

Introduction to Machine Learning

TUM Praktikum:
Sensorgestützte intelligente Umgebungen

Oscar Martinez Mozos
Robotics Group, University of Zaragoza, Spain



Example classification problem

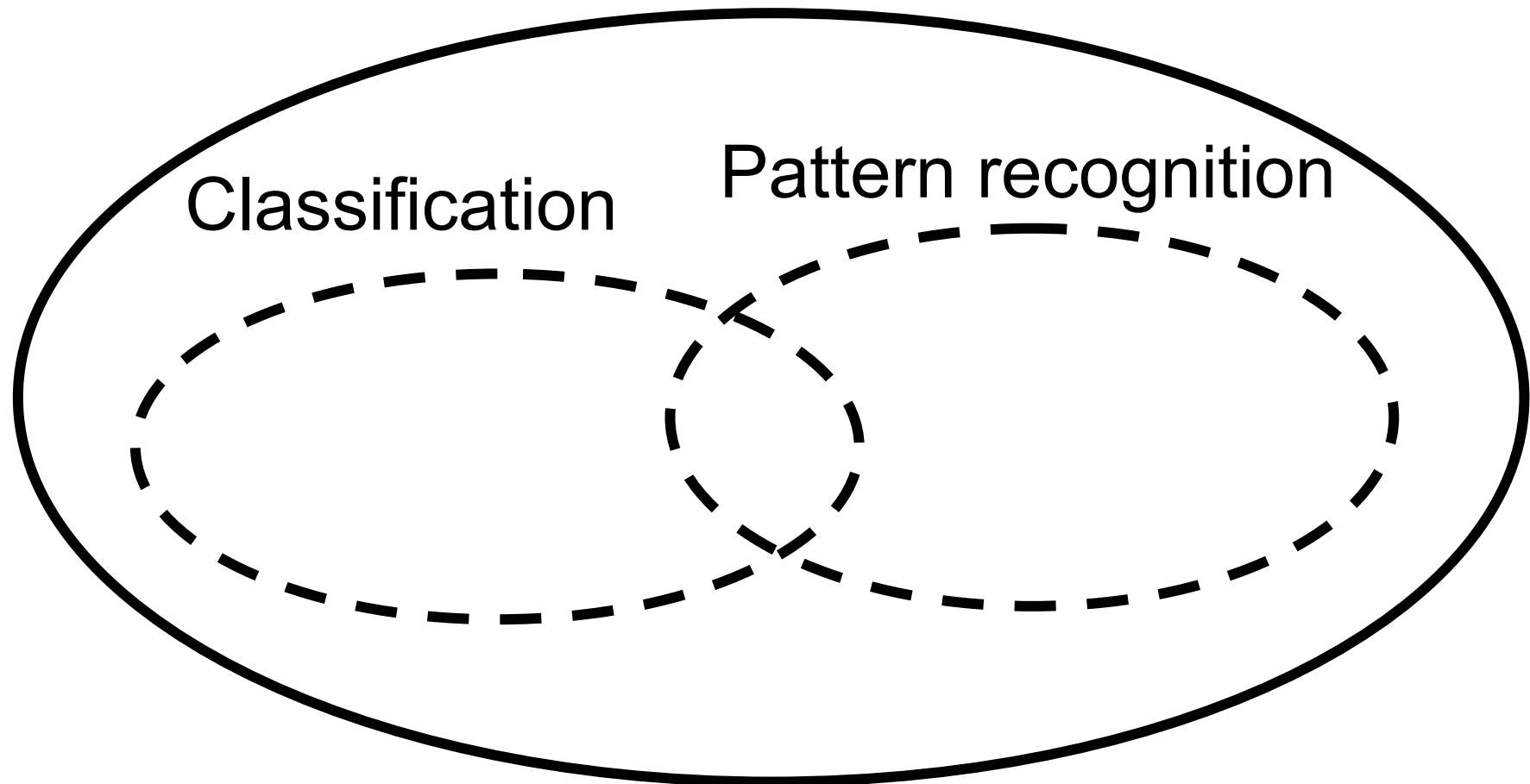
- This is the weather prediction of two cities for the next days:

Sat May 29	 Sunny	26° 16°
Sun May 30	 Sunny	25° 16°
Mon May 31	 Partly Cloudy	25° 17°
Tue Jun 1	 Mostly Sunny	26° 17°
Wed Jun 2	 Partly Cloudy	25° 17°

Sat May 29	 Partly Cloudy	21° 11°
Sun May 30	 Rain	16° 10°
Mon May 31	 Light Rain	13° 8°
Tue Jun 1	 Rain	15° 8°
Wed Jun 2	 Light Rain	17° 9°

Classify the prediction into **Alicante** or **Munich...**

Machine Learning



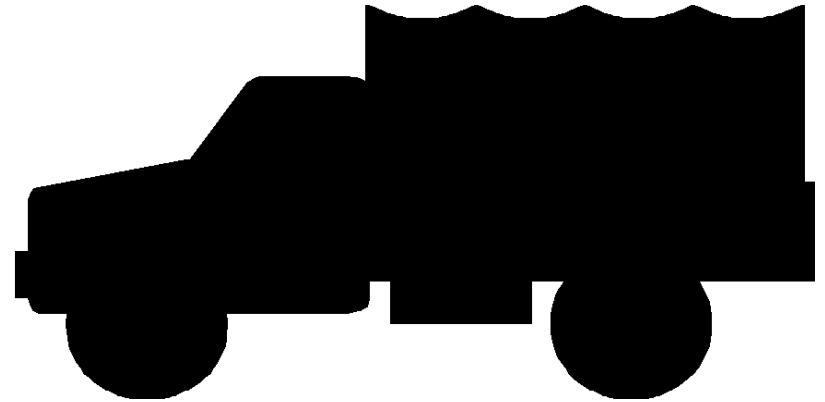
- Sometimes the differences are unclear. You can use the three definitions indistinctly.
- In this course we concentrate on labeling observations -> Pure classification problem.

What does classification mean?

- We give labels to examples representing something:



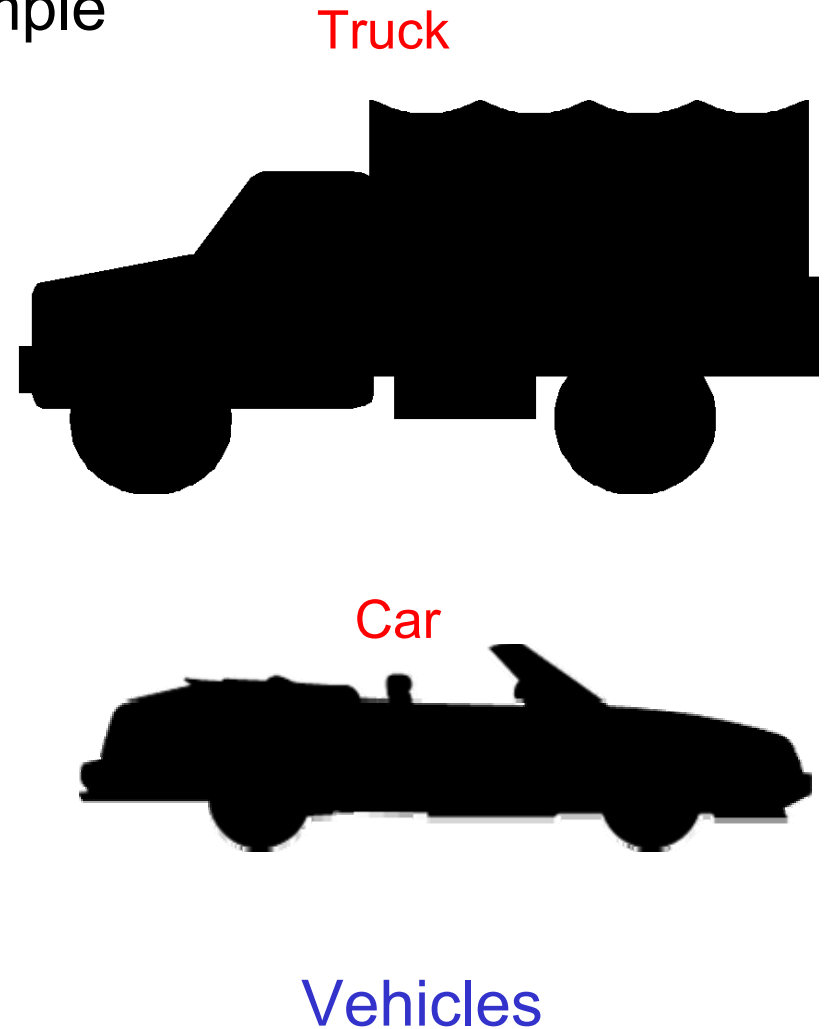
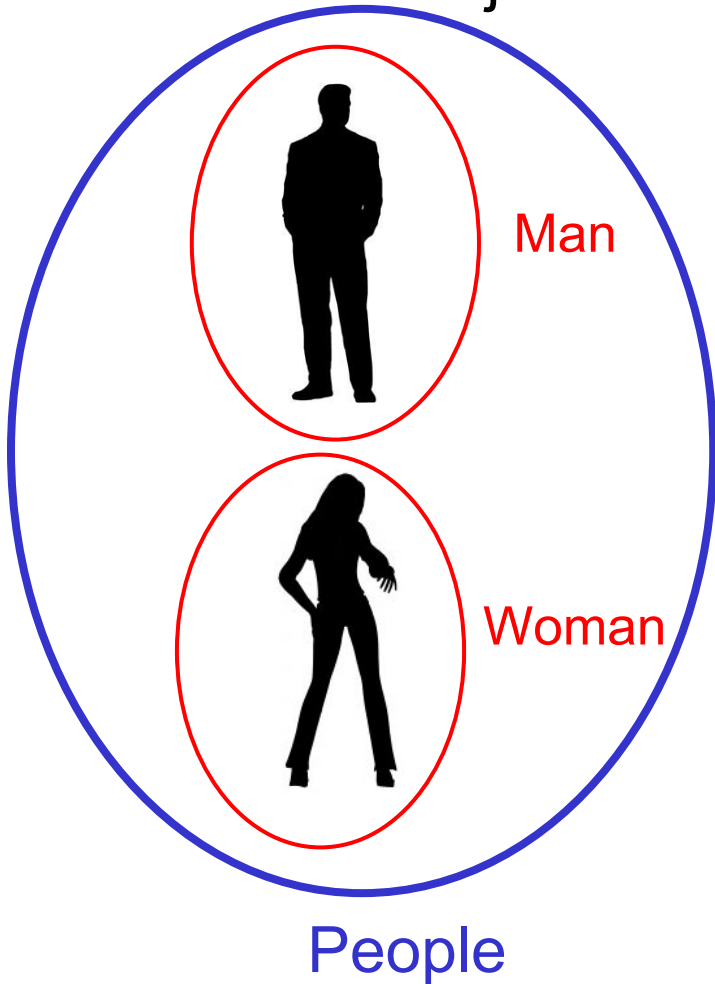
Label: People



Label: Vehicles

Classes and instances

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



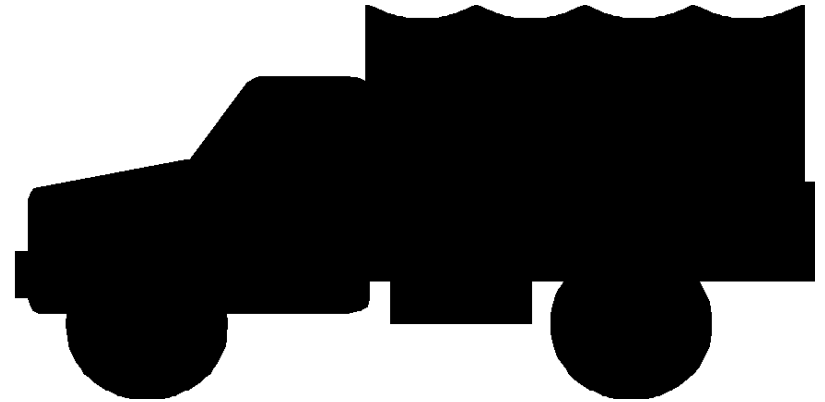
In this talk we concentrate on **classes**.

Representing the examples

- Computers understand numbers. Transform examples into numbers. How? → **Feature extraction**



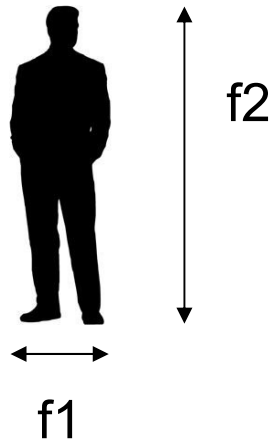
People



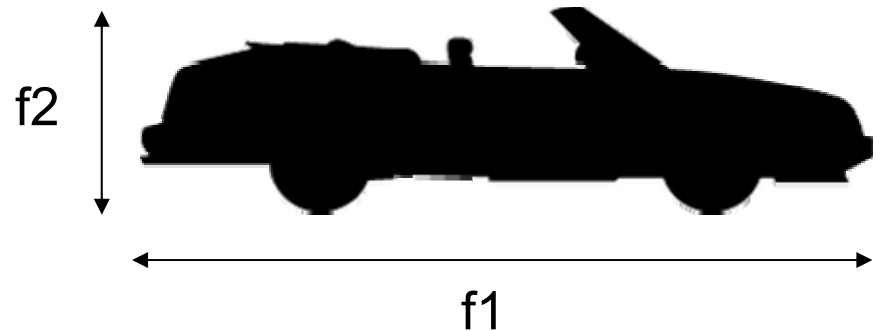
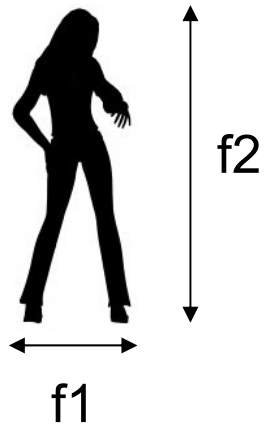
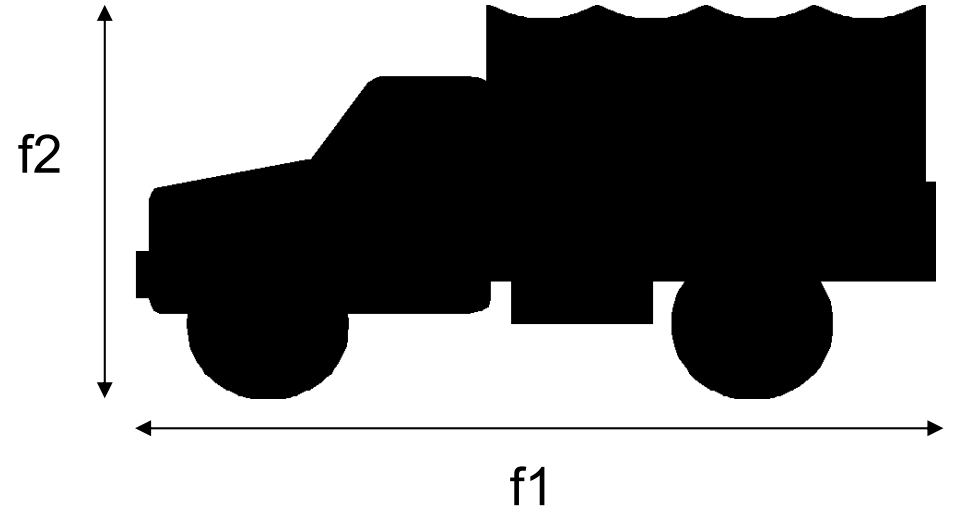
Vehicles

Representing the examples

- Computers understand numbers. Transform examples into numbers. How? → **Feature extraction**



f1: width
f2: height



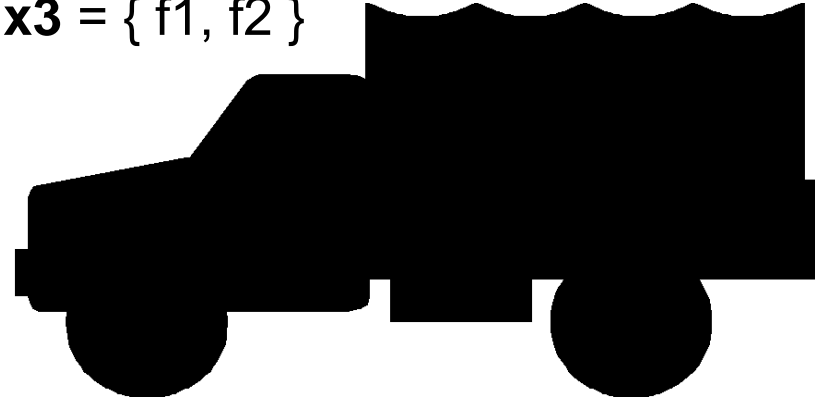
Feature vector

- Each example is now represented by its set of features

$$x_1 = \{ f_1, f_2 \}$$



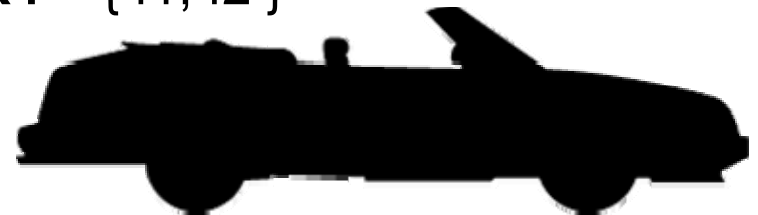
$$x_3 = \{ f_1, f_2 \}$$



$$x_2 = \{ f_1, f_2 \}$$

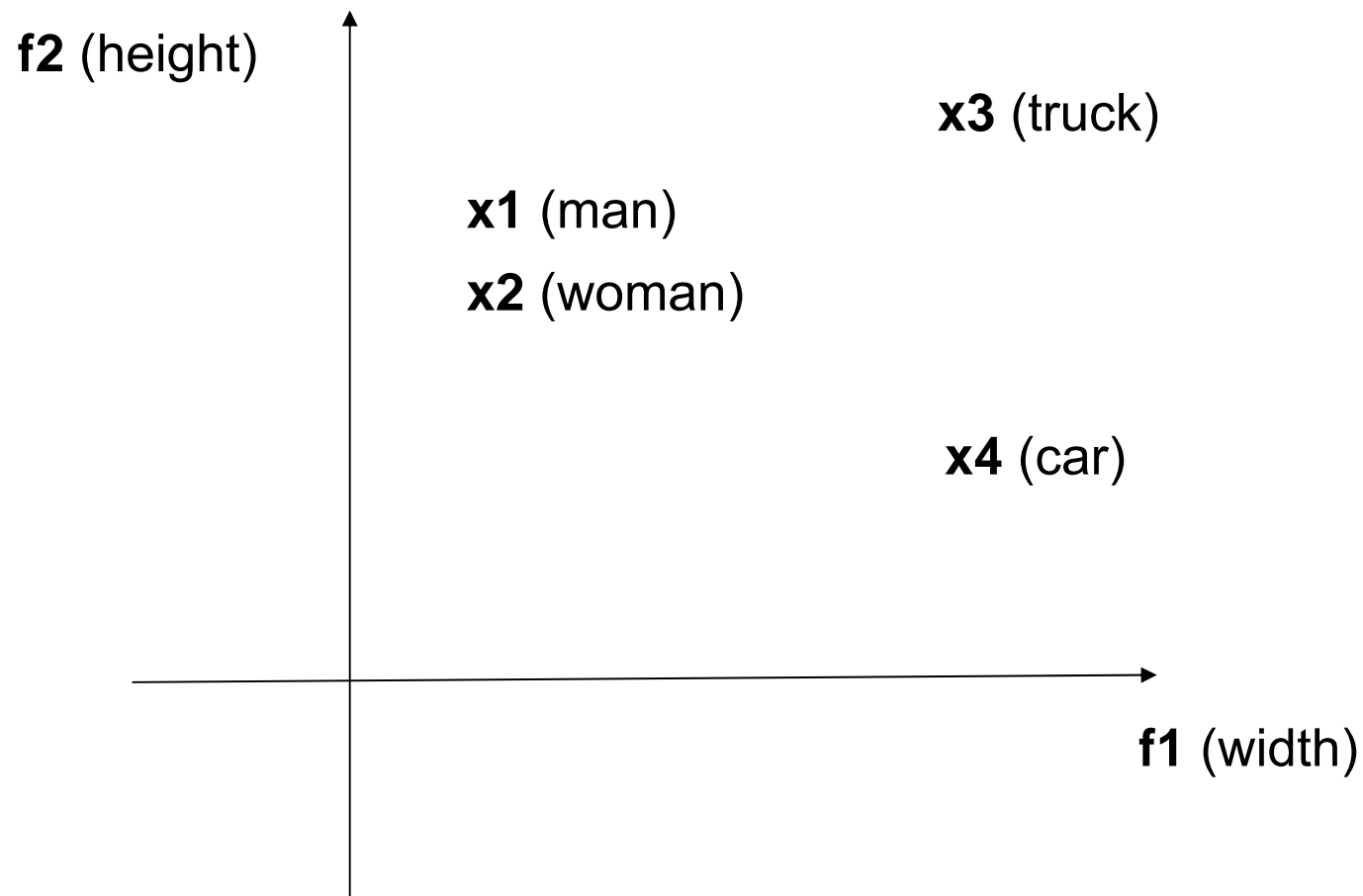


$$x_4 = \{ f_1, f_2 \}$$



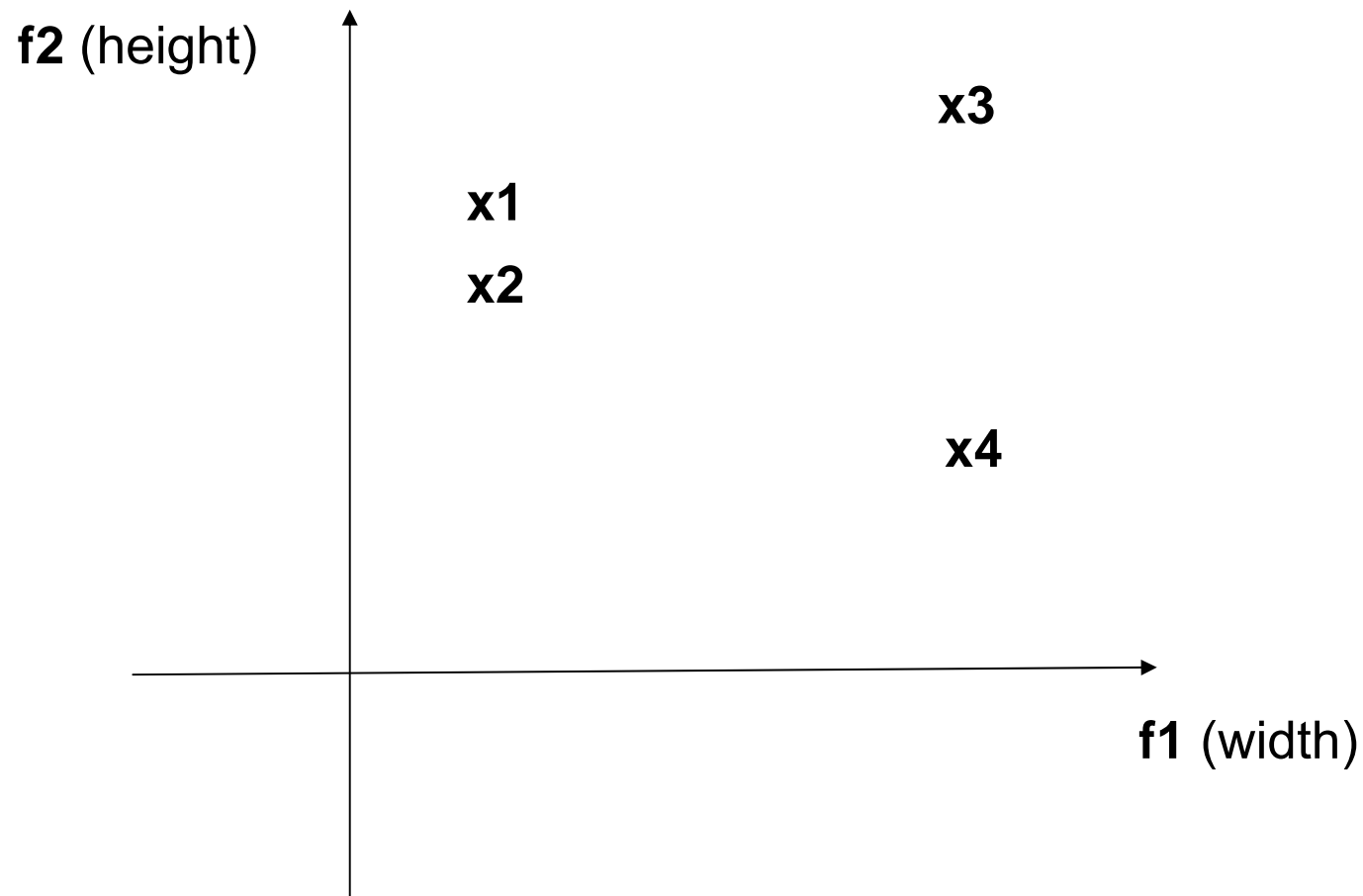
Feature space

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



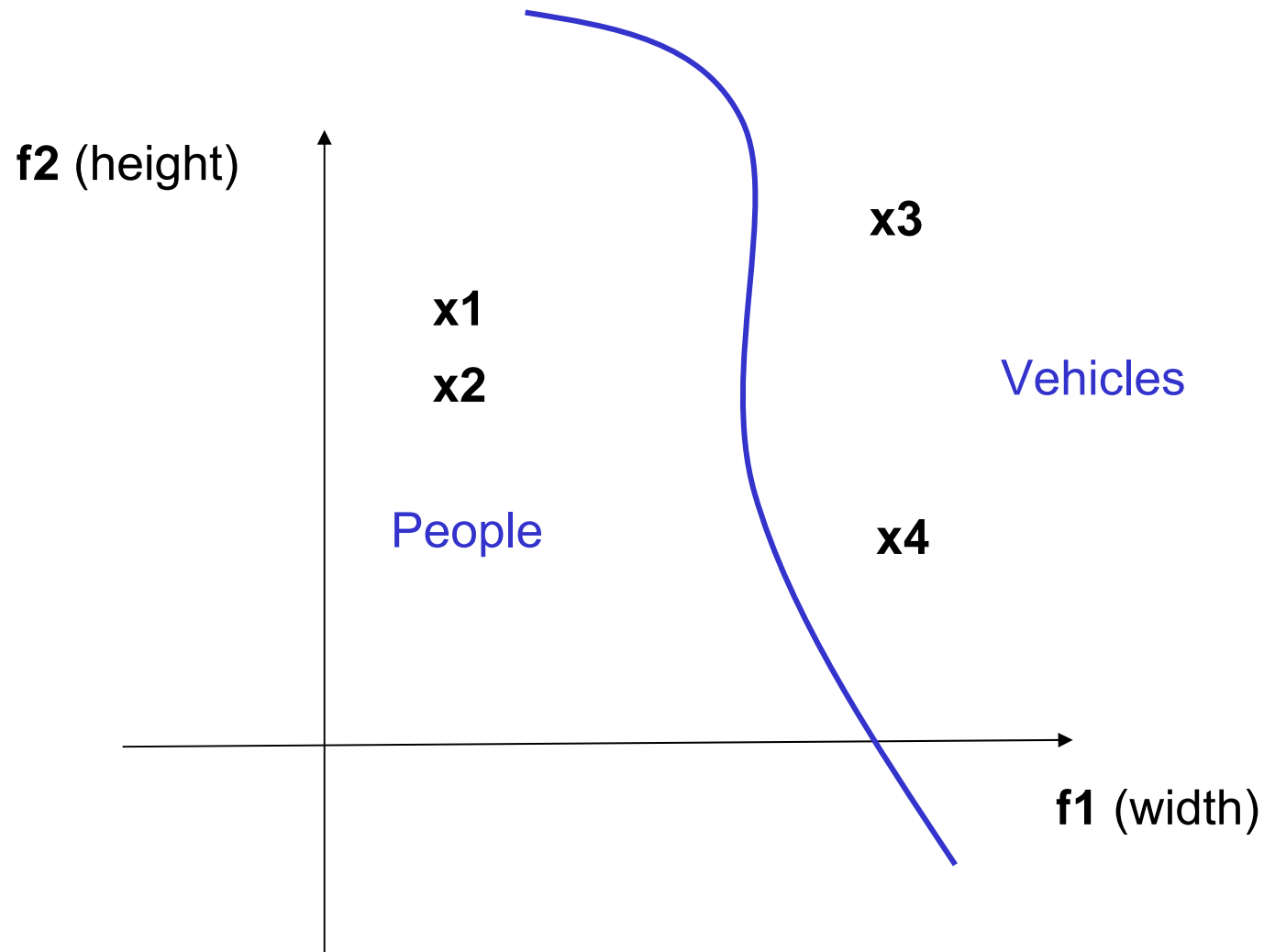
General idea of classification

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



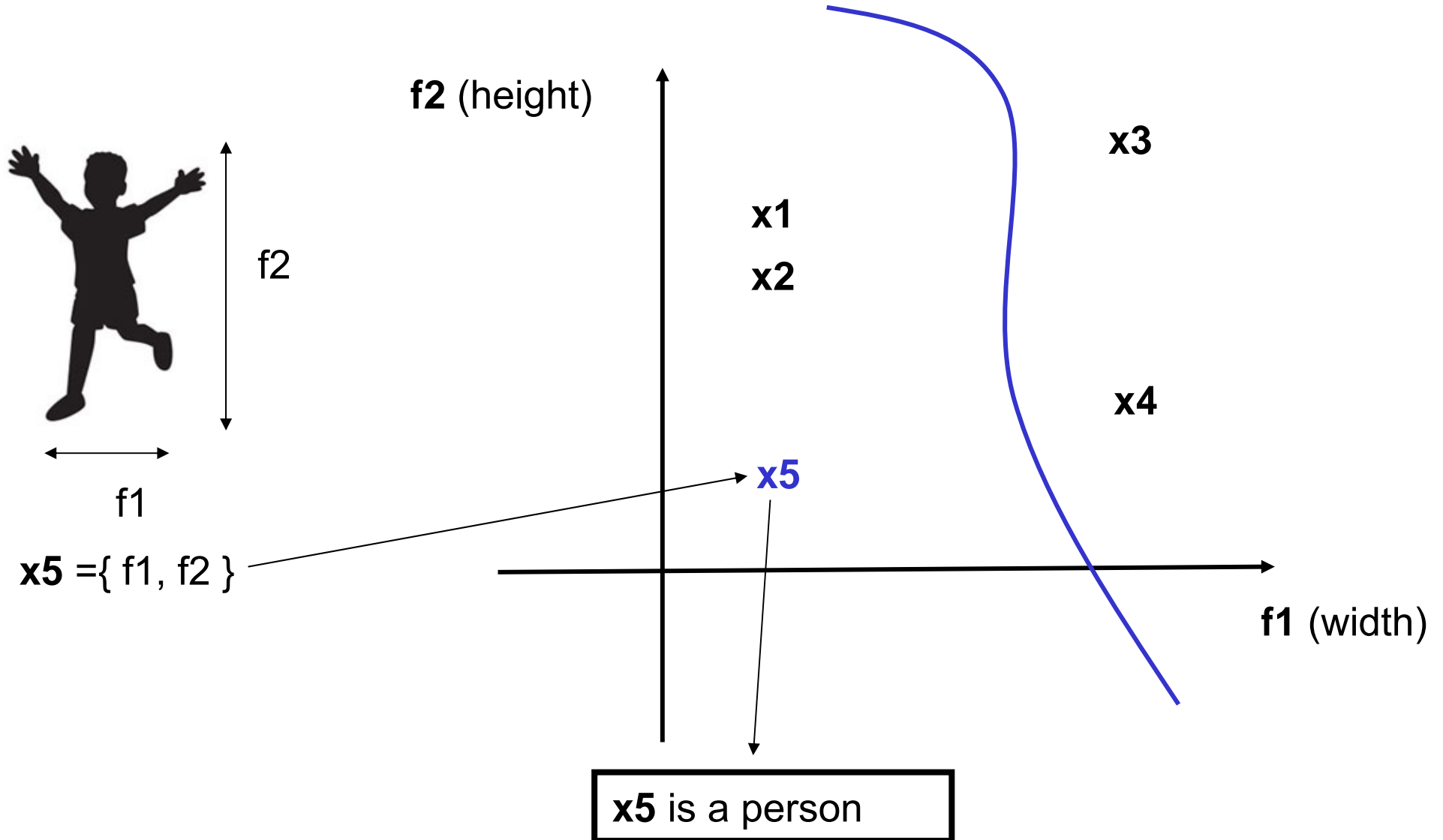
General idea of classification

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



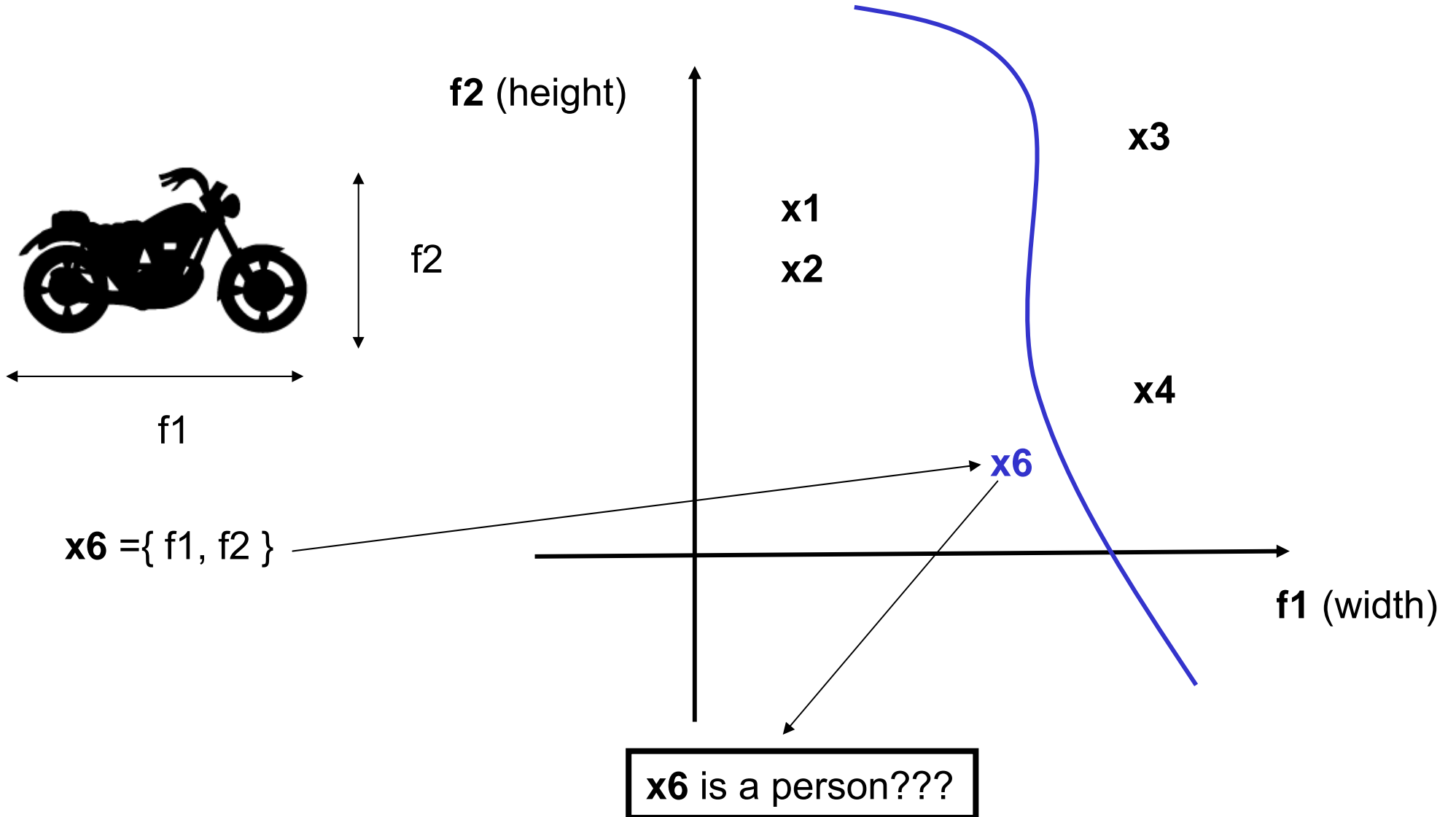
New classifications

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



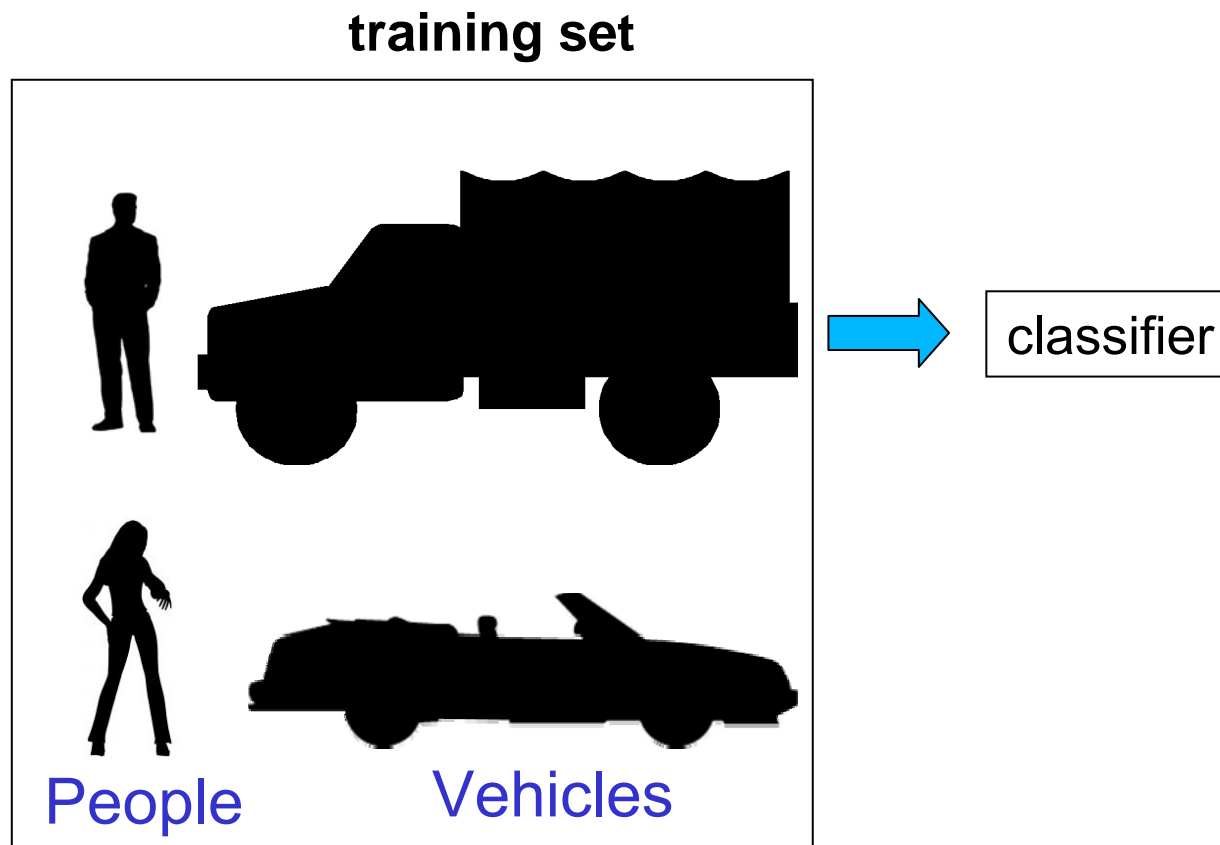
New classifications

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



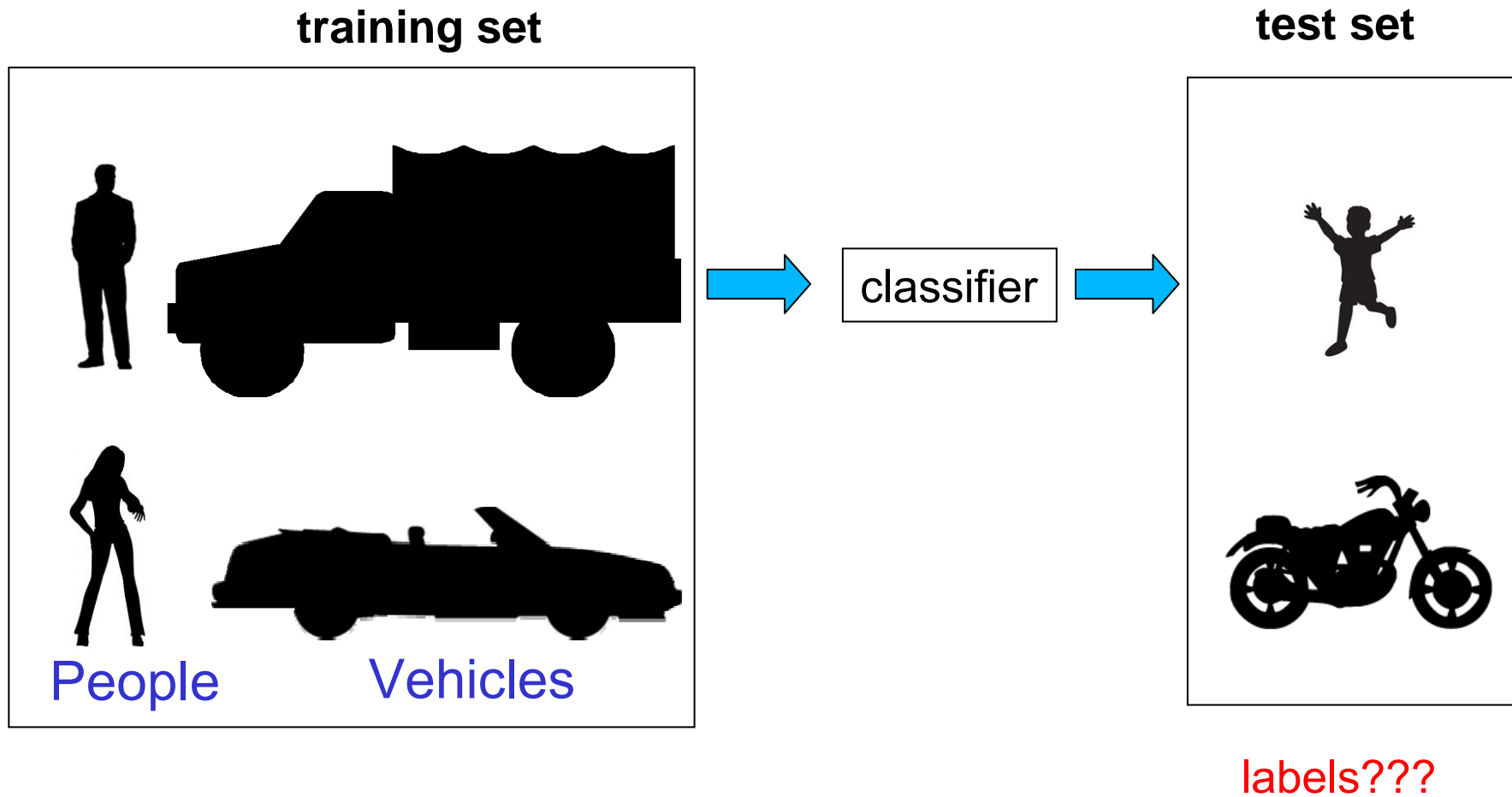
Training set

- Is the set of already labeled examples that is used to calculate the classifier.
- This process is called training.



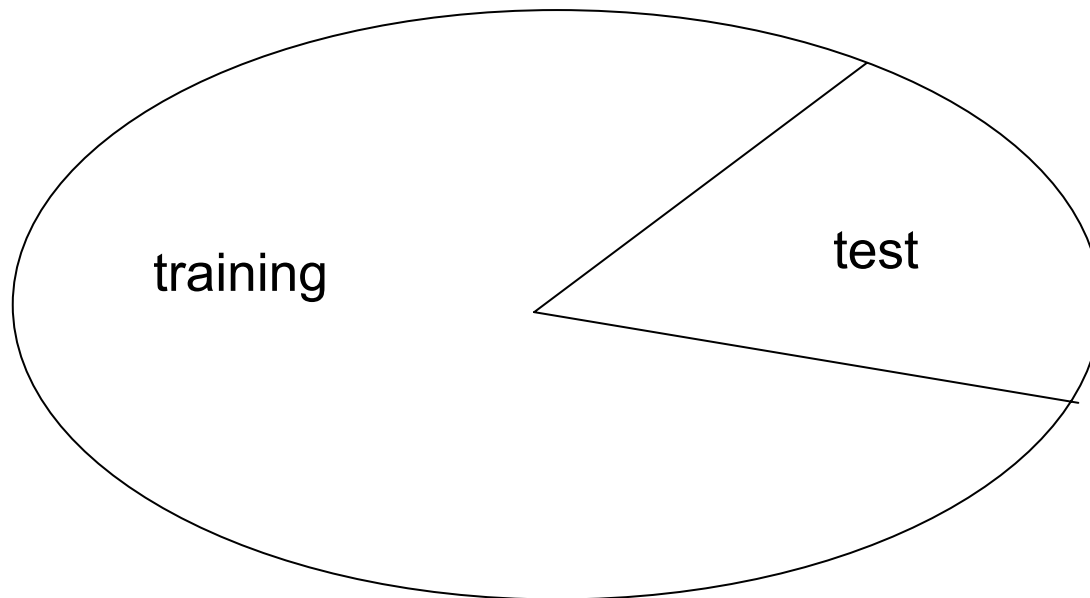
Test set

- 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 labeled to be able to get statistics.
- Usually a set of labeled examples is divided into training (~70%) and test (~30%) sets.



Measuring the quality of a classifier

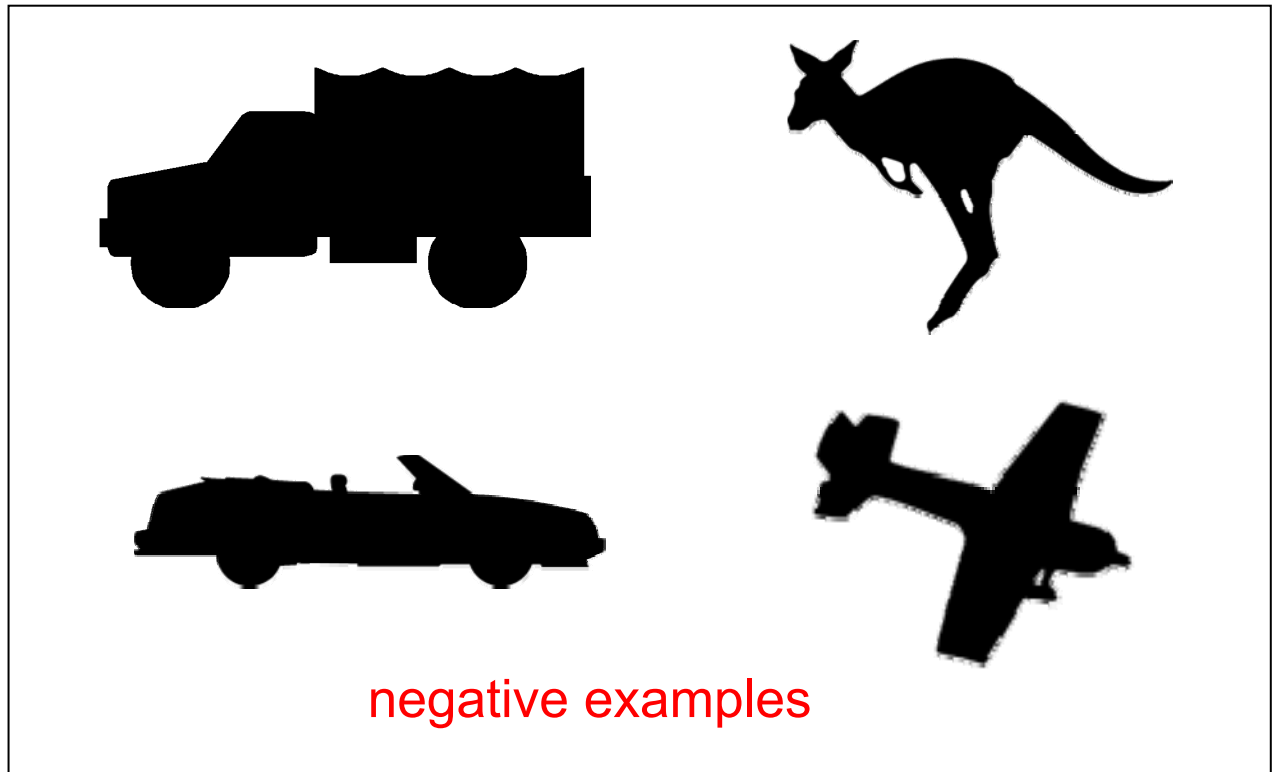
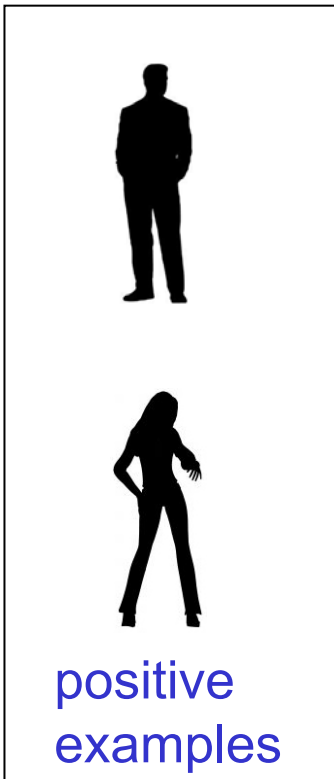
- The confusion matrix shows the confusions between classes when classifying the test:

	People	Vehicles
People	90%	10%
Vehicles	30%	70%

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

Detectors

- A detector is a classifier which try to distinguish only one class of objects from the rest of the world.
- In this case, the class to detect is called **positive** and the rest of classes in the world are called **negative**.
- Example: **people detector**.



Measuring the quality of a classifier

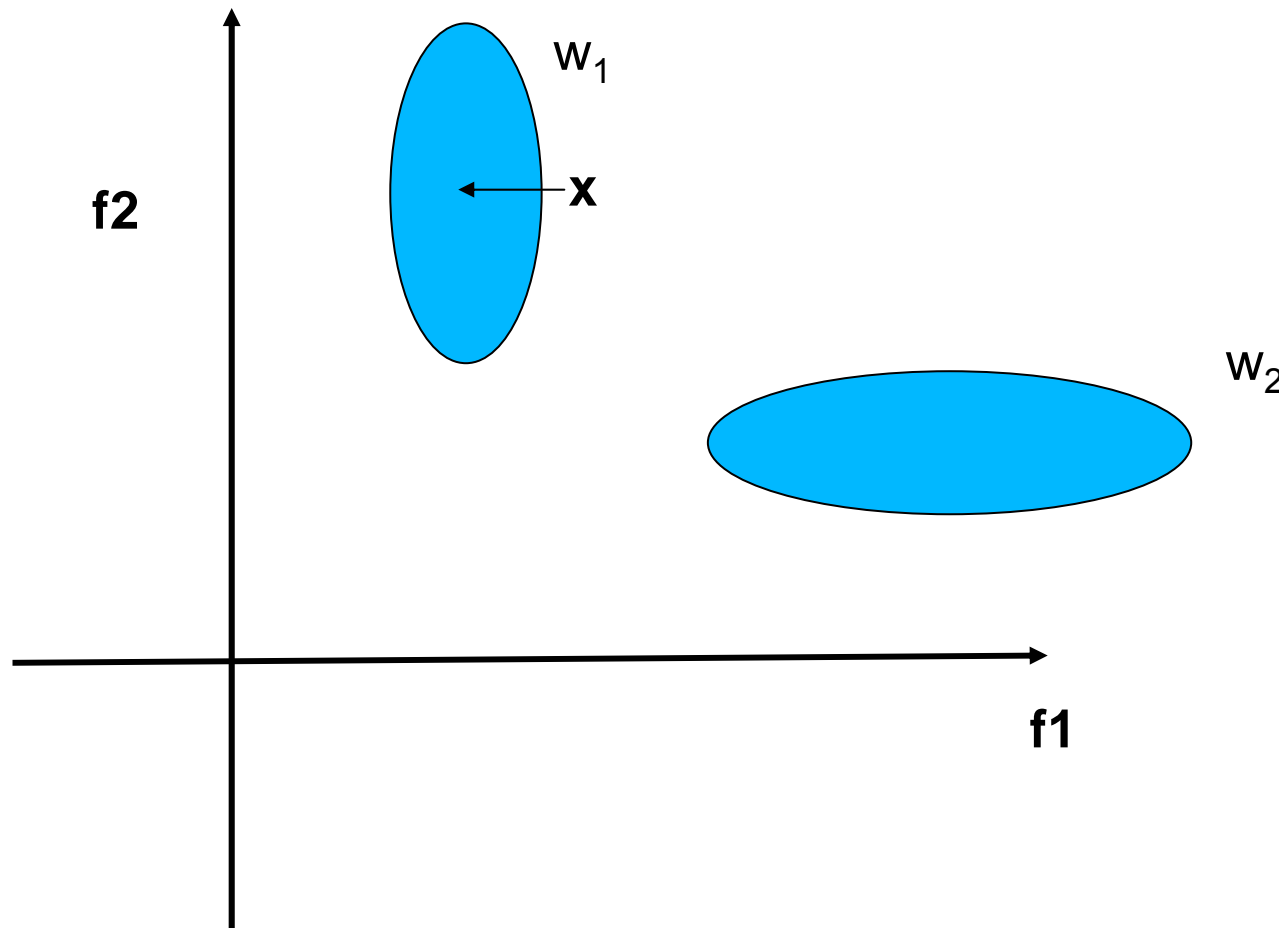
- In the case of a detector, the confusions have especial names: True positives(TP), True negatives (TN), False positives(FP), False Negatives(FN)

	True	False
True	TP	FN
False	FP	TN

- **Good detector: high trues (T^*) and low falses (F^*).**

Example: Binary classifier based on Bayes

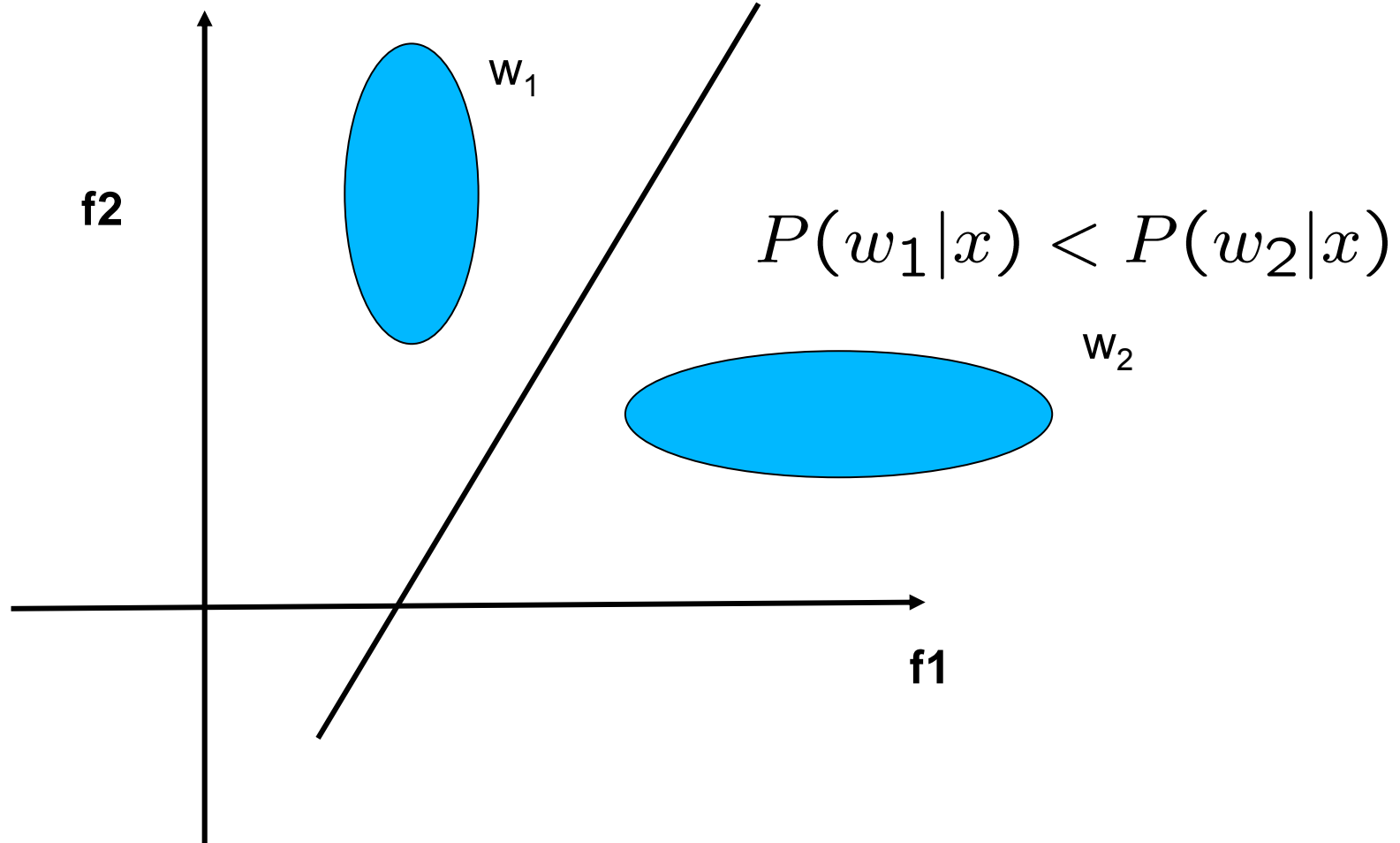
- Classes: w_1 , w_2 represented by Gaussians.
- Example x is assigned to class w_1 if $P(w_1|x) > P(w_2|x)$



Decision hyperplane

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

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



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

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

Assuming same covariance matrices $\Sigma_1 = \Sigma_2$
then x pertains to w_1 if:

$$-\frac{1}{2}(x - \mu_1)^T \Sigma_1 (x - \mu_1) > -\frac{1}{2}(x - \mu_2)^T \Sigma_1 (x - \mu_2)$$

which is a comparison of distances from point to gaussians.

Naive Bayes

$$-\frac{1}{2}(x - \mu_1)^T \Sigma_1 (x - \mu_1) > -\frac{1}{2}(x - \mu_2)^T \Sigma_1 (x - \mu_2)$$

Problem: Σ_i difficult to approximate when the dimension l of $x \in \mathbb{R}^l$ is very big. For l dimensions we need N^l to approximate the normal distribution.

Naive Bayes

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

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

Each feature is represented by a 1-dimensional Gaussian. Now we only need $l \cdot N$ examples.

Given a new point its classification assuming same priors is:

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

Estimating probabilities in Naive Bayes

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

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

If we have N training examples x_1, \dots, x_N then:

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

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

Classifying with Naive Bayes

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

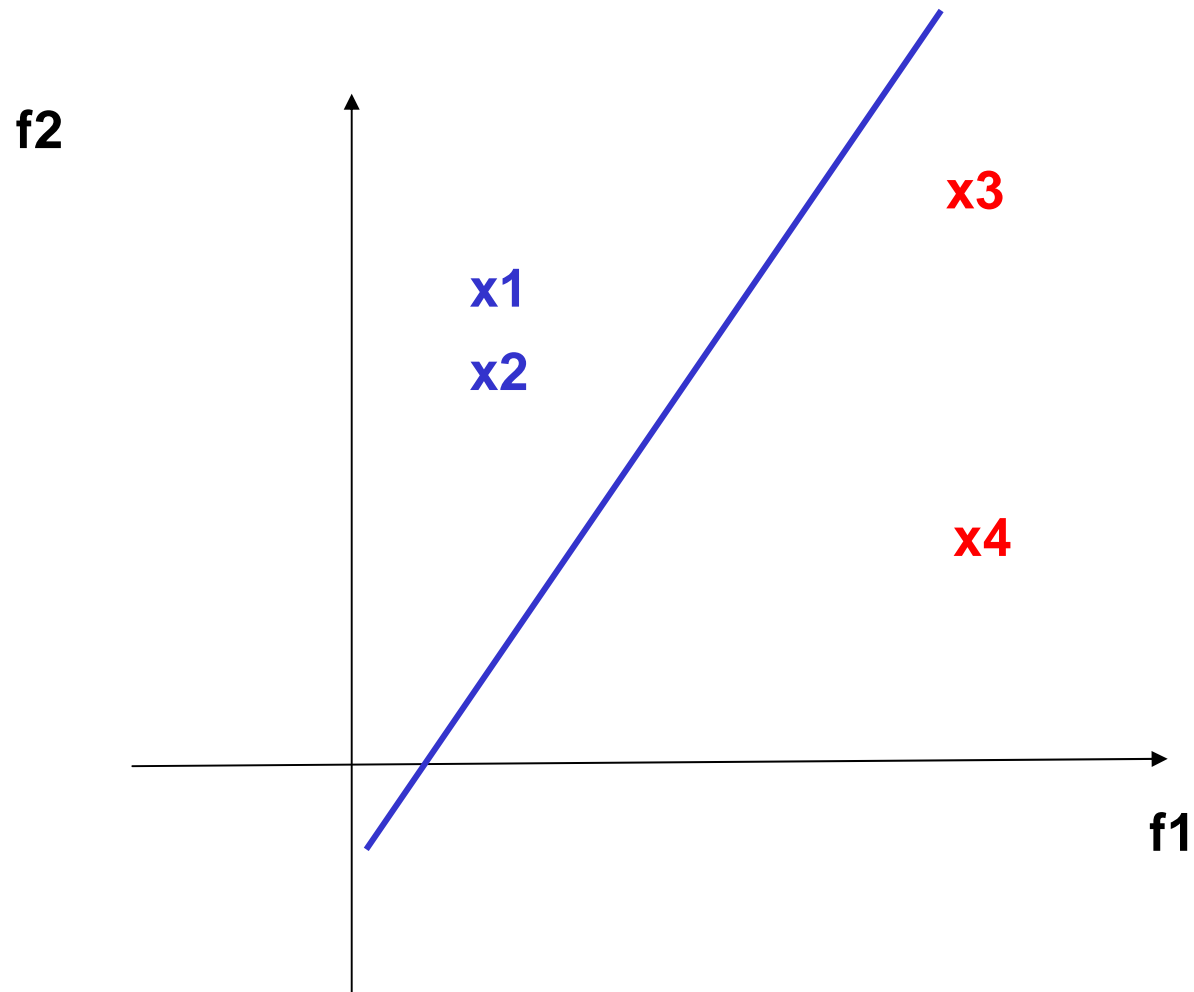
$$w = \operatorname{argmax}_{w_i} \prod_j^l P(f_j|w_i)$$

with

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

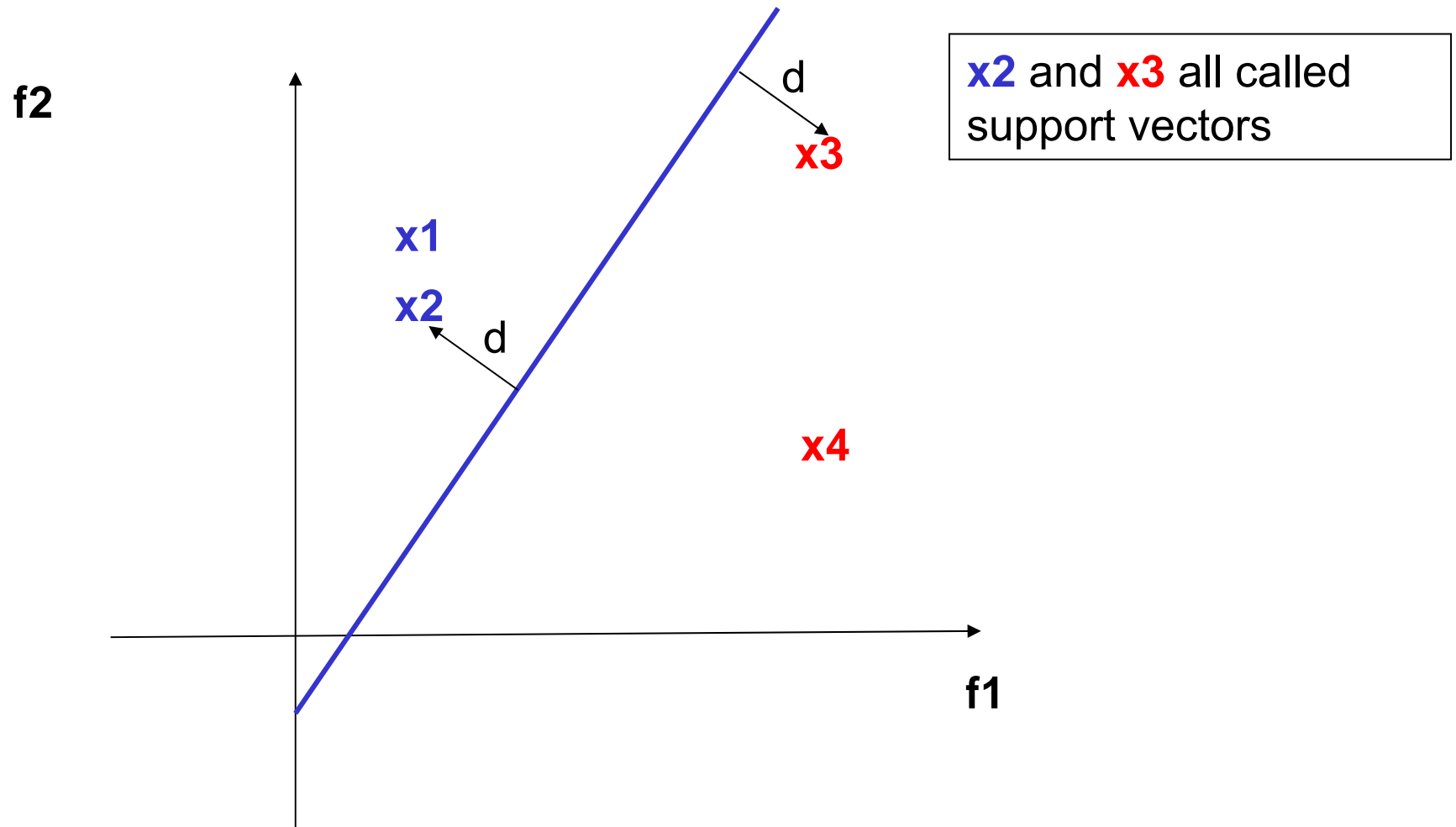
Linear classifiers

- 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



Nonlinear case

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

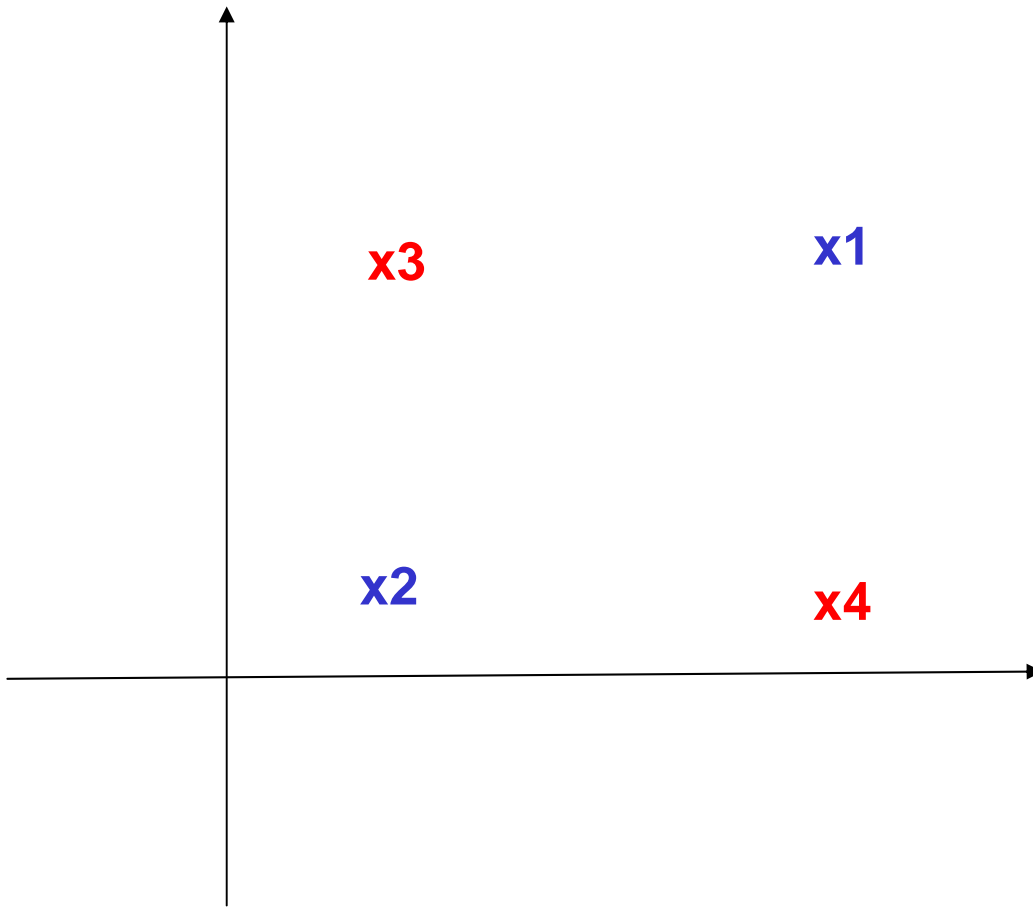
$$f(\|x - c_i\|)$$

- Which is actually a distance with respect to a center.
- Typical used function in SVM has a Gaussian form:

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

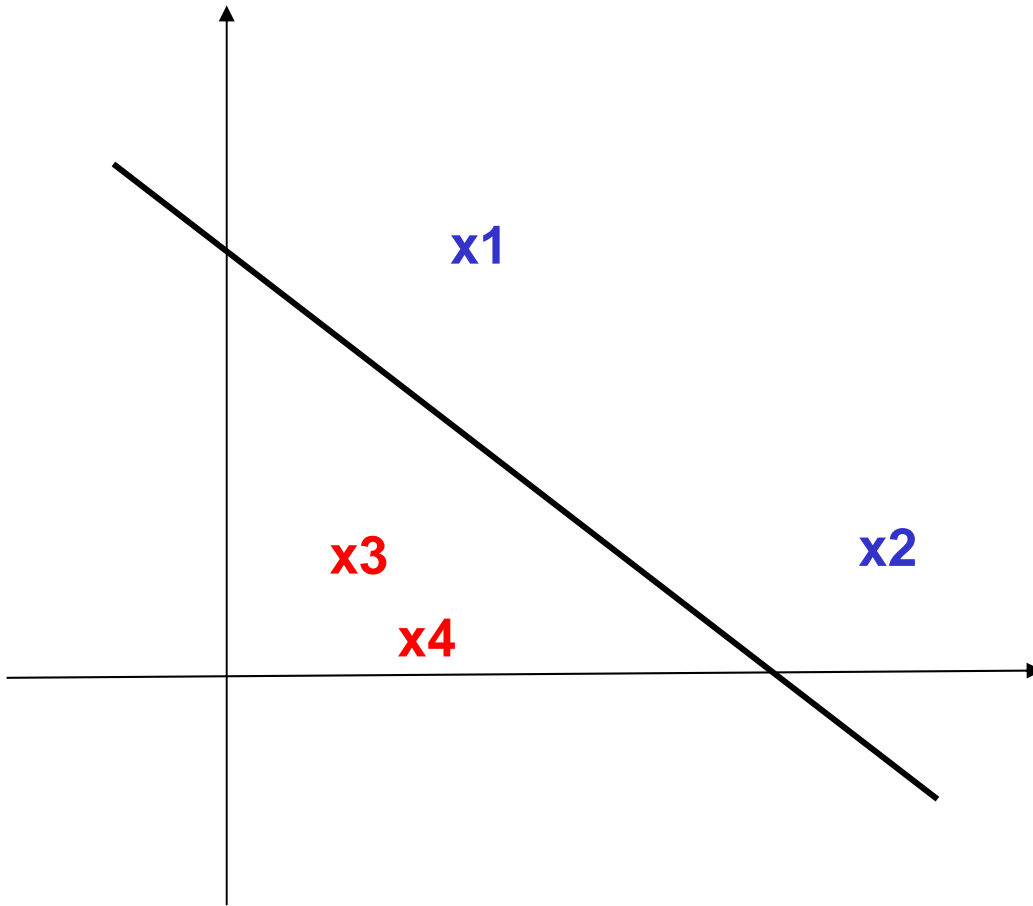
Nonlinear case

- Original feature space



Nonlinear case

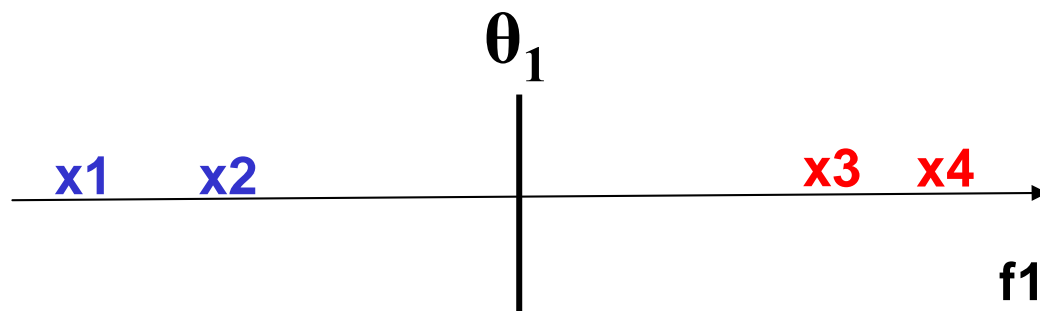
- Transformed feature space



Boosting

- Meta classifier that improves the result of simple classifiers.
- For each feature f_j we create a linear classifier:

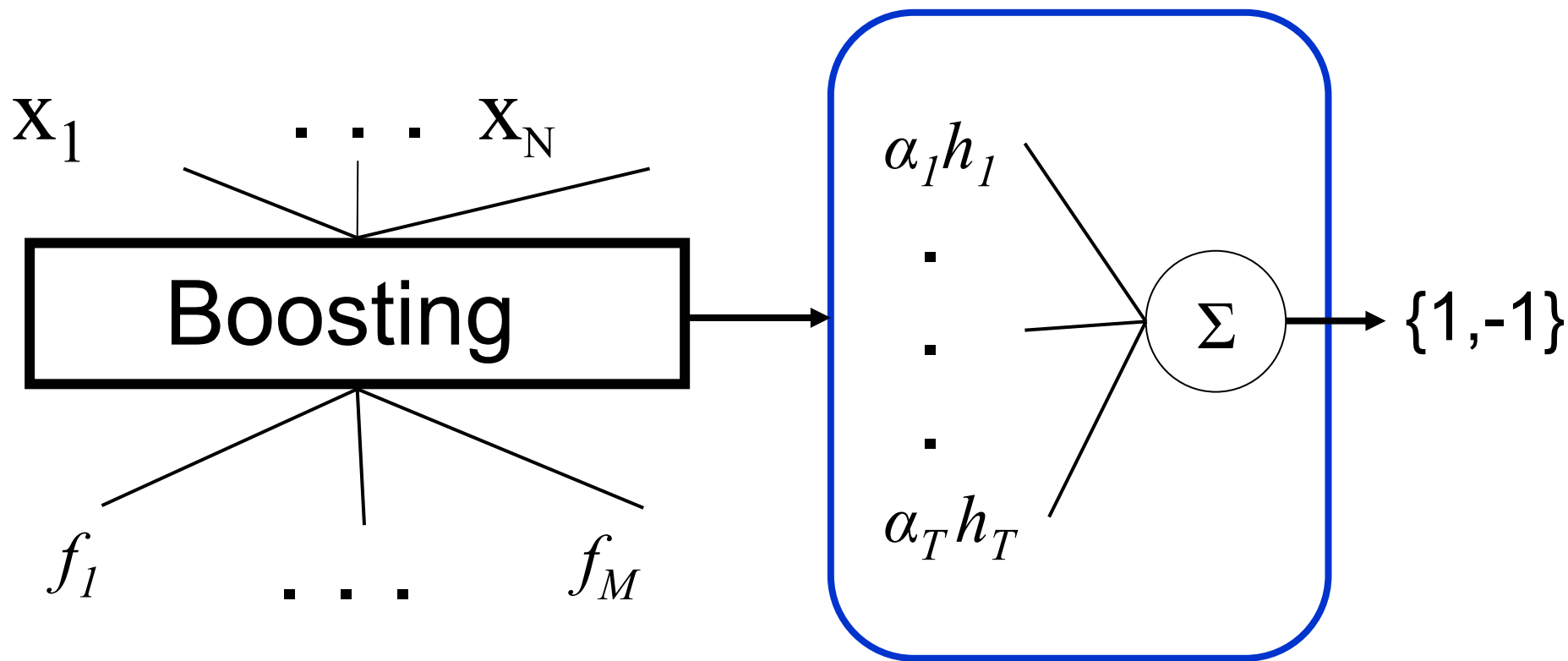
$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ -1 & \text{otherwise.} \end{cases}$$



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

Boosting

- Simple classifiers are combined to form a strong classifier



Binary Strong
Classifier H

- Input: set of examples $(x_1, y_1), \dots, (x_N, y_N)$.
- 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 $t = 1, \dots, T$:

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:

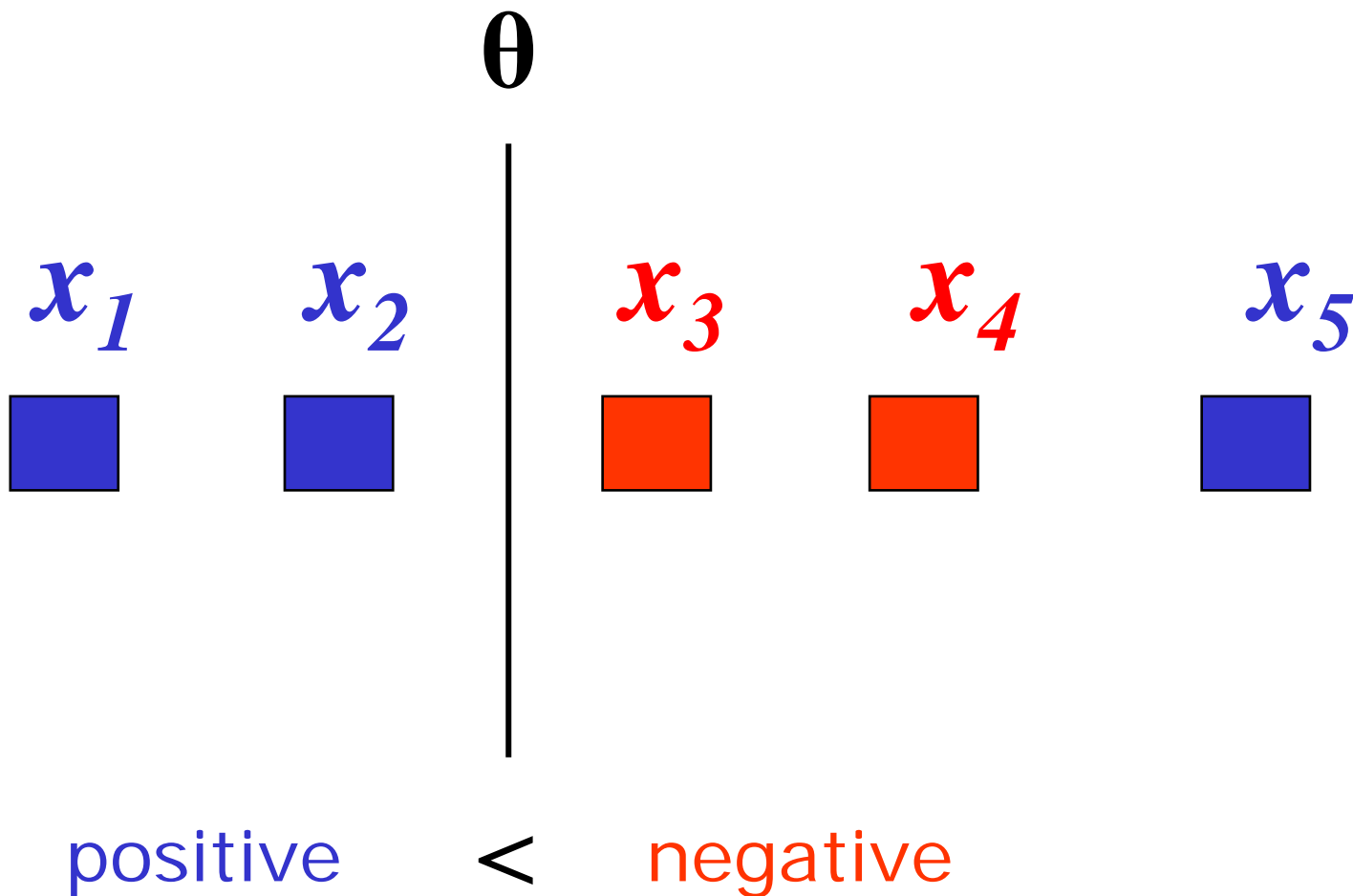
$$\epsilon_j = \sum_n^N w_{t,n} |h_j(x_n) - y_n|.$$

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:

$$h(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^T \log \frac{1}{\beta_t} h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \log \frac{1}{\beta_t} \\ 0 & \text{otherwise.} \end{cases}$$

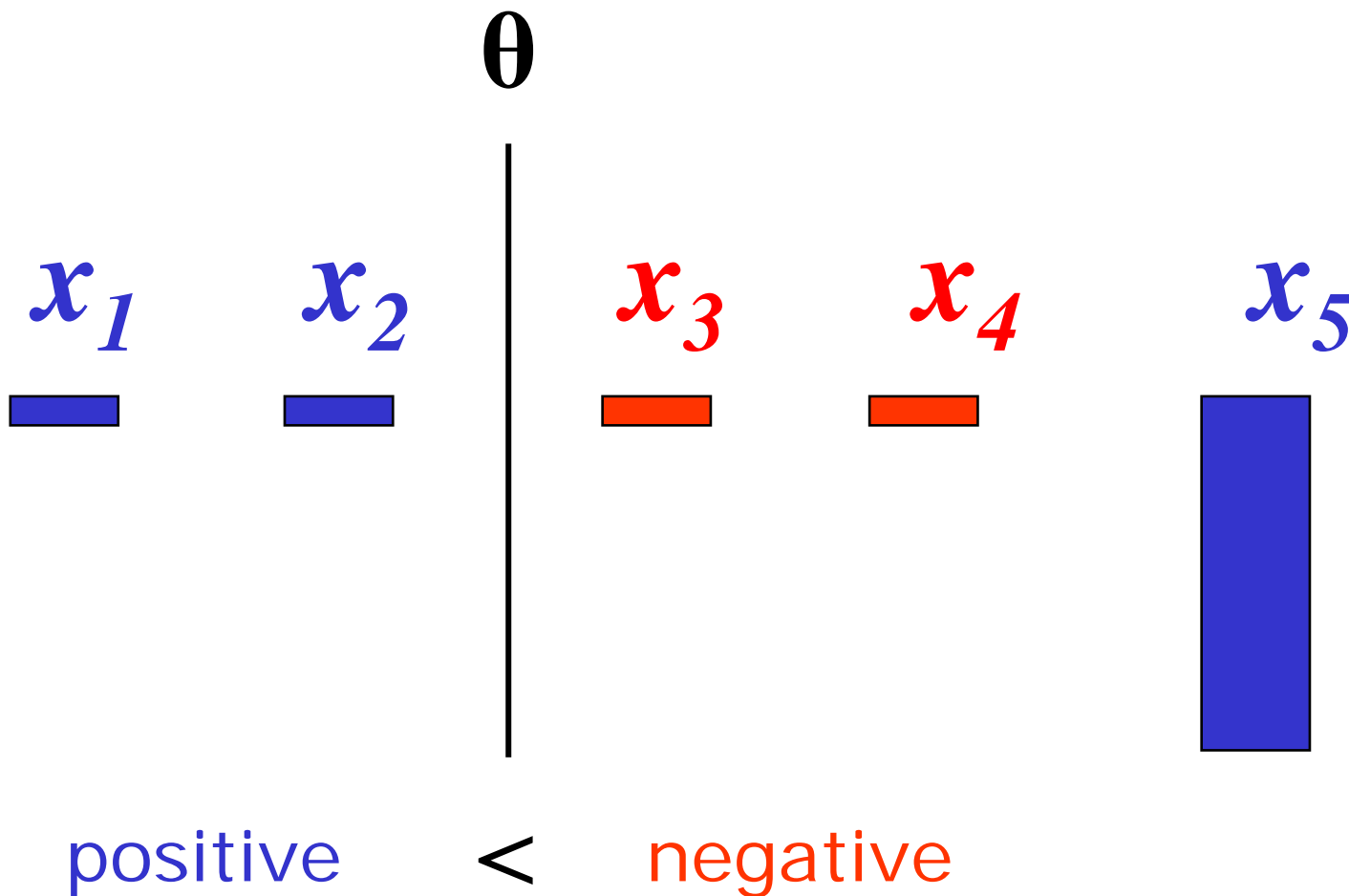
Main idea

positive
negative



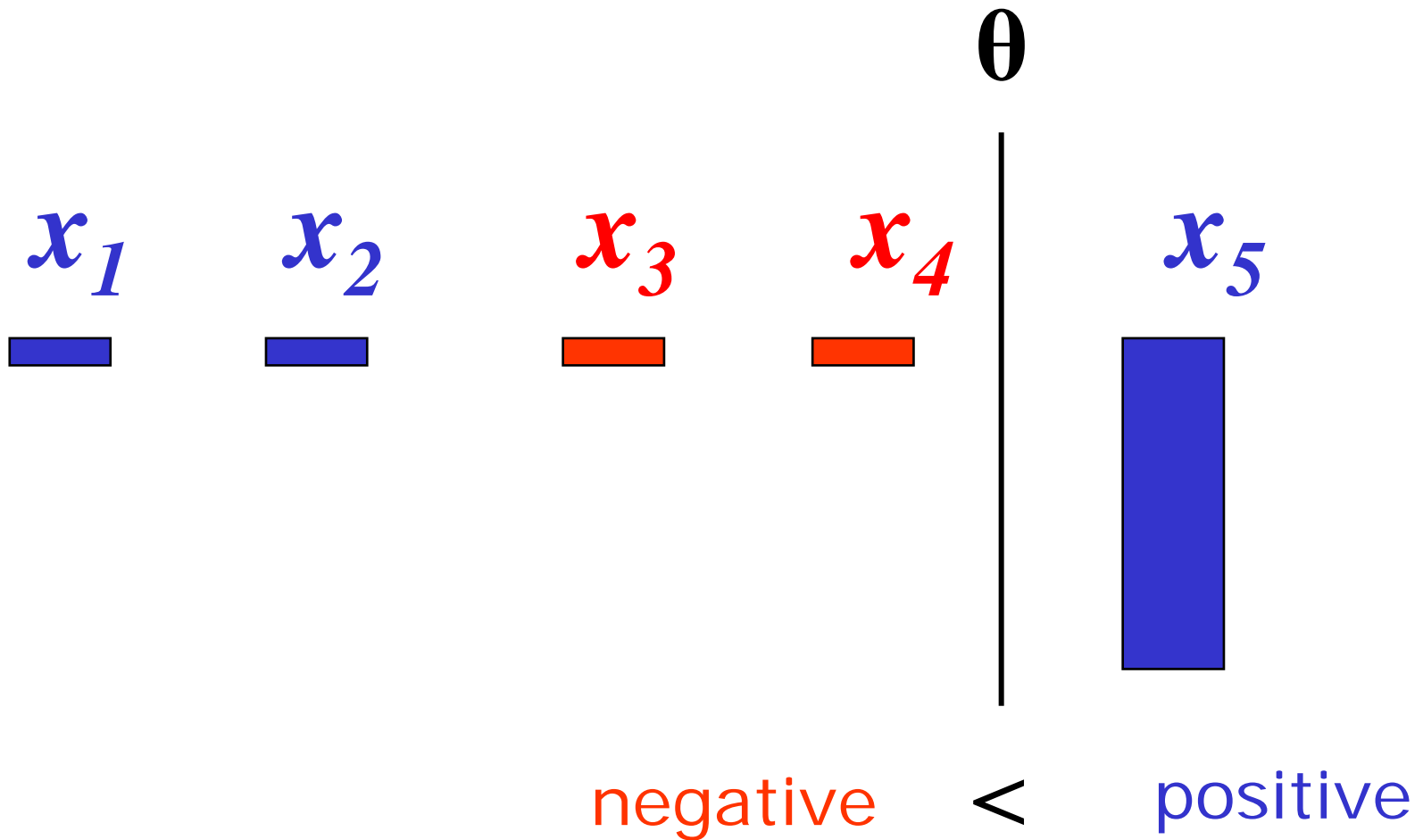
Main idea

positive
negative



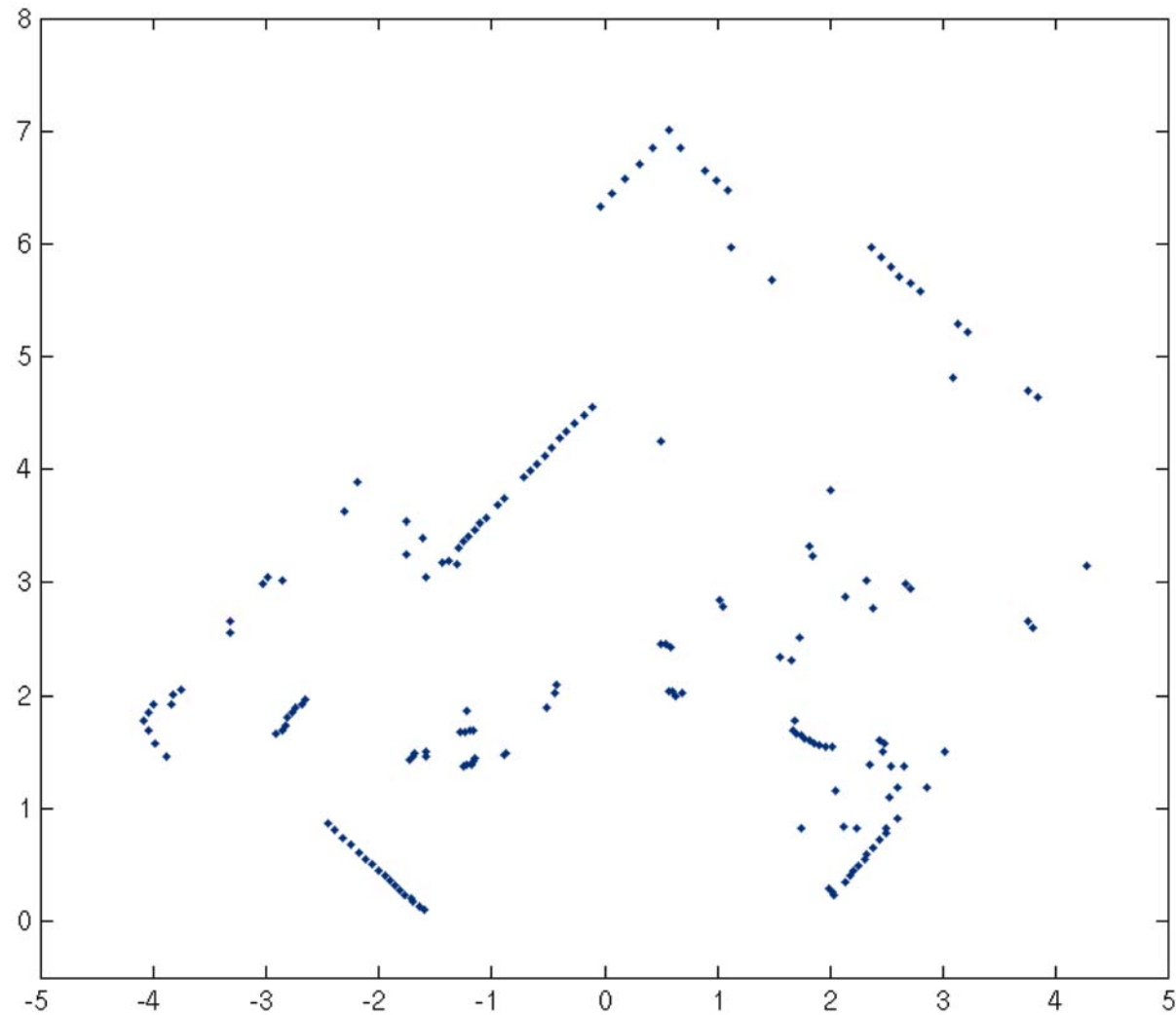
Main idea

positive
negative



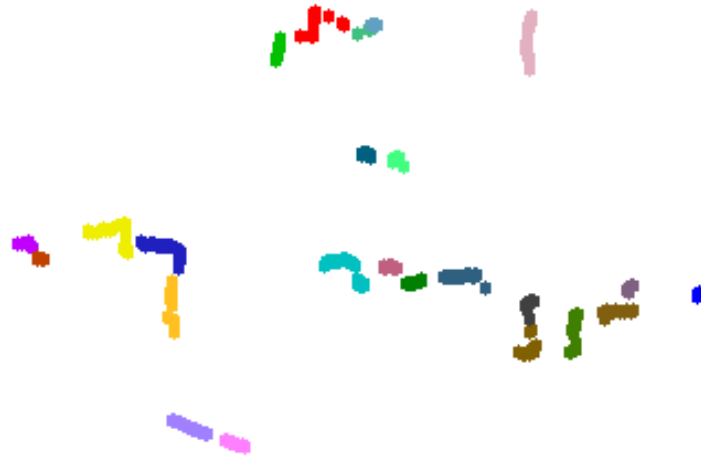
Video AdaBoost

Example application: 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.

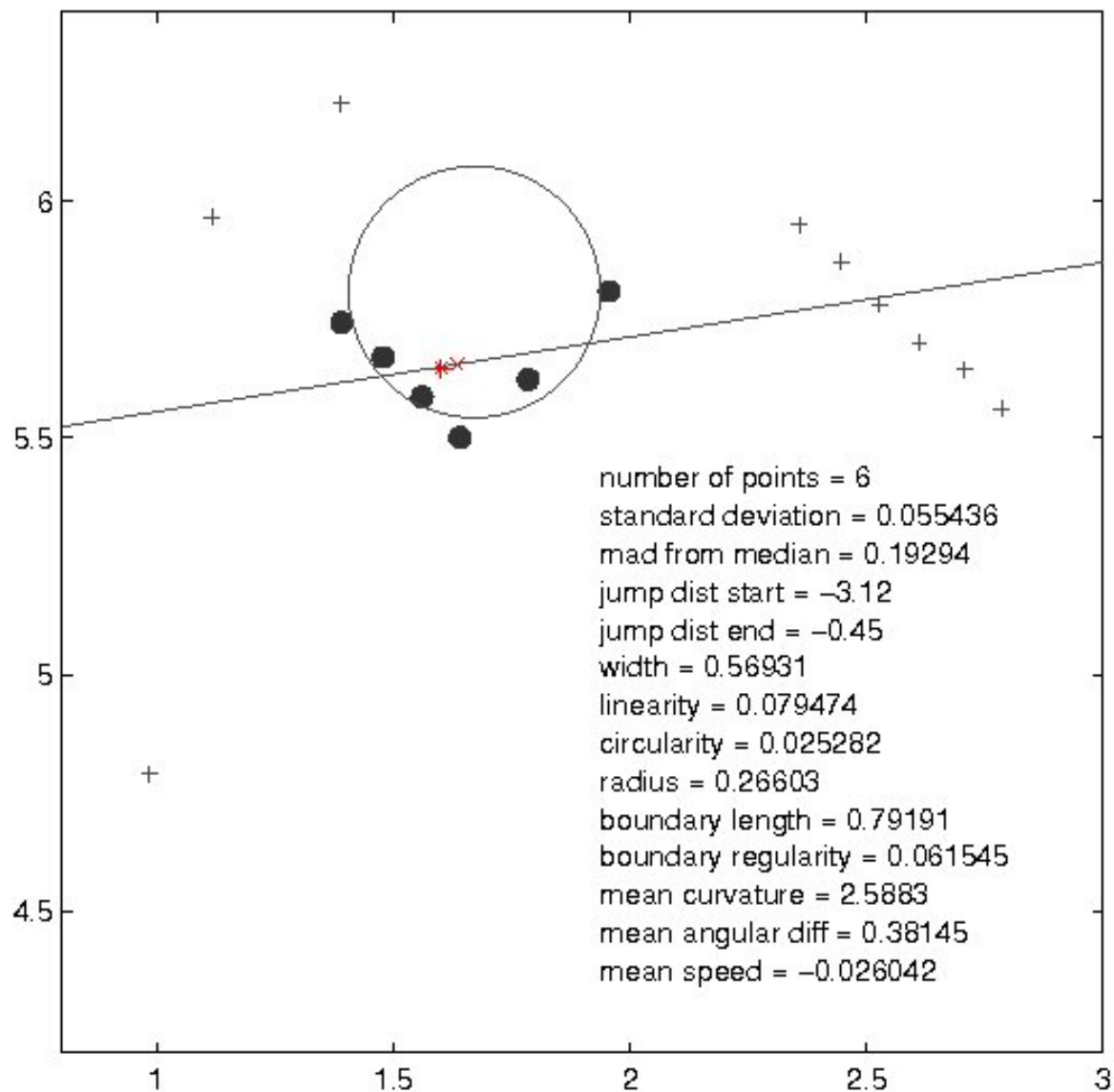
Detection in One Level

- Divide the scan into segments.

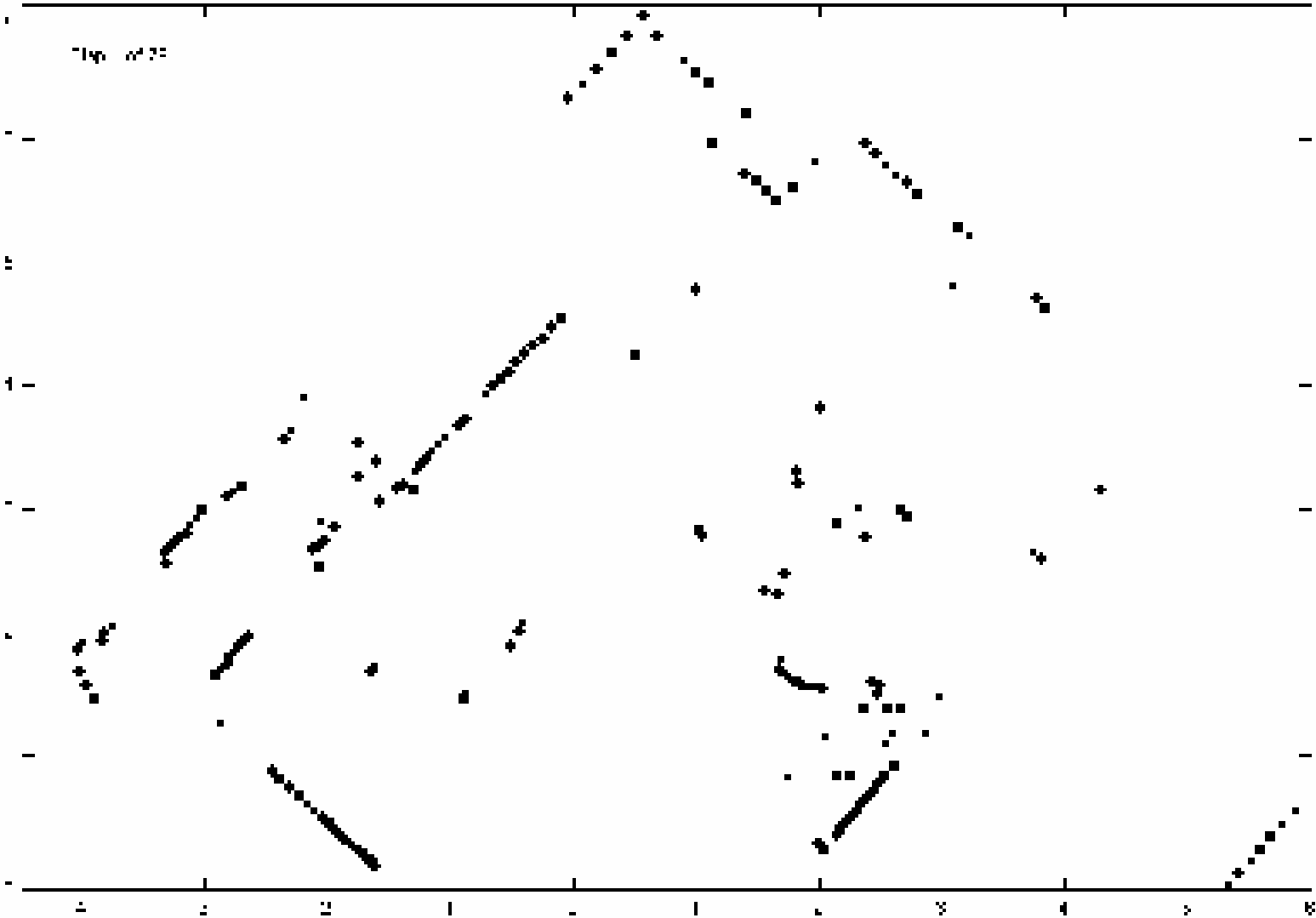


- Learn the segments corresponding to people.
- Classification problem:
 1. Select the **features** representing a segment.
 2. Learn a **classifier** using Boosting

Features from Segments



Results

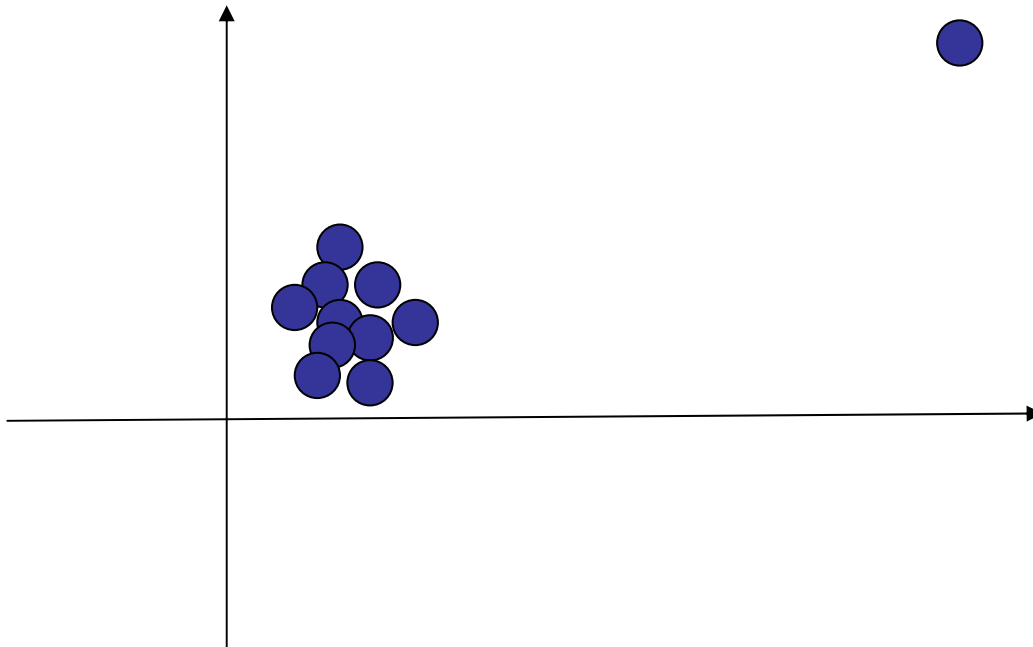


Final considerations when learning a classifier

- Outliers
- Overfitting
- Normalization of data
- Feature selection

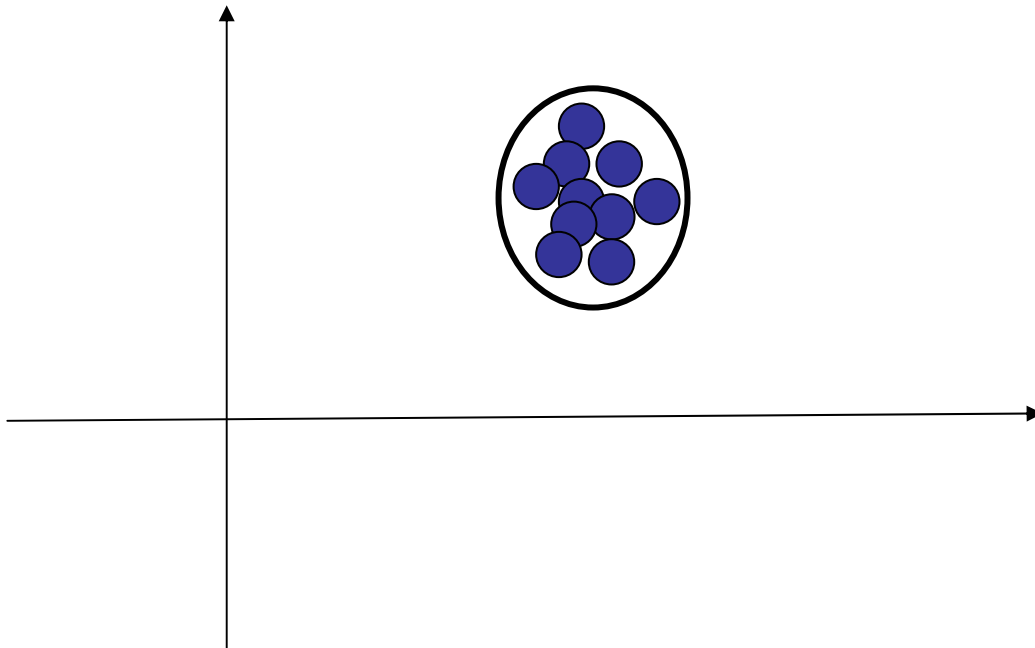
Outliers

- 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



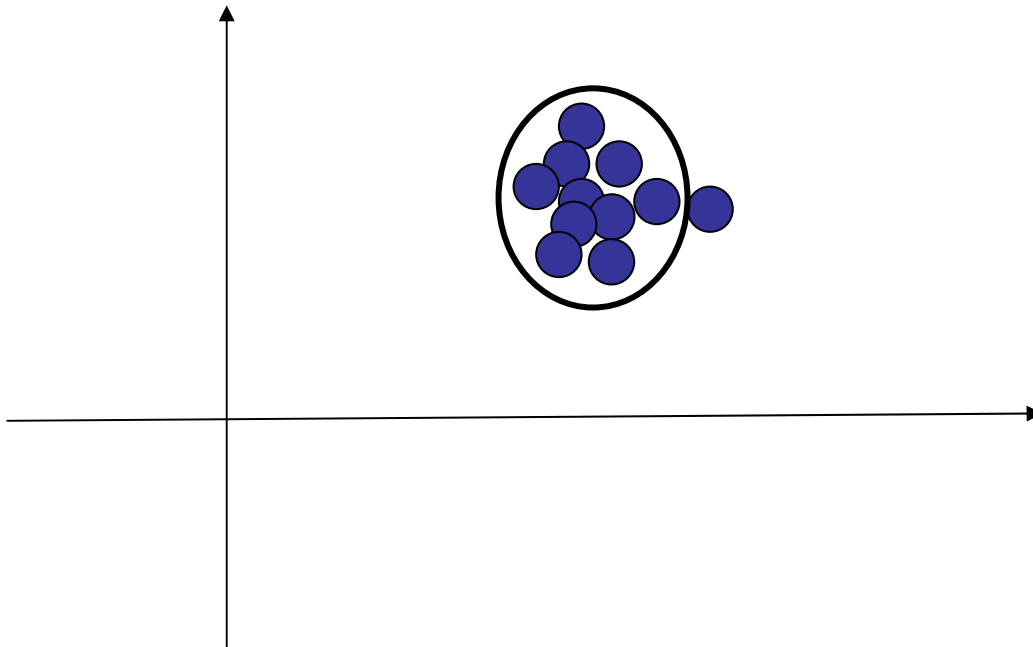
Overfitting

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



Overfitting

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

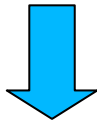


Normalization of data

- The ranges of the features could be very different.
- It is interesting to normalize them:

f1 [1 2]

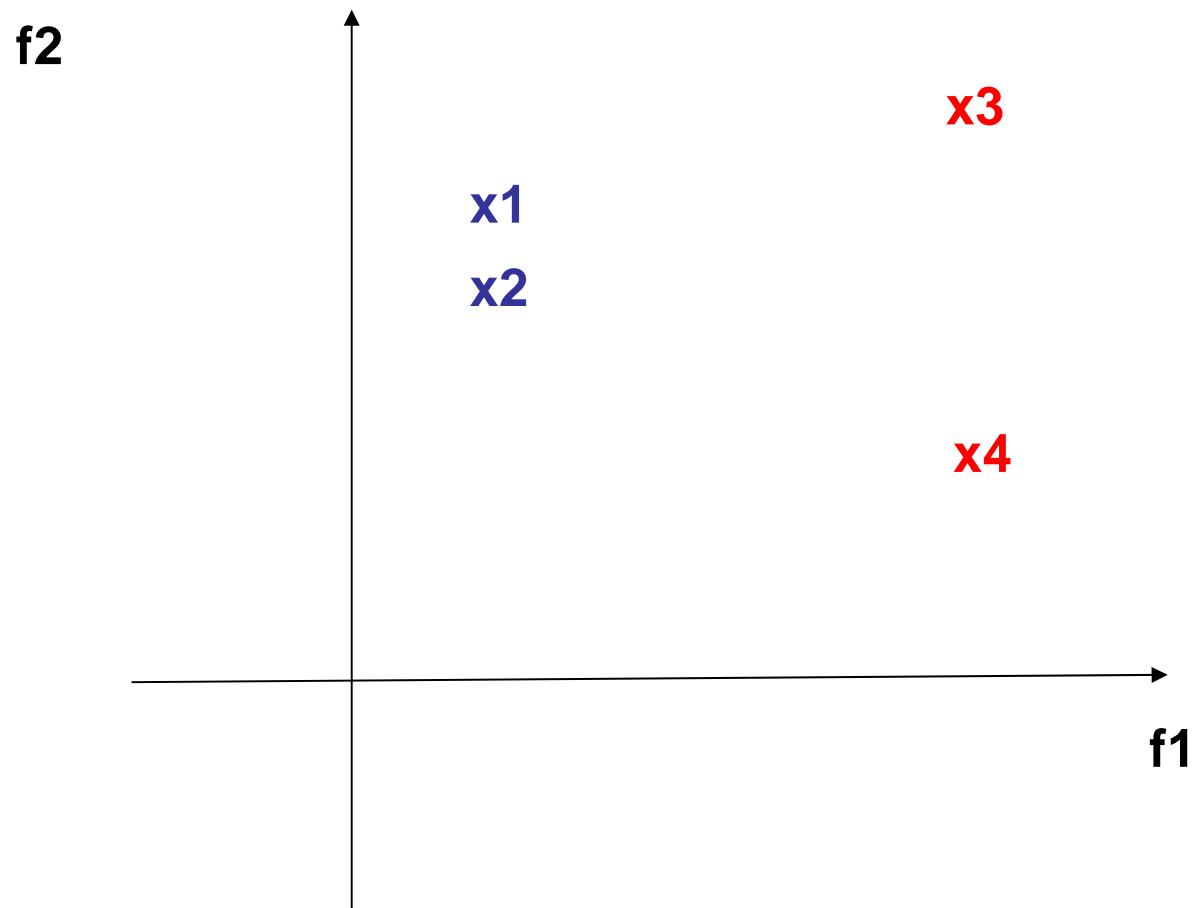
f2 [-100 100]



f1, f2 [0, 1]

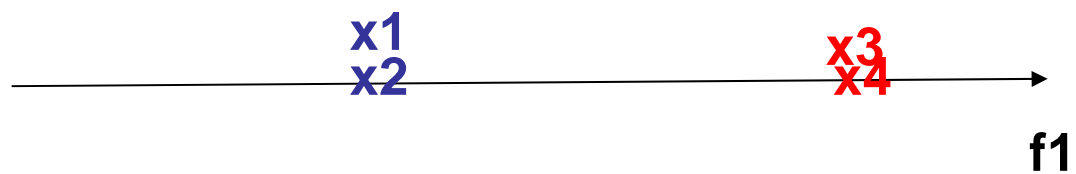
Feature selection

- Some features discriminate better than others.



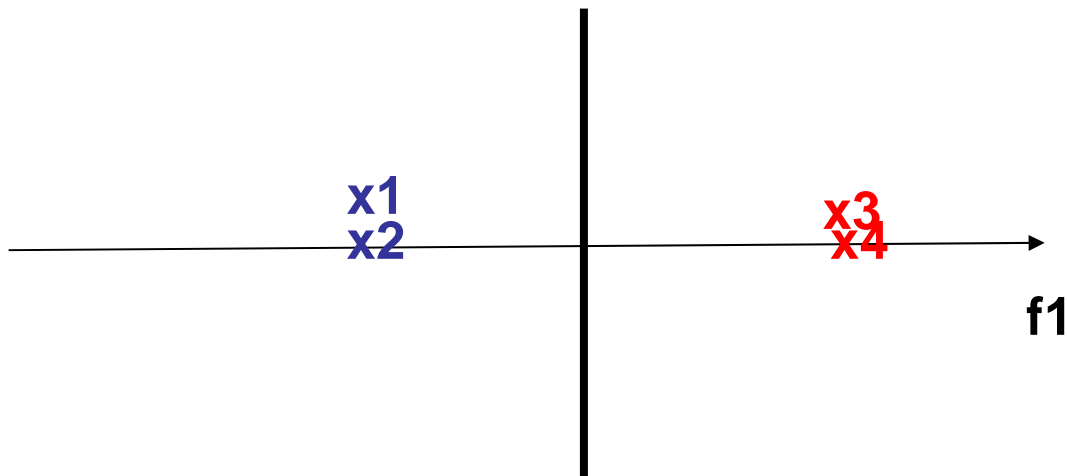
Feature selection

- Projection on **f1**



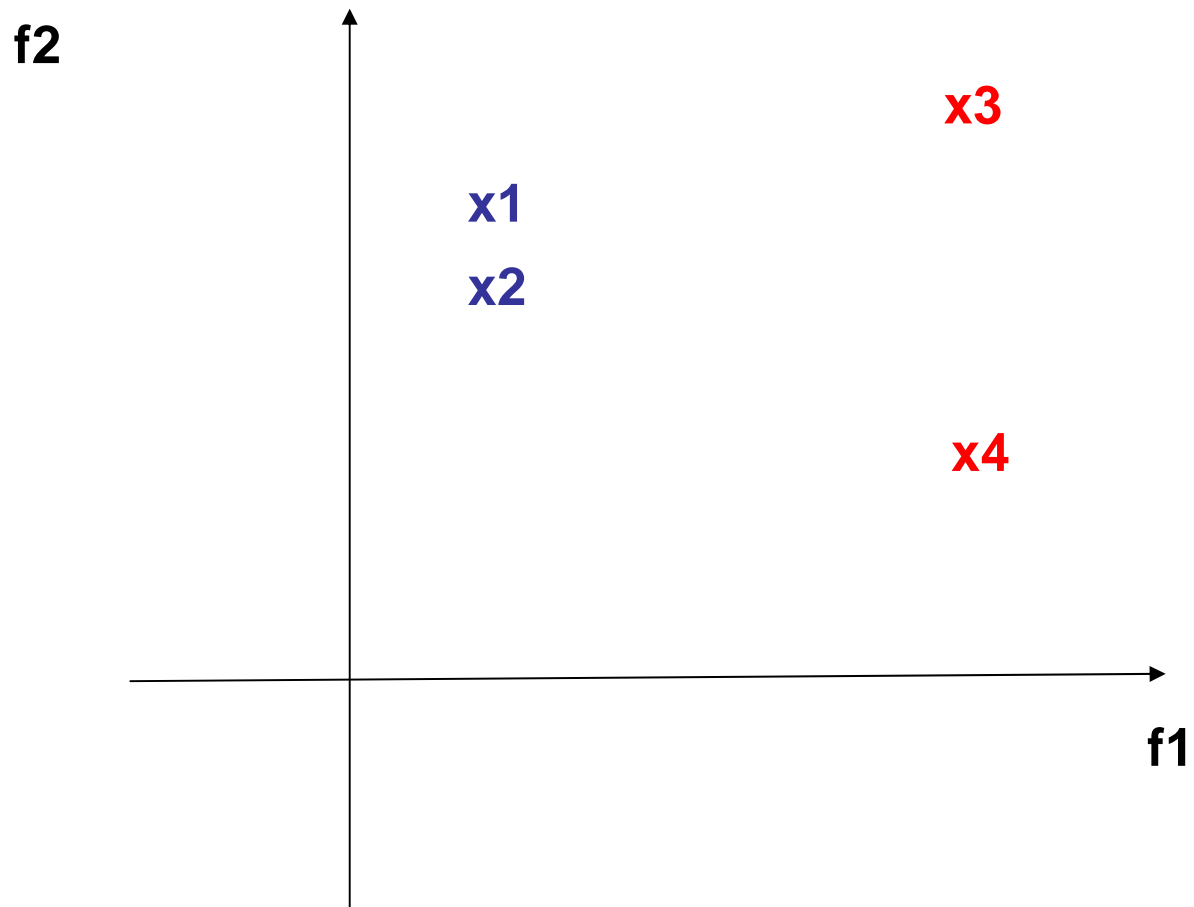
Feature selection

- Projection on **f1**, **easy classification**



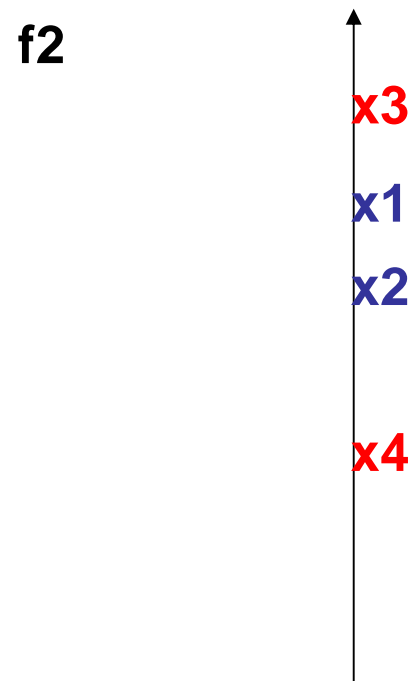
Feature selection

- Projection on **f2**



Feature selection

- Projection on **f2**, **difficult classification**



End