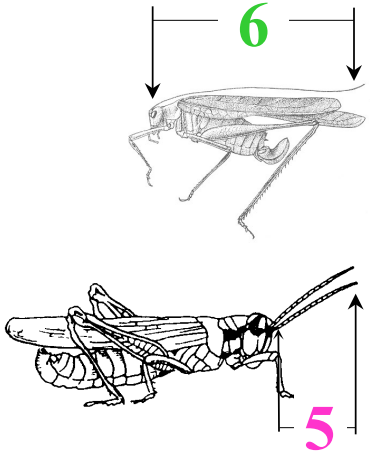


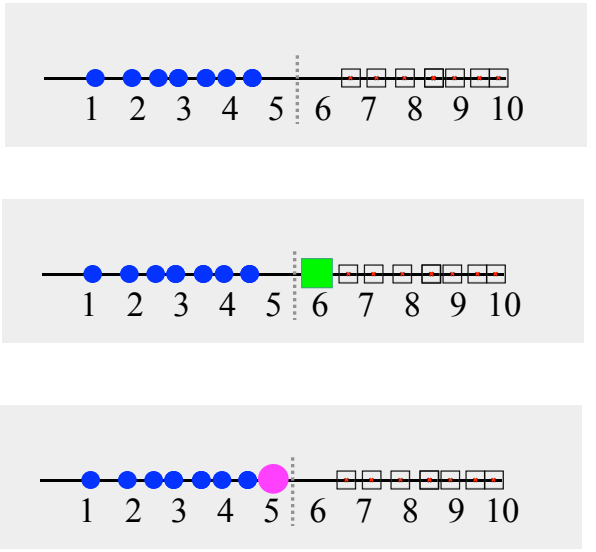
# The nearest neighbor algorithm is sensitive to irrelevant features…

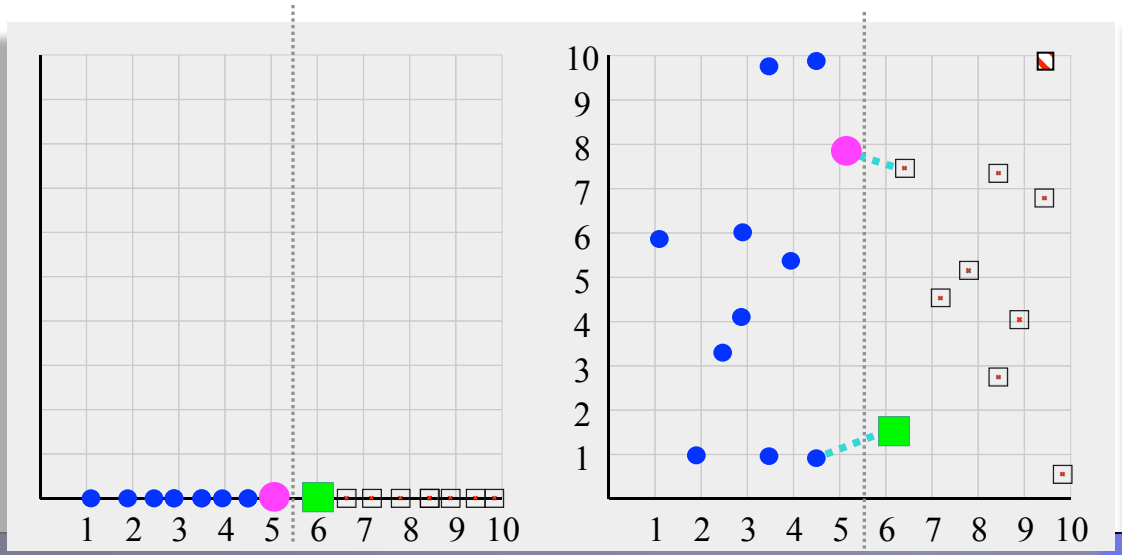
Suppose the following is true, if an insects antenna is longer than 5.5 it is a **Katydid**, otherwise it is a **Grasshopper**.

Using just the antenna length we get perfect classification!

Training data



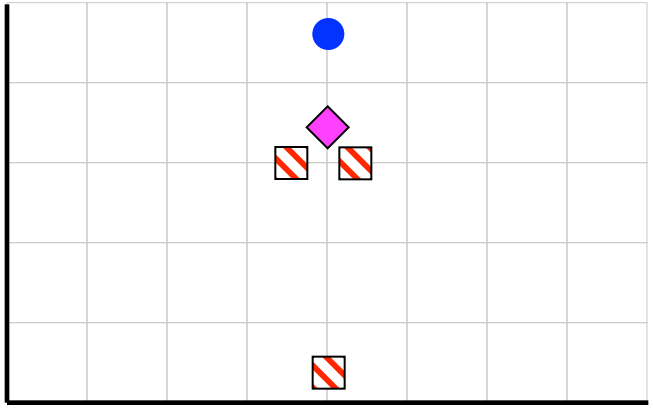

Suppose however, we add in an **irrelevant** feature, for example the insects mass.

Using both the antenna length and the insects mass with the 1-NN algorithm we get the wrong classification!

# How do we mitigate the nearest neighbor algorithms sensitivity to irrelevant features?

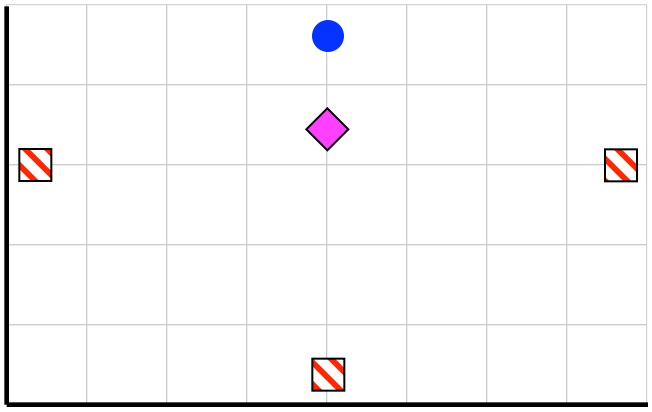
- Use more training instances
- Ask an expert what features are relevant to the task
- Use statistical tests to try to determine which features are useful
- Search over feature subsets

# The nearest neighbor algorithm is sensitive to the units of measurement



X axis measured in **centimeters**
Y axis measure in dollars

The nearest neighbor to the **pink** unknown instance is **red**.



X axis measured in **millimeters**
Y axis measure in dollars

The nearest neighbor to the **pink** unknown instance is **blue**.

One solution is to normalize the units to pure numbers.

# Scaling Issues

- Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
- Example:
  - height of a person may vary from 1.5m to 1.8m
  - weight of a person may vary from 90lb to 300lb
  - income of a person may vary from $10K to $1M

# Advantages/Disadvantages of Nearest Neighbor

- Advantages:
  - Simple to implement
  - Handles correlated features (Arbitrary class shapes)
  - Defined for any distance measure
  - Handles streaming data trivially
- Disadvantages:
  - Very sensitive to irrelevant features.
  - Slow classification time for large datasets
  - Works best for real valued datasets

– Does not build a model explicitly
  - "Lazy learners", as opposed to eager learners like decision tree induction