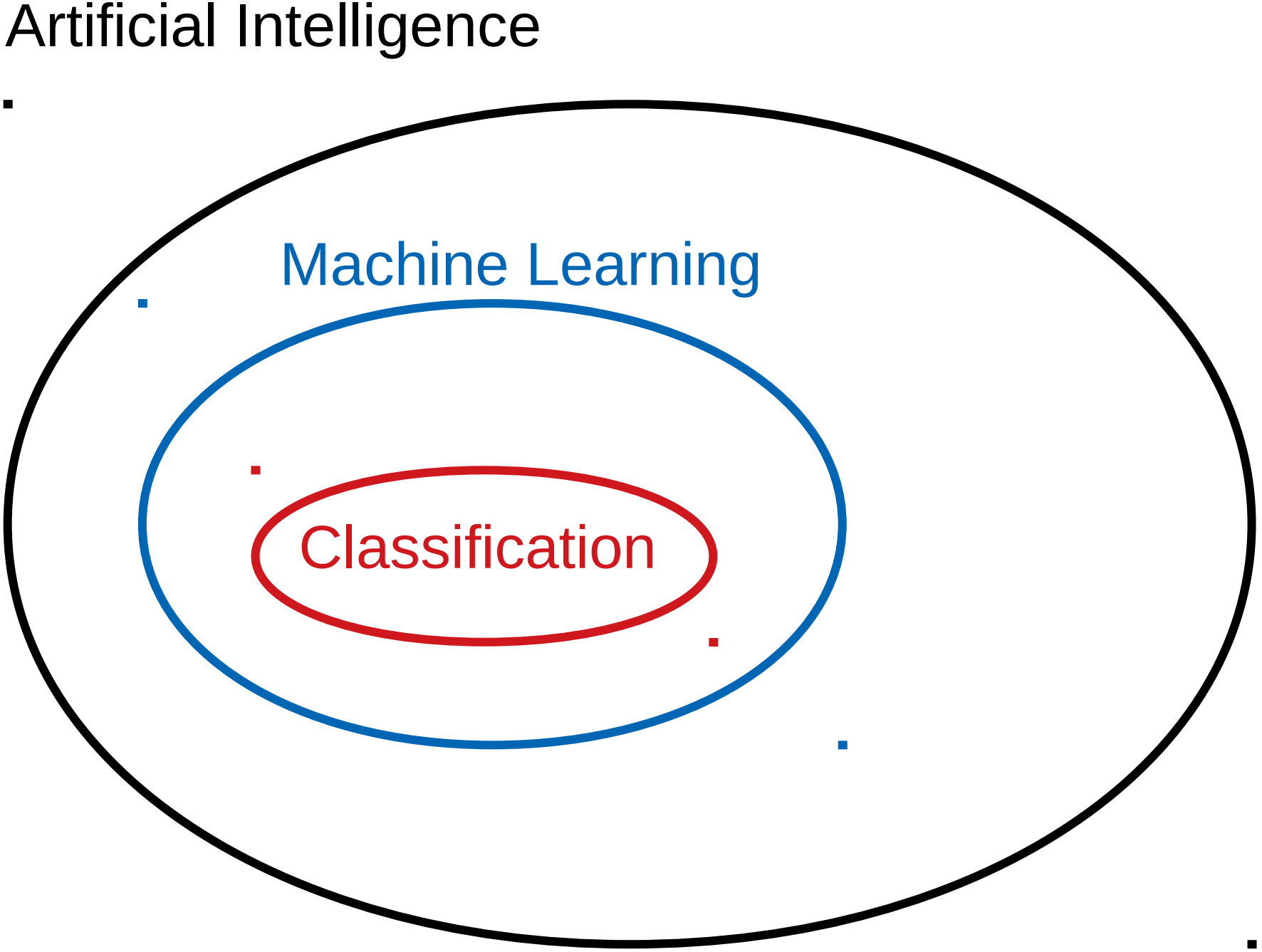
Introduction to Machine Learning:

Classification

An example classification problem

- This is the weather prediction of two cities for the next days:

Classify the prediction into **Alicante** or **Stockholm…**


Artificial Intelligence
Machine Learning
Classification

What part of this system is using Classification?

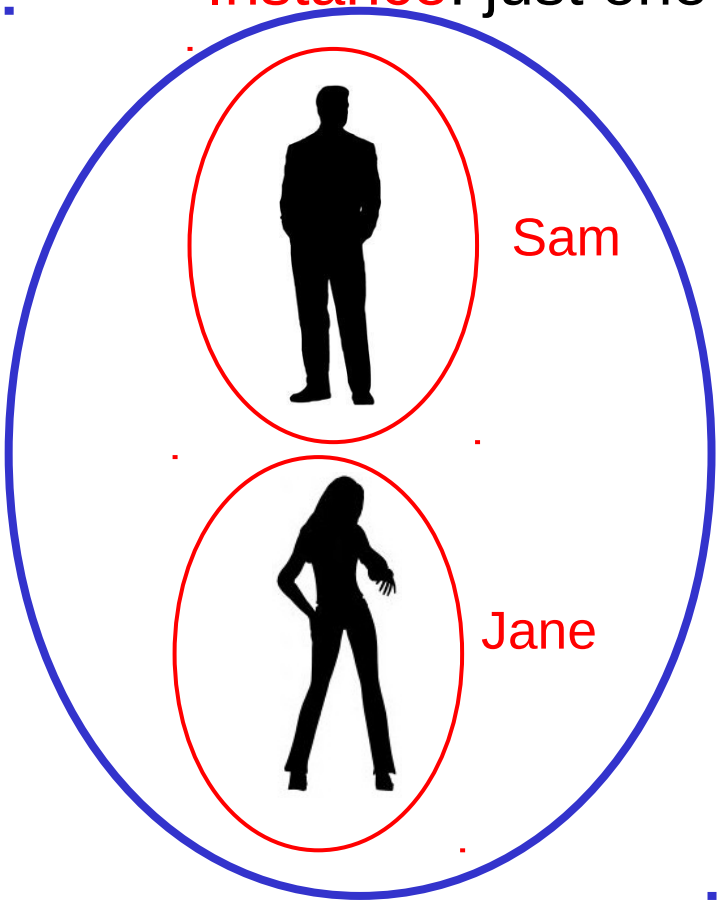
What does classification mean?

- We give labels to examples representing something:

Label: People

Classes and instances

- Class: group of examples sharing some characteristics
- Instance: just one concrete example



classes/categories

Representing the examples

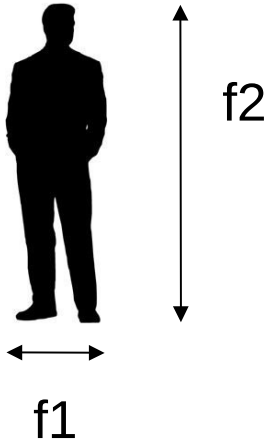
- Computers work with numbers. Transform examples into numbers. How? → Feature extraction

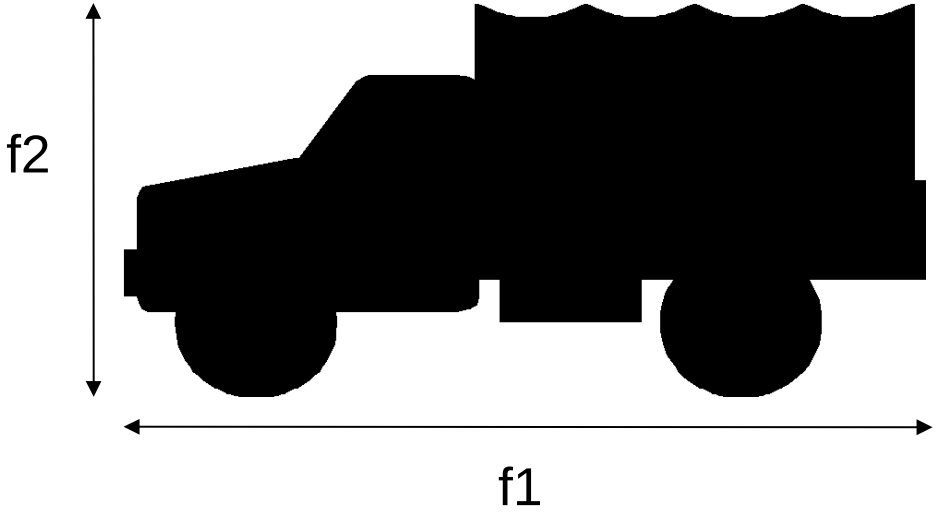
People

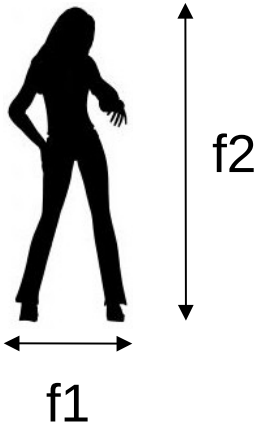
Vehicles

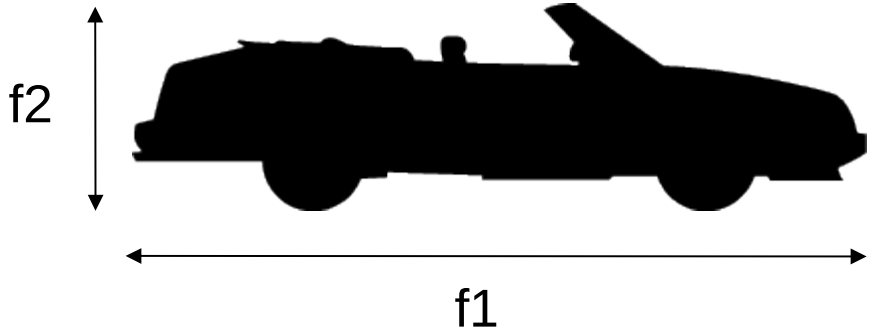
Representing the examples

- Computers work with numbers. Transform examples into numbers. How? → Feature extraction





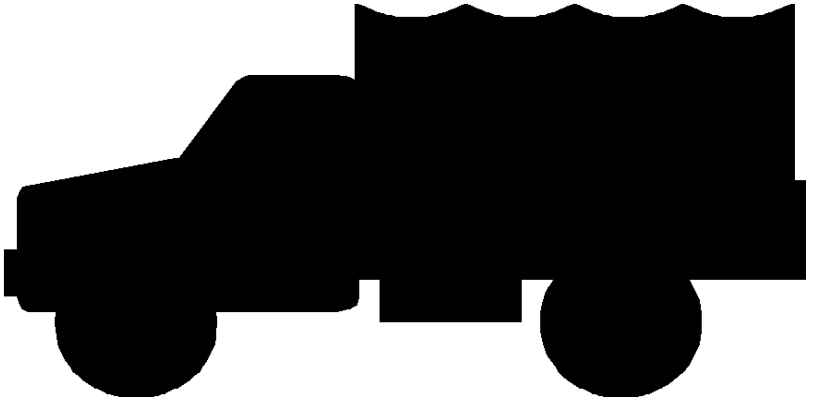




Feature vector

- Each example is now represented by its set of features

x1 = { f1, f2 }



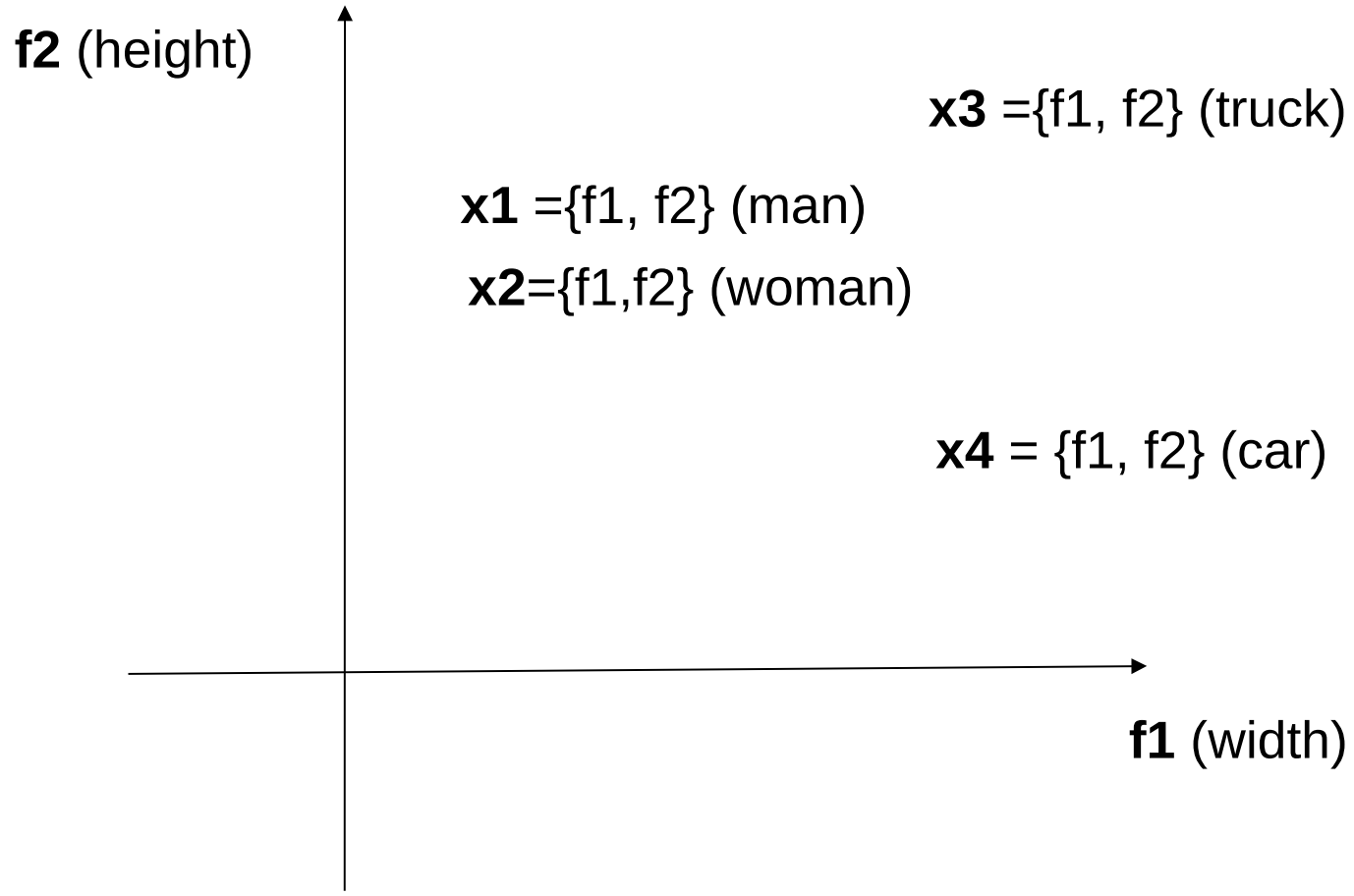
x3 = { f1, f2 }

x2 = { f1, f2 }

x4 = { f1, f2 }

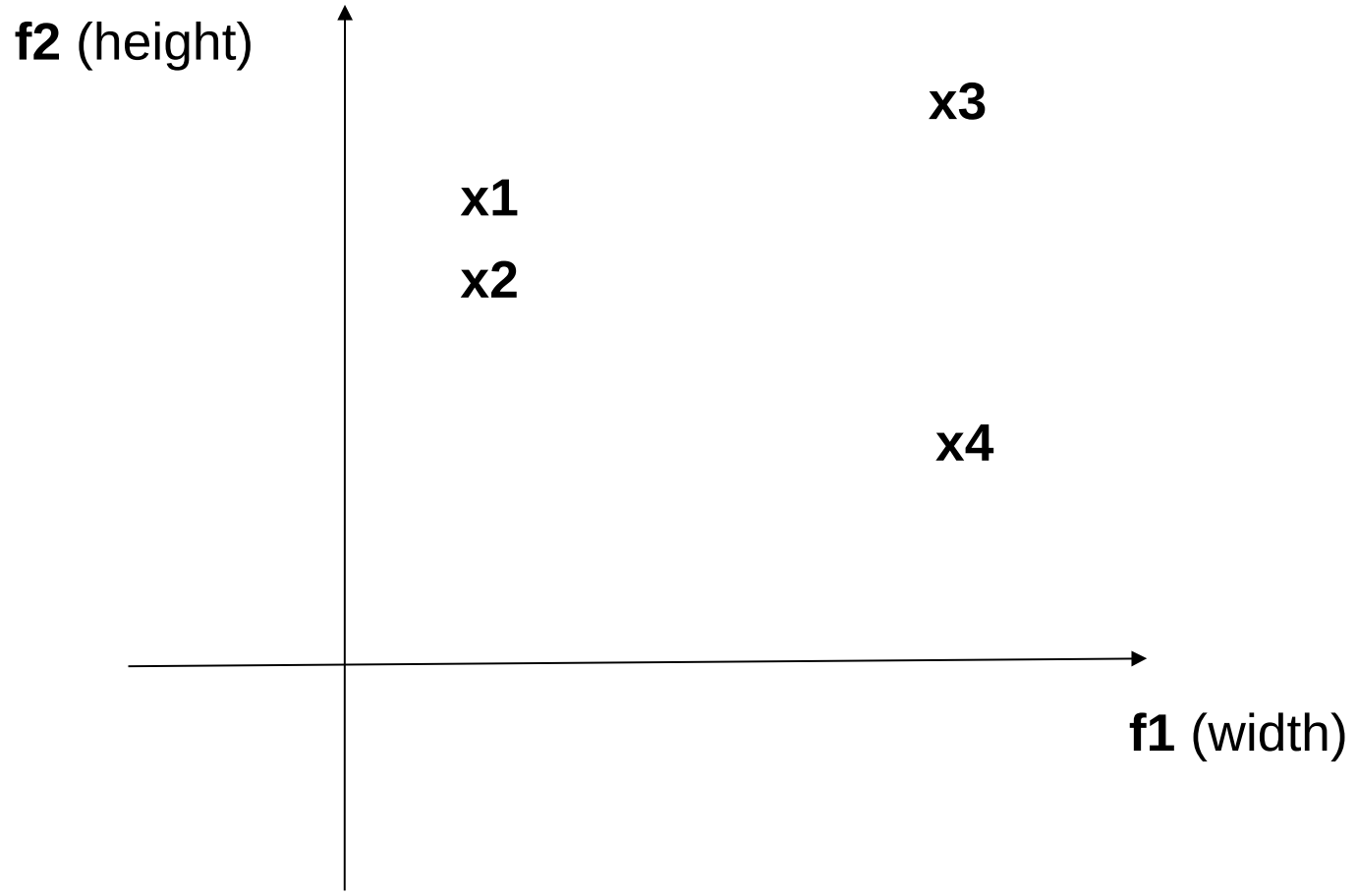
Feature space

- We plot the feature vectors into the feature space which will be used for the classification algorithms.



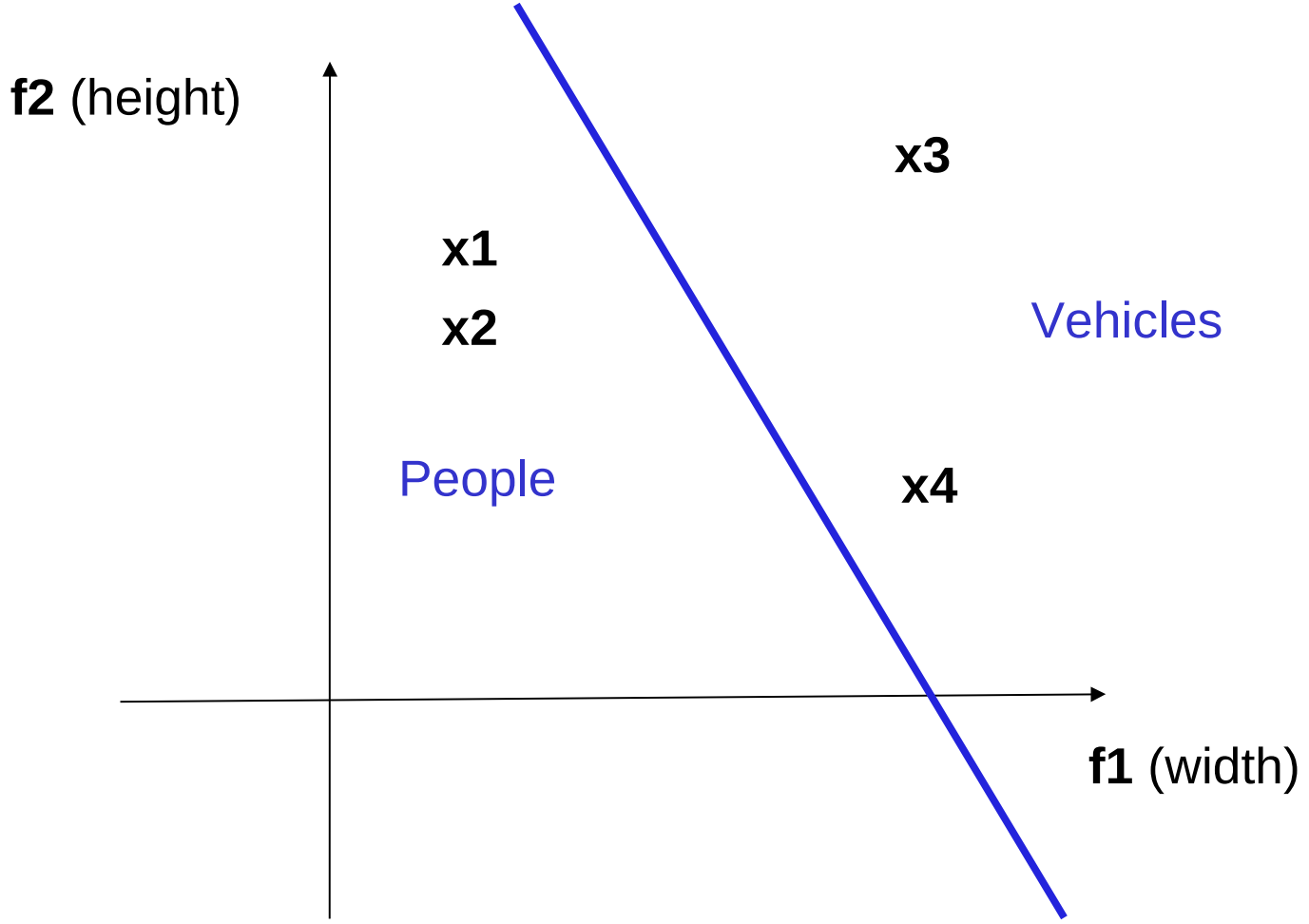
General idea of classification

- Find a function that separates the classes in the feature space.



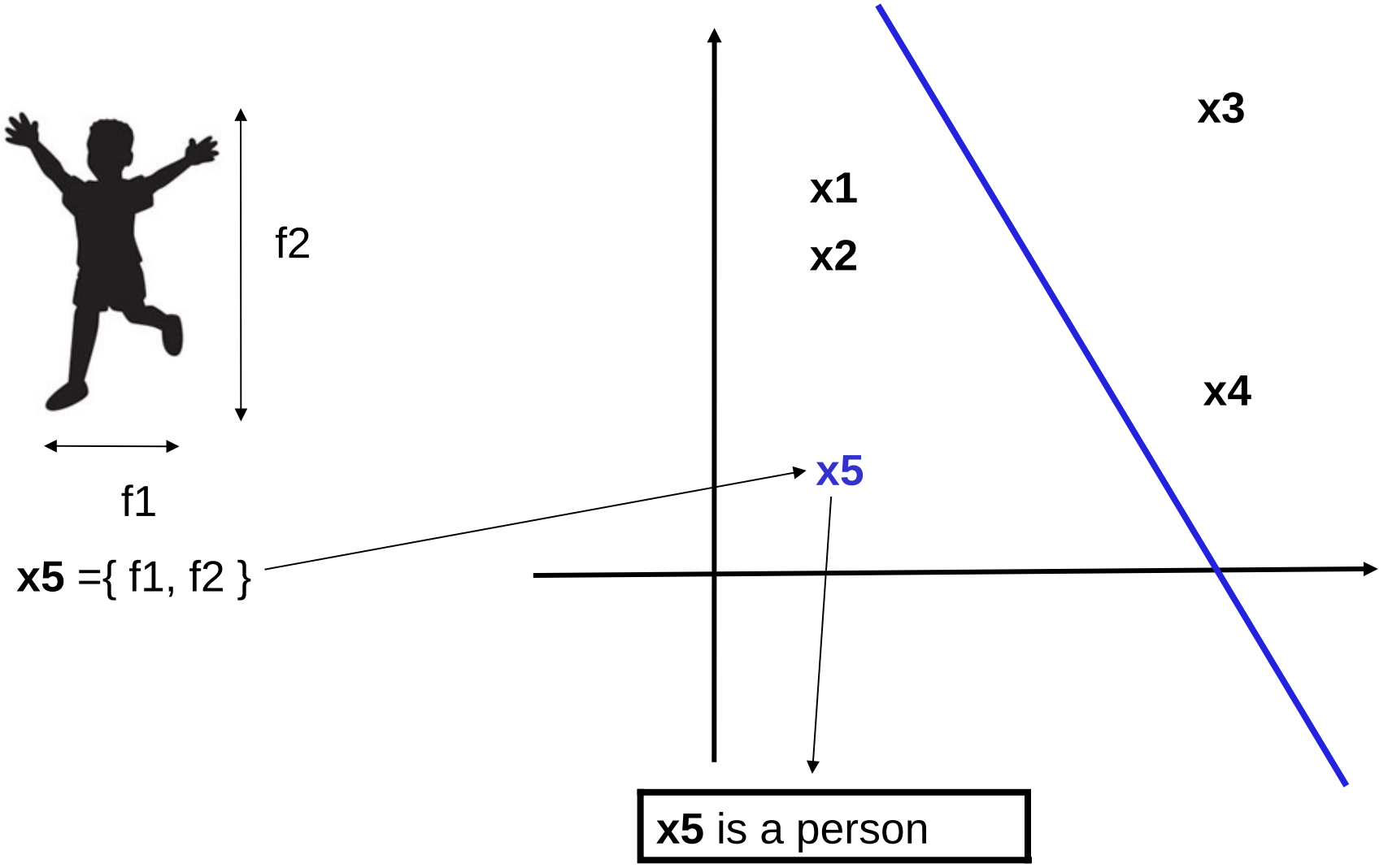
General idea of classification

- Find a function that separates the classes in the feature space.



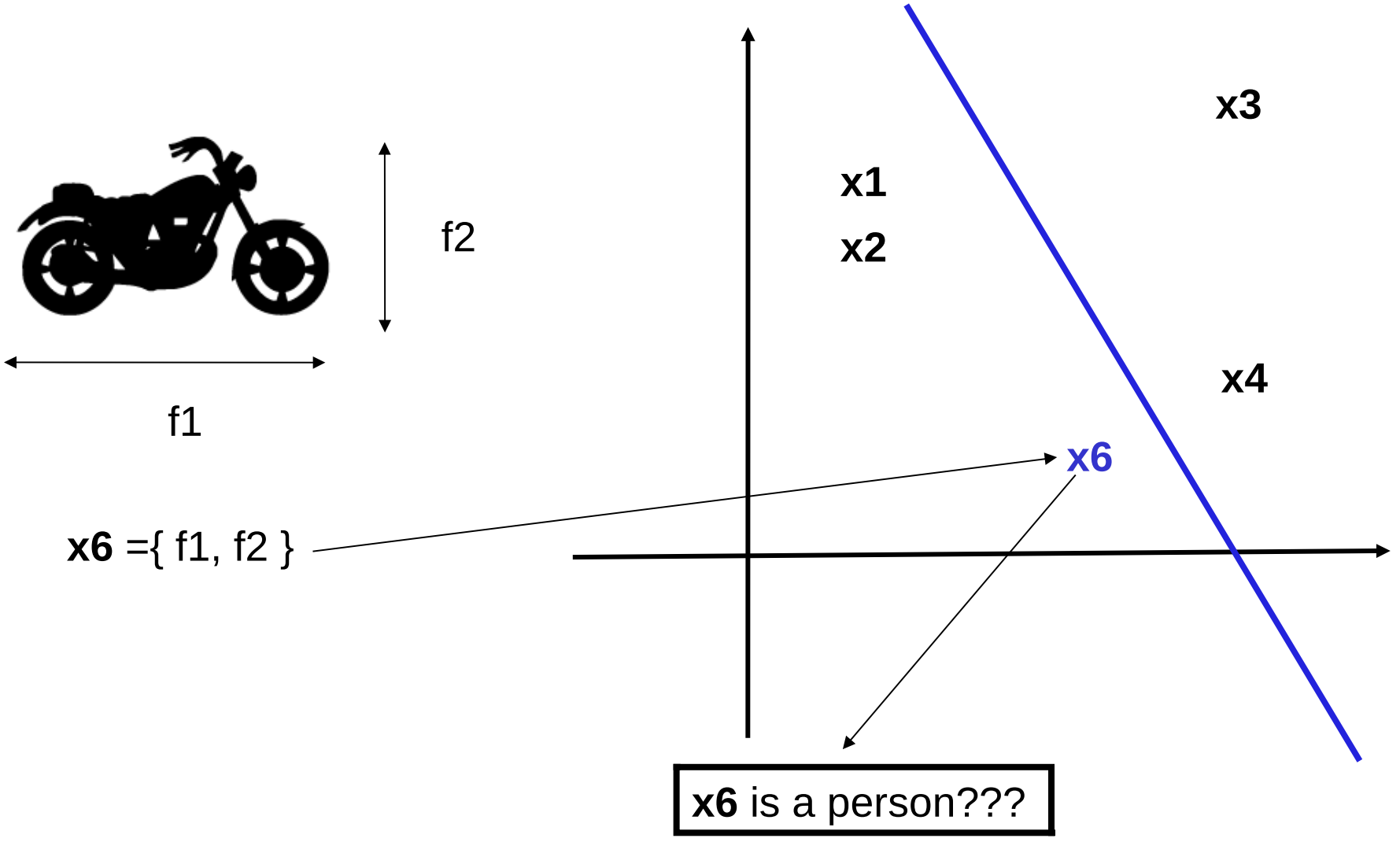
New classifications

- Once we have the function we try to classify new examples in the feature space



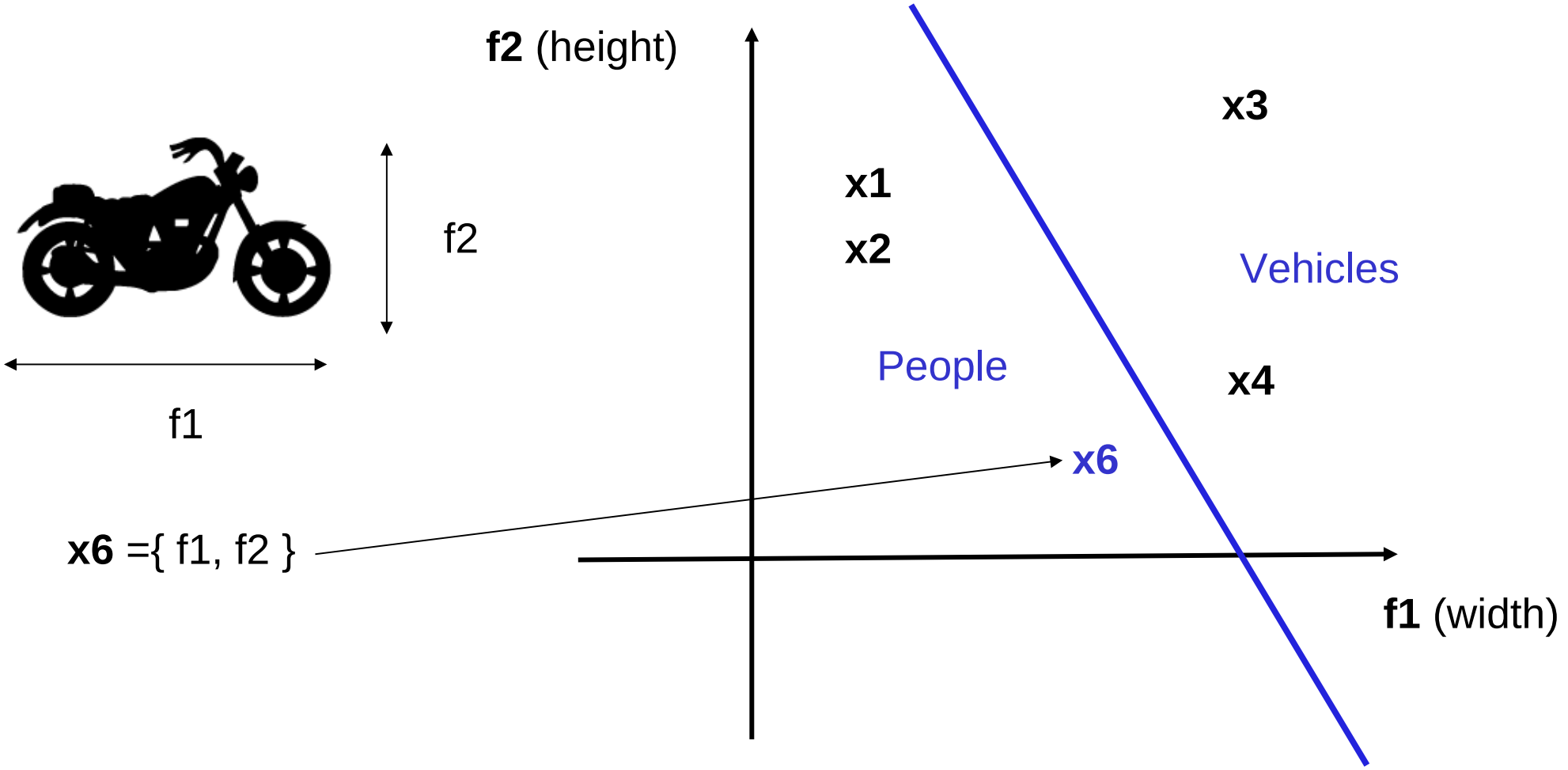
New classifications

- Classifiers are **not perfect**: errors in the classification



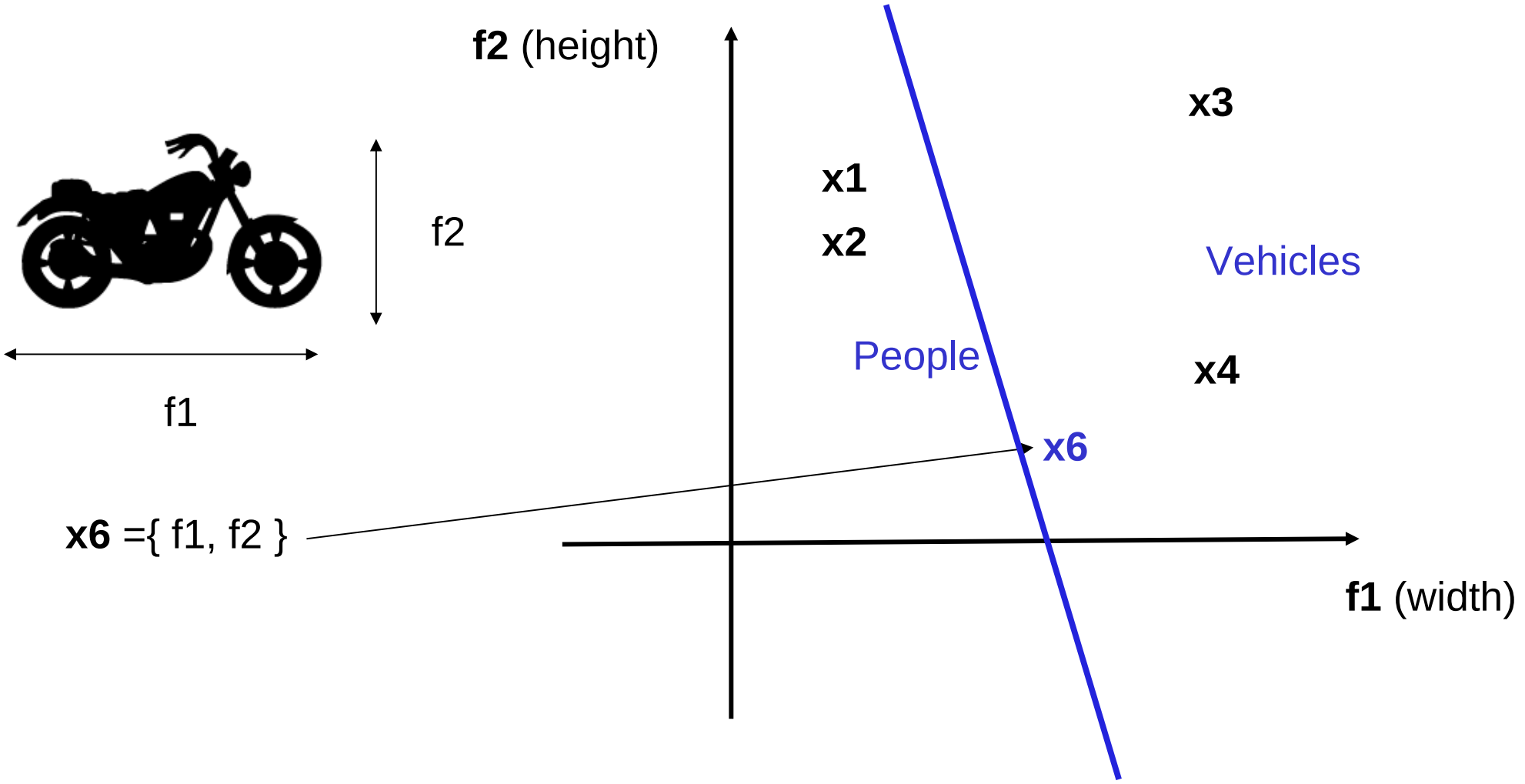
New classifications

- What can we do now?



New classifications

- Re-train using the wrong classifier examples



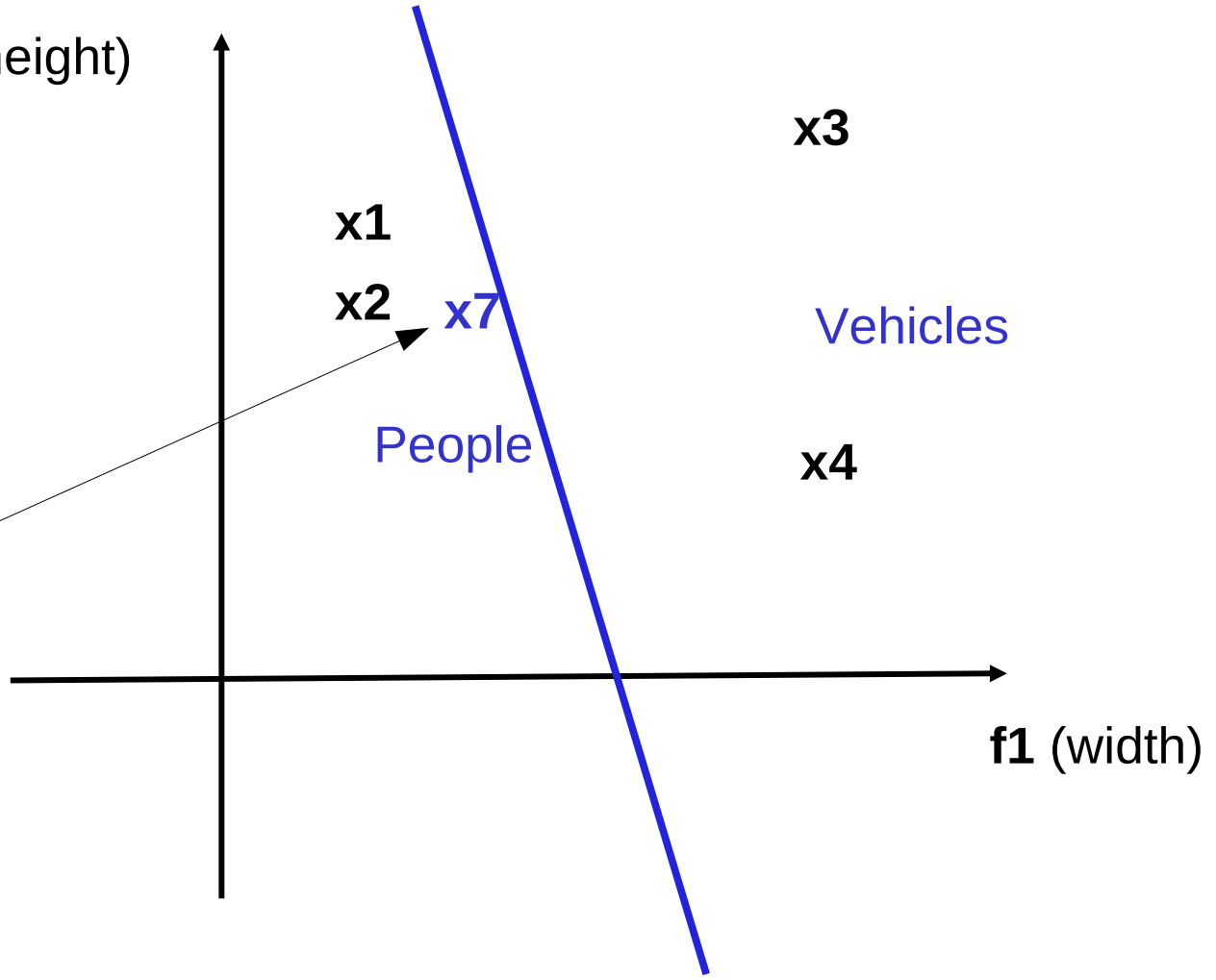
New classifications

- New thing

f2

f1

x7 ={ f1, f2 }



New classifications

- Impossible to reach a perfect classifier...Why?

f2

f1

x7 ={ f1, f2 }

f2 (height)

x1

x2

x7

x3

Vehicles

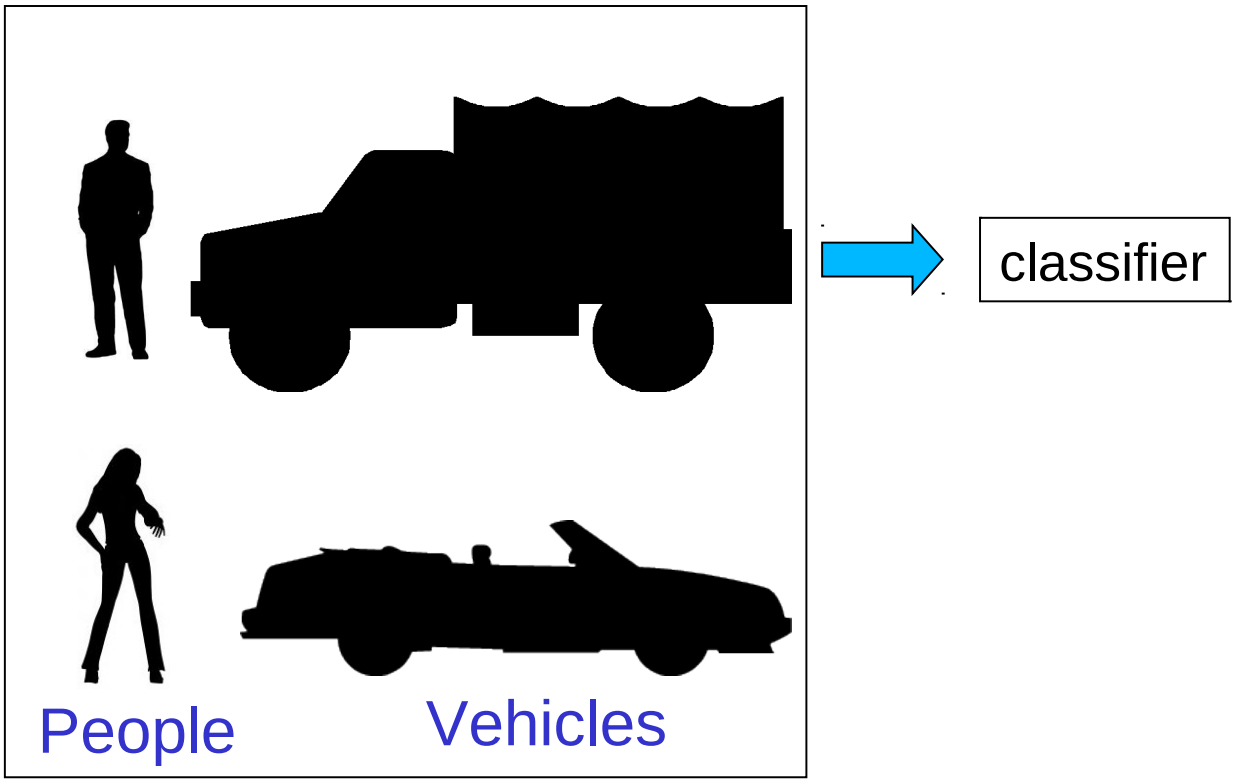
People

x4

f1 (width)

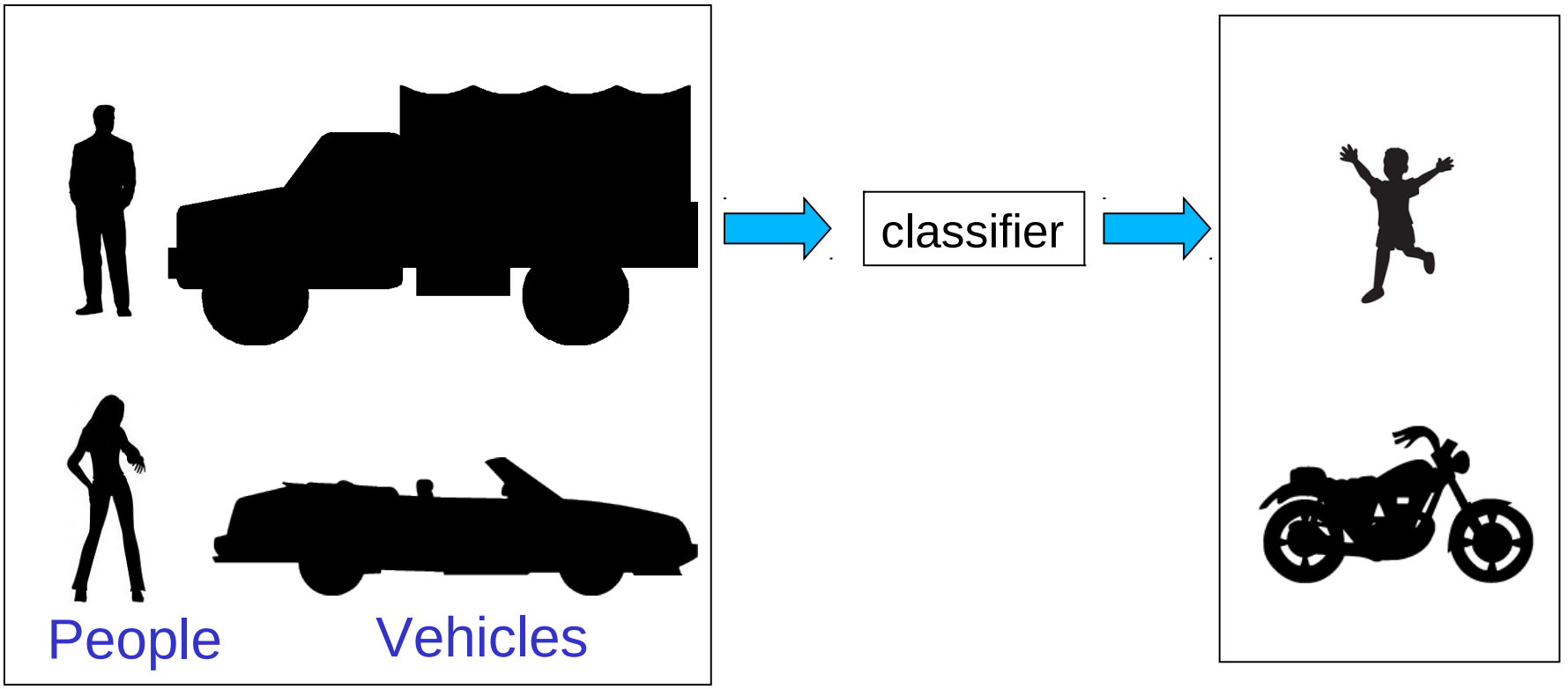
Training set

- Is the set of labelled examples that is used to create the classifier (obtain the function in feature space).
- This process is called training.



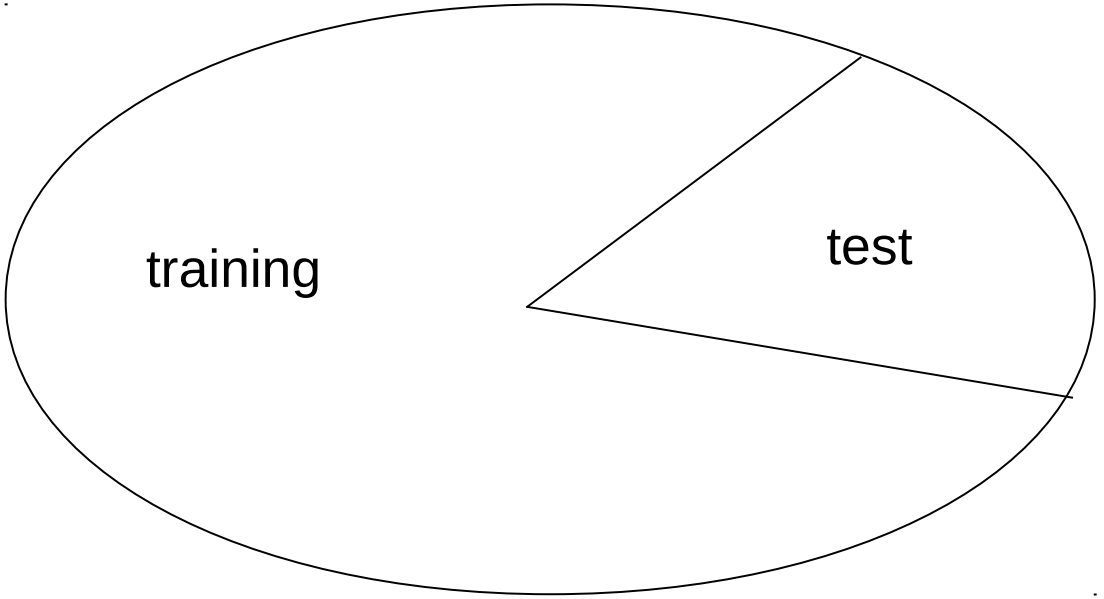
Test set

- Is the set of new examples that we want to classify (assign a label)



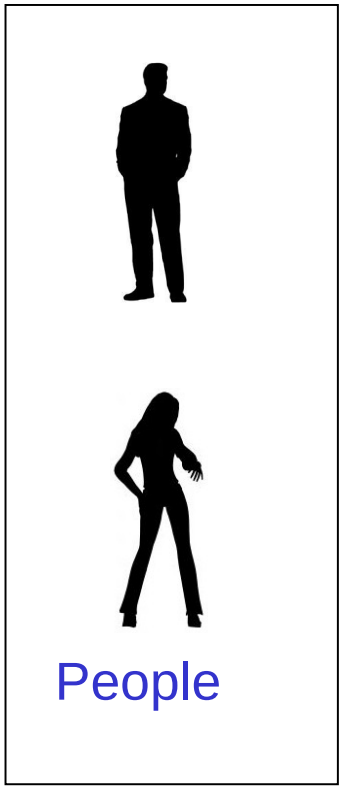
Measuring the quality of a classifier

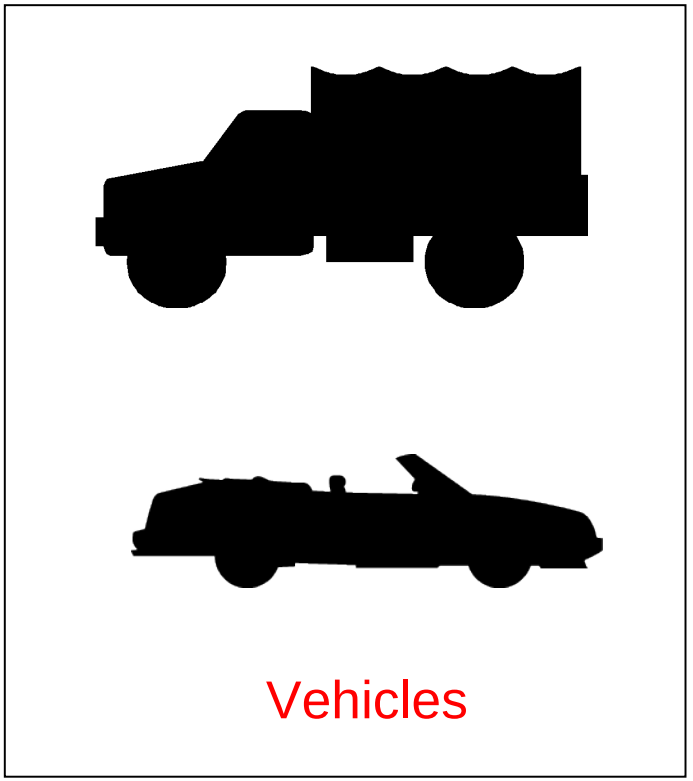
- We use the test set to measure the quality of the resulting classifier. Of course the examples of the test set need to be labelled to be able to get statistics.
- Usually a set of labelled examples is divided into training (~70%) and test (~30%) sets.



Binary classifier

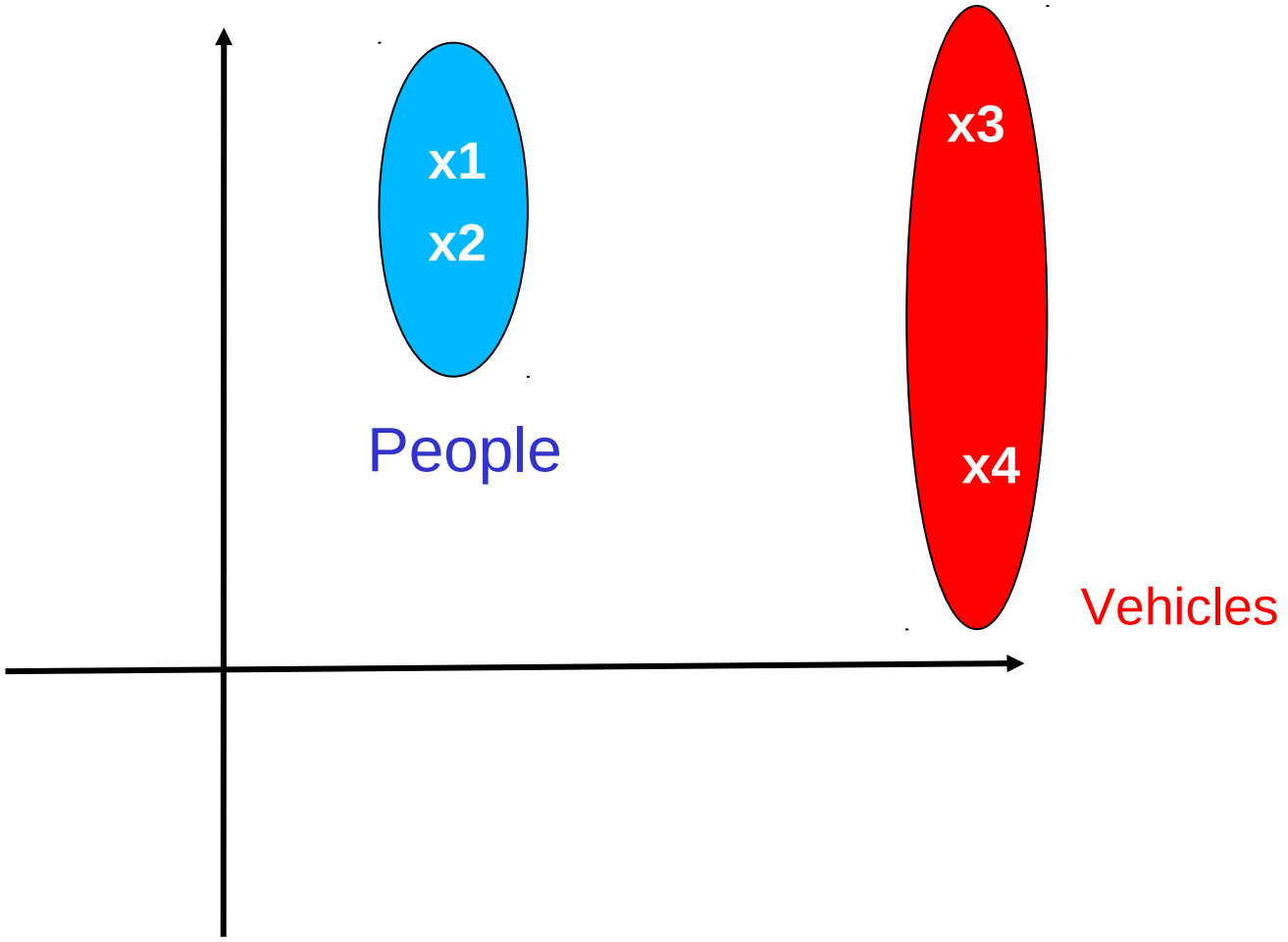
- A binary classifier distinguishes two classes
- Example: **people** vs **vehicle** classifier





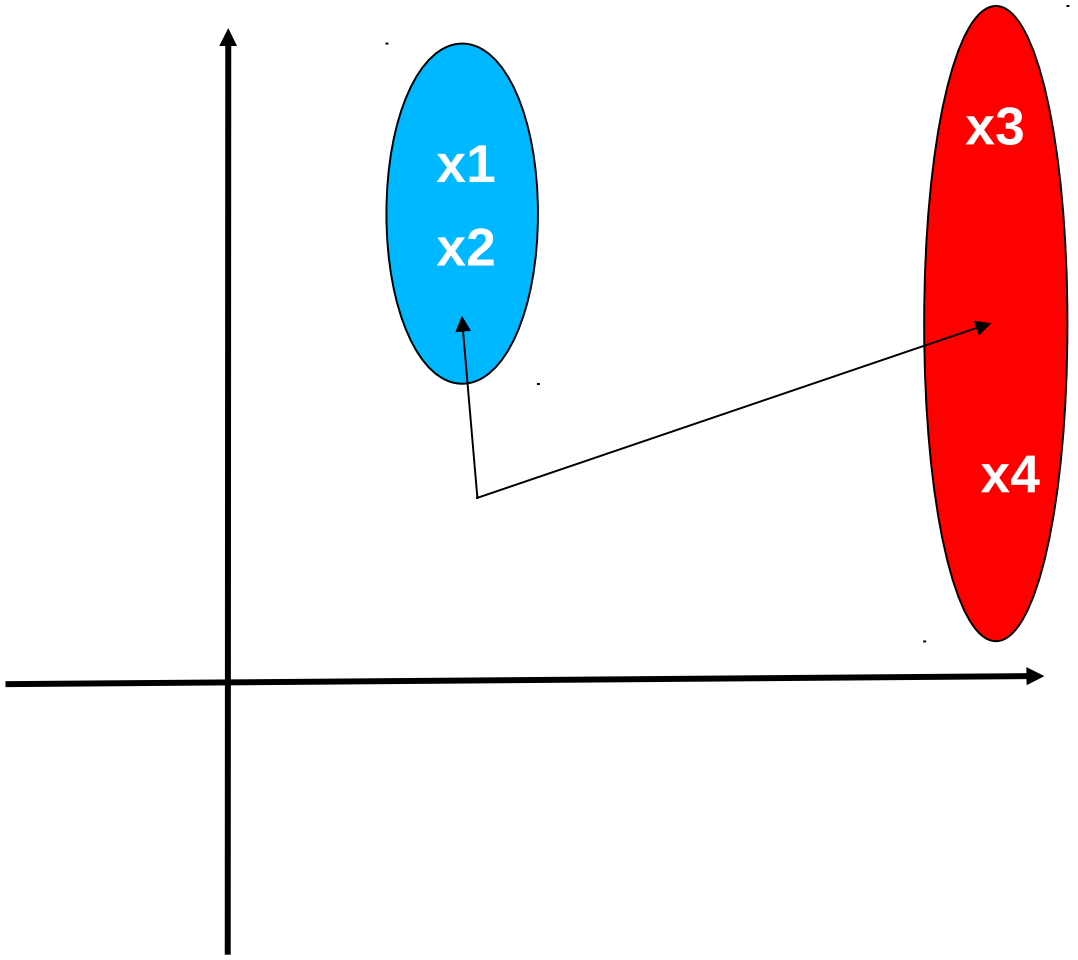
Example: Binary classifier based on Bayes

- Classes: w_1, w_2 represented by Gaussians.

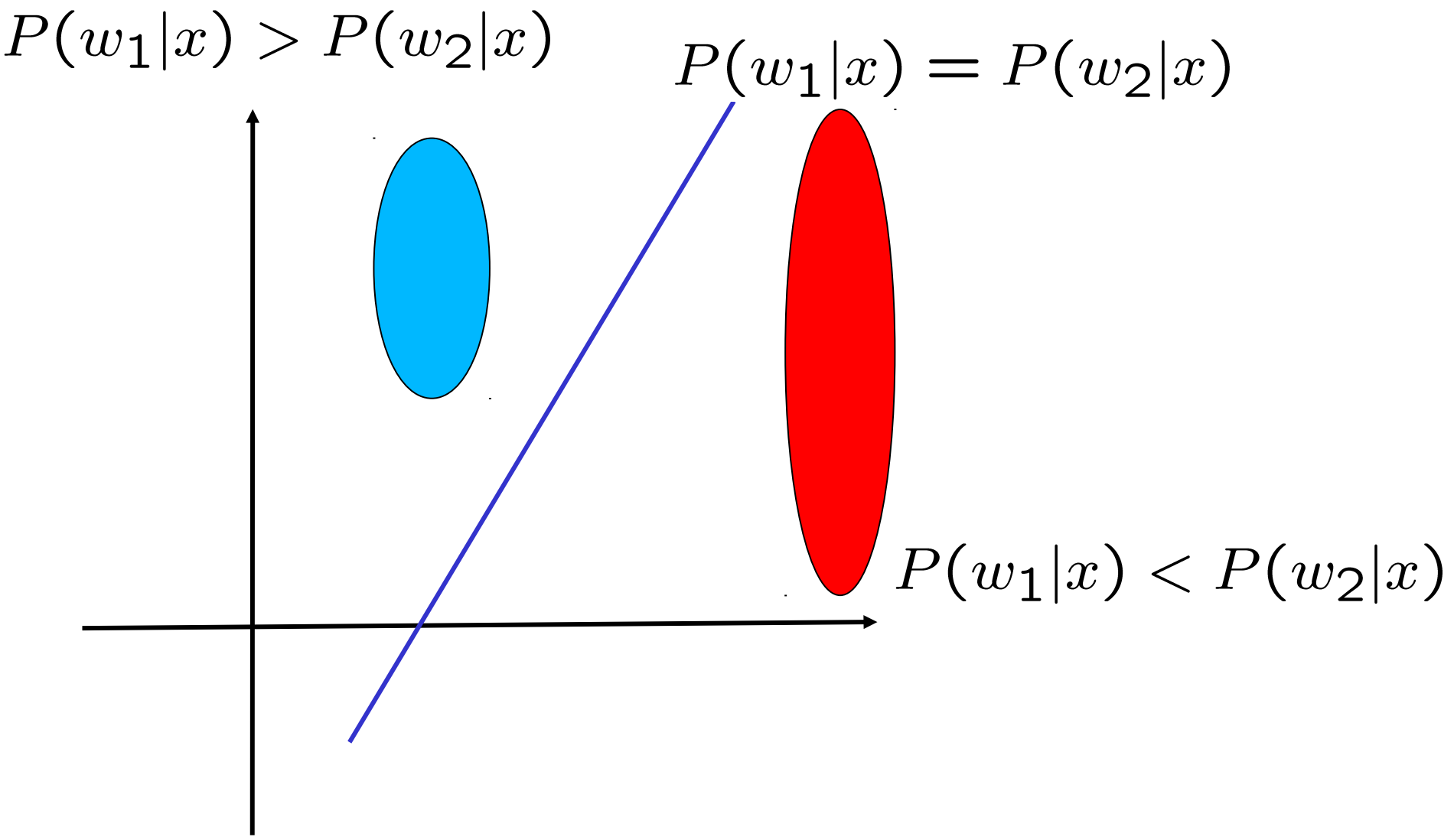


Example: Binary classifier based on Bayes

- Classes: w_1, w_2 represented by Gaussians.
- Example x is assigned to class w_1 if $P(|w_1|x) > P(w_2|x)$



Decision hyperplane



Bayesian classification

x belongs to w_1 if $P(w_1|x) > P(w_2|x)$

Applying Bayes' theorem:

$$\frac{P(x|w_1)P(w_1)}{P(x)} > \frac{P(x|w_2)P(w_2)}{P(x)}$$

$$P(x|w_1)P(w_1) > P(x|w_2)P(w_2)$$

Assuming same priors for w_i (equiprobable classes):

$$P(x|w_1) > P(x|w_2)$$

Bayesian classification using Gaussians

$$P(x|w_i) = \frac{1}{(2\pi)^{l/2}\,|\sum_i|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_i)^T \sum_i{}^{-1}(x-\mu_i)\right)$$

Assuming same covariance matrices $\sum_1 = \sum_2$
then x pertains to w_1 if:

$$-\frac{1}{2}(x-\mu_1)^T \sum_1 (x-\mu_1) > -\frac{1}{2}(x-\mu_2)^T \sum_1 (x-\mu_2)$$

which is a comparison of distances from point to gaussians.

Naive Bayes

$$-\tfrac{1}{2}(x-\mu_1)^T\sum_1(x-\mu_1) > -\tfrac{1}{2}(x-\mu_2)^T\sum_1(x-\mu_2)$$

Problem: $\sum_i$ difficult to approximate when the dimension l of $x \in \Re^l$ is very big.

Naive Bayes

Solution: assume independent features. We reduce one l-dimensional problem to l 1-dimensional problems:

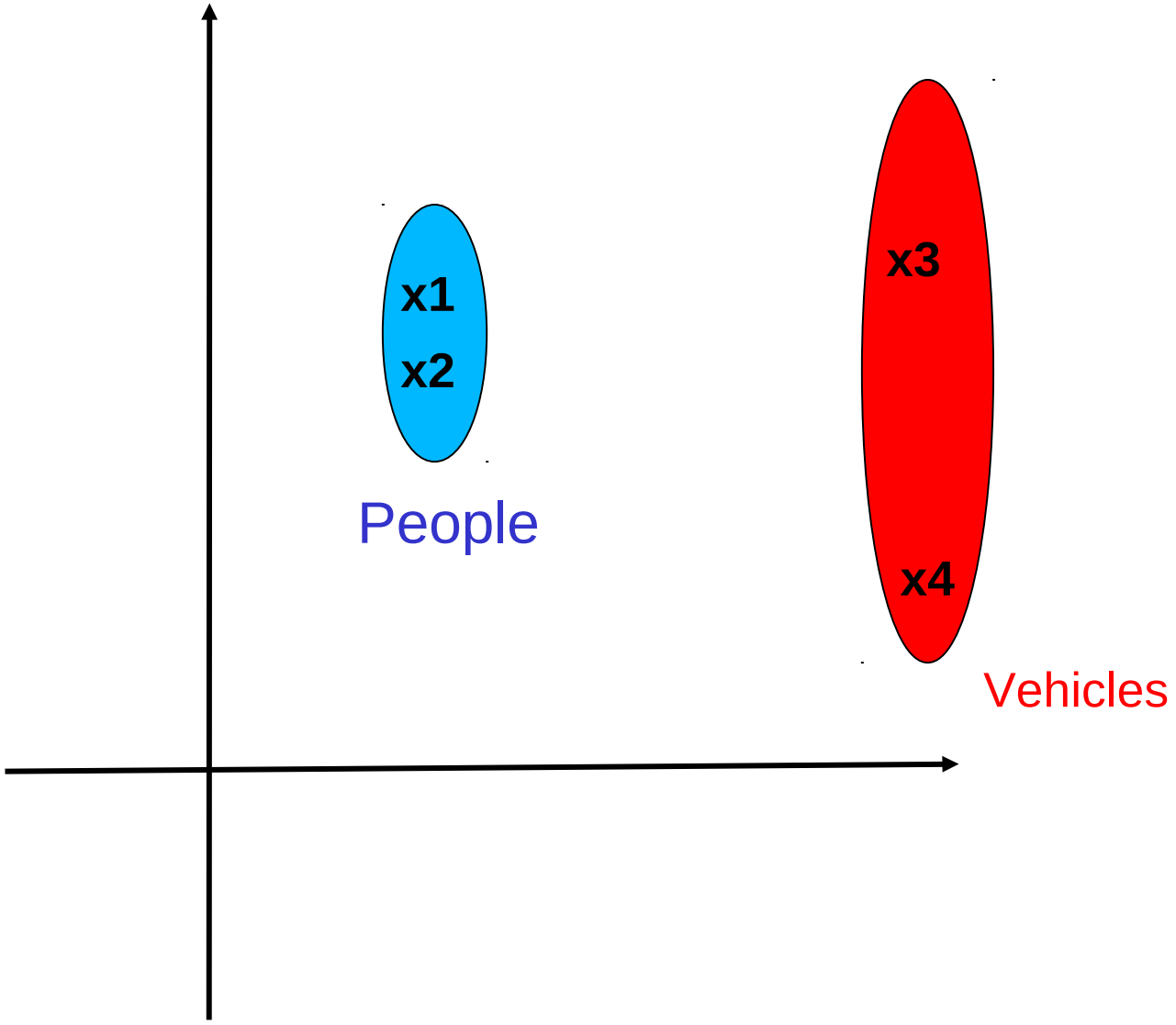
$$P(w_i|x) = P(w_i) \prod_j^l P(f_j|w_i), \quad x = \{f_1, f_2, ..., f_l\}$$

Each feature is represented by a 1-dimensional Gaussian.

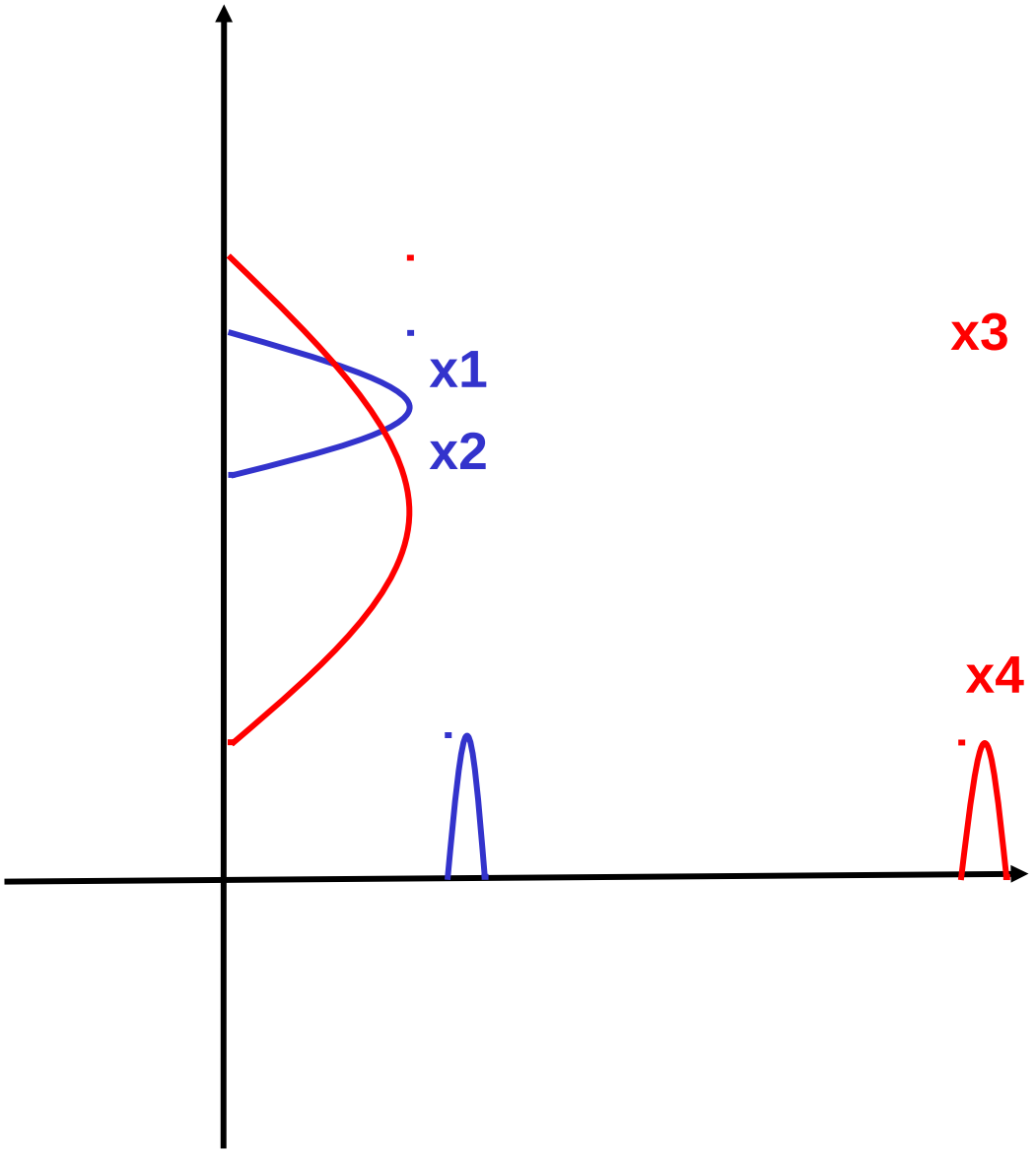
Given a new point its classification assuming same priors is:

$$w = \underset{w_i}{\text{argmax}} \prod_j^l P(f_j|w_i)$$

Naive Bayes



Naive Bayes



Estimating probabilities in Naive Bayes

Each class w_i is represented by l normal distributions in the form:

$$w_i \sim N(\mu_{i1}, \sigma_{i1}) \ldots N(\mu_{il}, \sigma_{il})$$

If we have N training examples $x_1, ..., x_N$ then:

$$\mu_{ij} = \frac{1}{N_i} \sum_{x_n} f_j, \quad \forall x_n, label(x_n) = w_i$$

where N_i is the number of elements in the training data with label w_i.

Classifying with Naive Bayes

Given a new example $x = \{f_1, ..., f_j\}$, it will be assigned the class w_i such as:

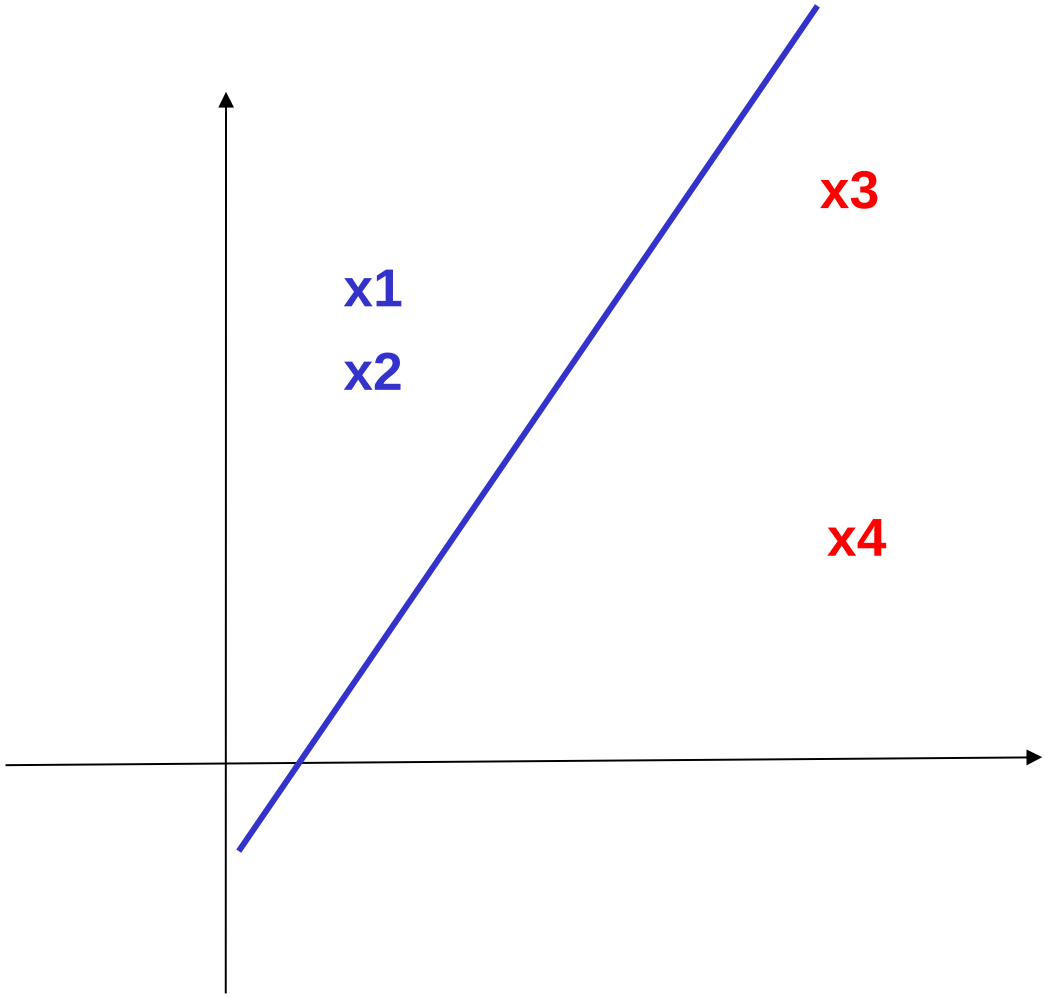
$$w = \underset{w_i}{\text{argmax}} \prod_{j}^{l} P(f_j|w_i)$$

with

$$P(f_j|w_i) = \frac{1}{\left(2\pi\sigma^2\right)^{\frac{1}{2}}} \exp -\frac{(x - \mu_{ij})^2}{2\sigma^2}$$

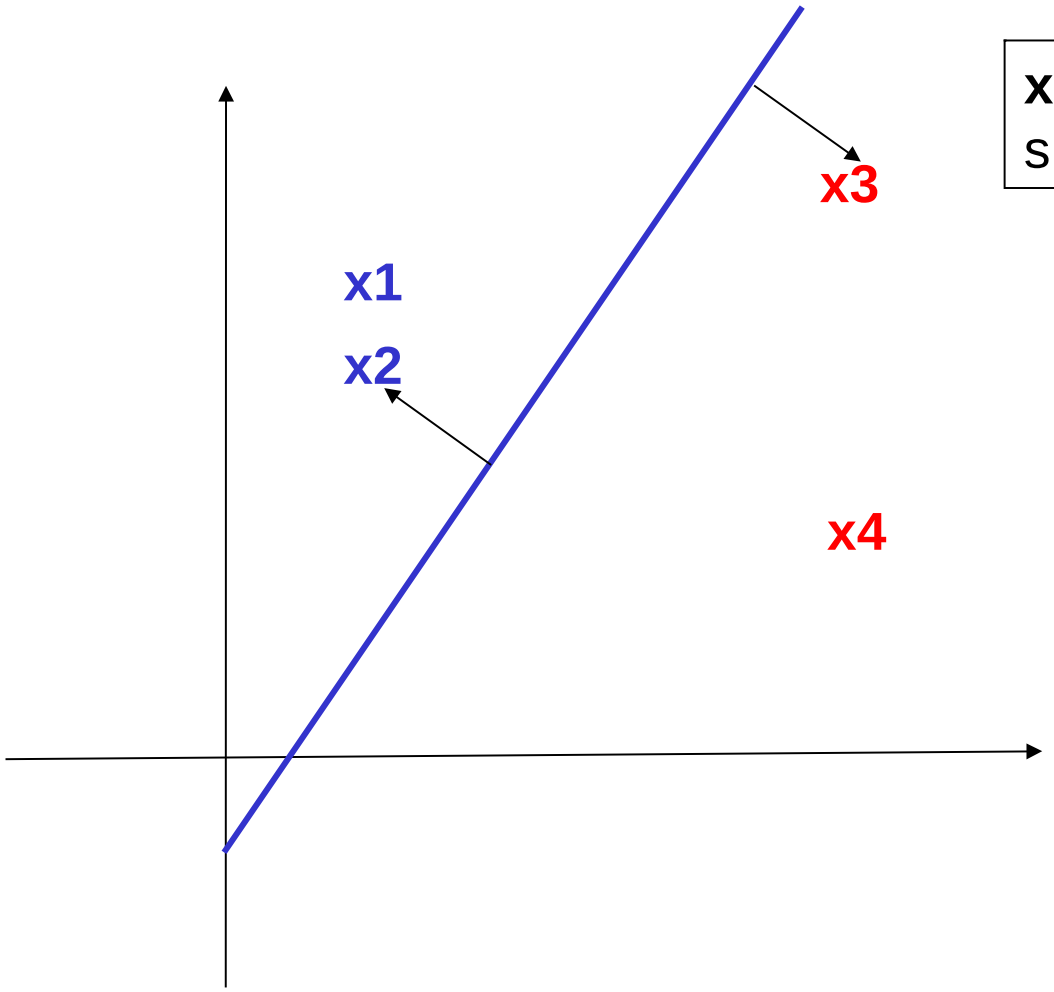
Linear classifiers

- Do not assume any model for the classes w_i, and just look for a hyperplane dividing the data



Support Vector Machines

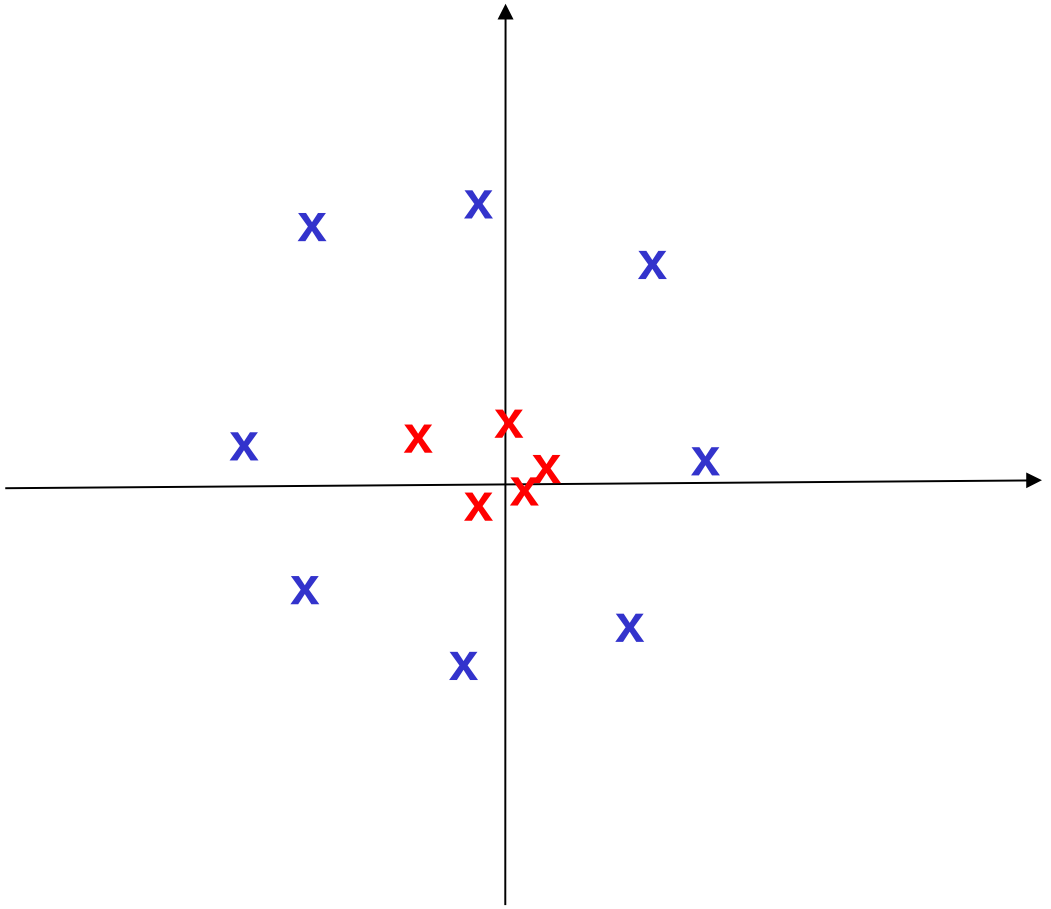
- Finds the hyperplane which optimizes the distance between the two classes



x2 and **x3** all called support vectors

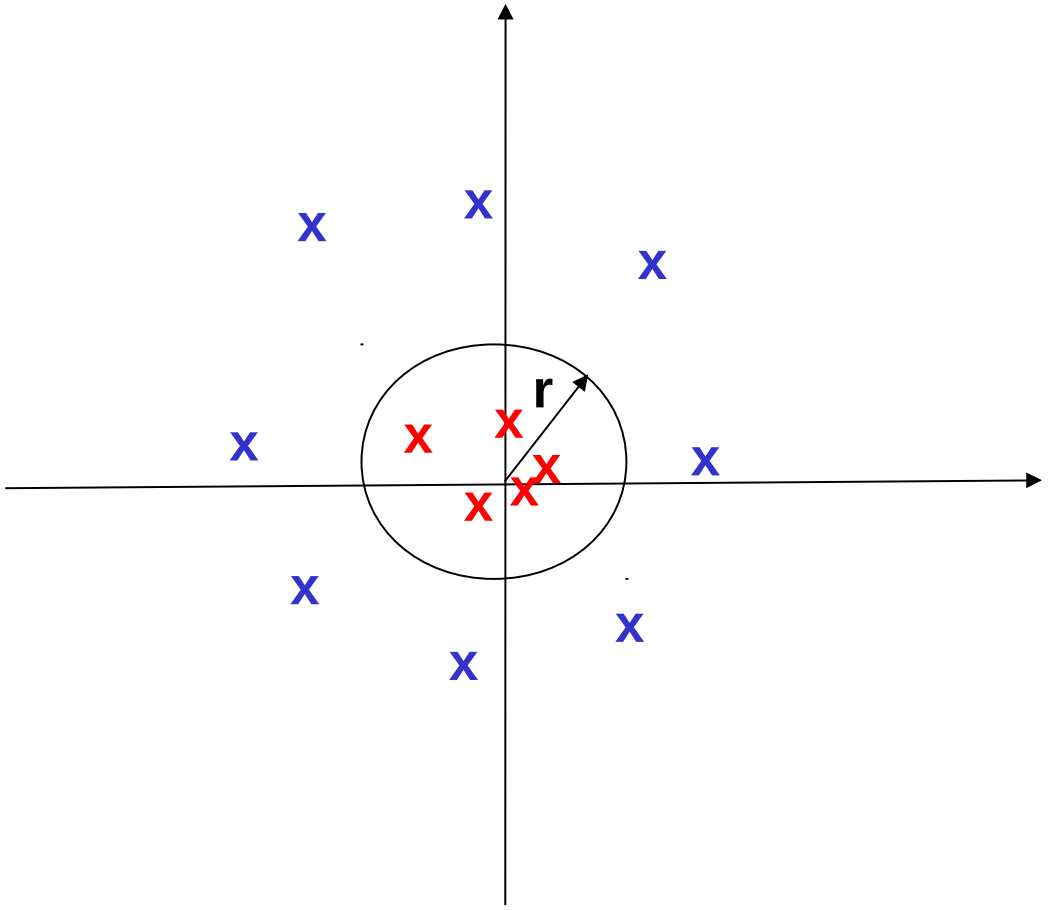
Nonlinear separable

- If the classes are nonlinear separable we can apply a transformation in the feature space.



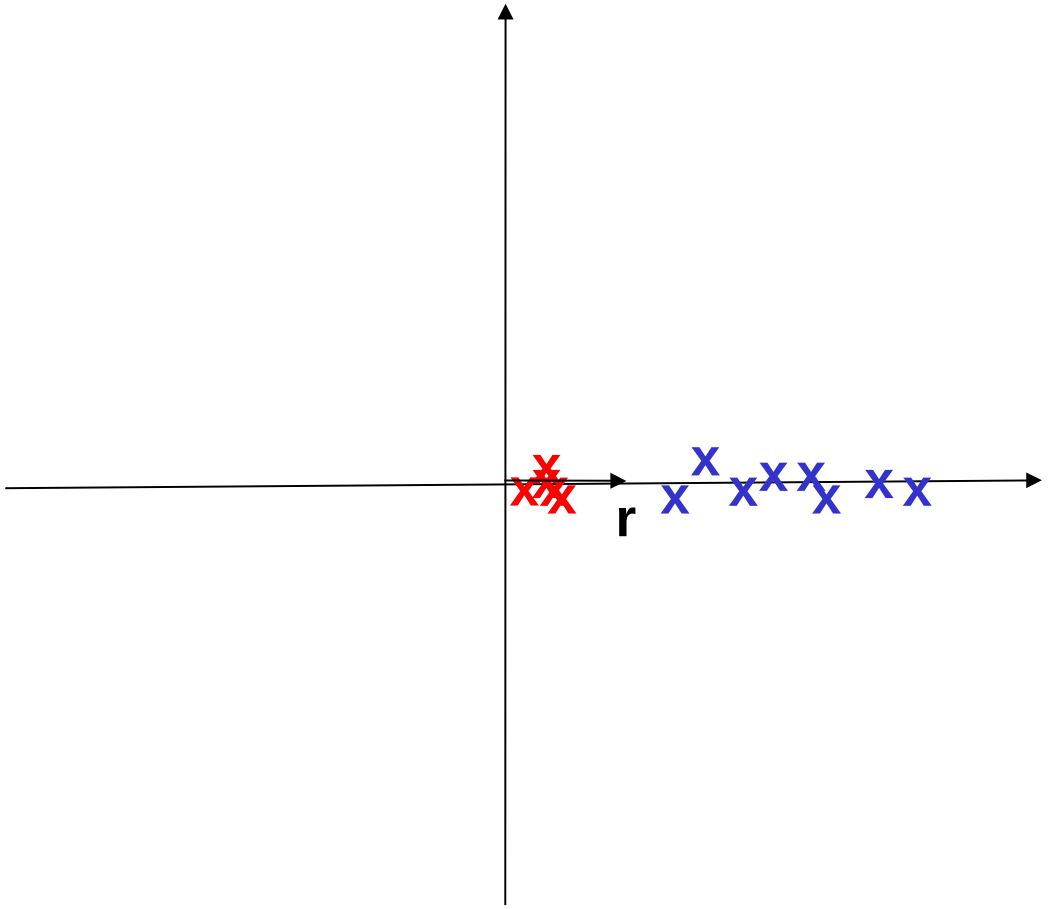
Simple transformation

- Euclidean distance to the centre



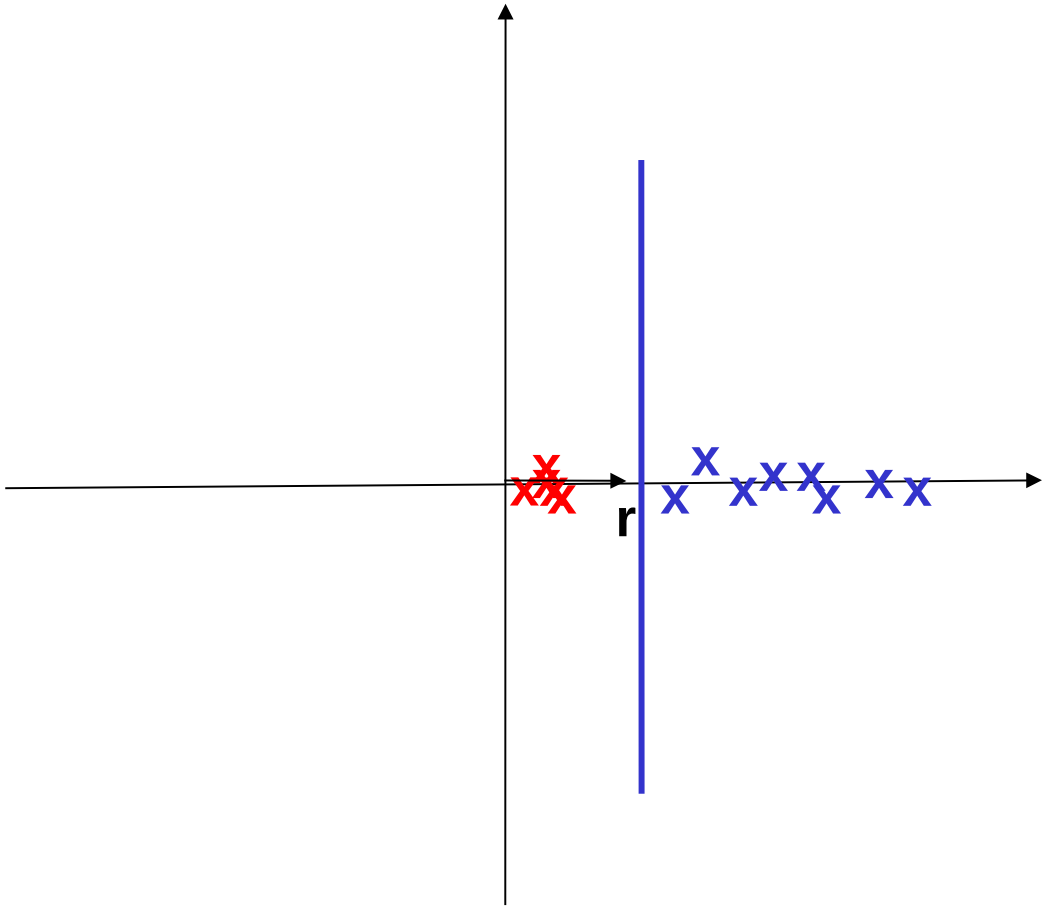
Simple transformation

- Euclidean distance to the centre



Simple transformation

- Euclidean distance to the centre



Nonlinear separable

- If the classes are nonlinear separable we can apply a transformation in the feature space.
- Usually a radial function is used in the form:

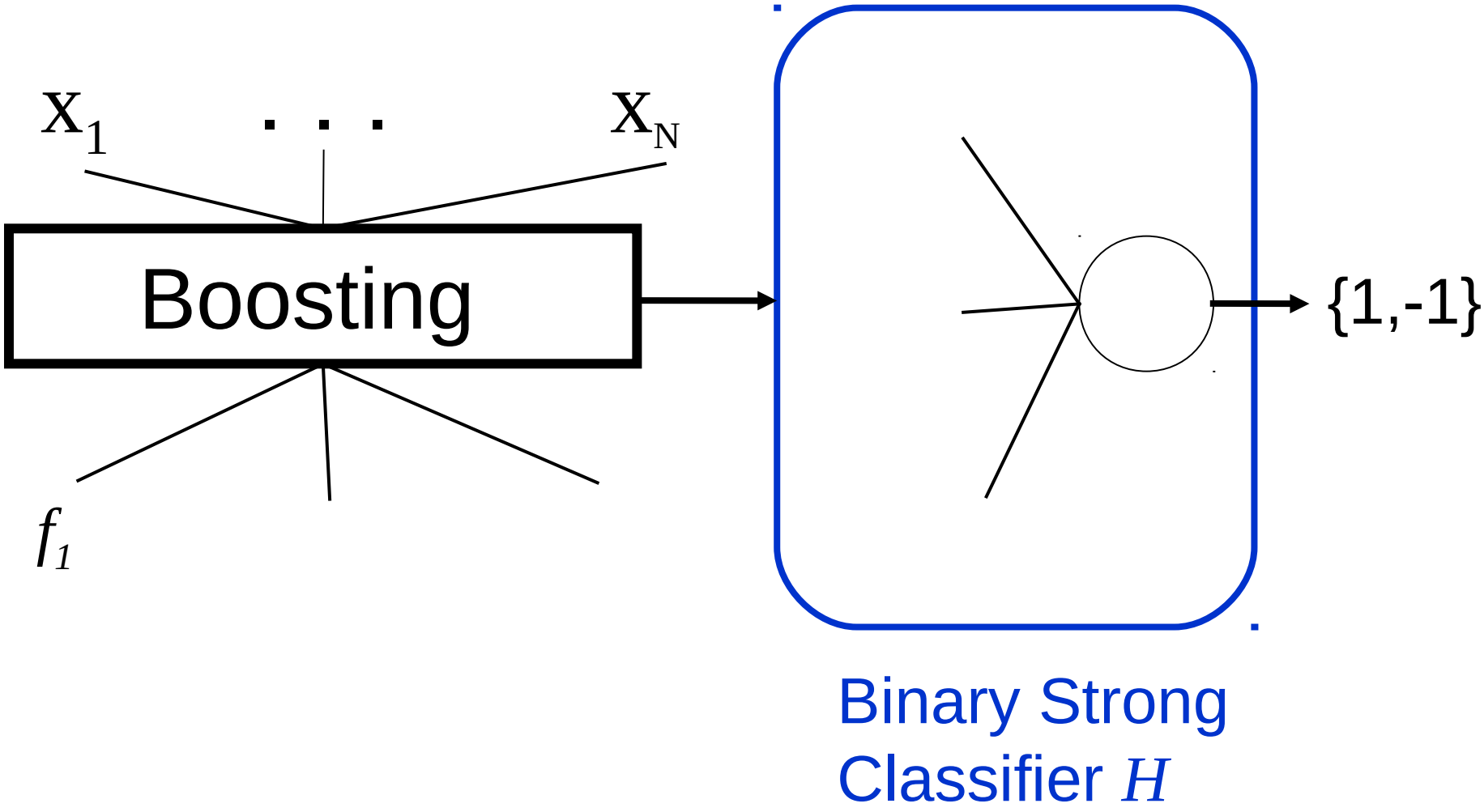
$$f(||x - c_i||)$$

- Typical used function in SVM has a Gaussian form:

$$f(x) = exp\left(-\frac{1}{2\sigma_i^2}||x - c_i||^2\right)$$

Boosting

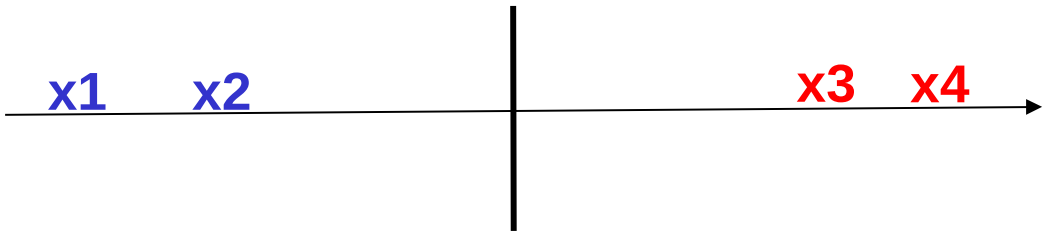
- Meta classifier that improves the result of other classifiers called weak classifiers h_i



Simple weak classifiers for boosting

- For each feature f_j we create a weak classifier h_j as

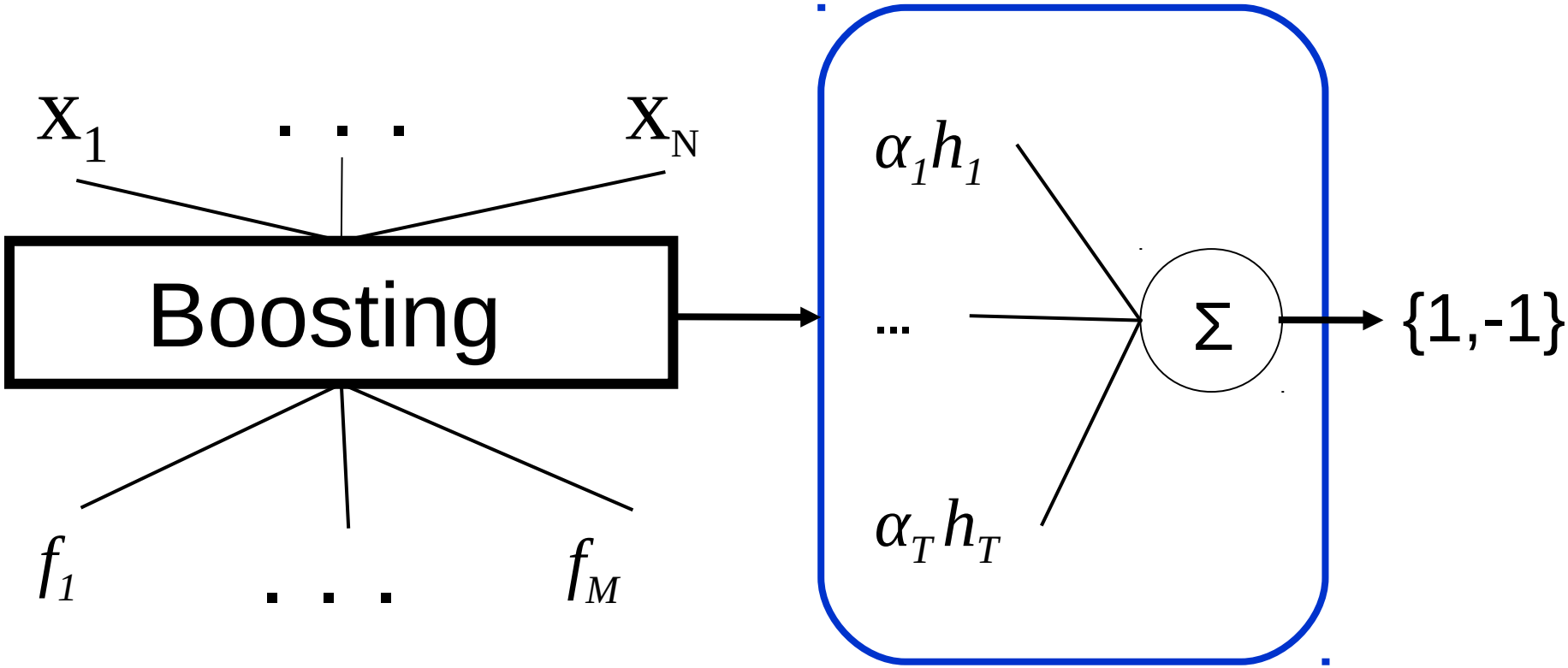
$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ -1 & \text{otherwise.} \end{cases}$$



- Where θ_j is a threshold and p_j indicates the direction of the inequality (> or <)

Boosting

- Weak classifiers are combined to form a strong classifier



$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ -1 & \text{otherwise.} \end{cases}$$

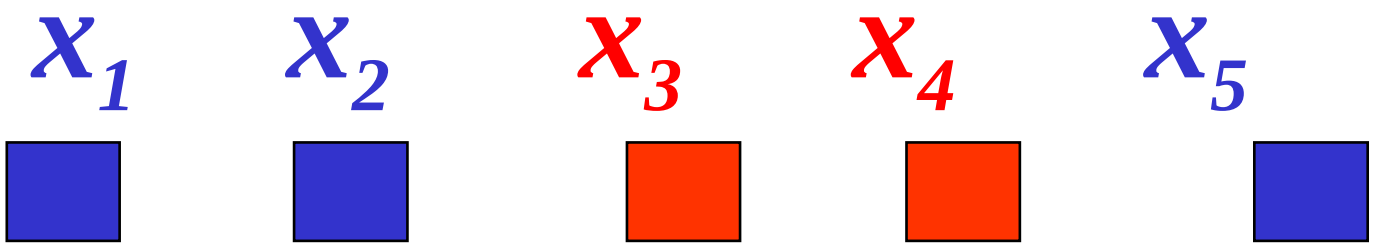
Binary Strong Classifier H

- Input: set of examples $(x_1, y_1), \ldots, (x_N, y_N)$.
- Let m be the number of negatives examples and l be the number of positive examples. Initialize weights $w_{1,n} = \frac{1}{2m}, \frac{1}{2l}$ depending on the value of y_n.
- For $t = 1, \ldots, T$:
  1. Normalize the weights $w_{t,n}$ so that $\sum_{n=1}^{N} w_{t,n} = 1$.
  2. For each feature f_j, train a weak classifier h_j.
  3. The error ϵ_j of a classifier h_j is determined with respect to the weights:
$$\epsilon_j = \sum_n^N w_{t,n} \left| h_j(x_n) - y_n \right|.$$
  4. Choose the classifier h_j with the lowest error ϵ_j and set $(h_t, \epsilon_t) = (h_j, \epsilon_j)$.
  5. Update the weights $w_{t+1,n} = w_{t,n}\beta_t^{1-e_n}$, where $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$ and $e_n = 0$, if example x_n is classified correctly by h_t and 1, otherwise.
- The final strong classifier is given by:
$$h(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \log \frac{1}{\beta_t} h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \log \frac{1}{\beta_t} \\ 0 & \text{otherwise.} \end{cases}$$

Main idea - 1st iteration for feature *j*

positive

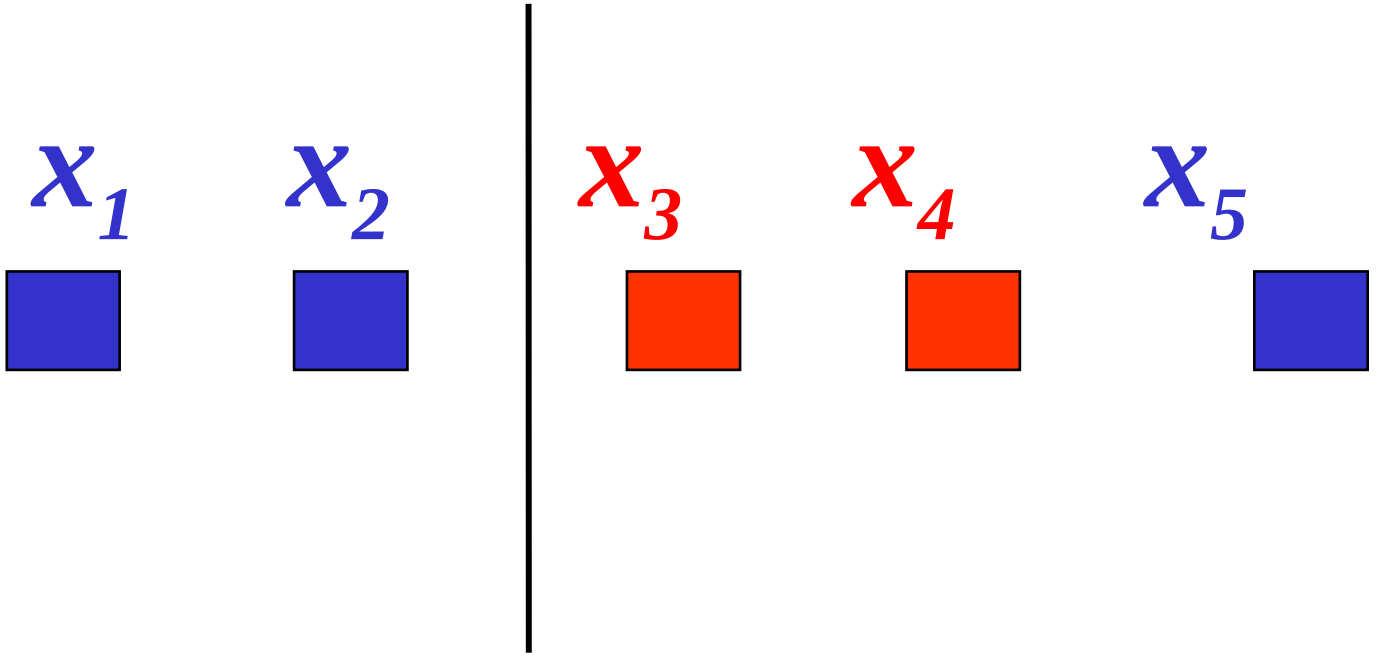
negative



Main idea - 1st iteration for feature *j*

positive

negative

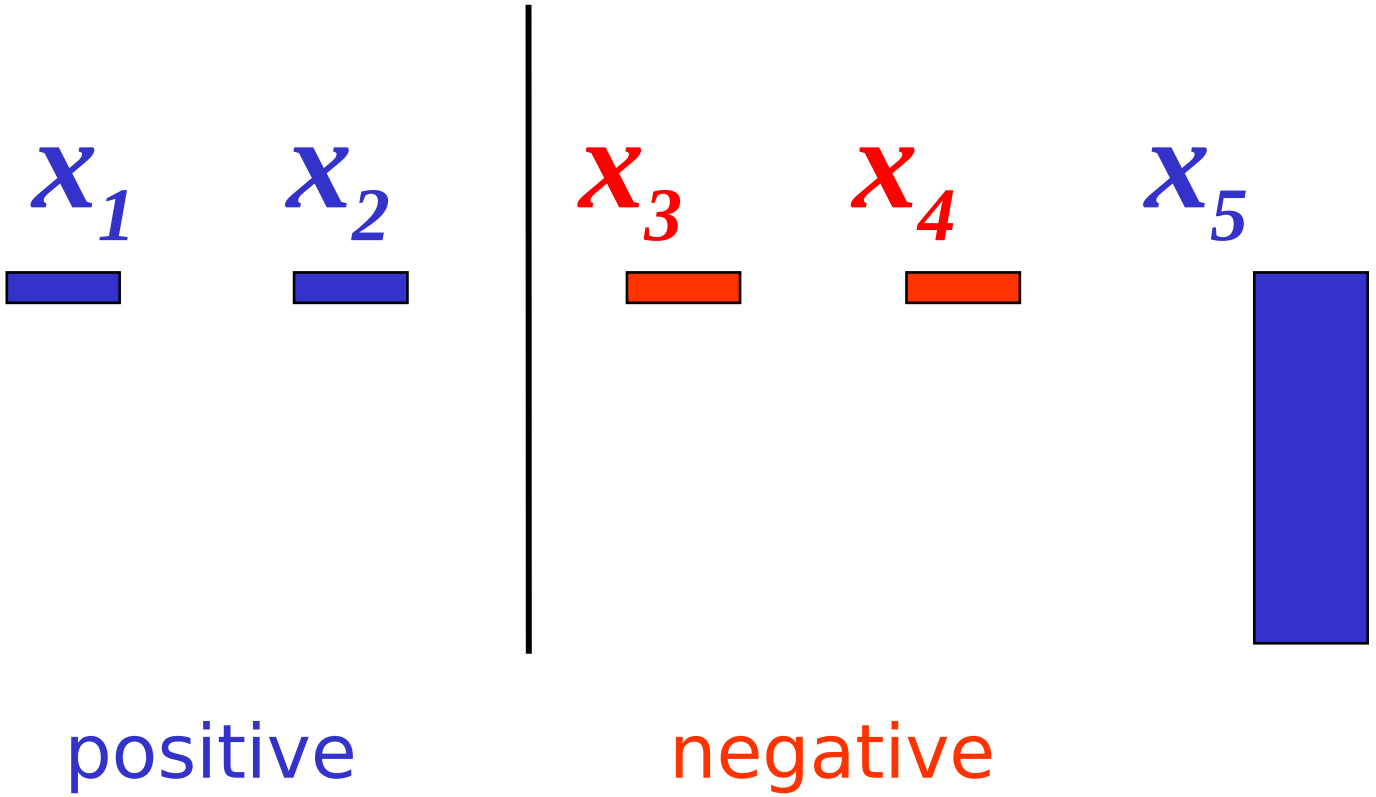


positive negative

Main idea - 1st iteration for feature *j*

positive

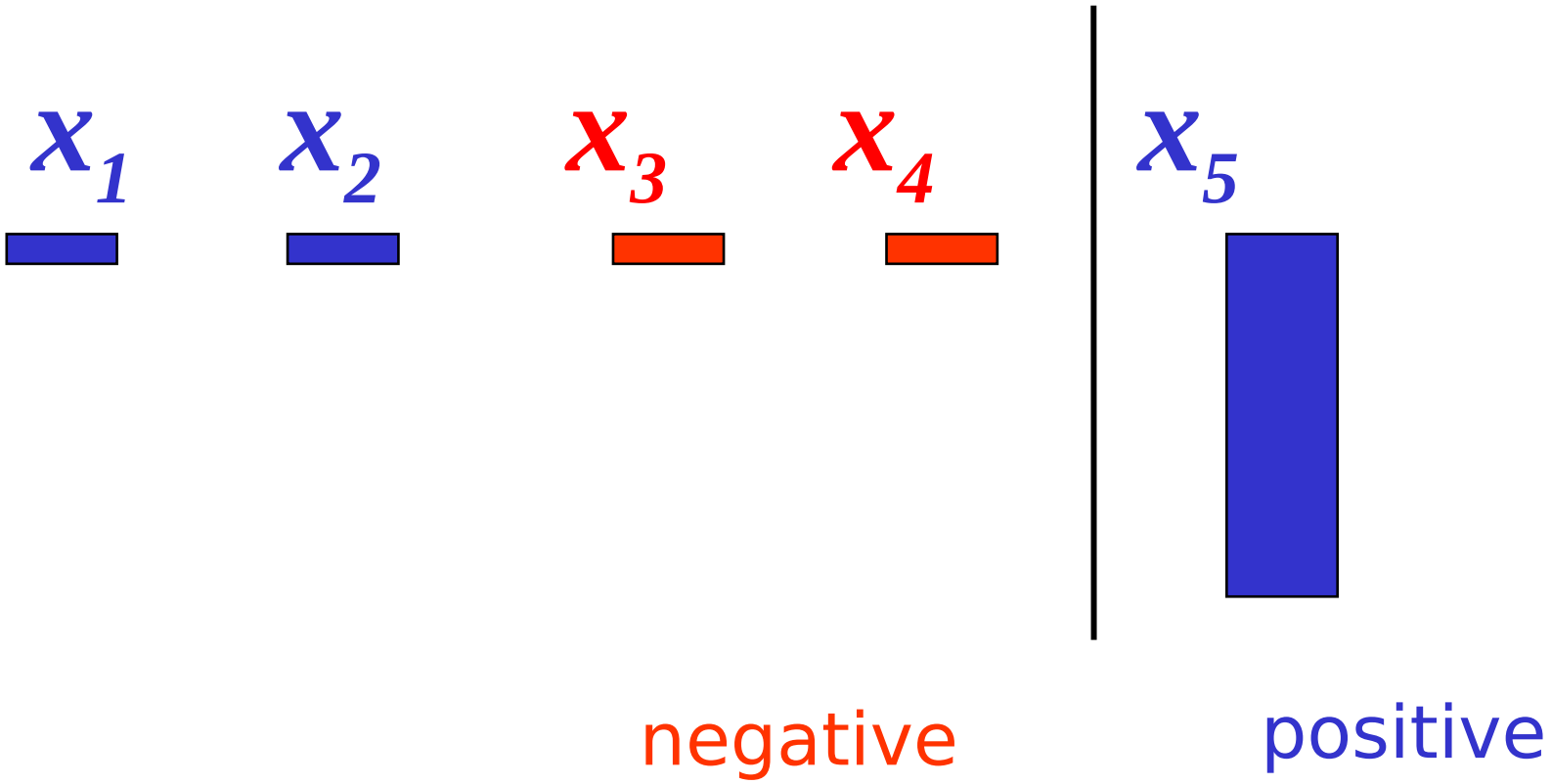
negative



Main idea – 2nd iteration for feature *j*

positive

negative



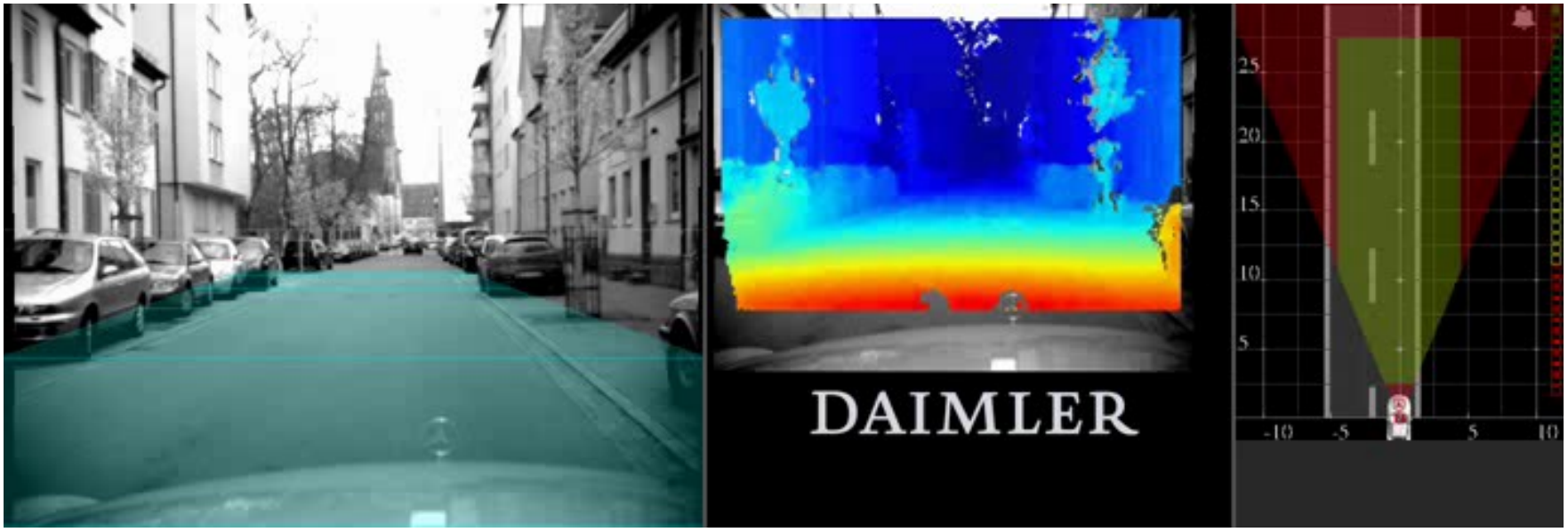
AdaBoost in Action

Kai Arras
Social Robotics Lab, University of Freiburg

Nov 2009 Social Robotics Laboratory

Question 1

- Daimler/Mercedes claims its new automatic system in cars for detecting people is 100% save (you will never run over a person).
- **How? Any trick?**

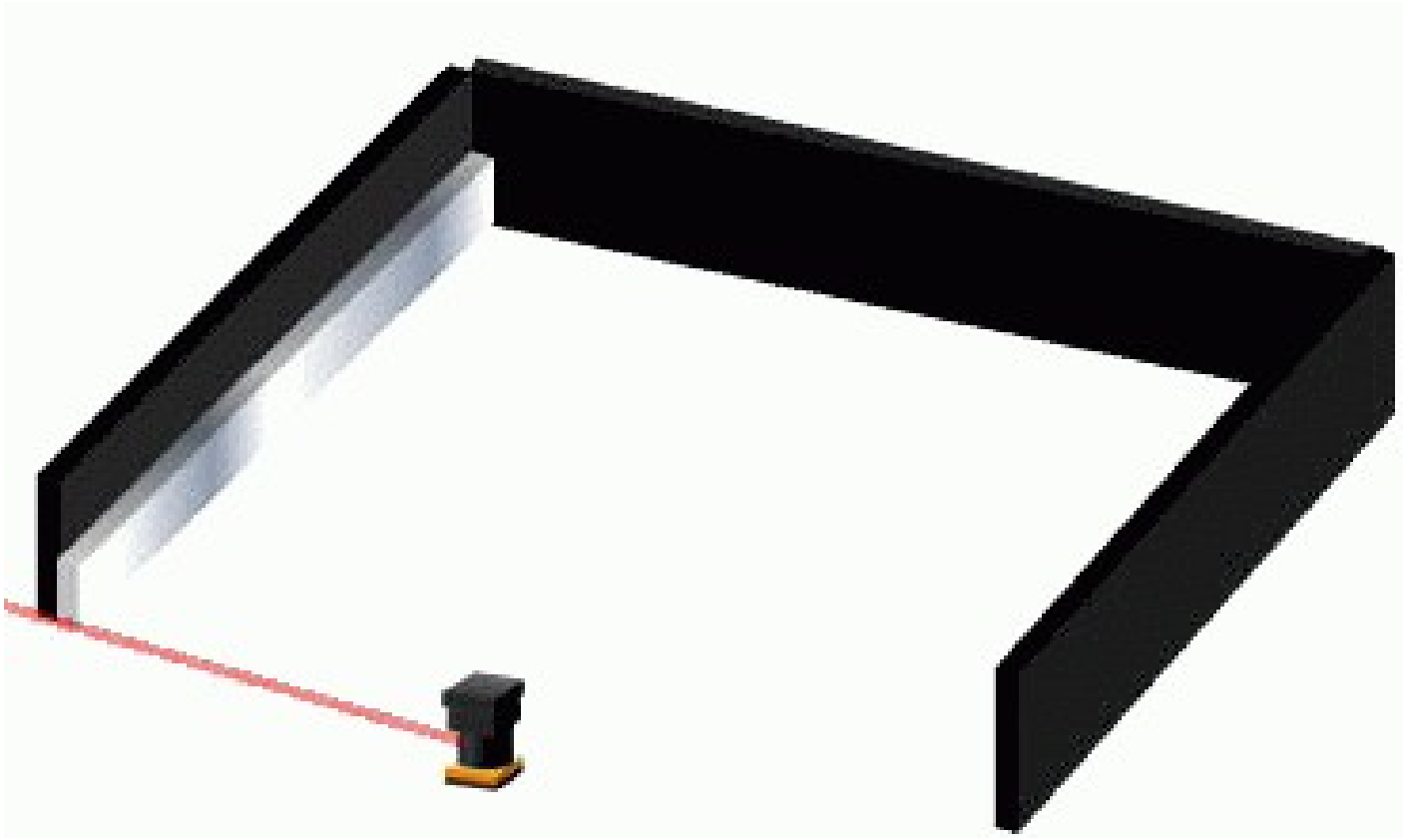


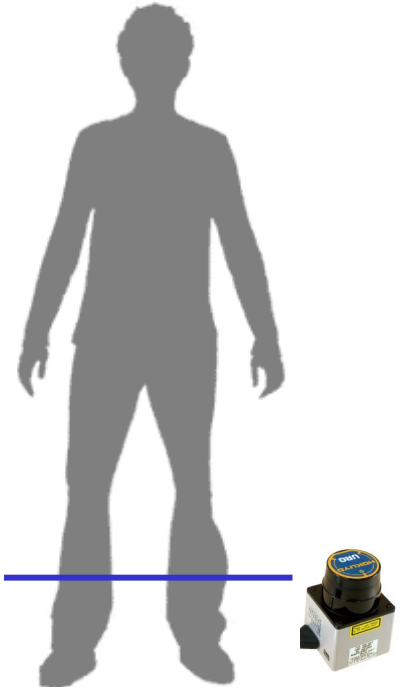
Question 2

- Linear classifiers (hyperplane) are only able to separate two classes
- Could you devise a strategy for classifying **three** classes using linear classifiers?, i.e., devise a system to classify **three** classes using classifiers for **two** classes.

- HINT: this is as a problem transformation/decomposition from a **3-class** problem into **2-class** problems

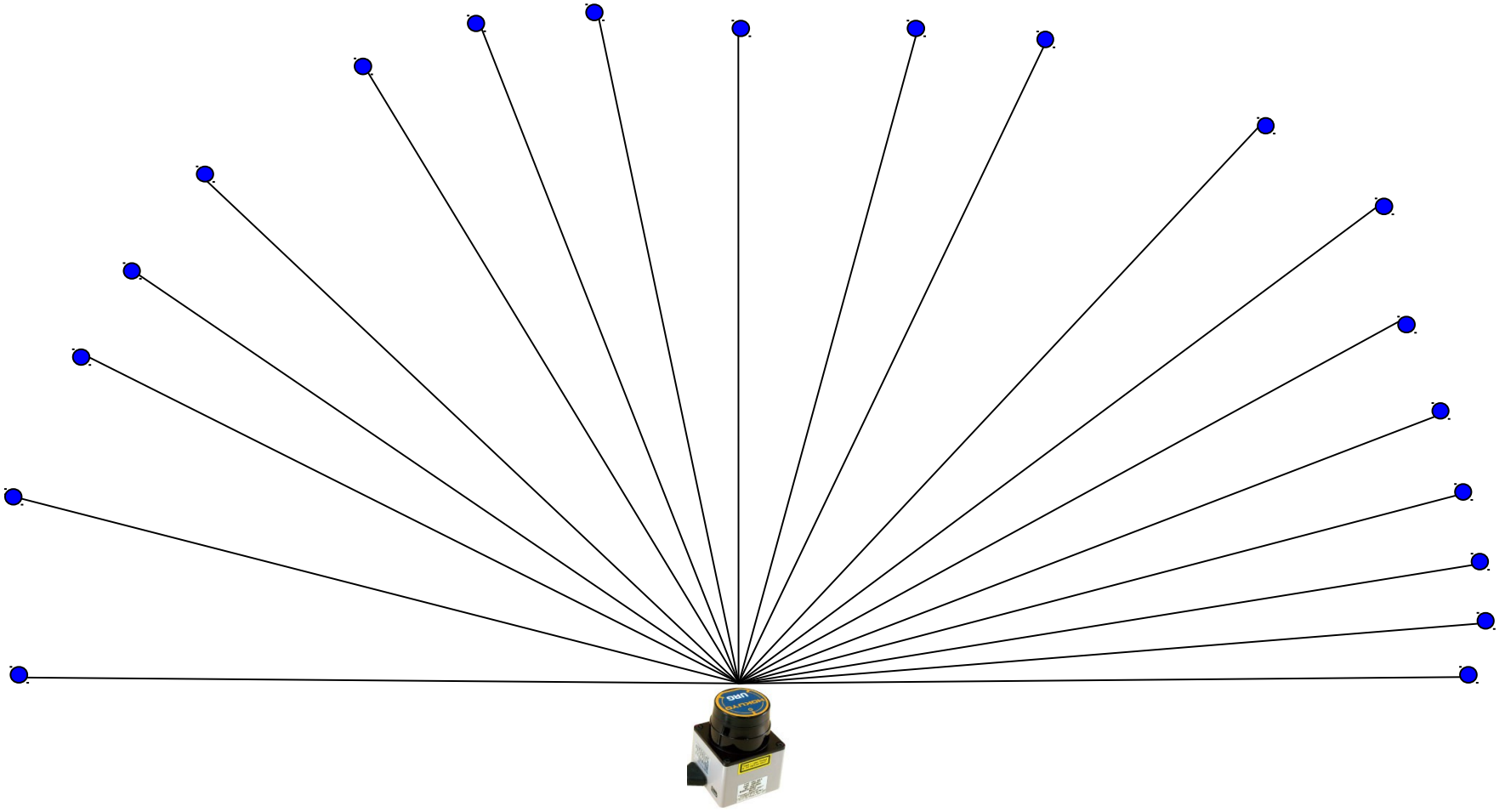
Example application:
People detection in 2D laser data





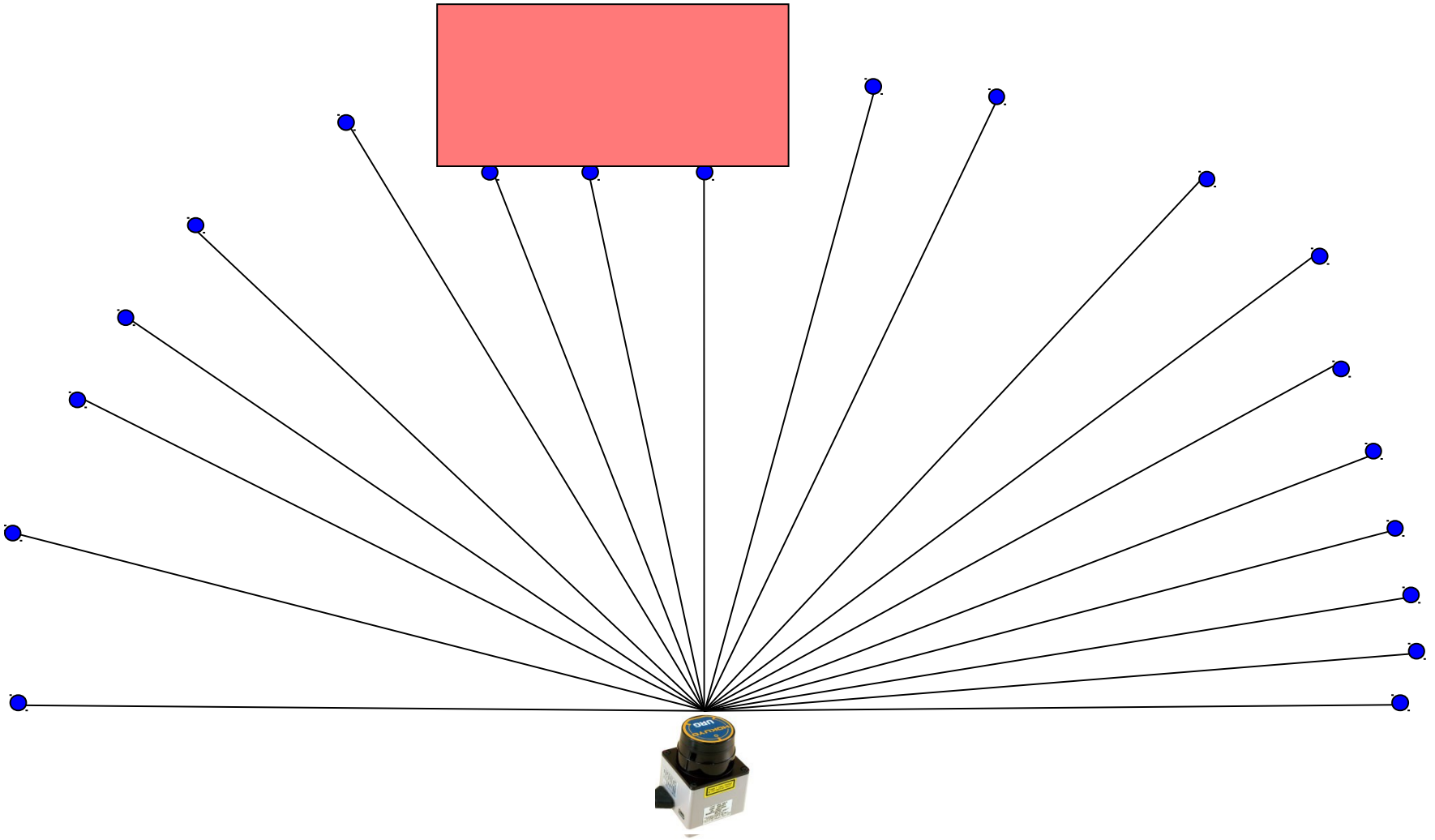
2D Laser Scan Overview

- A laser scan emits beams in different directions.



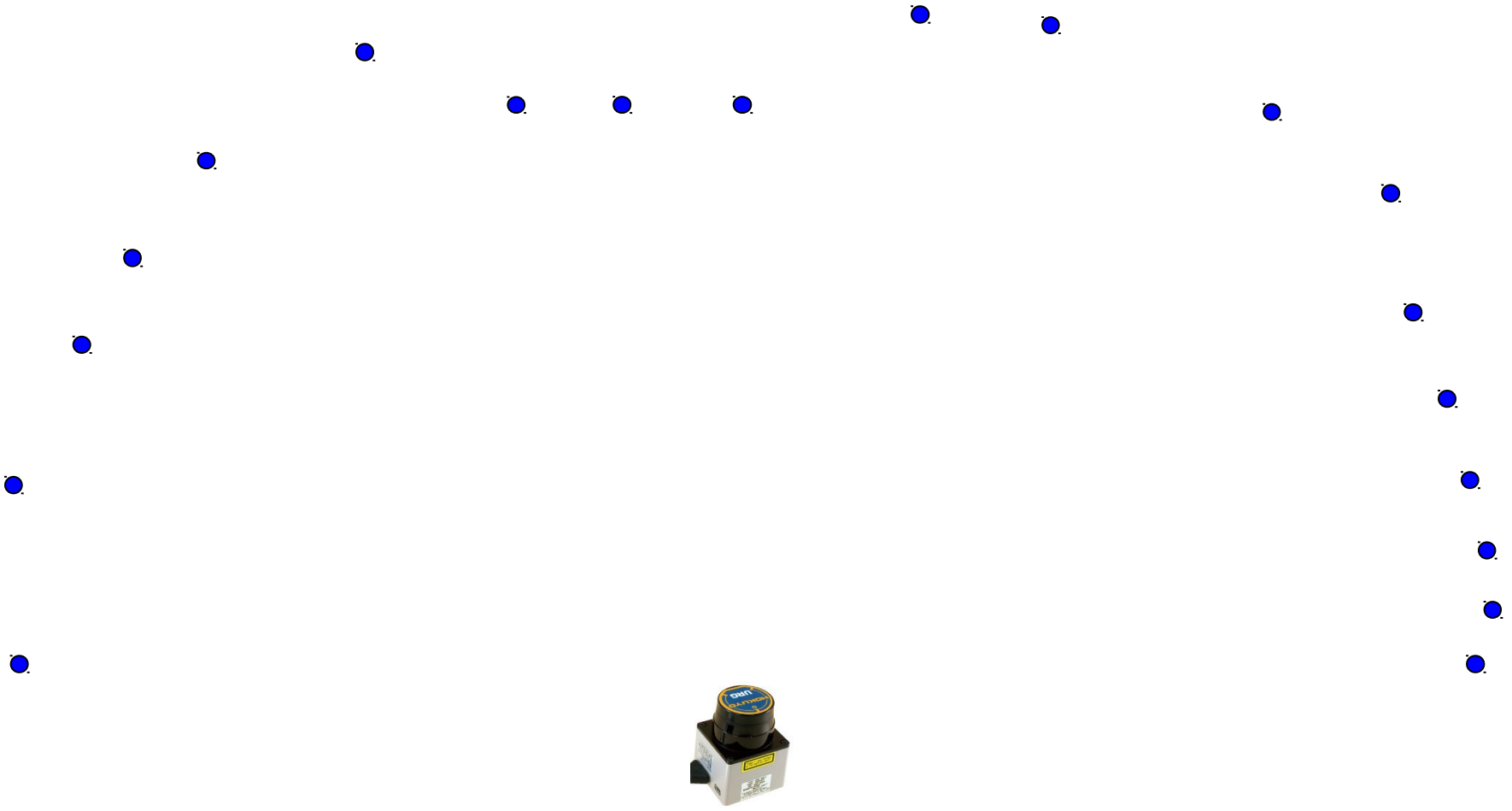
2D Laser Scan Overview

- The beams hit objects in the environment and we can obtain the corresponding distances.

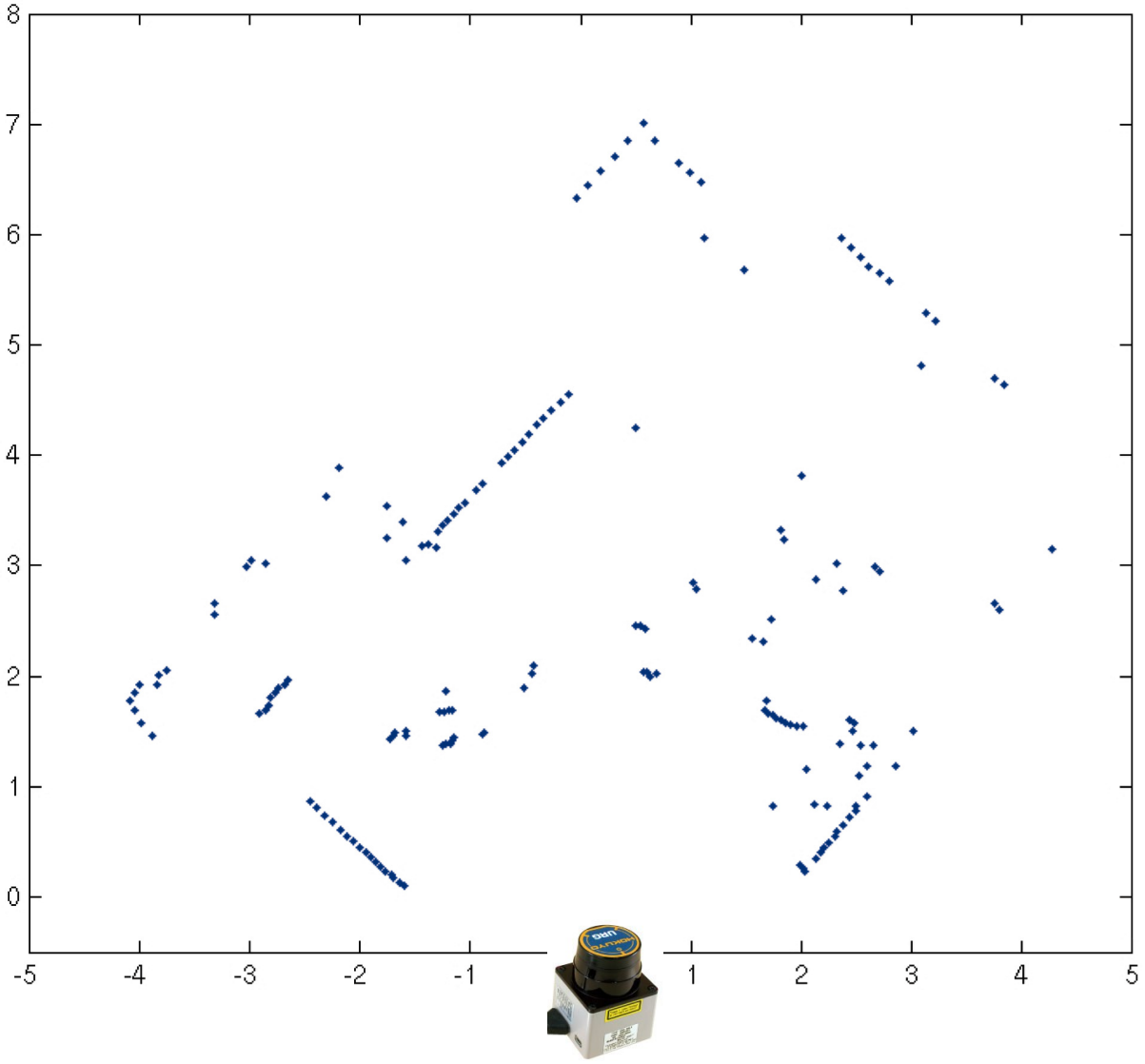


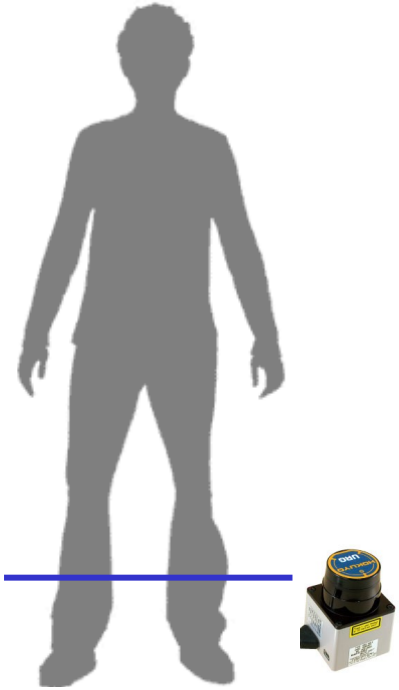
2D Laser Scan Overview

- Usually scans are represented by their end points.



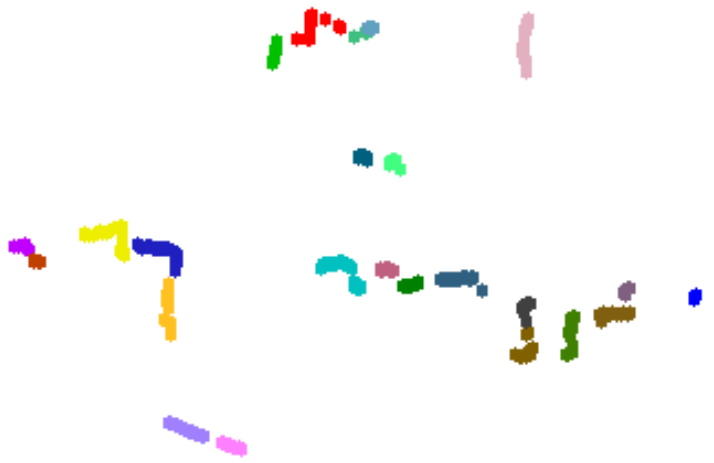
People detection in 2D laser data





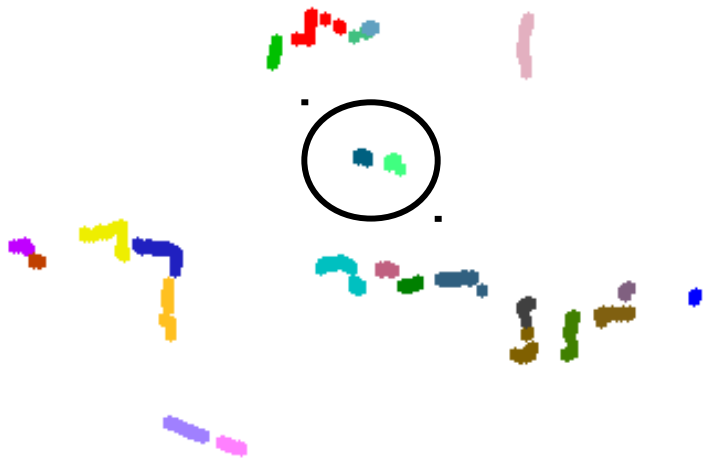
Detection in One Level

- Divide the scan into segments.



Detection in One Level

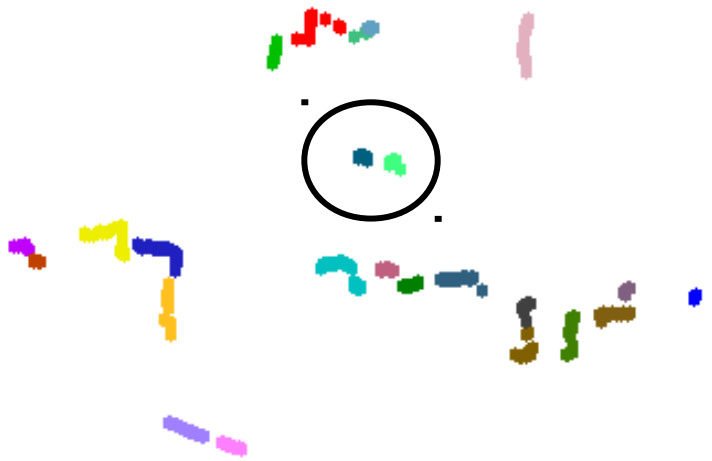
- Divide the scan into segments.



- Select the segments corresponding to people.

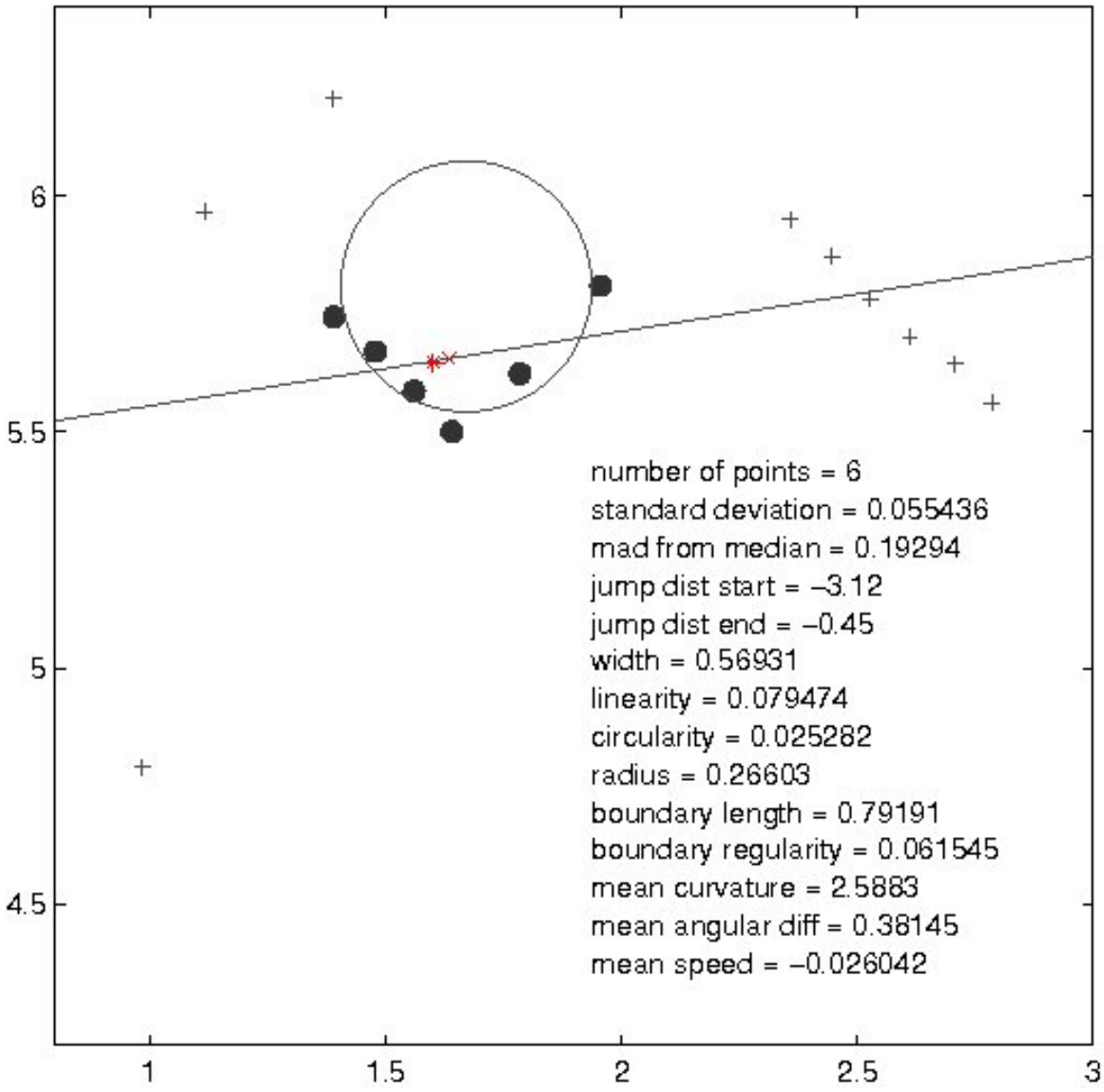
Detection in One Level

- Divide the scan into segments.



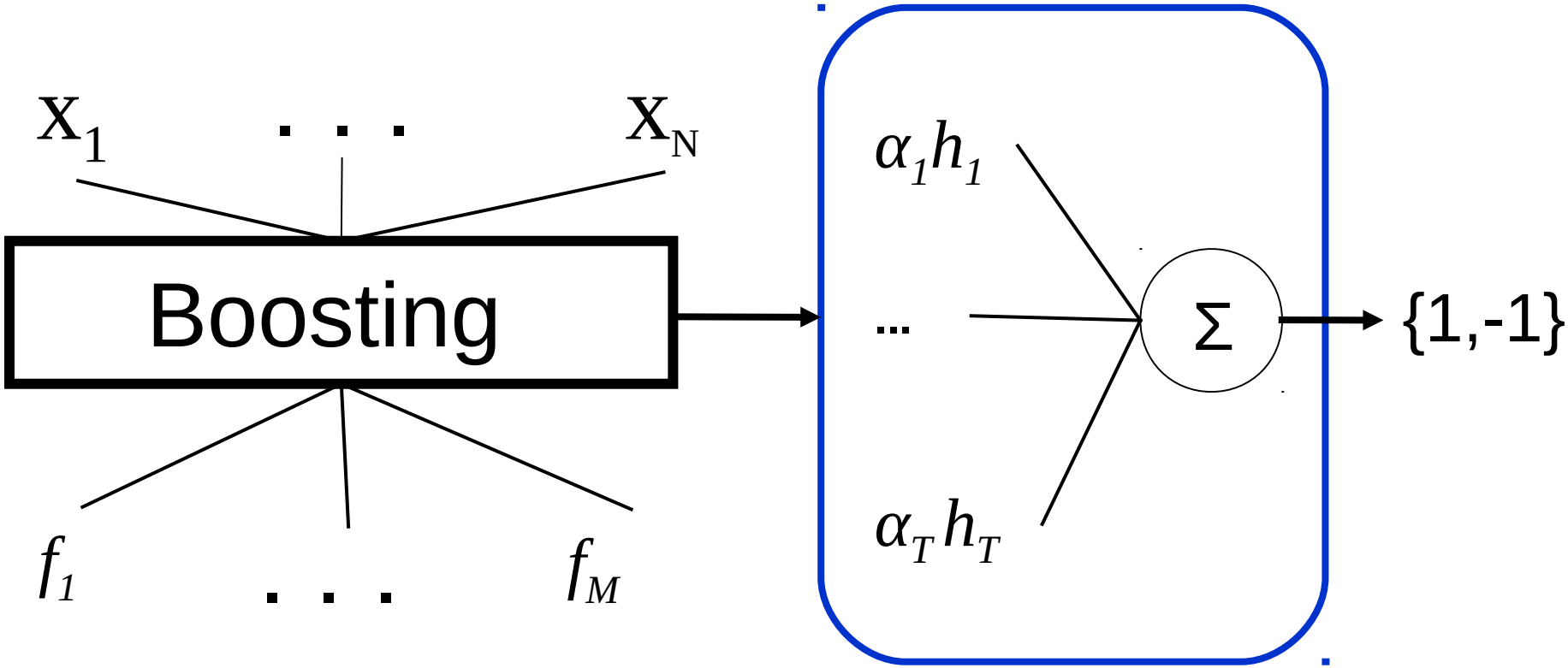
- Learn the segments corresponding to people.
- Classification problem:
  1. Select the features representing a segment.
  2. Learn a classifier using Boosting

Features from Segments



Boosting

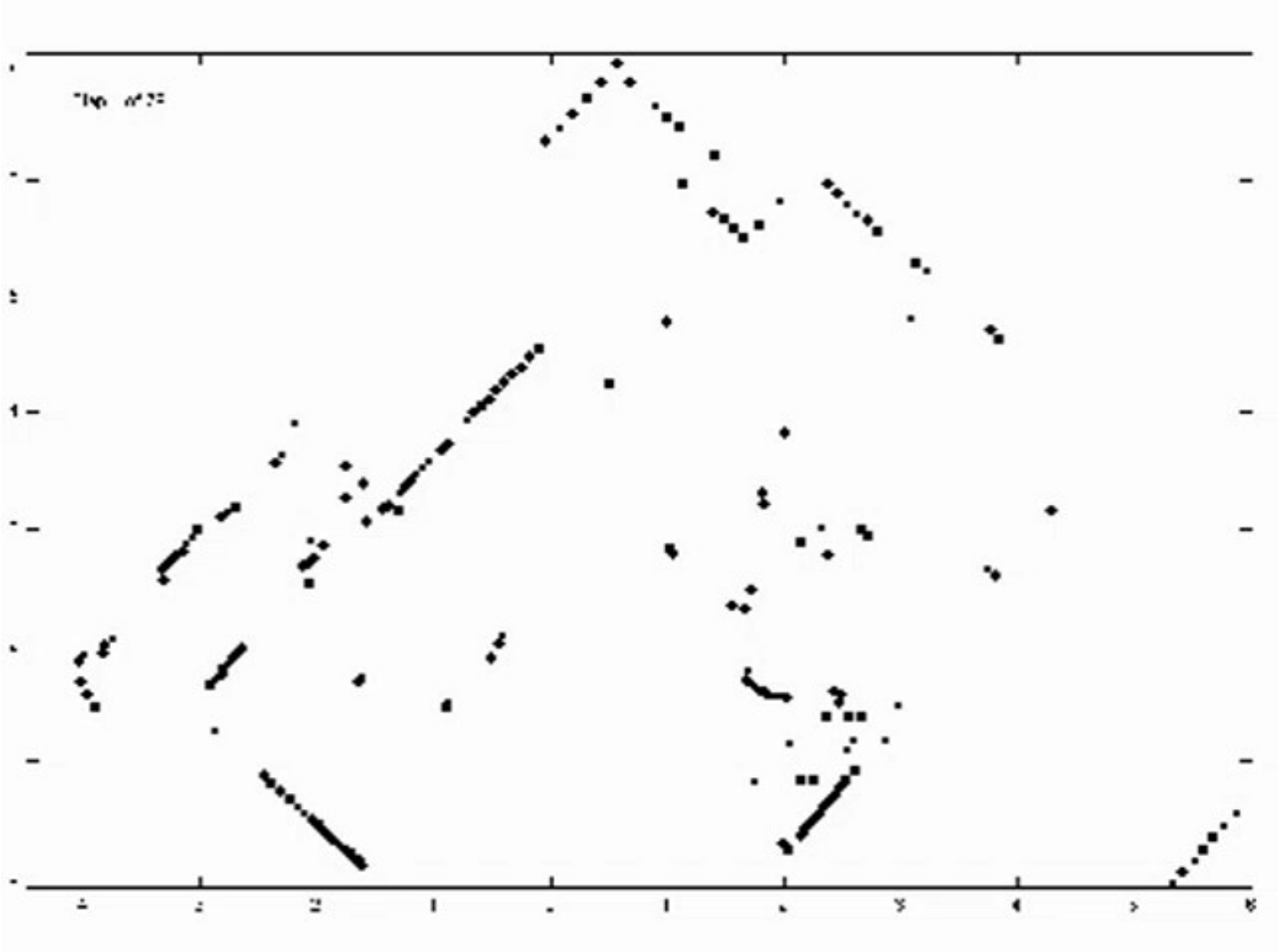
- Weak classifiers are combined to form a strong classifier



$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ -1 & \text{otherwise.} \end{cases}$$

Binary Strong Classifier H

Results

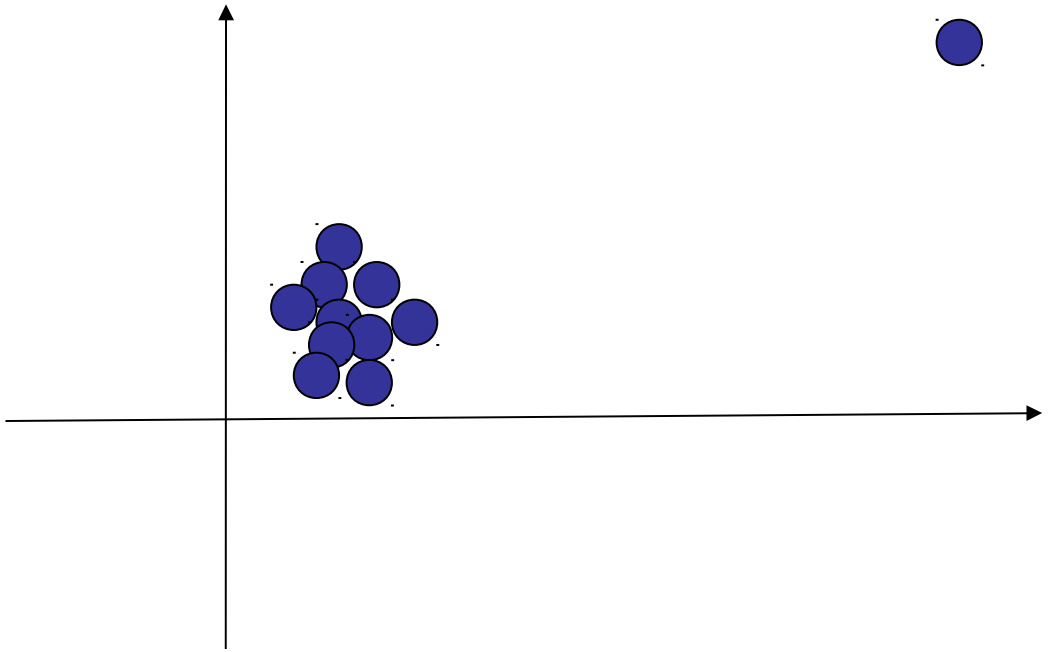


Some considerations when learning a classifier

- Outliers
- Overfitting
- Normalization of data
- Feature selection

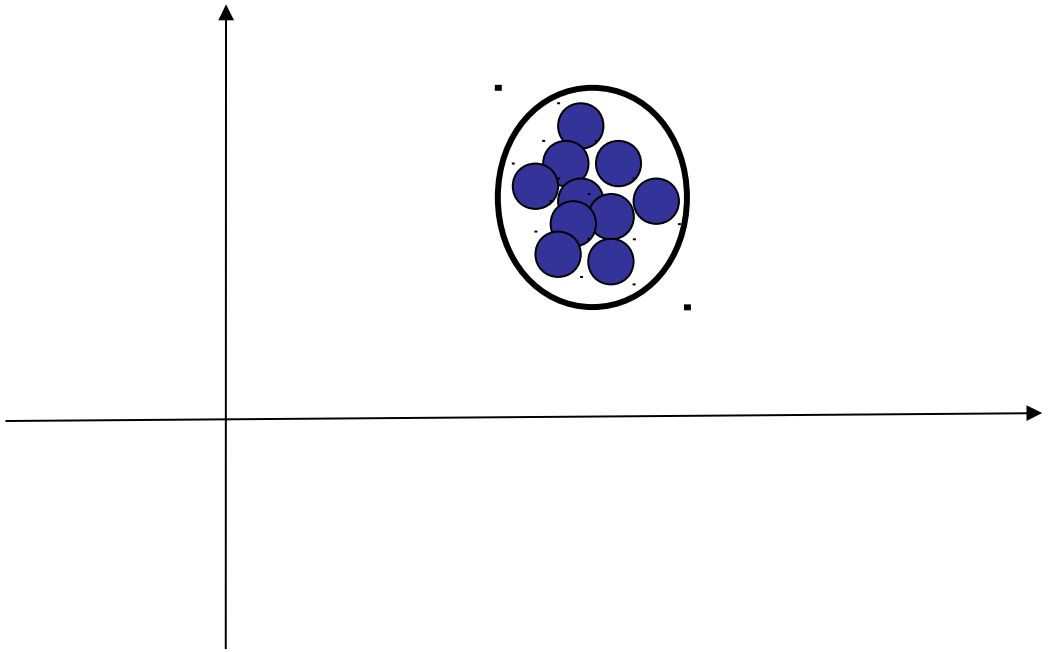
Outliers

- Examples with some features having extreme values
- Possible reasons:
  - Error during the extraction/calculation of the features
  - Exceptional example in the class
- Solution: ignore these examples



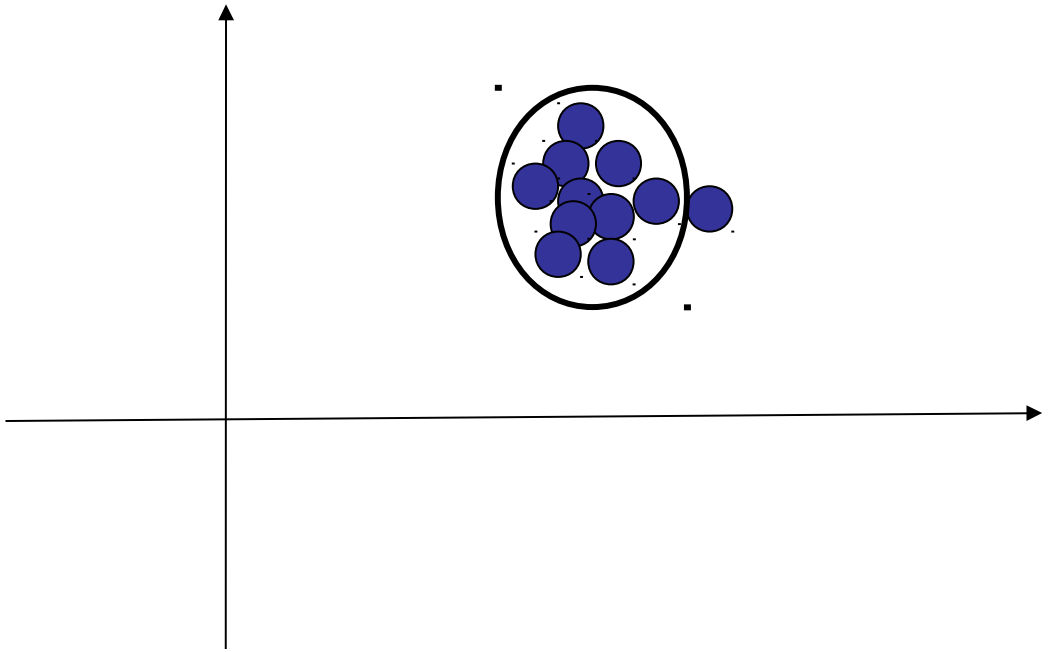
Overfitting

- The learned classifier is to close to the training examples.
- This reduces generalization if new examples are slightly different from the training.



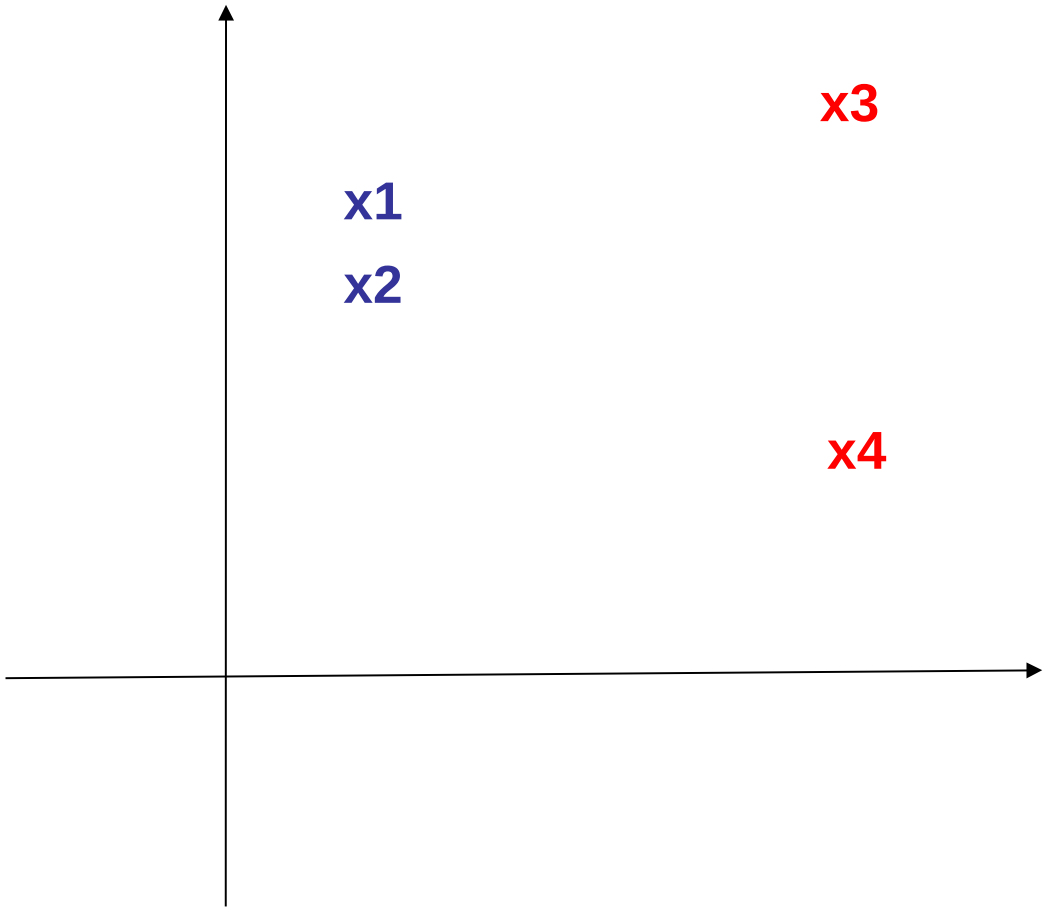
Overfitting

- The learned classifier is to close to the training examples.
- This reduces generalization if new examples are slightly different from the training.



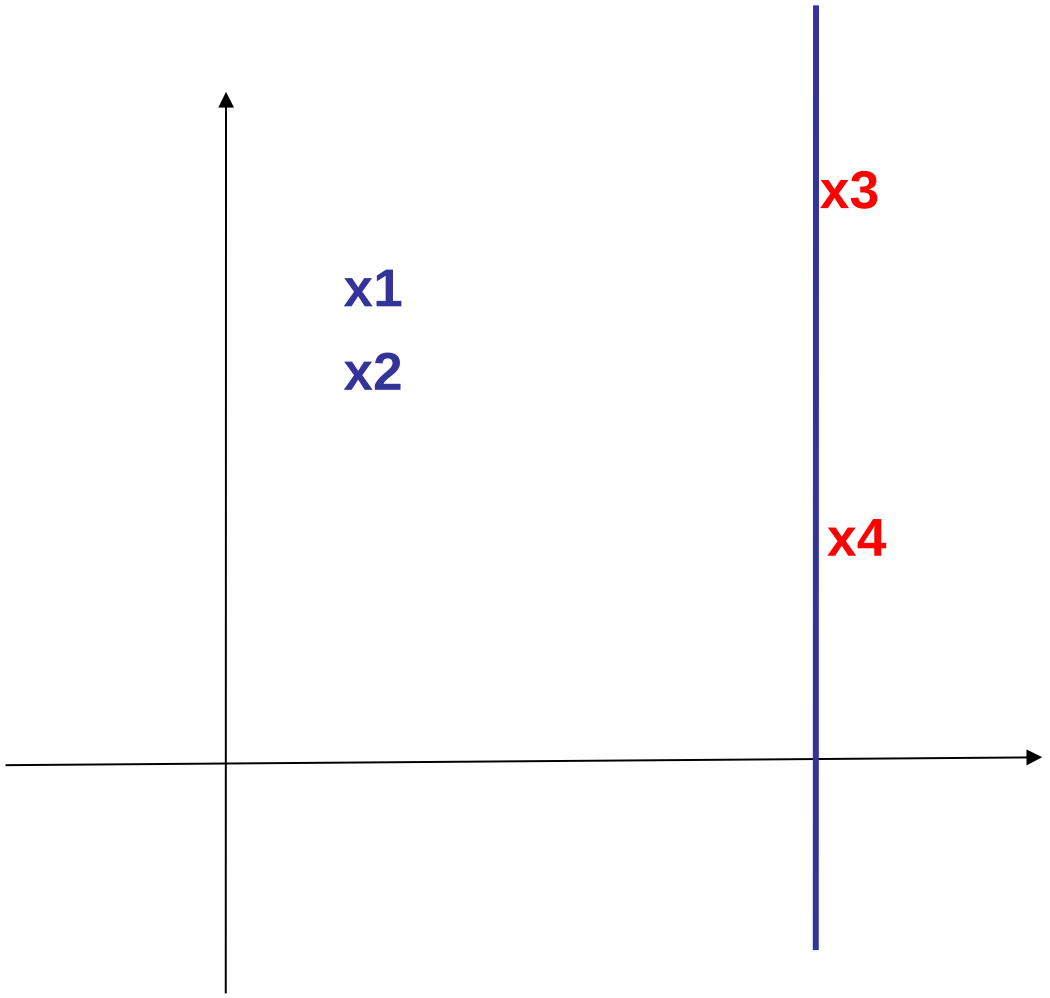
Overfitting

- Also in linear classifiers



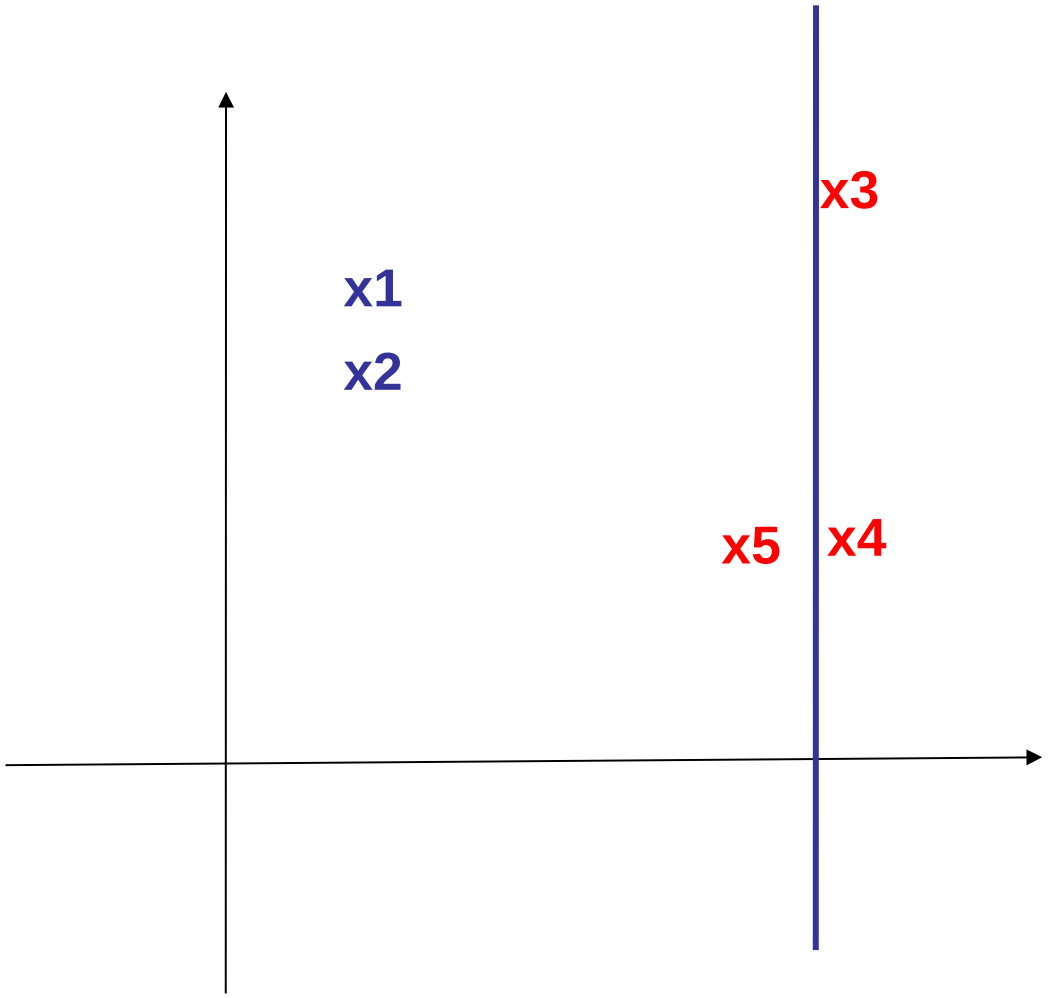
Overfitting

- Also in linear classifiers



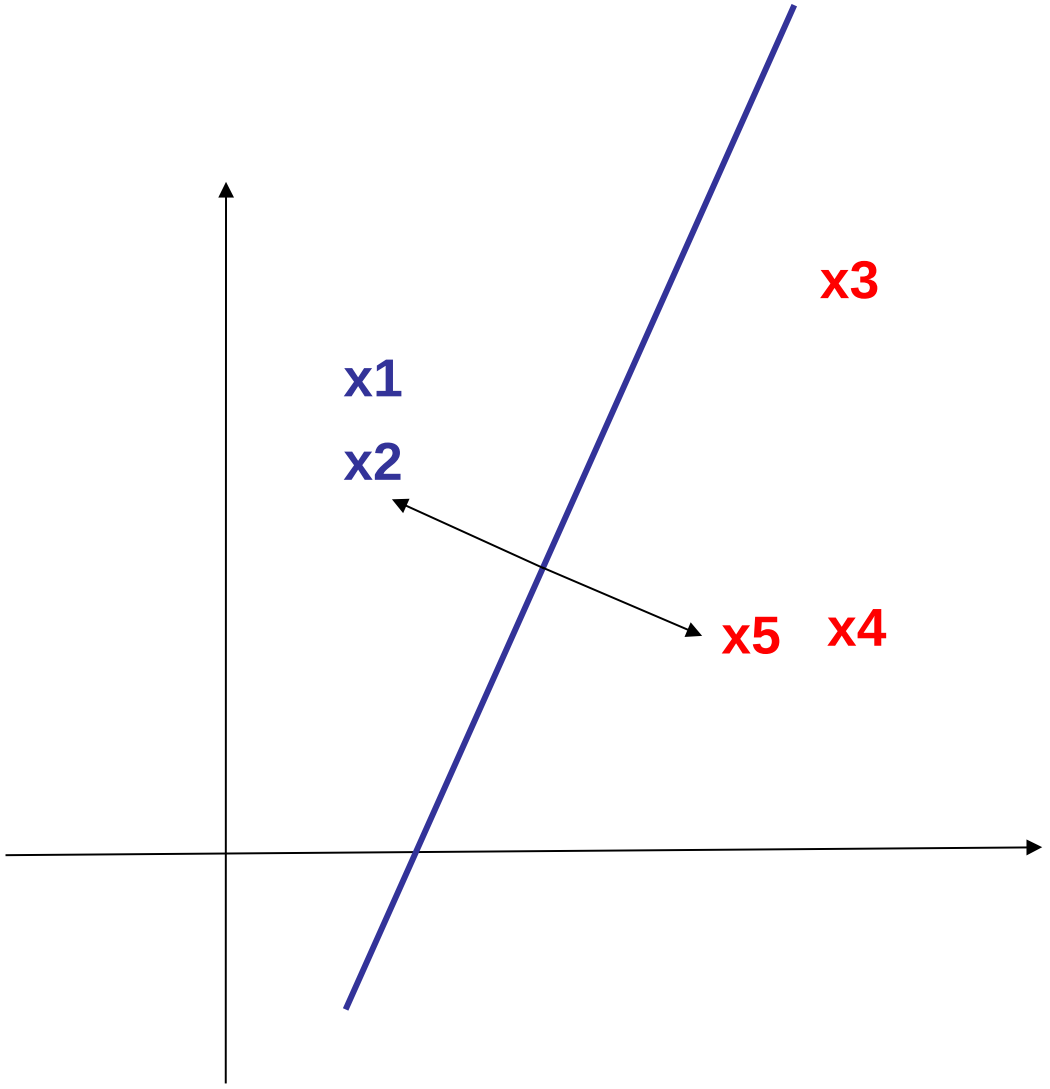
Overfitting

- Also in linear classifiers



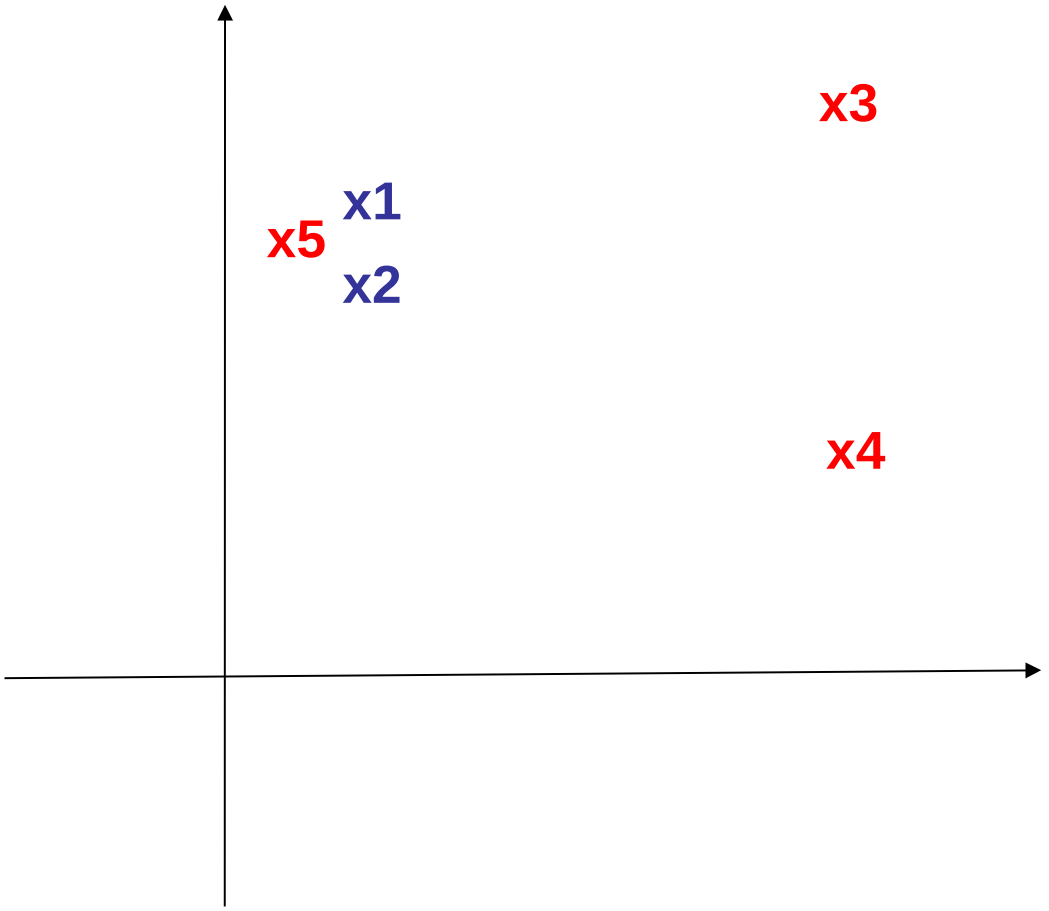
SVM revisited

- Maximises distances to classes



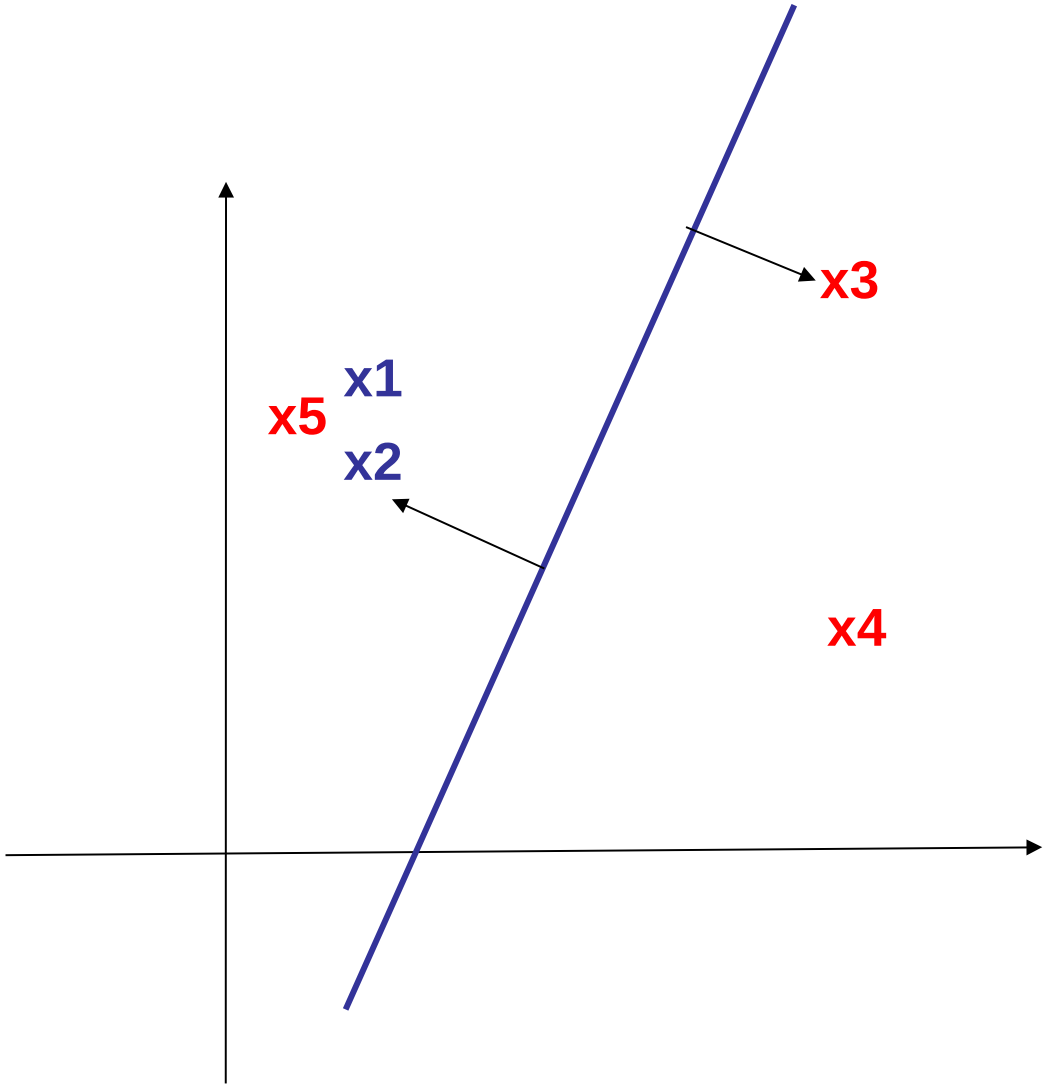
SVM revisited

- Allow misclassification



SVM revisited

- Allow misclassification

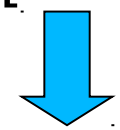


Normalization of data

- The ranges of the features could be very different.
- It is interesting to normalize them. This seems to improve some classifiers, e.g. SVM

f1 [1 2]

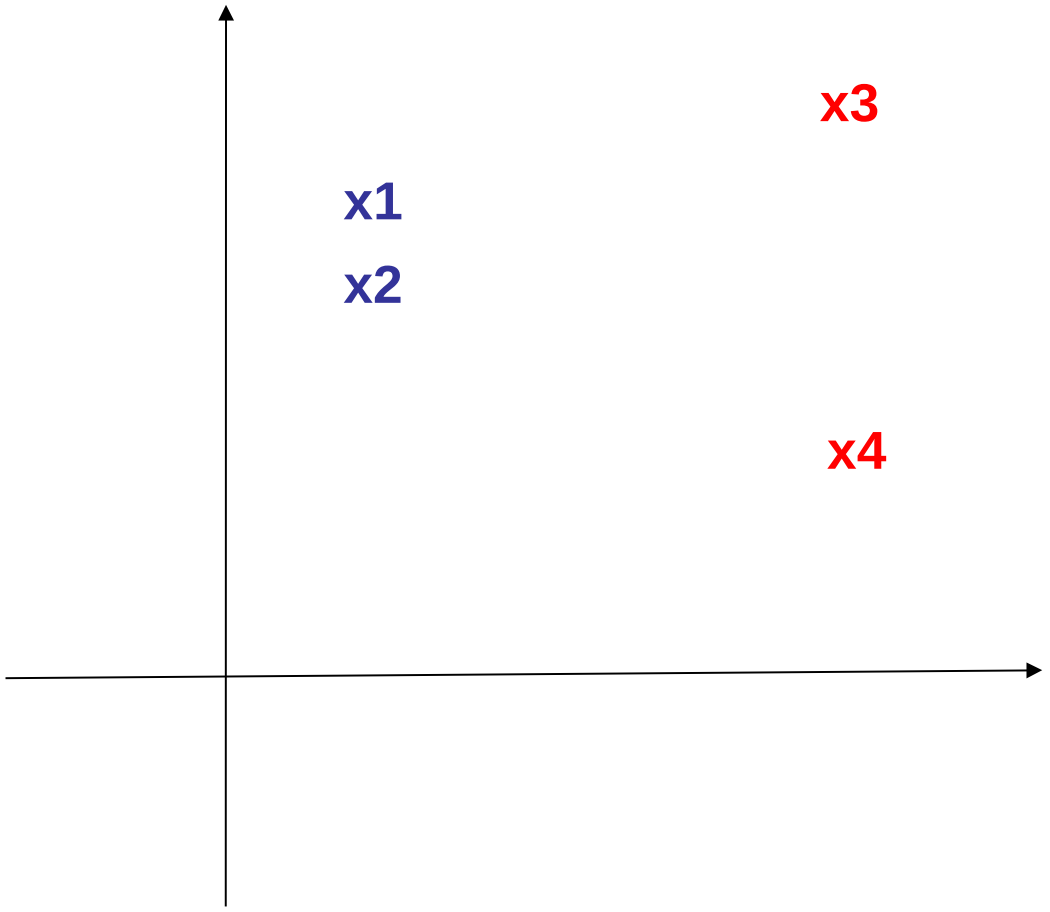
f2 [-100 100]



f1, f2 [0, 1]

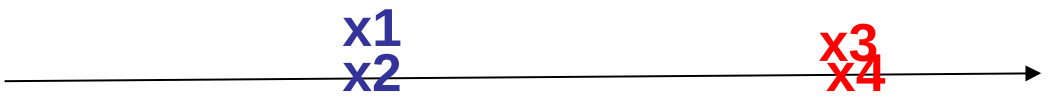
Feature selection

- Some features discriminate better than others.



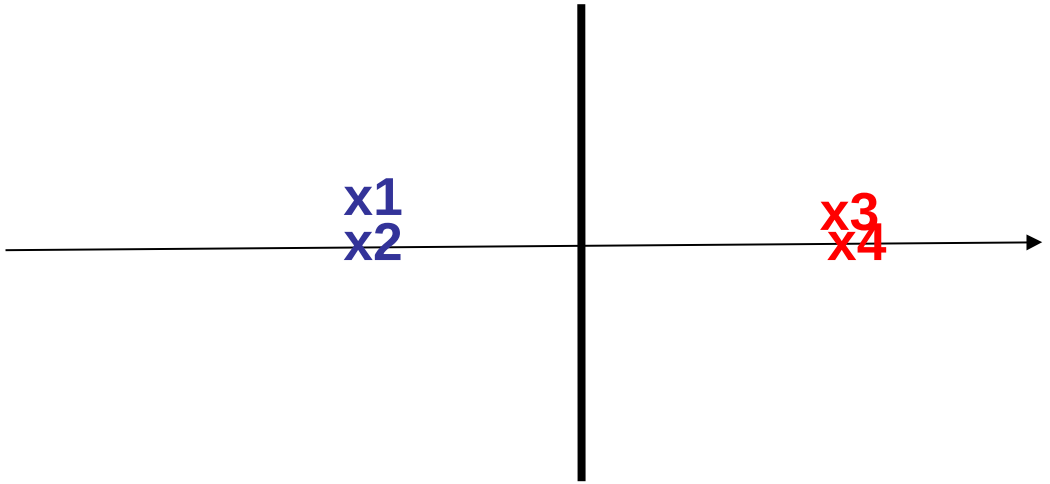
Feature selection

- Projection on **f1**



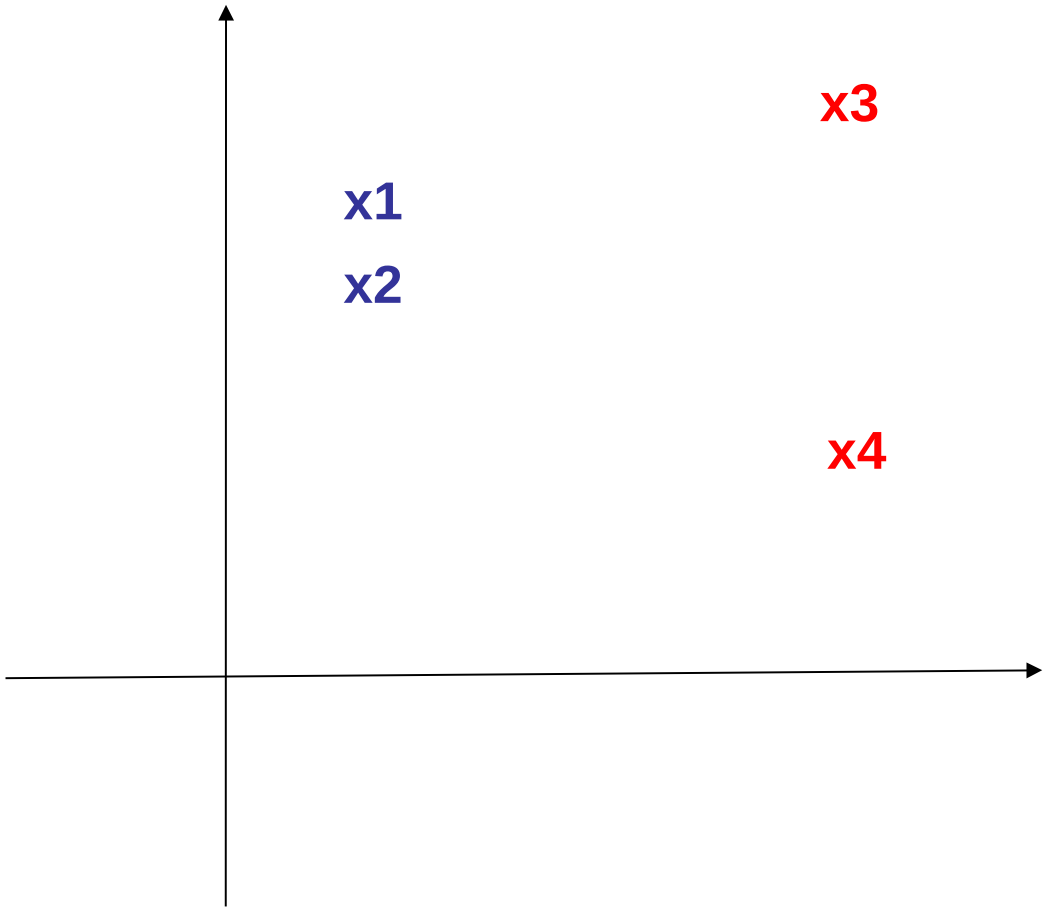
Feature selection

- Projection on **f1, easy classification**



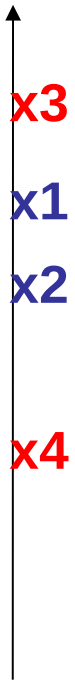
Feature selection

- Projection on **f2**



Feature selection

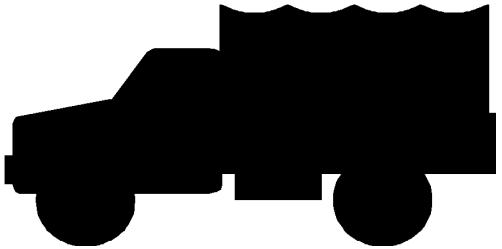
- Projection on **f2, difficult classification**



Multi-classification

- So far we have dealt with two classes, but there is usually more than two

people



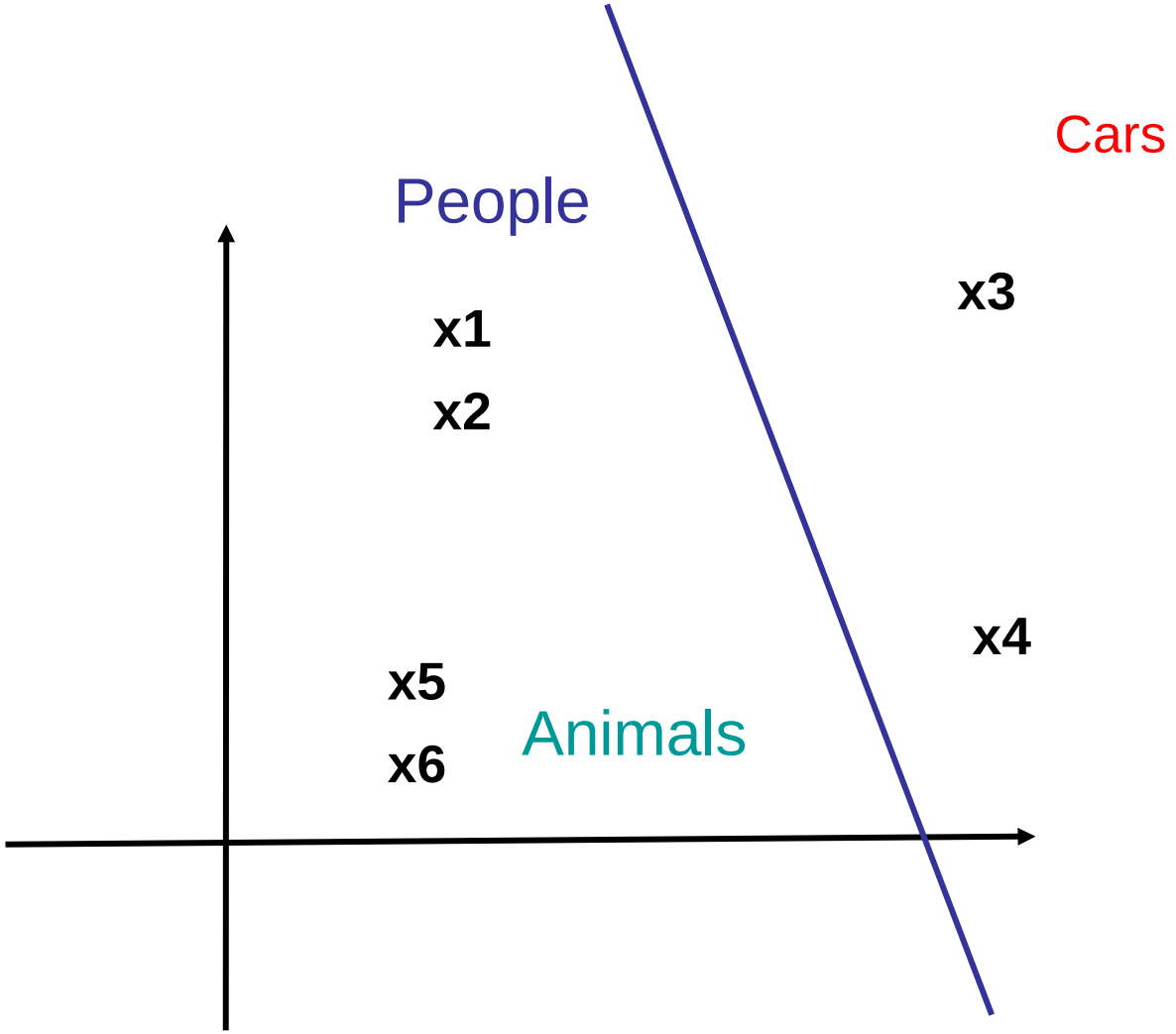
cars

animals

Planes

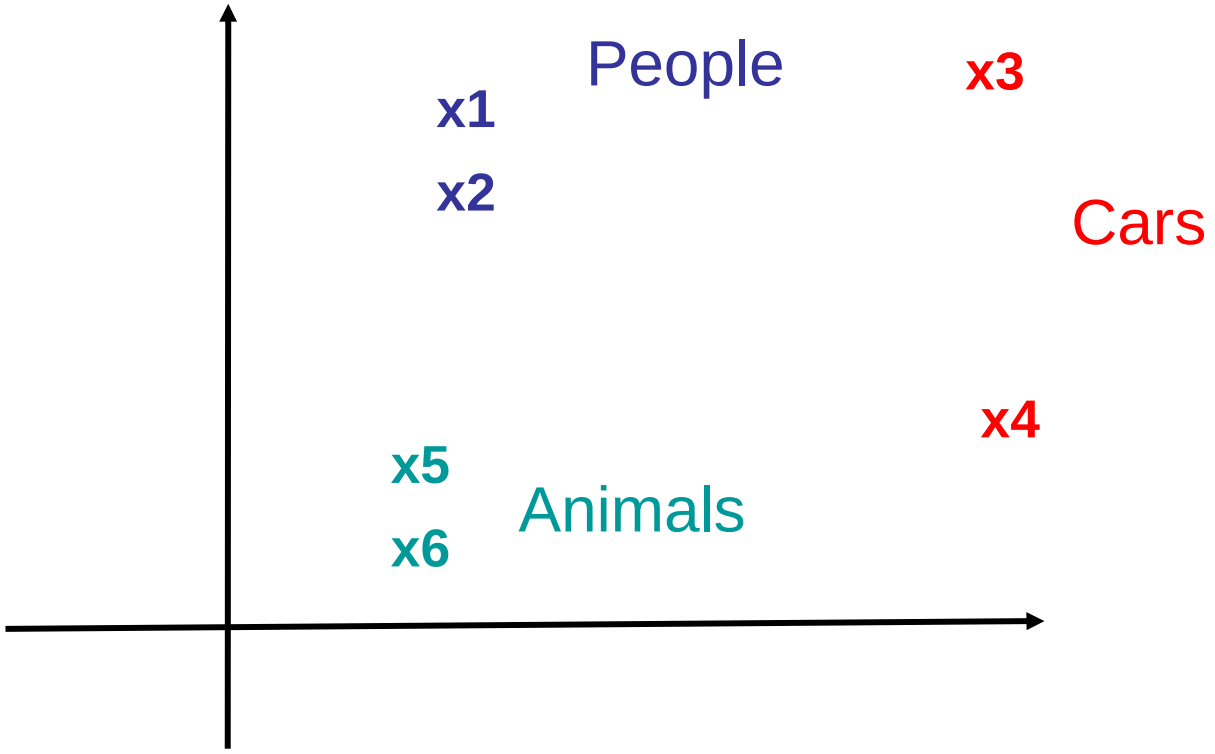
Multi-class classifiers based on hyperplanes

- Hyperplanes are discriminative but do not model -> binary classifiers



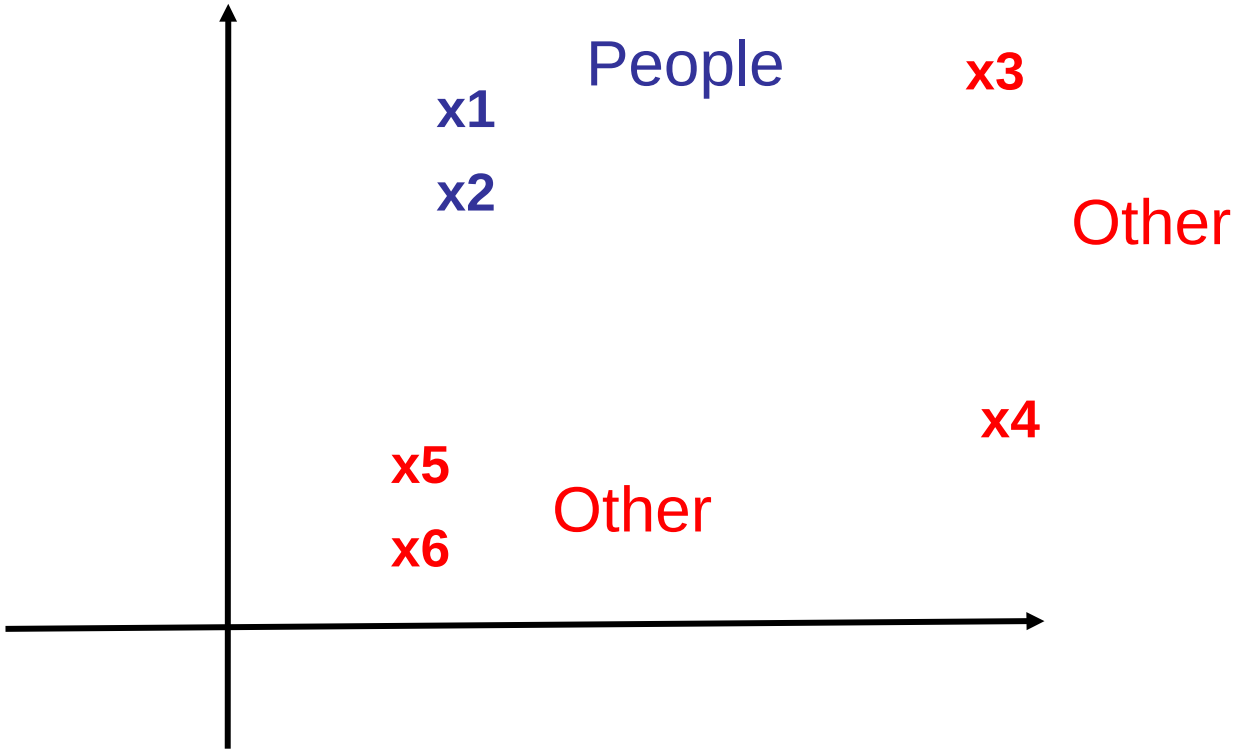
Multi-class with general classifiers

- If we have M classes, we train M binary classifiers (C)
- For each binary classifier C_i we label each example in the training data as 1 if the example corresponds to class i and -1 (or 0) otherwise (one-against-all)



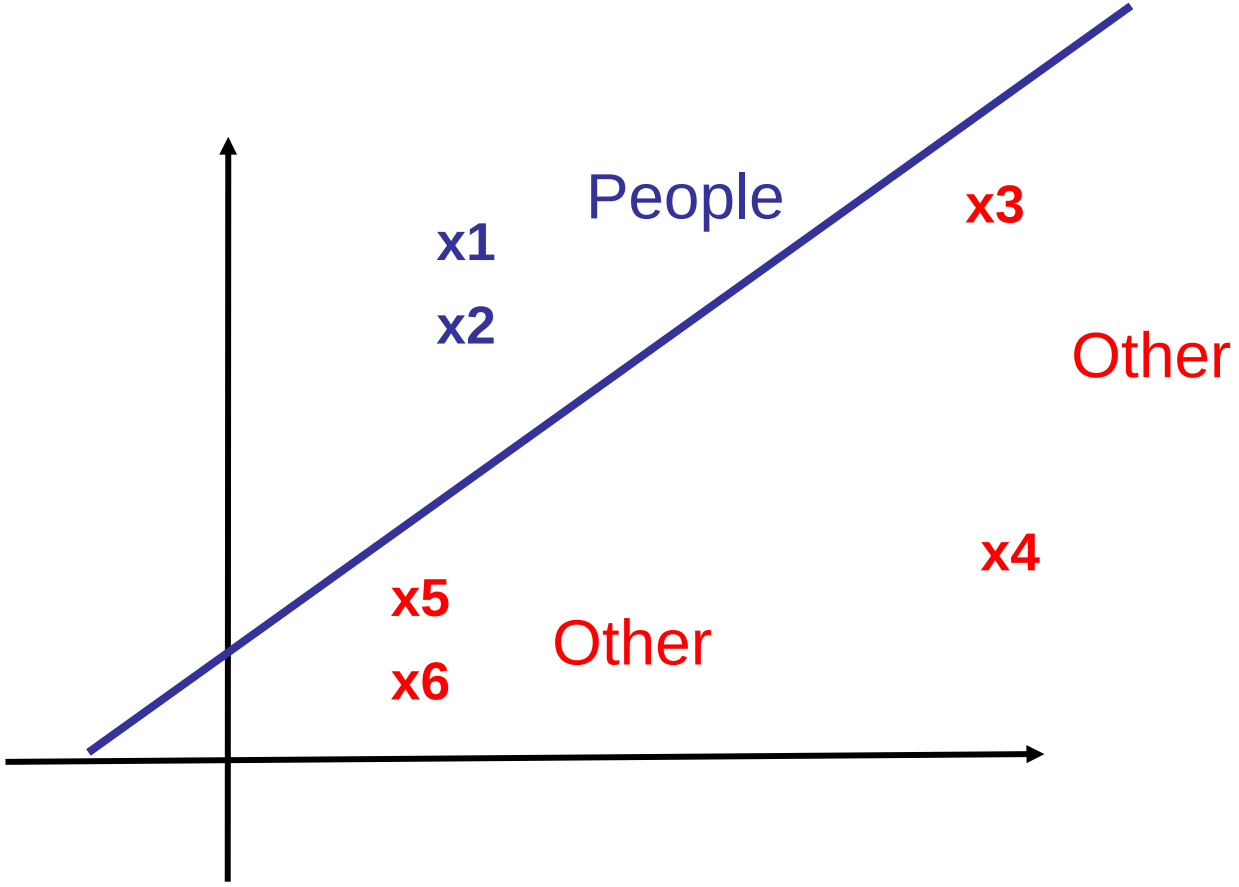
Multi-class with general classifiers

- If we have M classes, we train M binary classifiers (C)
- For each binary classifier C_i we label each example in the training data as 1 if the example corresponds to class i and -1 (or 0) otherwise (one-against-all)



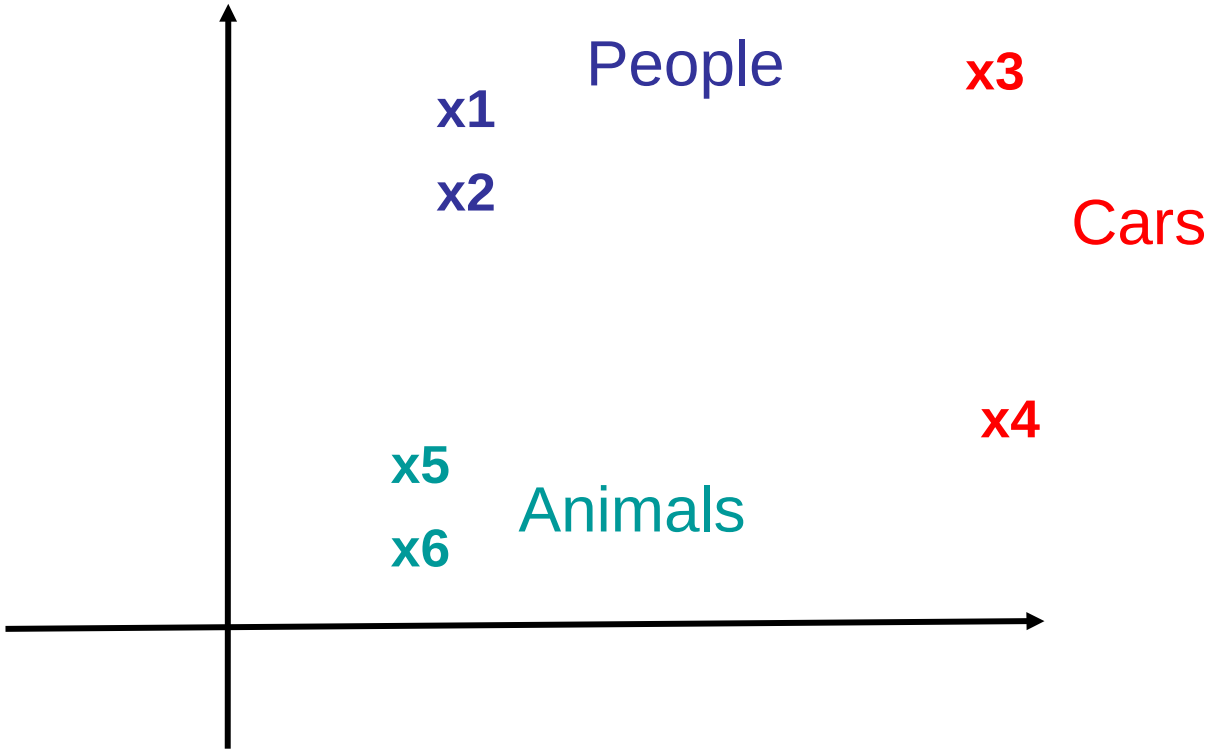
Multi-class with general classifiers

- If we have M classes, we train M binary classifiers (C)
- For each binary classifier C_i we label each example in the training data as 1 if the example corresponds to class i and -1 (or 0) otherwise (one-against-all)



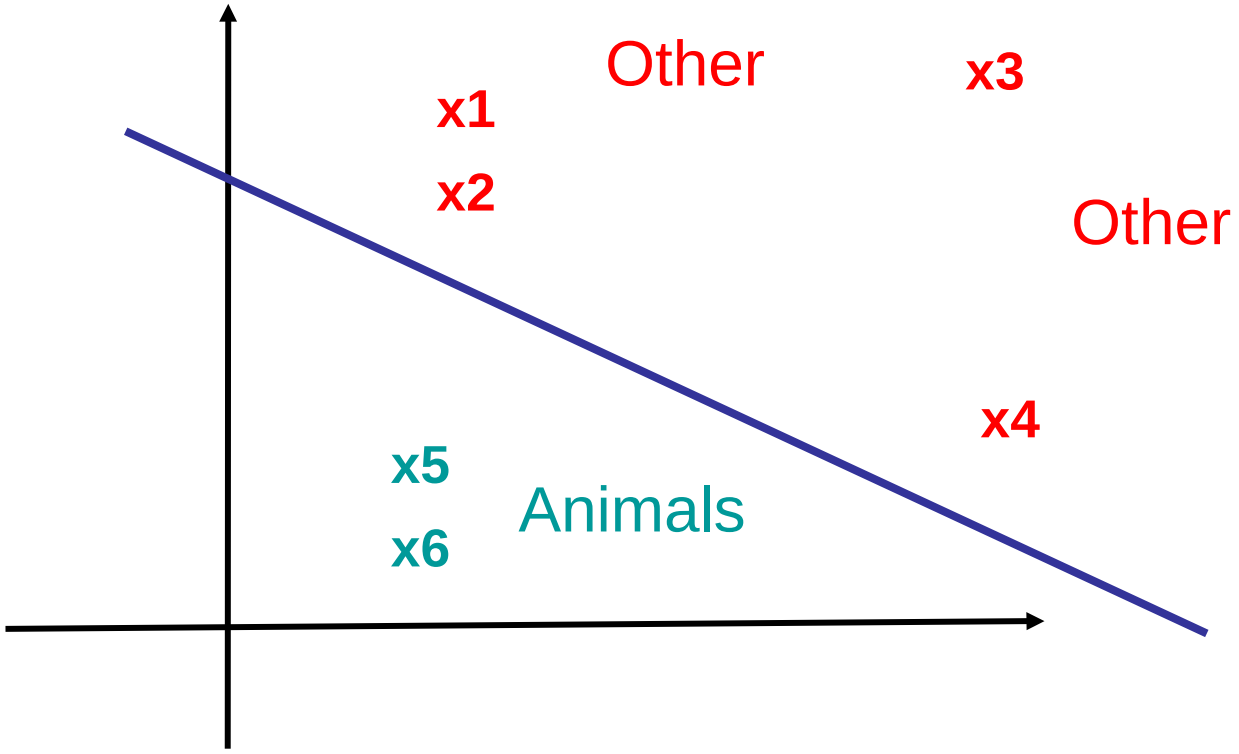
Multi-class with general classifiers

- If we have M classes, we train M binary classifiers (C)
- For each binary classifier C_i we label each example in the training data as 1 if the example corresponds to class i and -1 (or 0) otherwise (one-against-all)



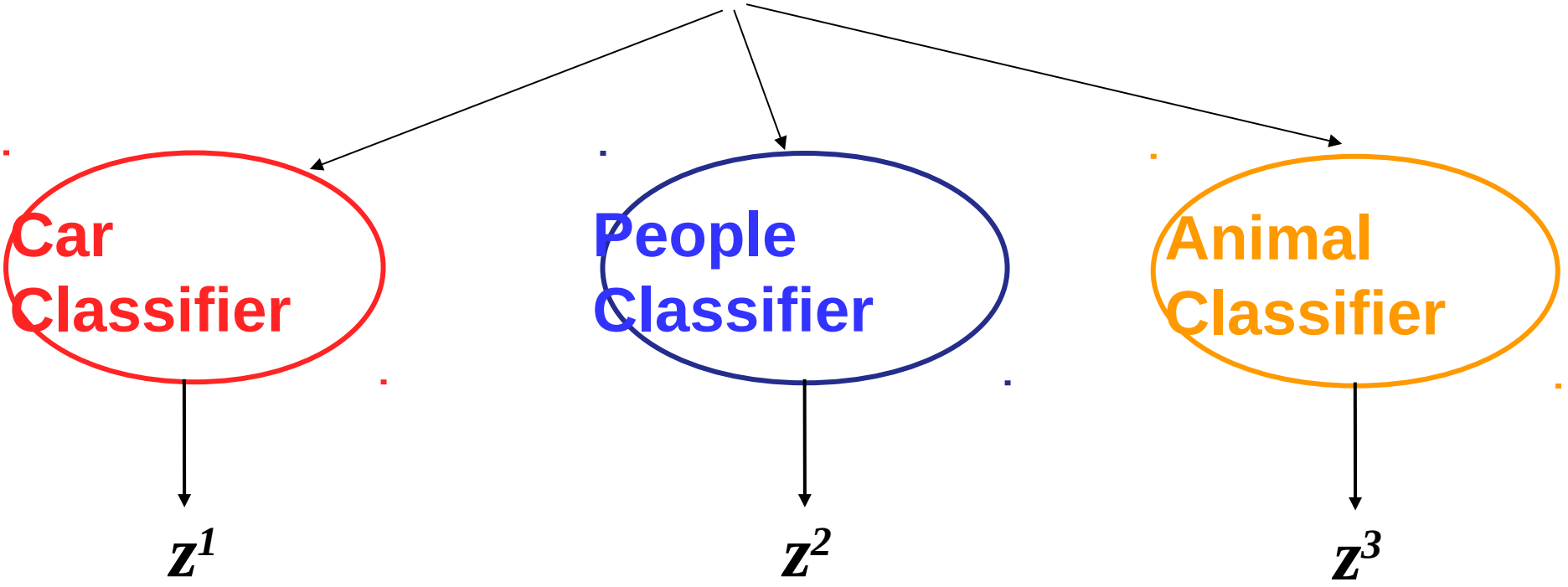
Multi-class with general classifiers

- If we have M classes, we train M binary classifiers (C)
- For each binary classifier C_i we label each example in the training data as 1 if the example corresponds to class i and -1 (or 0) otherwise (one-against-all)



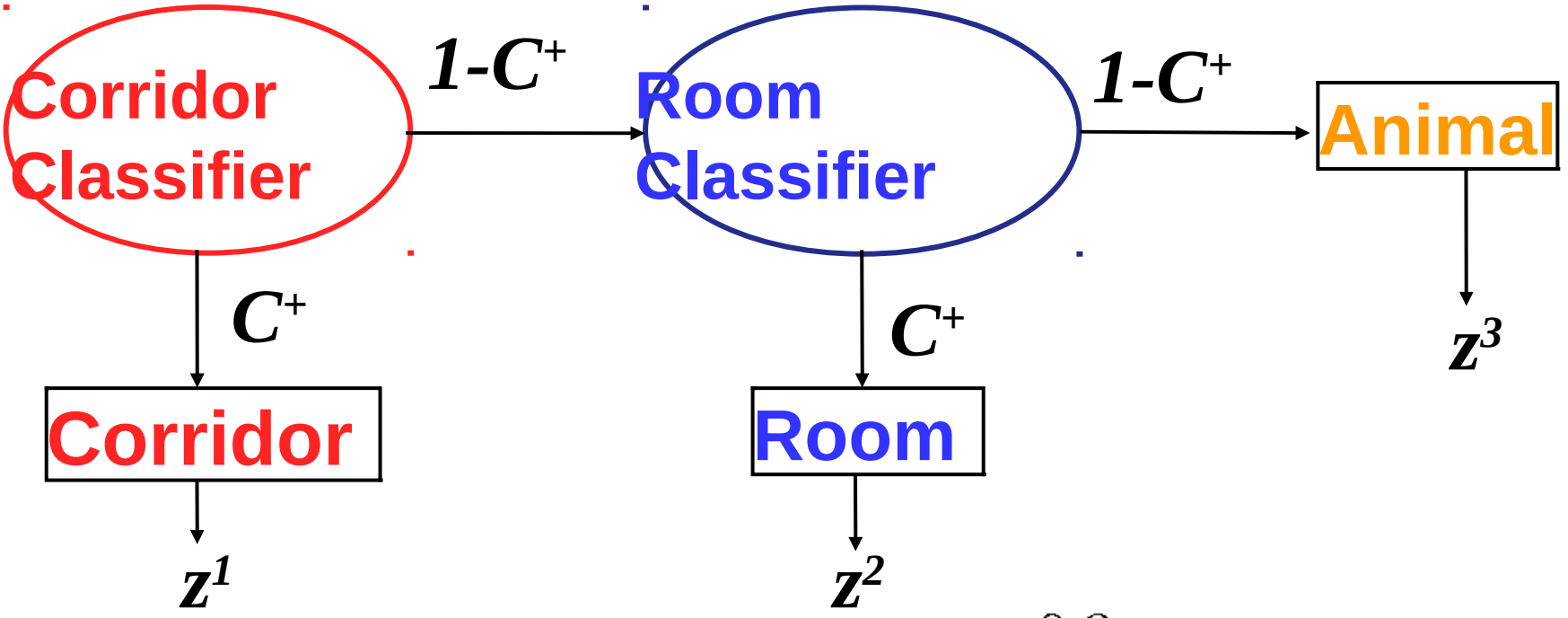
Multi-class with general classifiers

- We can run all classifiers in parallel and select the class with highest confidence. For this we need the classifier to output a confidence value **z**

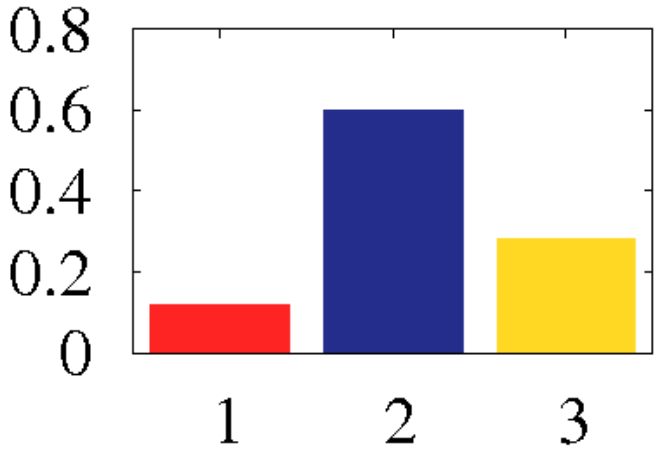


Multiple Classes

- We can also create a decision list

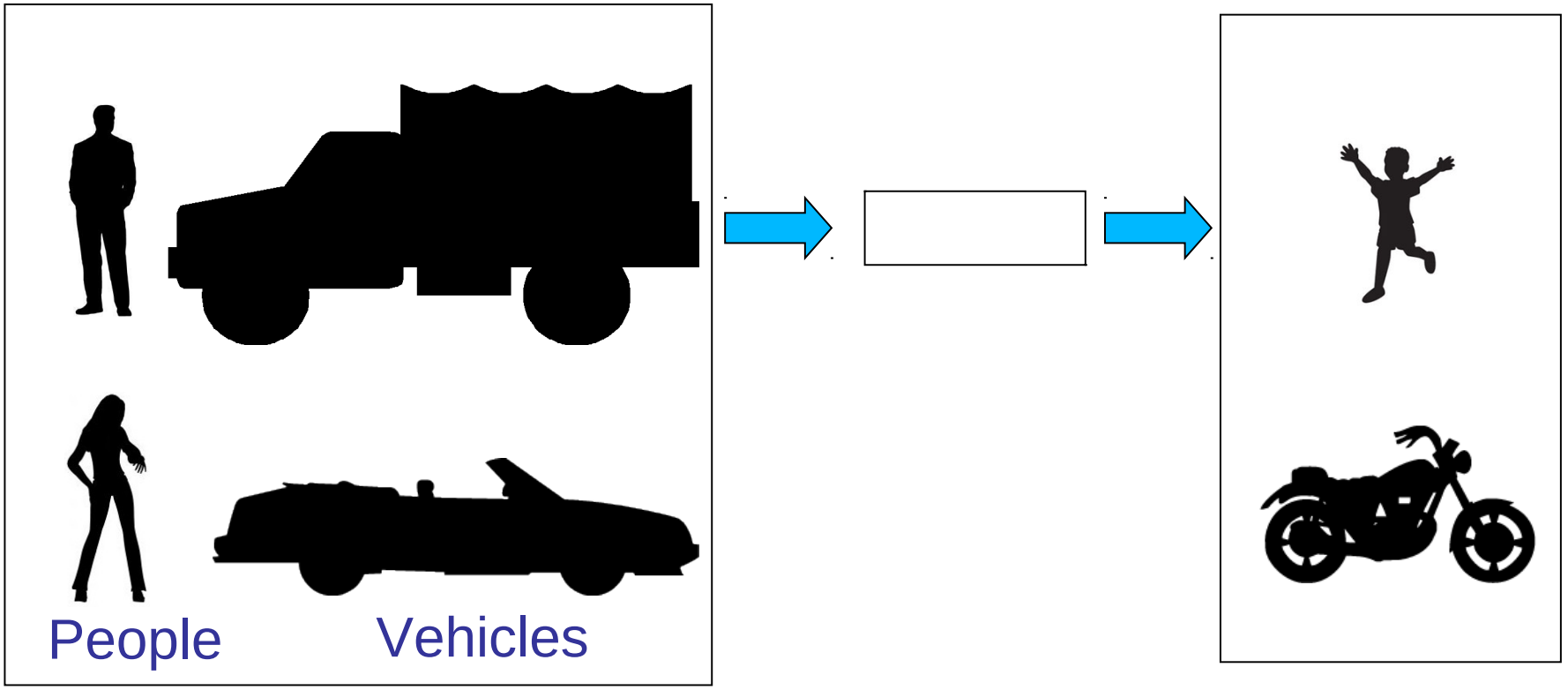


$$z^{[k]} = C_k^+ \prod_{j=1}^{k-1} (1 - C_j^+)$$



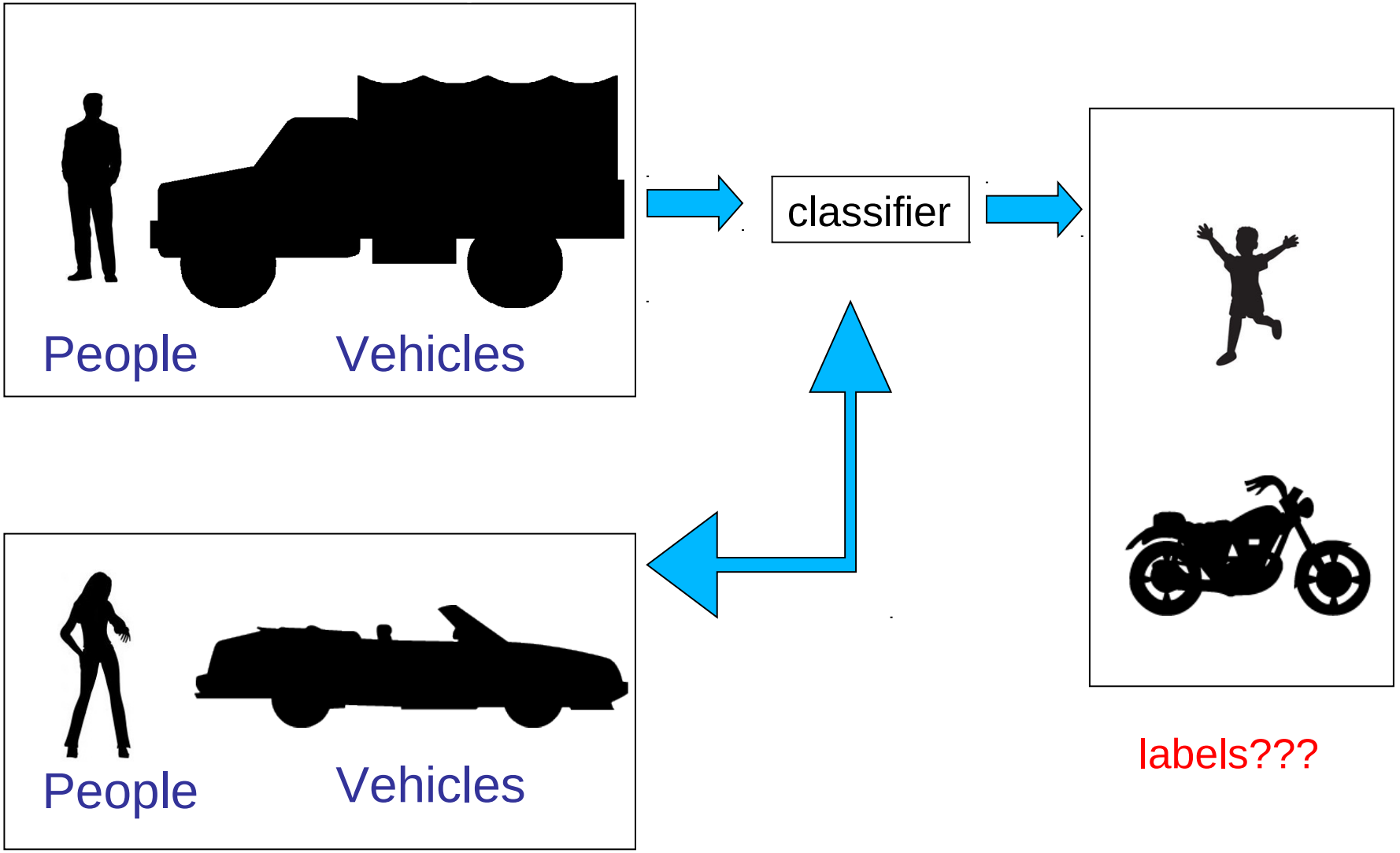
Training

- We create (train) our classifier using a training set
- We evaluate our classifier in a test set.



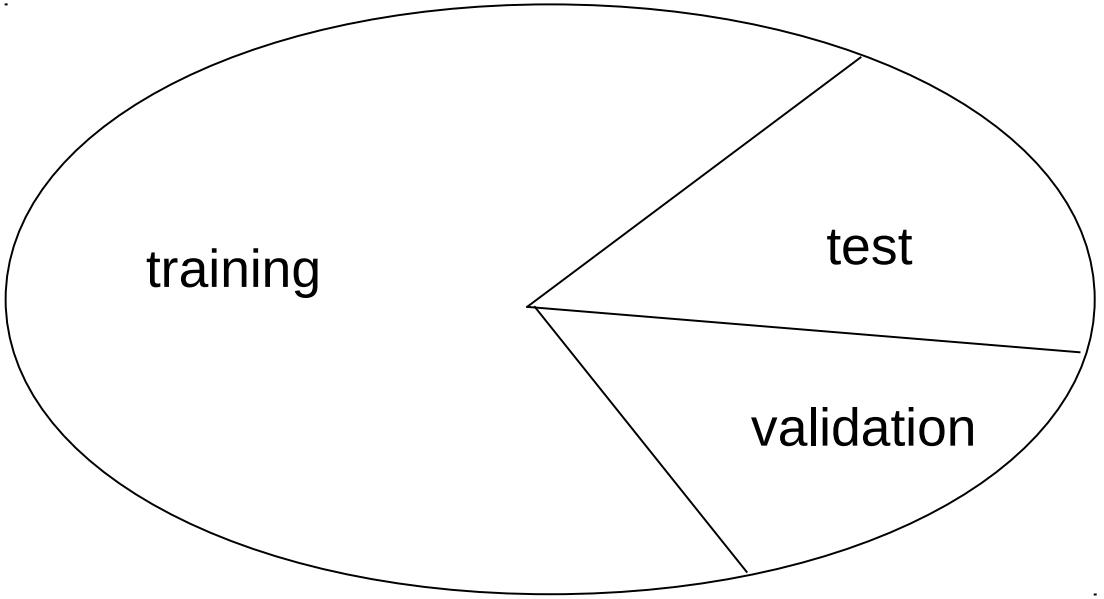
Training

- We can use a **validation set** to **tune** our classifier before testing, e.g. SVMs need parameter tuning



Sets

- Usually a set of labelled examples is divided into training (~60%) and validation (~20%), and test (~20%) sets.



Evaluation

- The **confusion matrix** shows the classification rates
- Rows indicate original labels
- Columns indicate outputs from the classifier

%	People	Cars	Animals
People	**80**	10	10
Car	5	**90**	5
Animals	12	10	**78**

- The diagonals indicate the **true classifications**.
- The rest are **false classifications**.
- A **good classifier** has **high true** classifications and **low false** classifications.

Evaluation

- In a binary classifier we two classes **positives** and **negatives**

%	Positive	Negative
Positive	**80**	20
Negative	10	**90**

- Each cell has an specific name
  - Positive - Positive **True Positives (TP)**
  - Positive - Negative **False Negative (TN)**
  - Negative - Positive **False Positive (FP)**
  - Negative - Negative **True Negative (TN)**

Evaluation

- Most of the evaluations are for binary classification:

$$\frac{TP+TN}{TP+TN+FP+FN}$$

$$\frac{FP+FN}{TP+TN+FP+FN}$$

$$\frac{TP}{TP+FN}$$

$$\frac{TN}{TN+FP}$$

Evaluation

- Most of the evaluations are for binary classification:

$$\frac{TP+TN}{TP+TN+FP+FN}$$

$$\frac{FP+FN}{TP+TN+FP+FN}$$

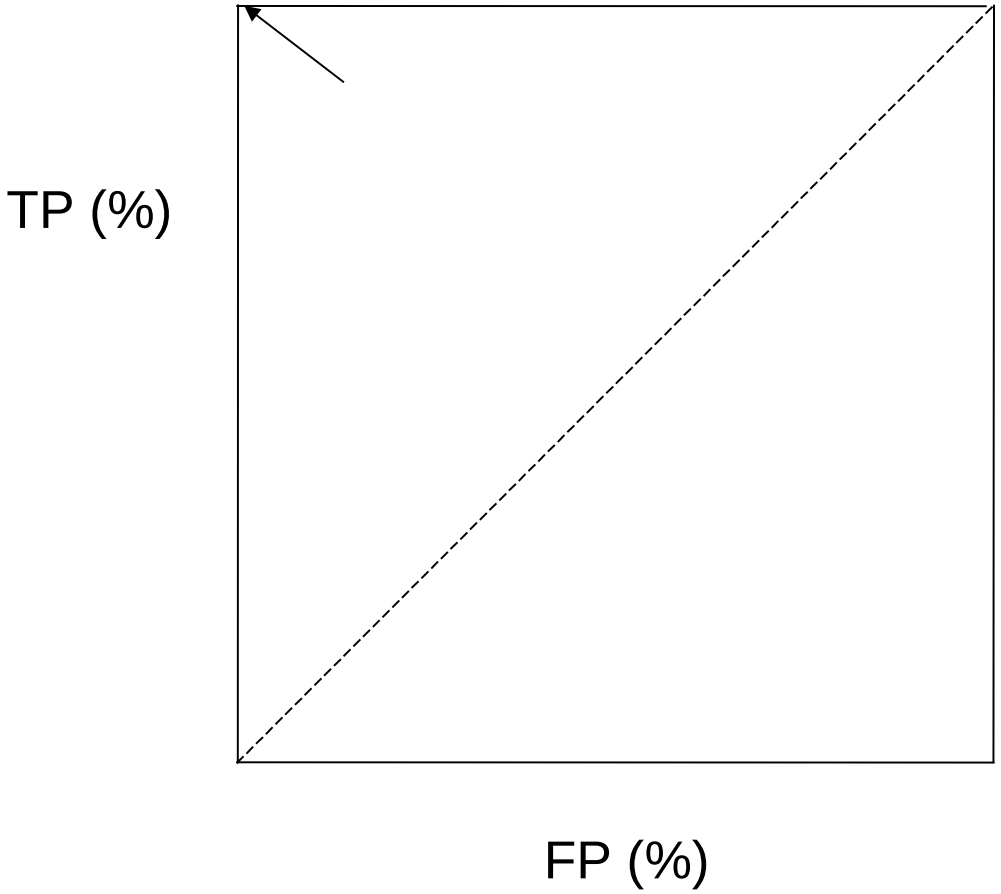
$$\frac{TP}{TP+FN}$$

$$\frac{TN}{TN+FP}$$

Evaluation

- **Receiver operating characteristic** (ROC) curve needs confidence values. Based on positive predictions

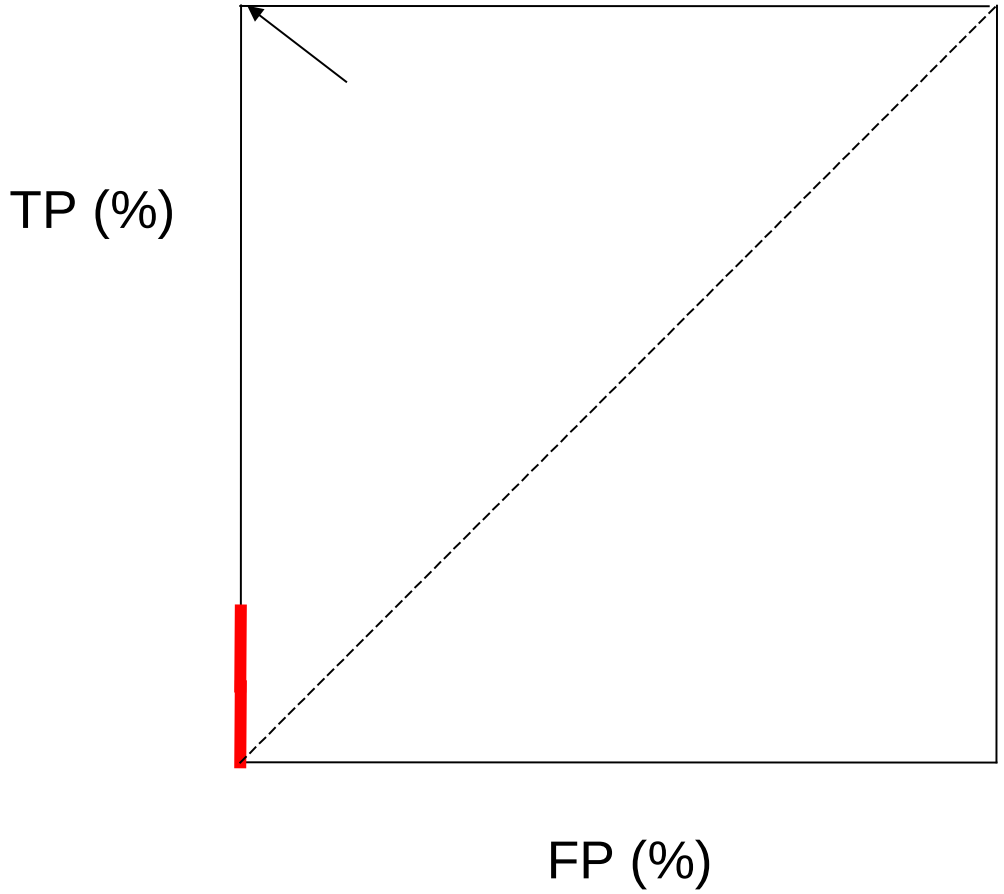
Original	Predicted	Score
P	P	0.96
P	p	0.82
N	P	0.75
N	P	0.62
P	P	0.52
P	P	0.36
N	P	0.20
N	P	0.12
P	P	0.6



Evaluation

- **Receiver operating characteristic** (ROC) curve needs confidence values. Based on positive predictions

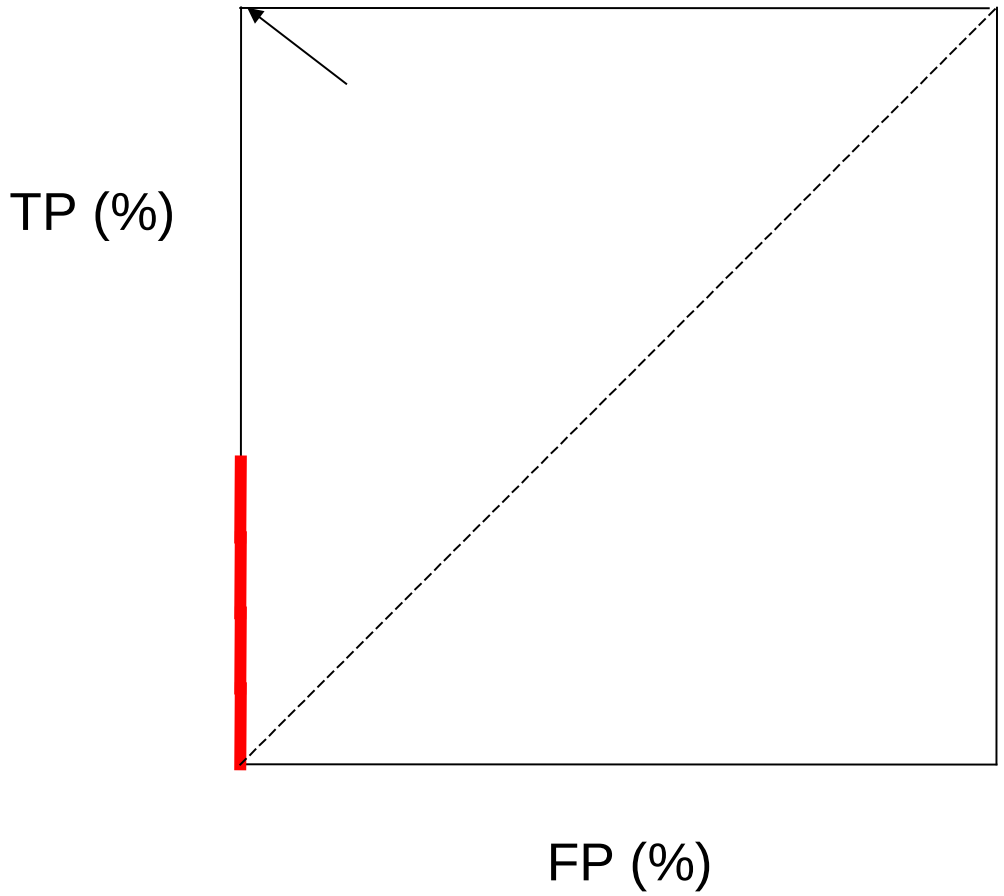
Original	Predicted	Score
P	**P**	0.96
P	P	0.82
N	P	0.75
N	P	0.62
P	P	0.52
P	P	0.36
N	P	0.25
N	P	0.20
N	P	0.12
P	P	0.06



Evaluation

- **Receiver operating characteristic** (ROC) curve needs confidence values. Based on positive predictions

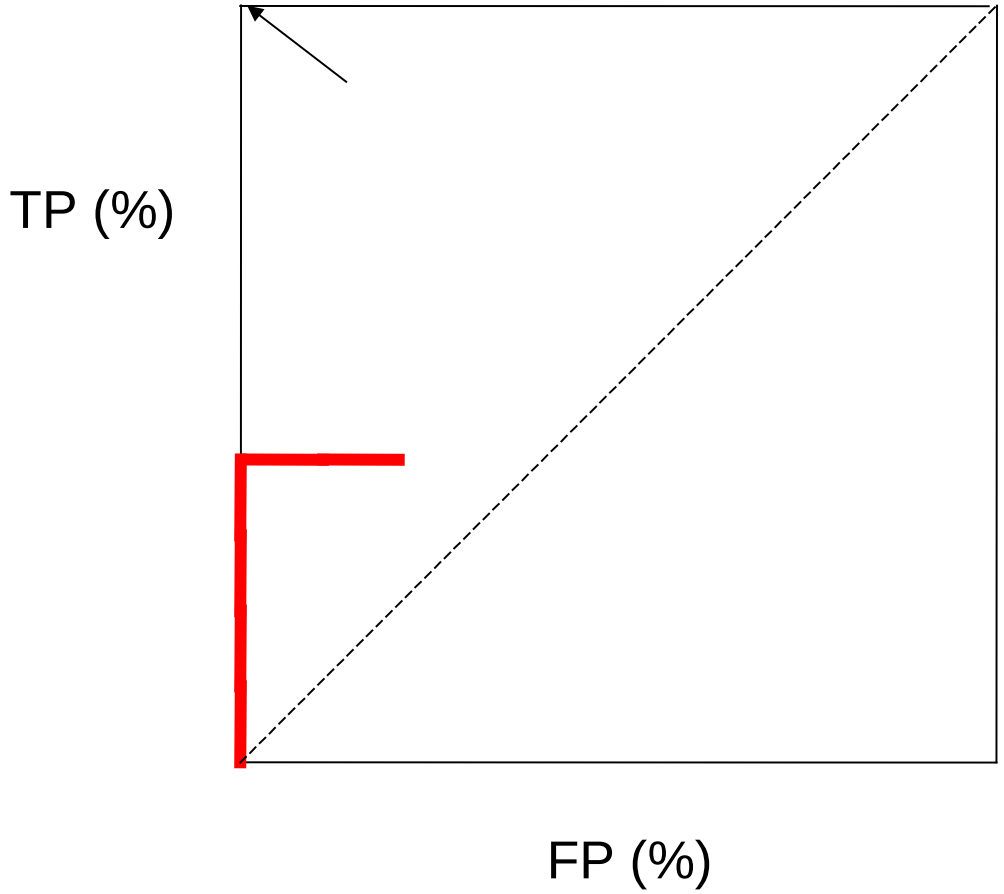
Original	Predicted	Score
P	**P**	0.96
P	**P**	0.82
N	P	0.75
N	P	0.62
P	P	0.52
P	P	0.36
N	P	0.25
N	P	0.20
N	P	0.12
P	P	0.06



Evaluation

- **Receiver operating characteristic** (ROC) curve needs confidence values. Based on positive predictions

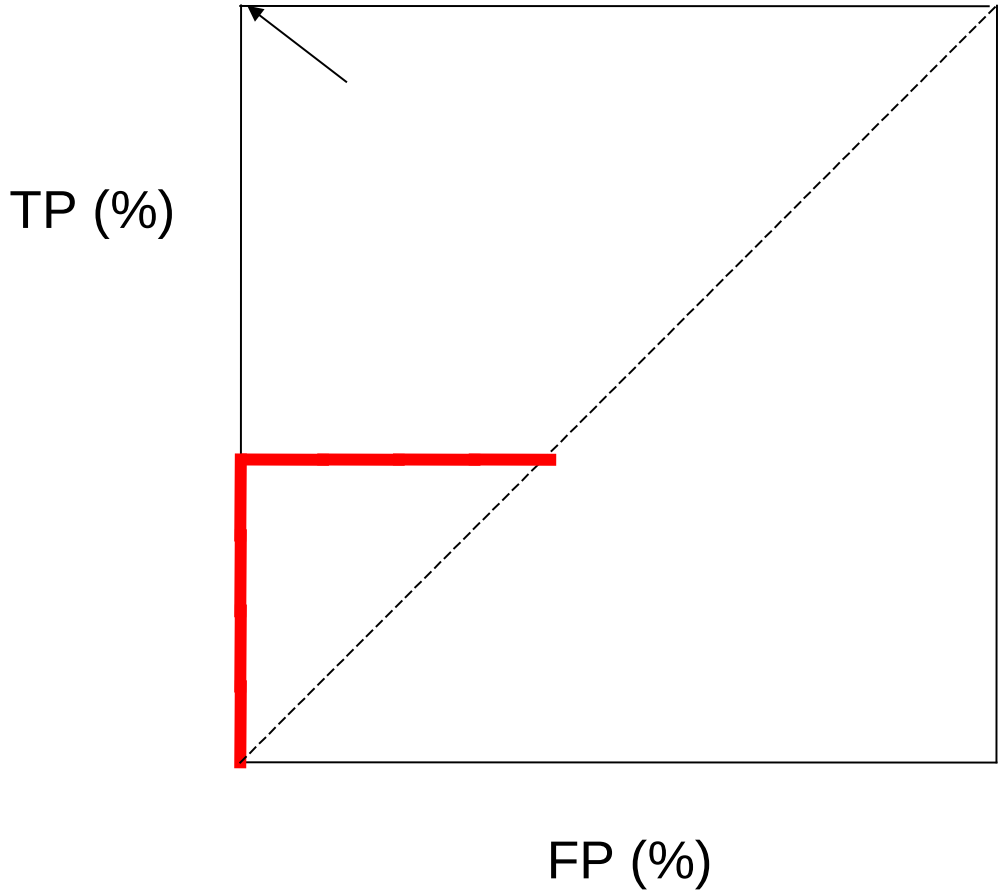
Original	Predicted	Score
P	**P**	0.96
P	**P**	0.82
N	**P**	0.75
N	P	0.62
P	P	0.52
P	P	0.36
N	P	0.25
N	P	0.20
N	P	0.12
P	P	0.06



Evaluation

- **Receiver operating characteristic** (ROC) curve needs confidence values. Based on positive predictions

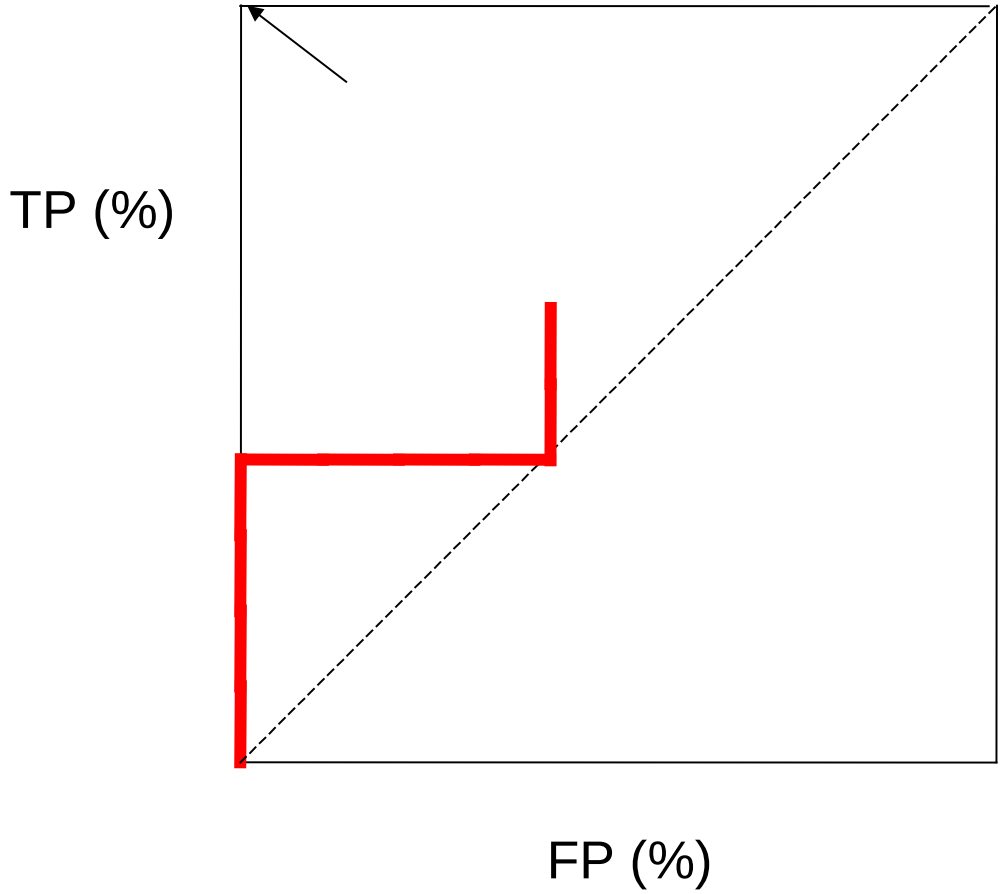
Original	Predicted	Score
P	**P**	0.96
P	**P**	0.82
N	**P**	0.75
N	**P**	0.62
P	P	0.52
P	P	0.36
N	P	0.25
N	P	0.20
N	P	0.12
P	P	0.06



Evaluation

- **Receiver operating characteristic** (ROC) curve needs confidence values. Based on positive predictions

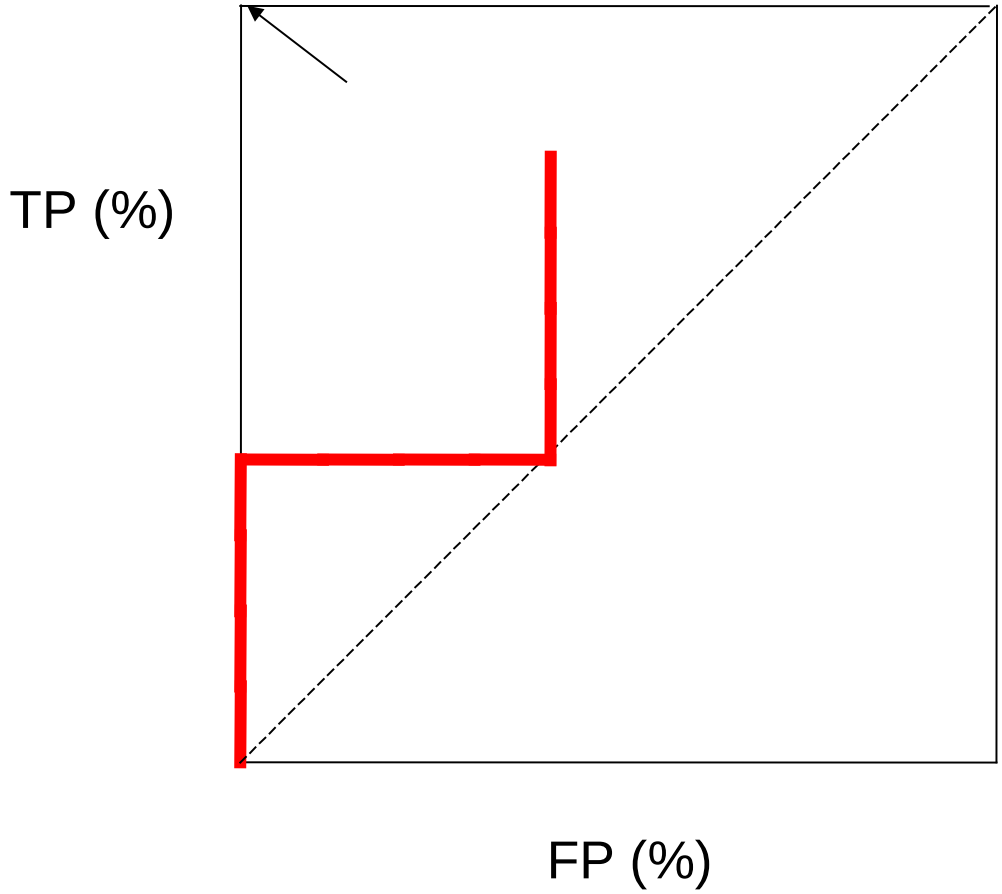
Original	Predicted	Score
P	**P**	0.96
P	**P**	0.82
N	**P**	0.75
N	**P**	0.62
P	**P**	0.52
P	P	0.36
N	P	0.25
N	P	0.20
N	P	0.12
P	P	0.06



Evaluation

- **Receiver operating characteristic** (ROC) curve needs confidence values. Based on positive predictions

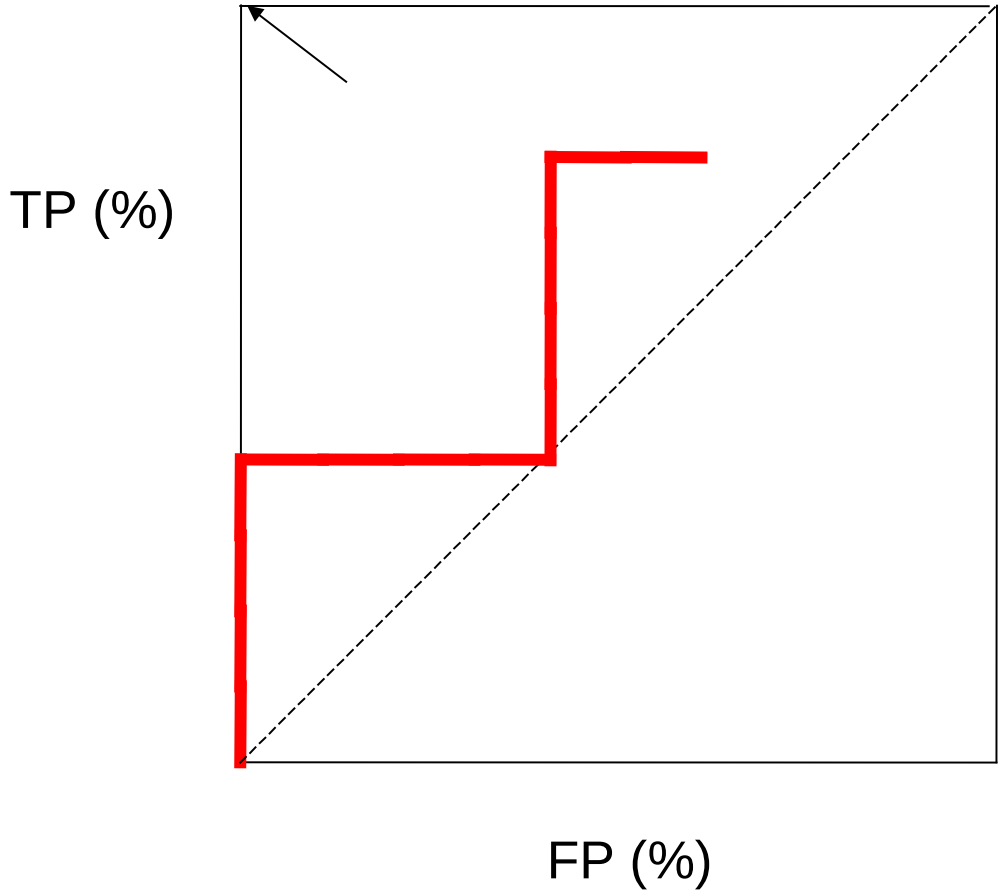
Original	Predicted	Score
P	**P**	0.96
P	**P**	0.82
N	**P**	0.75
N	**P**	0.62
P	**P**	0.52
P	**P**	0.36
N	P	0.25
N	P	0.20
N	P	0.12
P	P	0.06



Evaluation

- **Receiver operating characteristic** (ROC) curve needs confidence values. Based on positive predictions

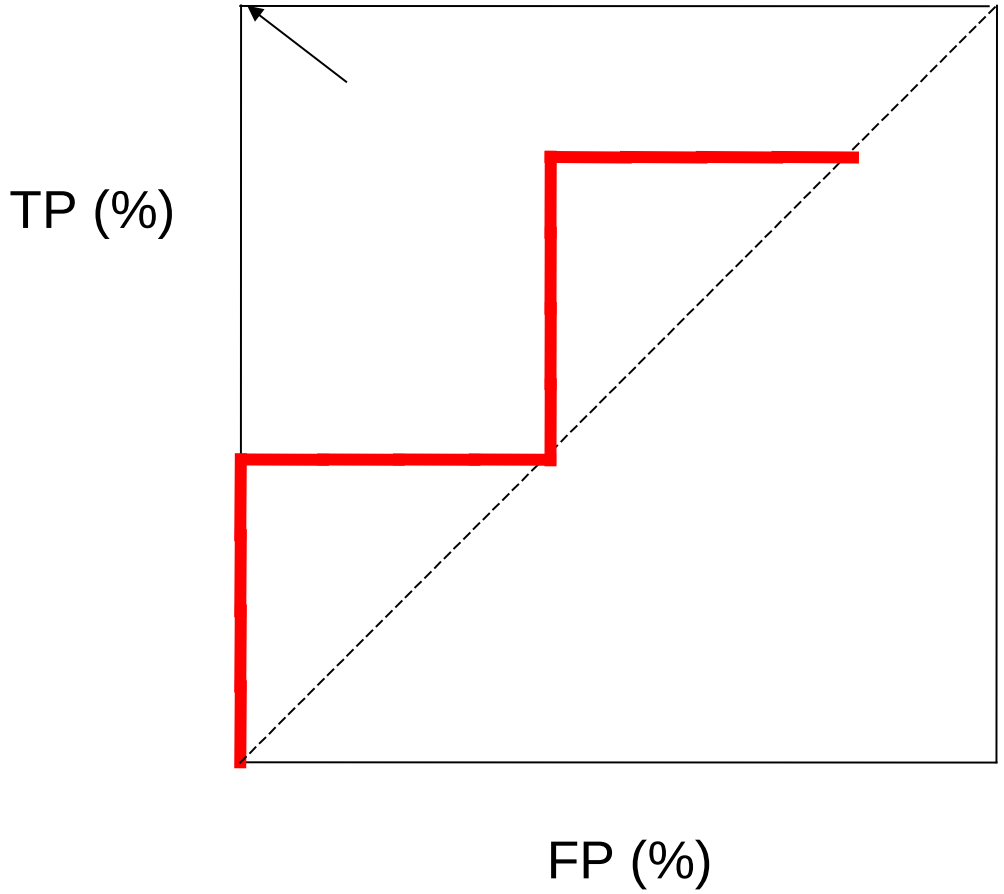
Original	Predicted	Score
P	**P**	0.96
P	**P**	0.82
N	**P**	0.75
N	**P**	0.62
P	**P**	0.52
P	**P**	0.36
N	**P**	0.25
N	P	0.20
N	P	0.12
P	P	0.06



Evaluation

- **Receiver operating characteristic** (ROC) curve needs confidence values. Based on positive predictions

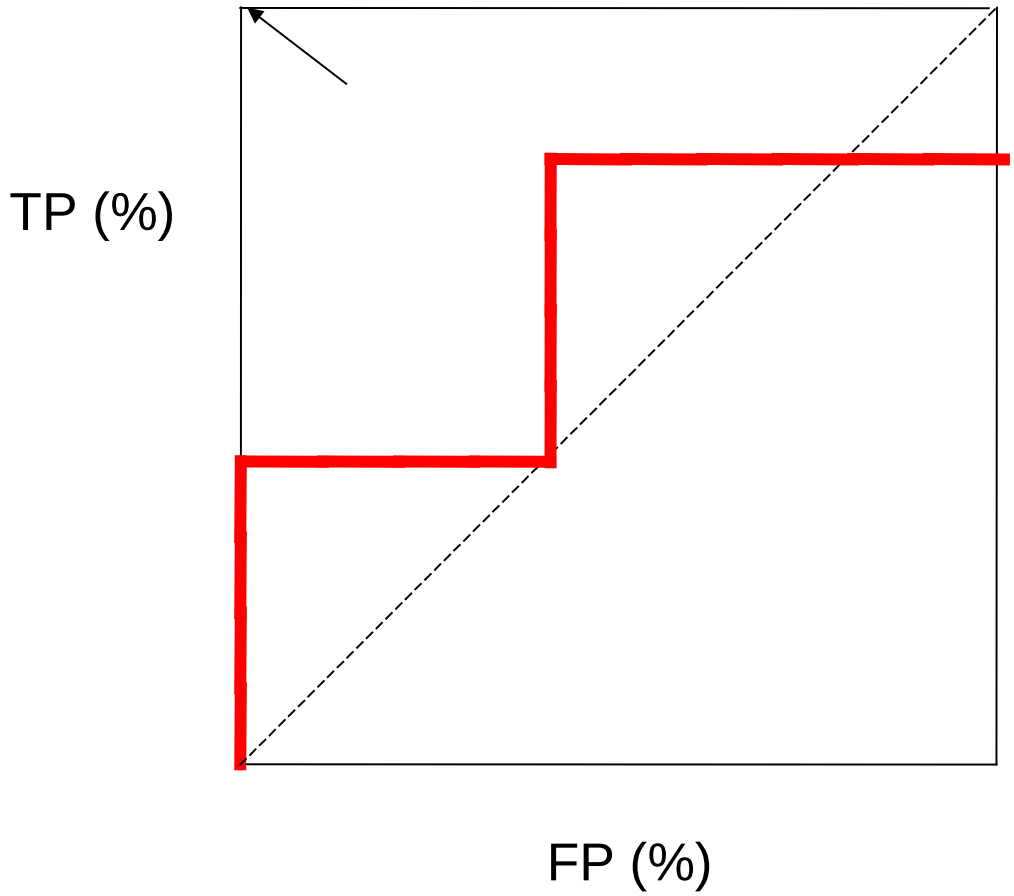
Original	Predicted	Score
P	**P**	0.96
P	**P**	0.82
N	**P**	0.75
N	**P**	0.62
P	**P**	0.52
P	**P**	0.36
N	**P**	0.25
N	**P**	0.20
N	P	0.12
P	P	0.06



Evaluation

- **Receiver operating characteristic** (ROC) curve needs confidence values. Based on positive predictions

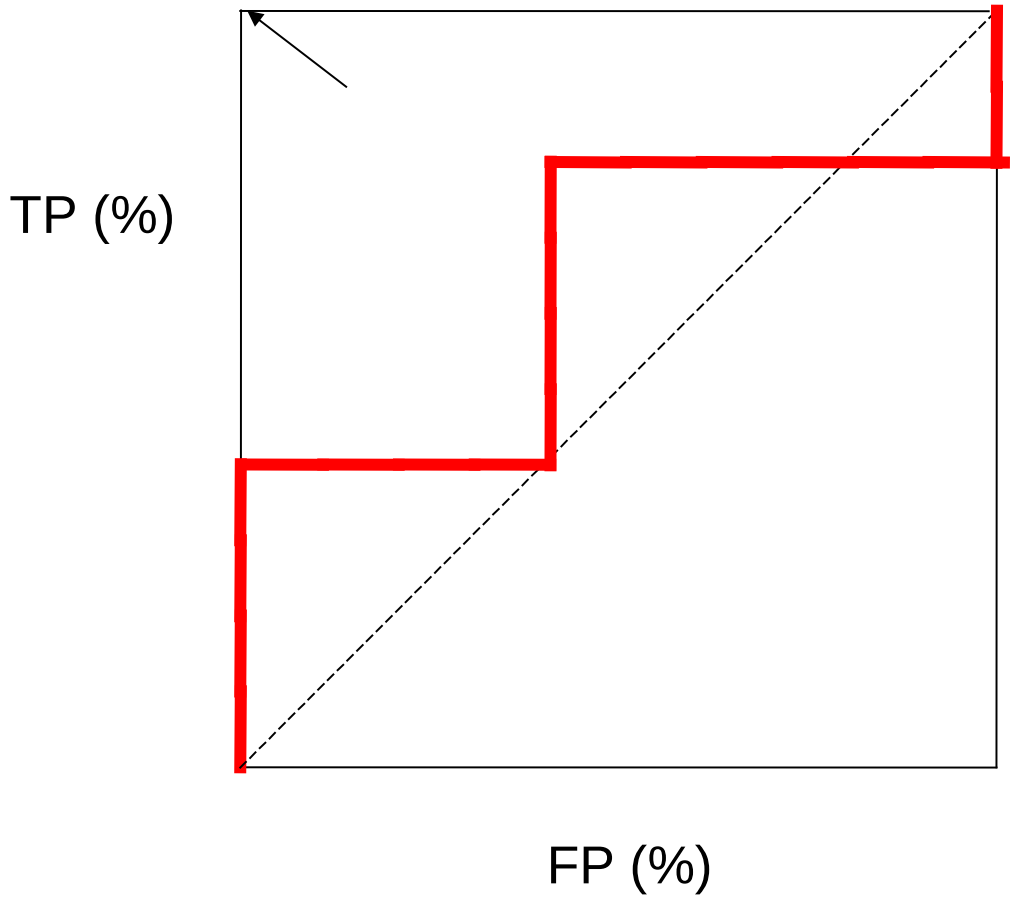
Original	Predicted	Score
P	**P**	0.96
P	**P**	0.82
N	**P**	0.75
N	**P**	0.62
P	**P**	0.52
P	**P**	0.36
N	**P**	0.25
N	**P**	0.20
N	**P**	0.12
P	P	0.06



Evaluation

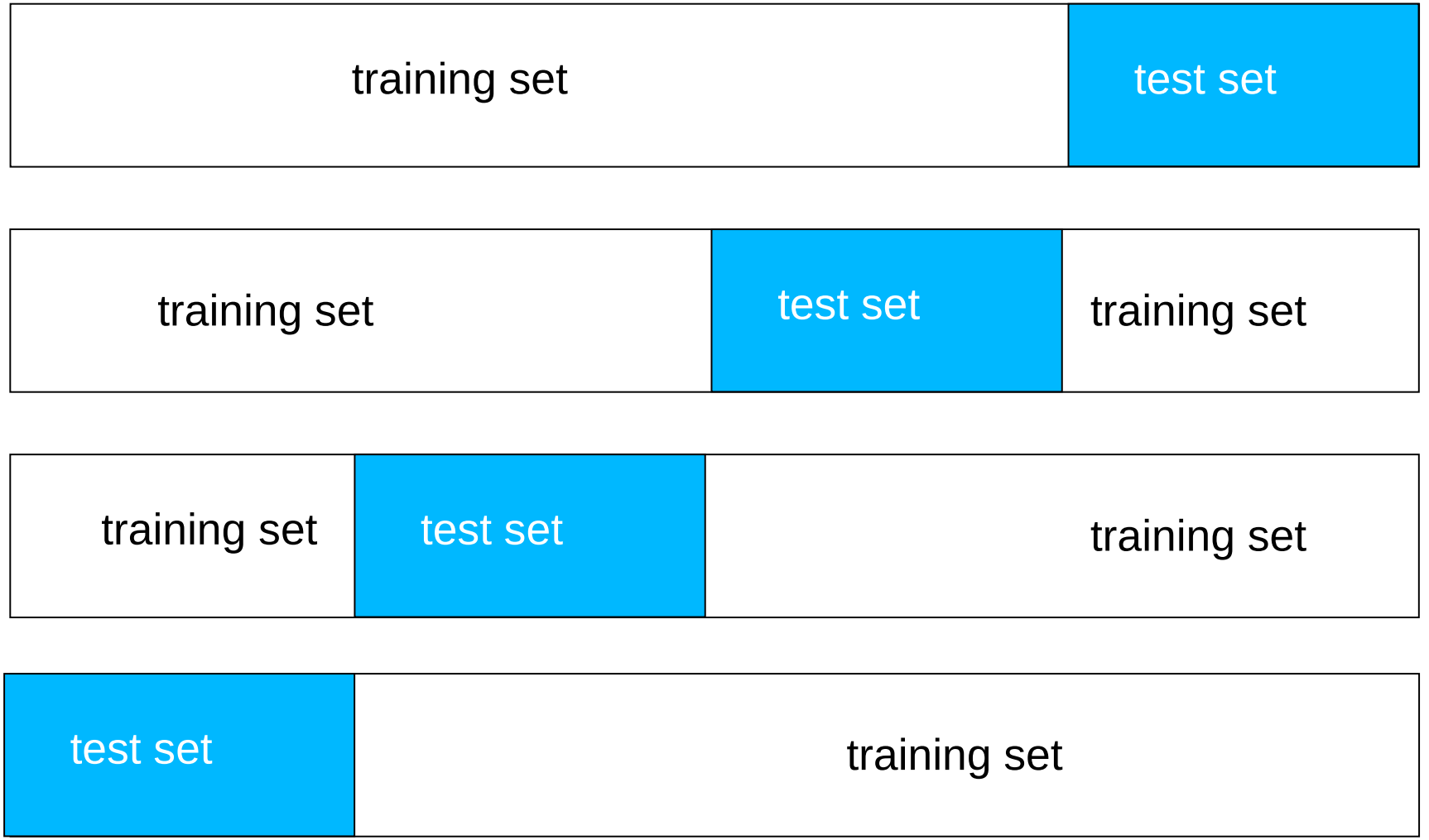
- **Receiver operating characteristic** (ROC) curve needs confidence values. Based on positive predictions

Original	Predicted	Score
P	**P**	0.96
P	**P**	0.82
N	**P**	0.75
N	**P**	0.62
P	**P**	0.52
P	**P**	0.36
N	**P**	0.25
N	**P**	0.20
N	**P**	0.12
P	**P**	0.06



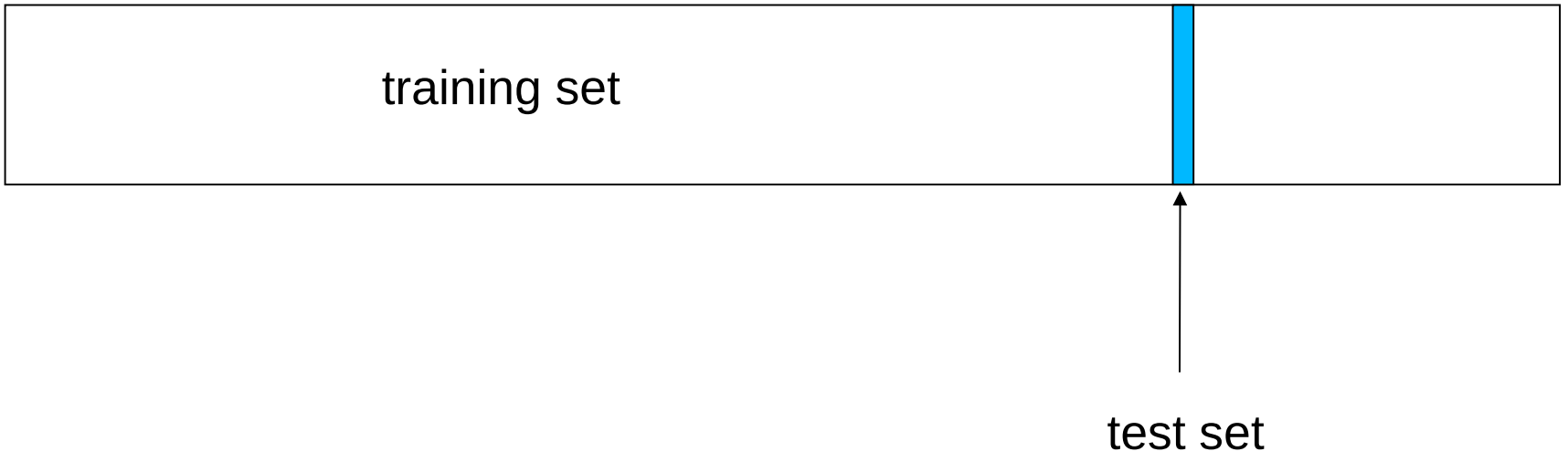
Repeating experiments

- If our set is big enough we can repeat the experiment using different training and test sets K times: **K-fold cross validation**



Repeating experiments

- We then calculate the **average** and **std** of the accuracies over the K experiments
- An extreme case is when in each experiment we use only **one example** in the test set: **leave-one-out**
- If N is the total number of examples, then we run N experiments



Tools

- Weka http://www.cs.waikato.ac.nz/ml/weka/
- Libsvm http://www.csie.ntu.edu.tw/~cjlin/libsvm/
- OpenCV opencv.org/

Some references

- **Clustering + Bayes:**
  - "Pattern Recognition," Sergios Theodoridis , Konstantinos Koutroumbas

- **SVM + generative models + etc:**
  - "Pattern Recognition and Machine," Learning Christopher M. Bishop
  - Libsvm tutorial

- **Naive Bayes:**
  - "Machine learning," Tom Mitchel

- **Everything:**
  - "Data Mining," Ian Witten, Eibe Frank, Mark Hall (Weka Book)

Thank You!