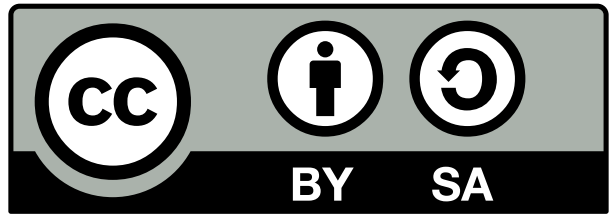
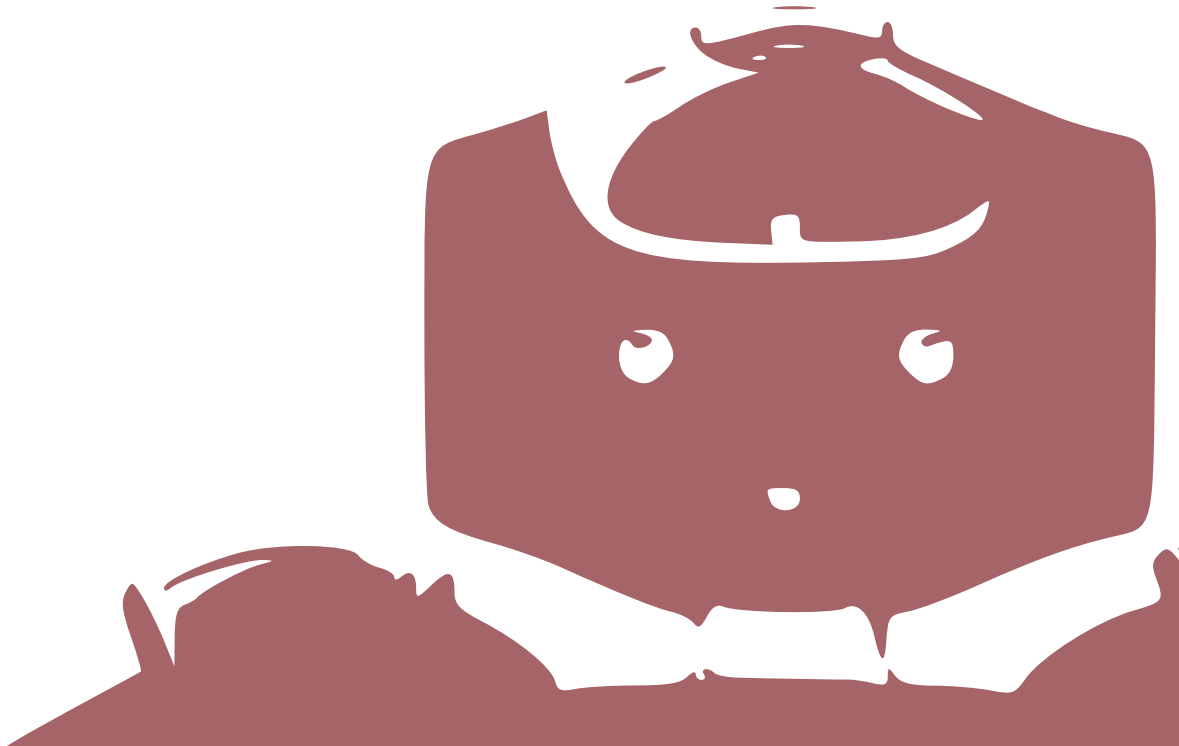


You can download the sources of this presentation here:

github.com/severin-lemaignan/lecture-hri-social-signal-processing

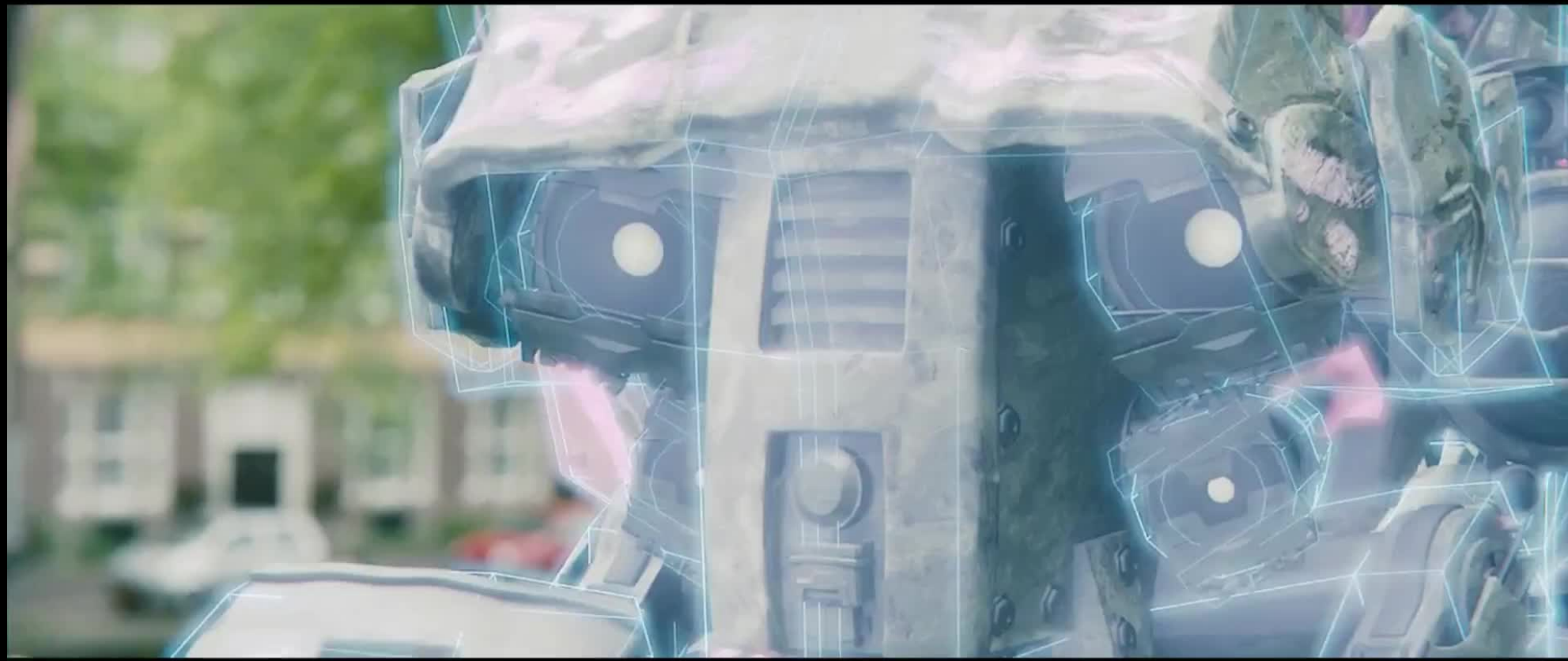
Social Signal Processing



Séverin Lemaignan

Bristol Robotics Lab
University of the West of England

WHAT ARE SOCIAL SIGNALS?




Gestures
Emotions
Face identification
Gaze
Prosody
Verbal communication
Proxemics
Body posture

IN THIS LECTURE

- What/why social sigal processing?
- Classification (k-nearest neighbours and Support Vector Machines)
- Case study: emotions (aka: let's build an emotion classifier from scratch)

WHAT ARE SOCIAL SIGNALS?

- Social signals are ***observable*** behaviours that people display during social interactions

[Burgoon, Magnenat-Thalmann, Pantic, Vinciarelli, **Social Signal Processing**, 2017]

WHAT ARE SOCIAL SIGNALS?

- Social signals are ***observable*** behaviours that people display during social interactions
- Social signals from an individual ***produces changes*** in others (like creating a belief about the person, generating an appropriate social response, perform an actions)

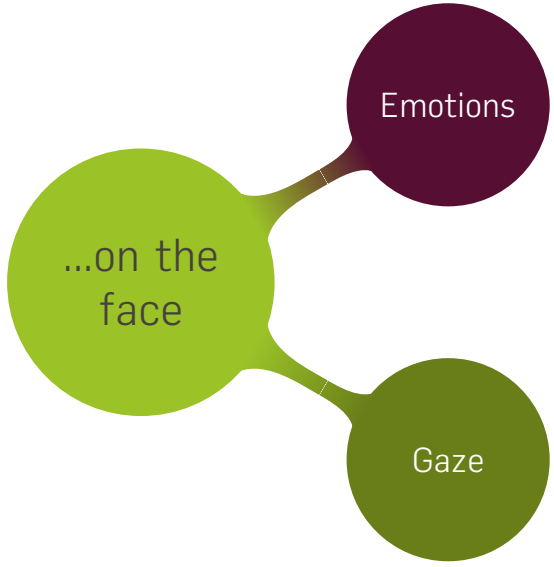
[Burgoon, Magnenat-Thalmann, Pantic, Vinciarelli, **Social Signal Processing**, 2017]

WHAT ARE SOCIAL SIGNALS?

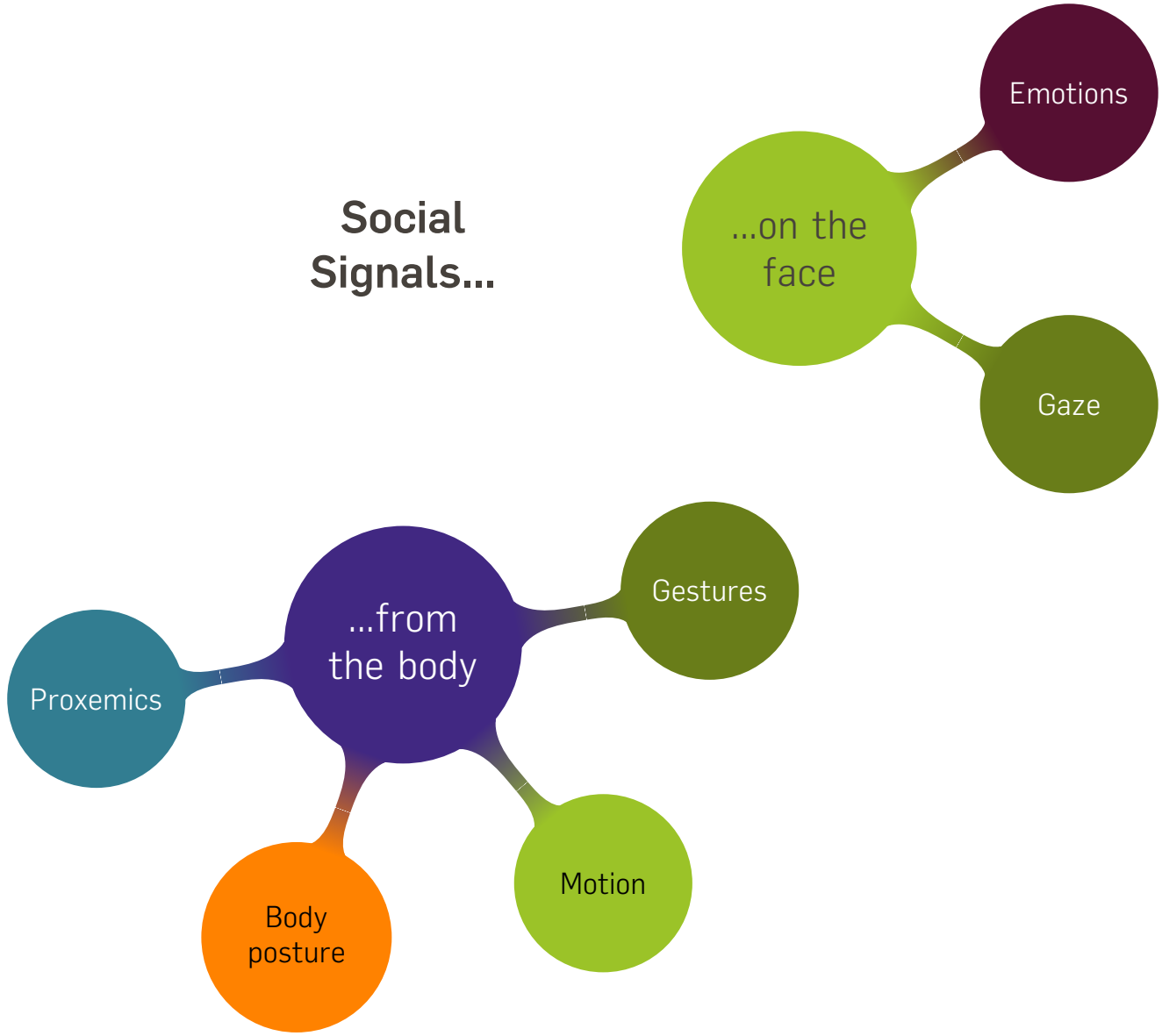
- Social signals are ***observable*** behaviours that people display during social interactions
- Social signals from an individual ***produces changes*** in others (like creating a belief about the person, generating an appropriate social response, perform an actions)
- the changes are not random, they follow ***principles and laws*** (in particular, ***social norms***)

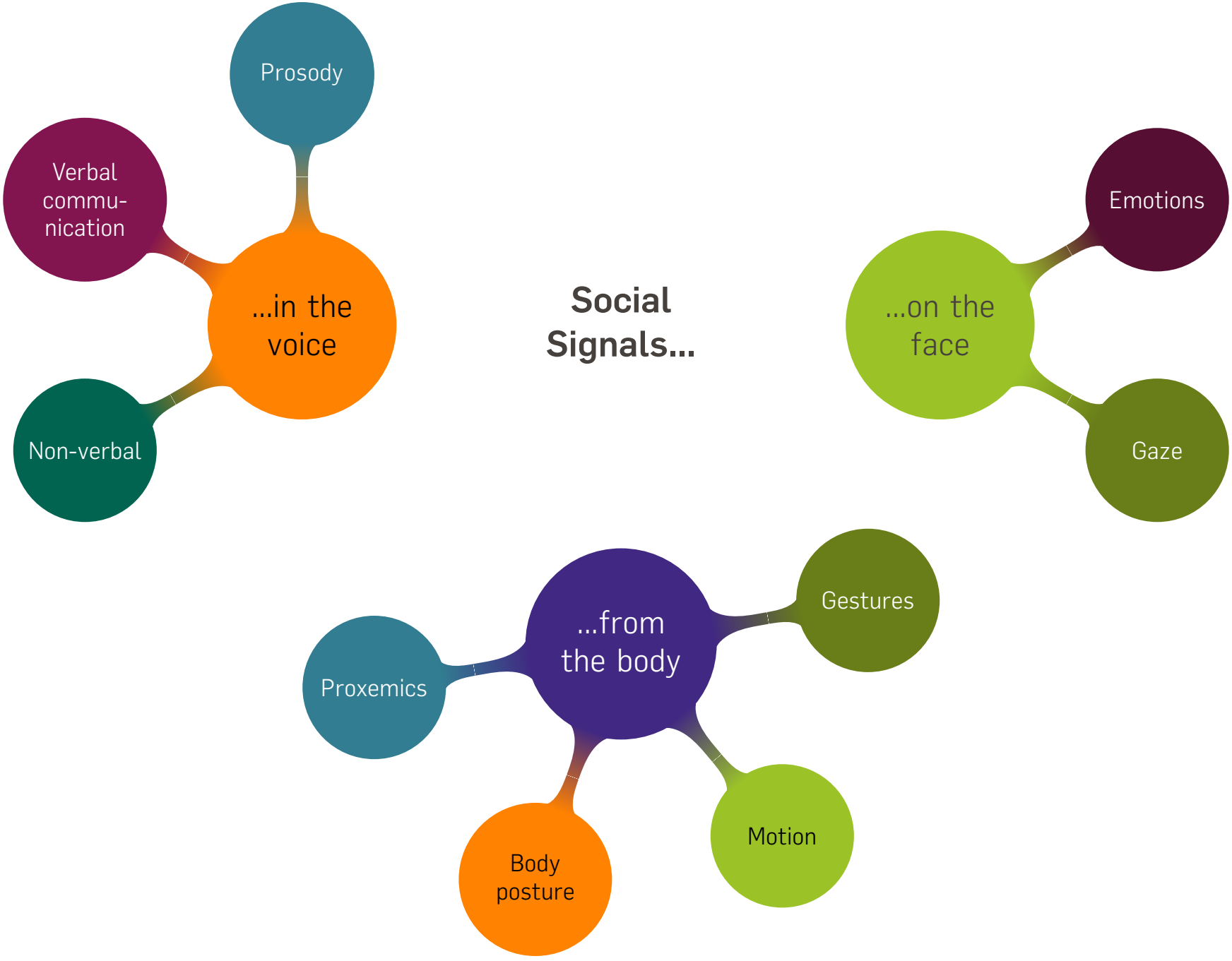
[Burgoon, Magnenat-Thalmann, Pantic, Vinciarelli, **Social Signal Processing**, 2017]

Social Signals...



Social Signals...




Prosody
Verbal commu-nication
...in the voice
Non-verbal
Social Signals...
Emotions
...on the face
Gaze
Gestures
...from the body
Proxemics
Body posture
Motion

WHY?

The ability to recognize human social signals and social behaviours like turn taking, politeness, and disagreement is essential when building social robots, human-robot interaction, or interactive systems

WHY?

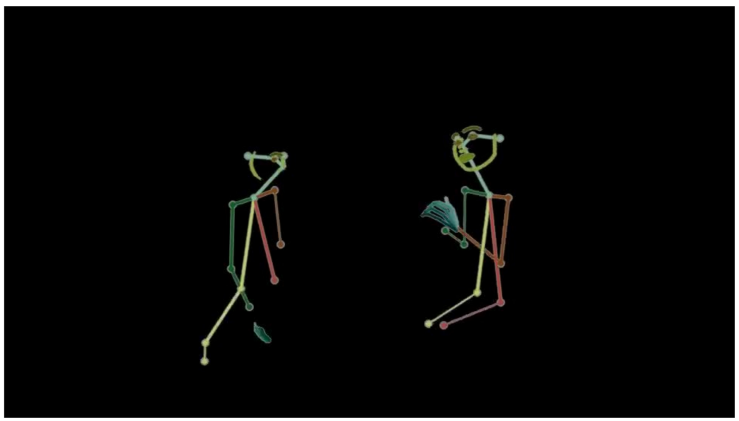
The ability to recognize human social signals and social behaviours like turn taking, politeness, and disagreement is essential when building social robots, human-robot interaction, or interactive systems

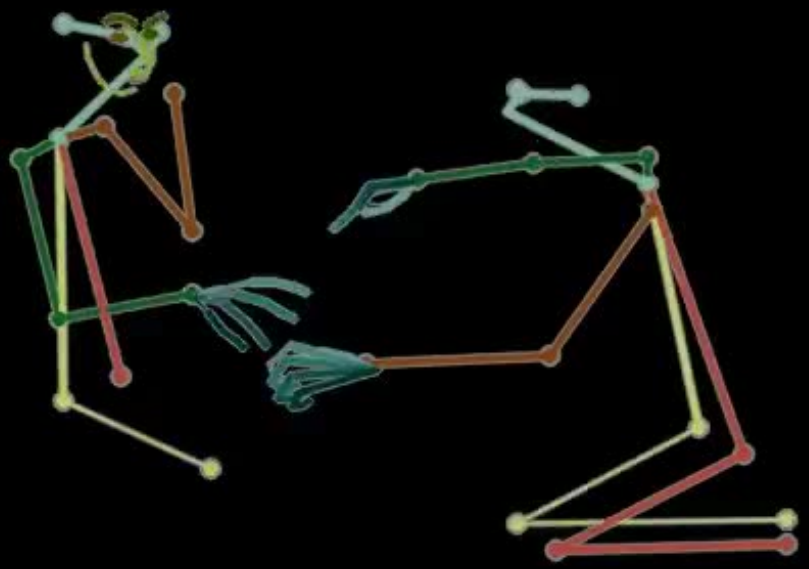
3 main problems

- *Modeling*: identification of the principles and laws
- *Analysis*: automatic detection and interpretation
- *Synthesis*: automatic generation of artificial social signals

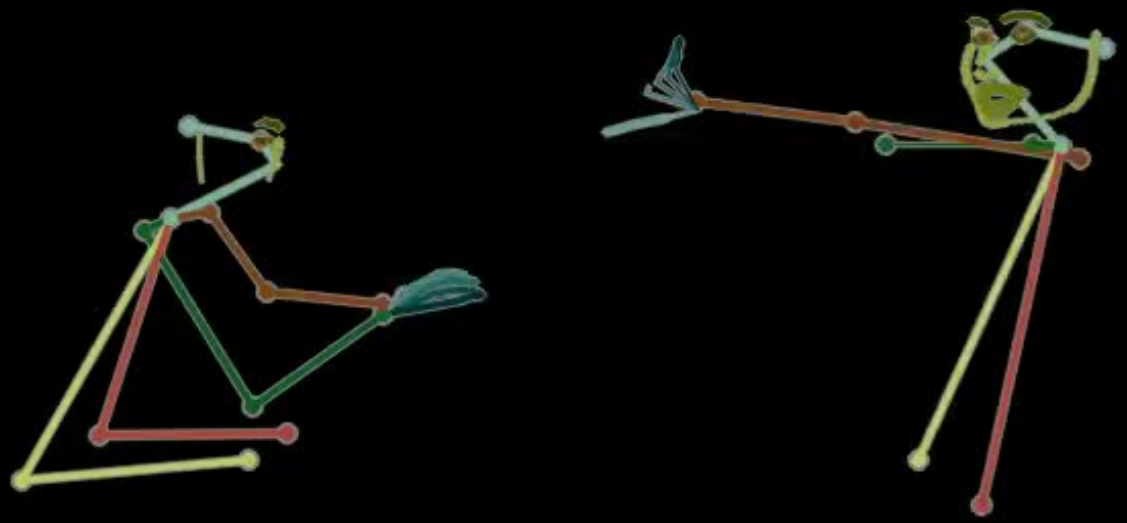
HOW HARD?

In order to understand “what’s going on?” (***classification***), we first need to build ***representations*** by the mean of raw data pre-processing (***conditionning***). This typically relies on ***models*** (skeletons, language, eye, etc.)






Focused!
ILLUMINATED POLITICAL
& PHYSICAL




open me

HOW HARD?

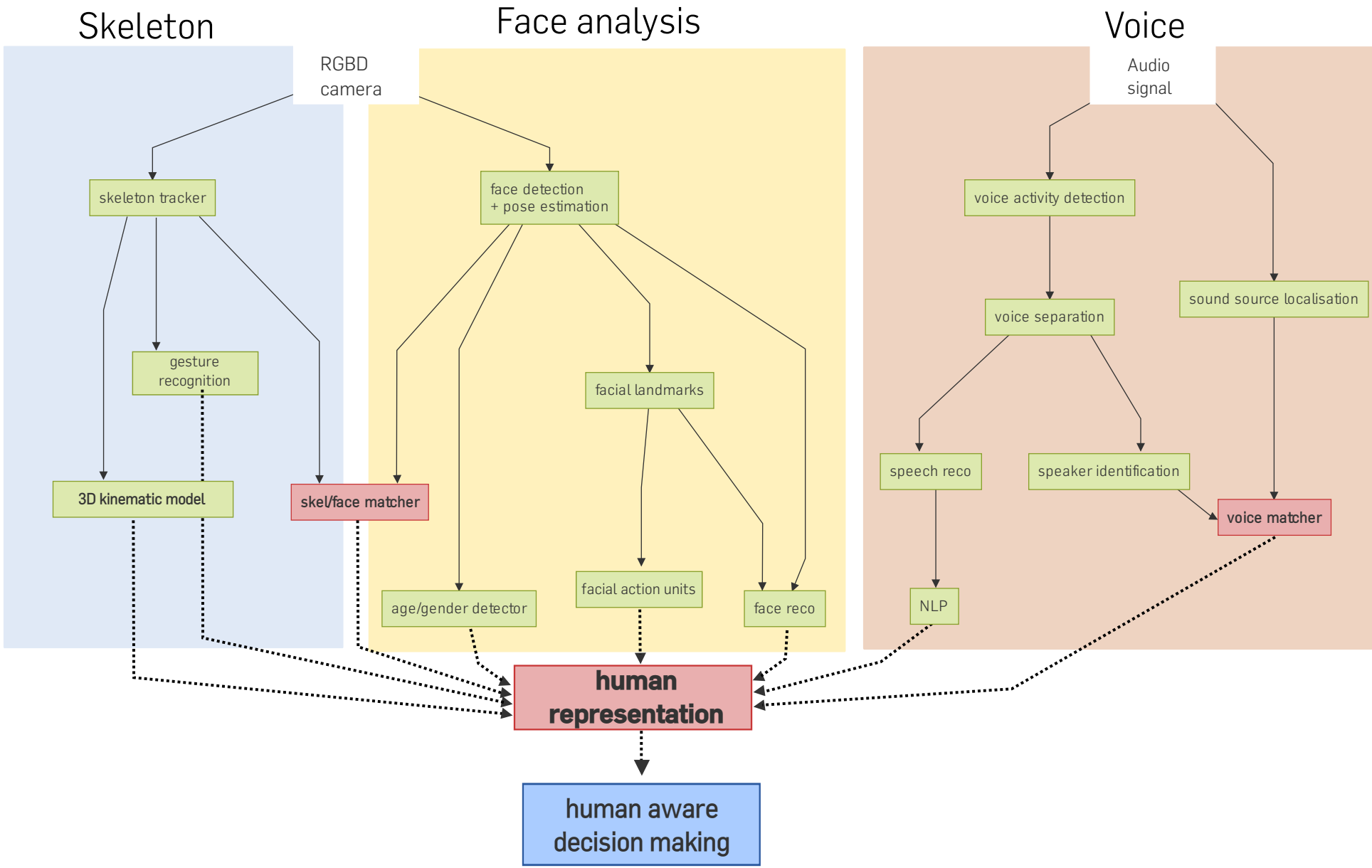
Building the right representation is hard, especially for multi-modal social signals (essentially, an open research question).

HOW HARD?

Building the right representation is hard, especially for multi-modal social signals (essentially, an open research question).

Social signal processing is typically broken down in smaller, more manageable, tasks:

- People detection
- Face detection
- Face recognition
- Gesture recognition
- Gaze detection
- Facial expression reading (wink, blink, talking, ...)
- Detection of social signals from verbal communication
- Emotion recognition (from faces, movement, speech, ...)
- ...


Skeleton
Face analysis
Voice
RGBD camera
Audio signal
skeleton tracker
face detection + pose estimation
voice activity detection
gesture recognition
facial landmarks
voice separation
sound source localisation
3D kinematic model
skel/face matcher
speech reco
speaker identification
voice matcher
age/gender detector
facial action units
face reco
NLP
human representation
human aware decision making

CASE STUDY: EMOTIONS

PAUL EKMAN AND BASIC EMOTIONS

Paul Ekman, a psychologist, found that when shown facial expressions people across the world all recognised six **basic emotions**.

Source: emotionresearcher.com

PAUL EKMAN AND BASIC EMOTIONS

Paul Ekman, a psychologist, found that when shown facial expressions people across the world all recognised six **basic emotions**.

Anger, disgust, fear, happiness, sadness and surprise.

Some other emotion are less universally recognised, e.g. contempt.

PAUL EKMAN AND BASIC EMOTIONS

Paul Ekman, a psychologist, found that when shown facial expressions people across the world all recognised six **basic emotions**.

Anger, disgust, fear, happiness, sadness and surprise.

Some other emotion are less universally recognised, e.g. contempt.

How can we process this social signal?

CLASSIFICATION

CLASSIFICATION

Social signal processing often relies on **classification.**

- **Classification** is deciding on which **category** a new observation belongs to based on training data.
- The training data contains data and known categories.
- Classification is a **supervised** learning algorithm.

CLASSIFICATION

Social signal processing often relies on **classification.**

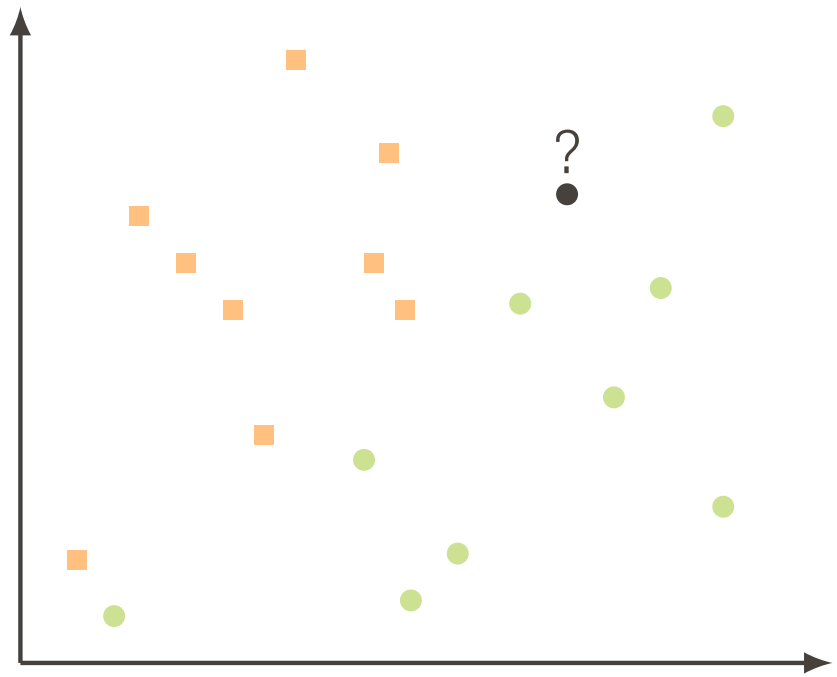
- **Classification** is deciding on which **category** a new observation belongs to based on training data.
- The training data contains data and known categories.
- Classification is a **supervised** learning algorithm.

Examples:

- An incoming tweet needs to classified as being positive or negative.
- An radar ping of a flying objects needs to be classified as belonging to one of ***n*** possible planes.
- The prosody of speech needs to be classified as belonging to one of six basic categories of emotion (happy, sad, angry, bored, surprised, neutral).
- A gesture filmed through a camera needs to be classified as meaning stop, go, left or right.

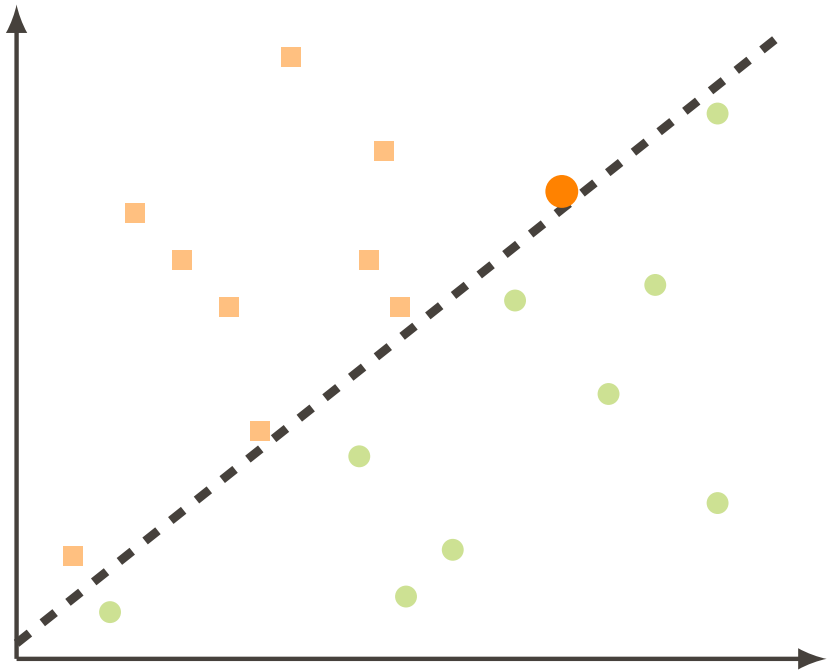
CLASSIFICATION: EXAMPLE

- Two dimensional problem
- Two categories, with training data for both categories
- To which category does a new observation belong?



CLASSIFICATION: EXAMPLE

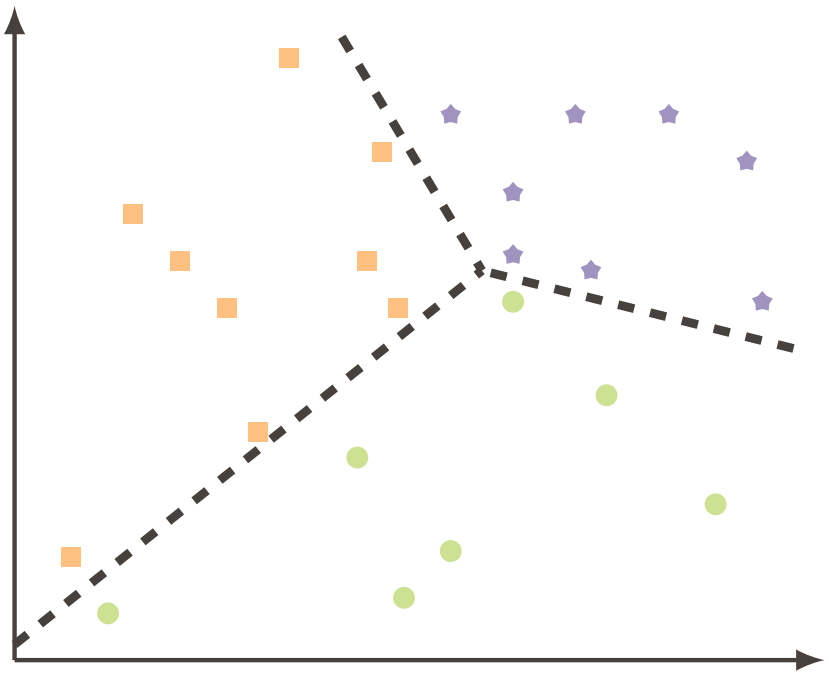
- Two dimensional problem
- Two categories, with training data for both categories
- To which category does a new observation belong?



When categories can be linearly separated, we have a very easy classification problem.

CLASSIFICATION: EXAMPLE

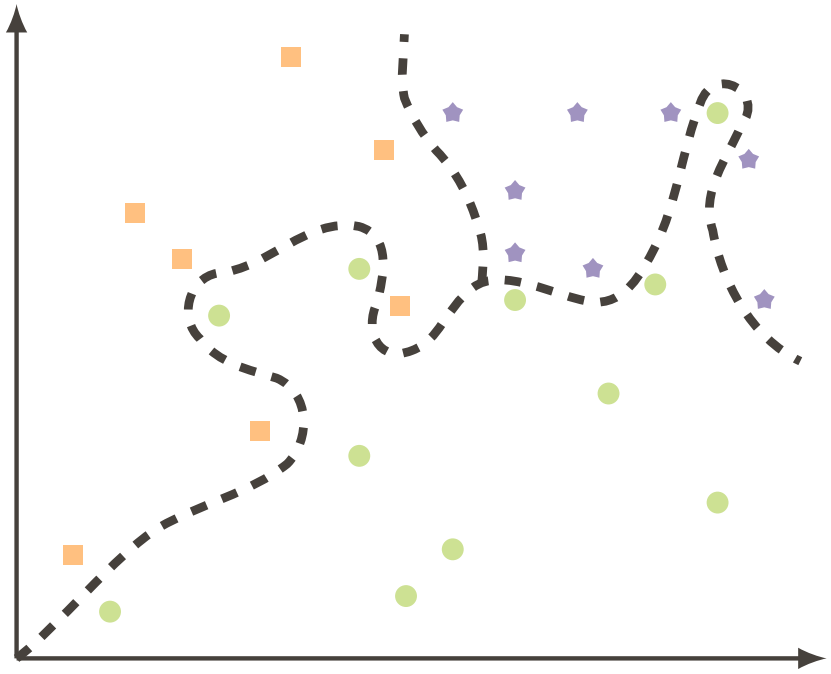
- Two dimensional problem
- Two categories, with training data for both categories
- To which category does a new observation belong?



Three or more can still be linearly separated.

CLASSIFICATION: EXAMPLE

- Two dimensional problem
- Two categories, with training data for both categories
- To which category does a new observation belong?



But what if categories are not linearly separable?

TWO CLASSIFICATION METHODS

Two methods will be explained here:

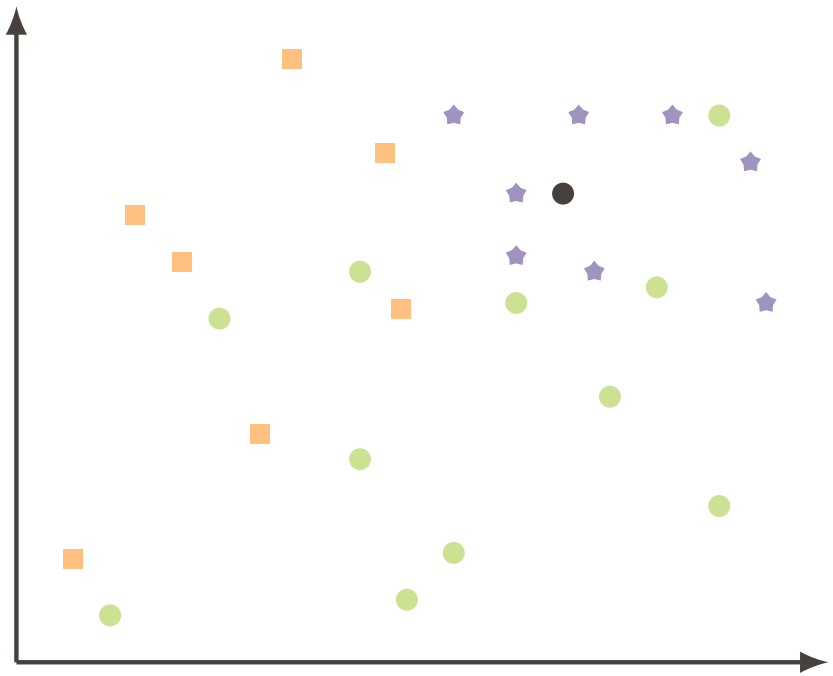
- *k*-nearest Neighbours (kNN)
- Support Vector Machines (SVM)

But there are hundreds of classifiers

- Decision trees
- Random forest
- Bayes classifiers
- Neural Networks
- ...

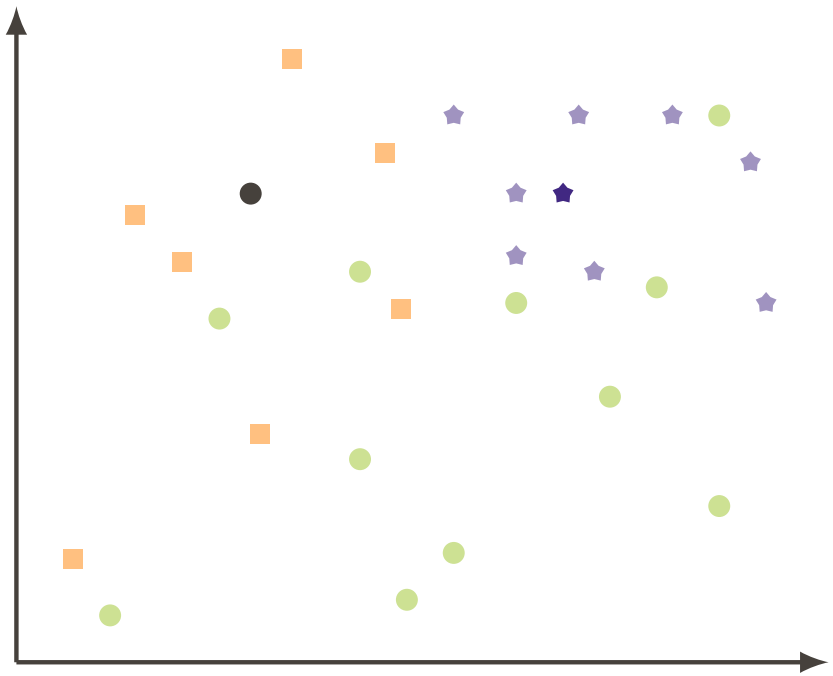
K-NEAREST NEIGHBOURS

When an observation comes in, calculate the distance to the *k* nearest neighbours. The observation belongs to the class the most frequent amongst neighbours.



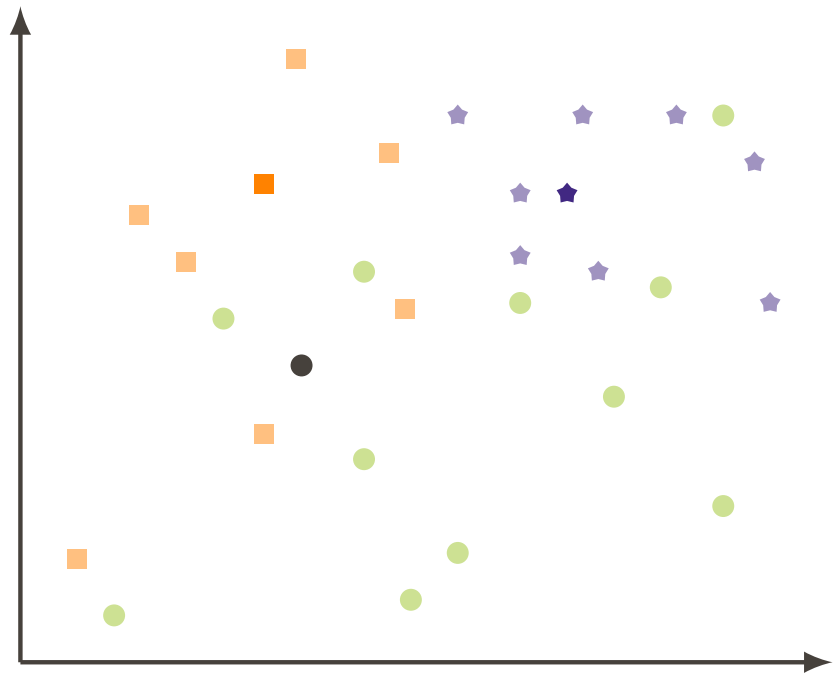
K-NEAREST NEIGHBOURS

When an observation comes in, calculate the distance to the *k* nearest neighbours. The observation belongs to the class the most frequent amongst neighbours.



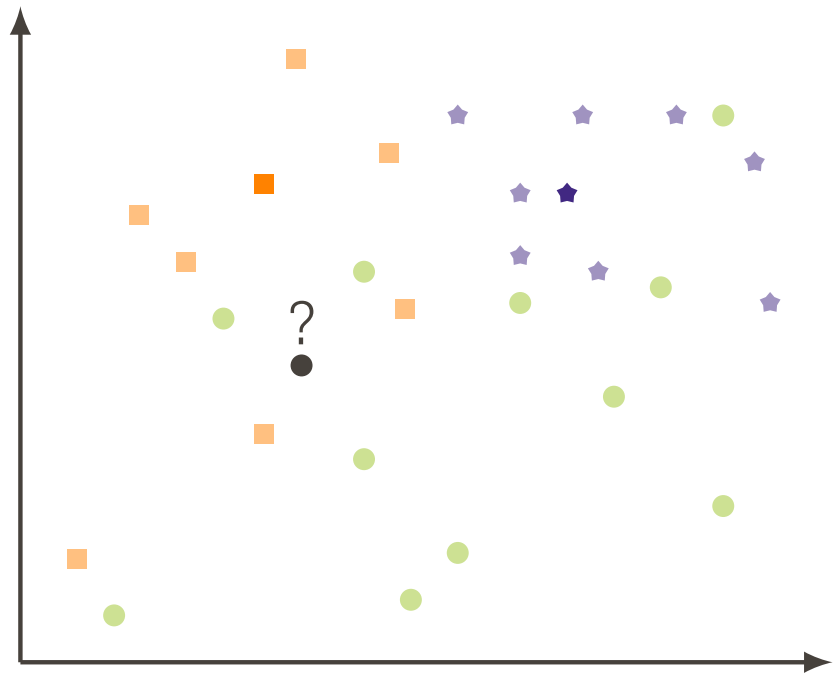
K-NEAREST NEIGHBOURS

When an observation comes in, calculate the distance to the *k* nearest neighbours. The observation belongs to the class the most frequent amongst neighbours.



K-NEAREST NEIGHBOURS

When an observation comes in, calculate the distance to the *k* nearest neighbours. The observation belongs to the class the most frequent amongst neighbours.



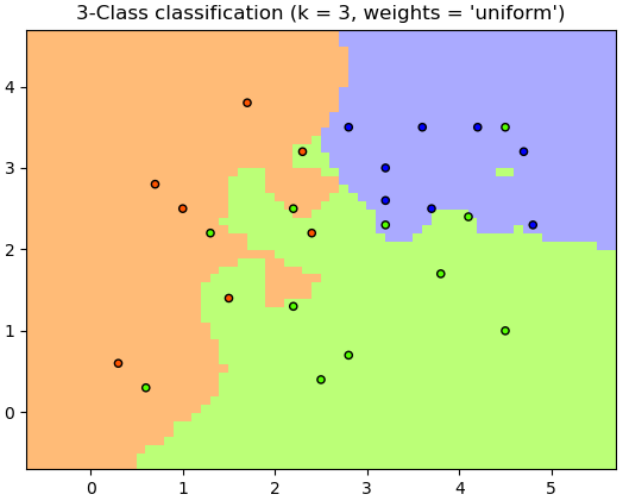
PYTHON IMPLEMENTATION OF *K*-NEAREST NEIGHBOURS

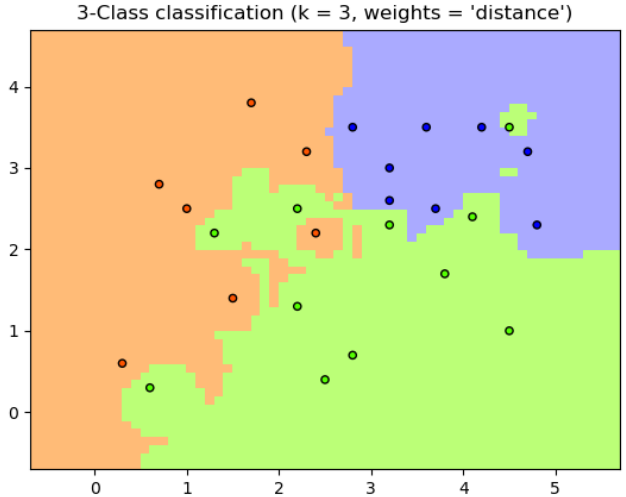
Complete example, with the data above:

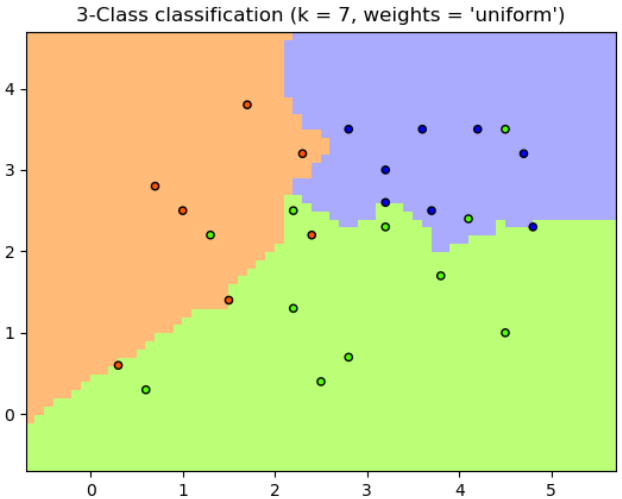
```
""" data.csv:
2.2,1.3,0 -> circles
3.2,2.3,0
...

2.3,3.2,1 -> squares
0.3,0.6,1
...

2.8,3.5,2 -> stars
3.2,2.6,2
...
"""
```

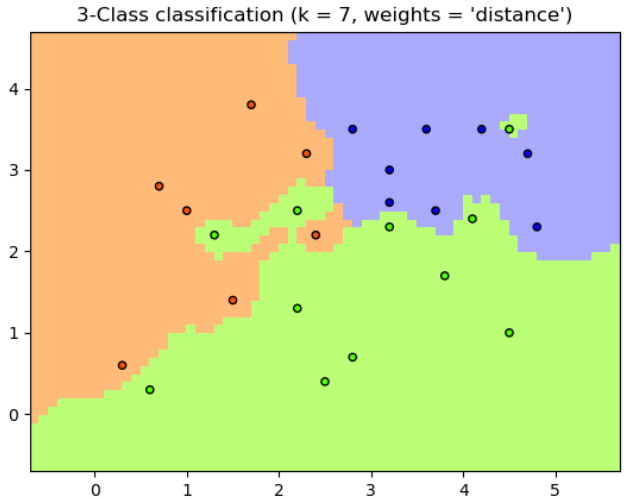
```
from numpy import genfromtxt
from sklearn import neighbors

csv = genfromtxt('data.csv', delimiter=',')
data = csv[:,:2]
categories = csv[:,2]

inputs = [ [3.5,3], [1.5,3], [1.8,1.9] ]

k = 3
knns = neighbors.KNeighborsClassifier(k)
knns.fit(data, categories)

predictions = knns.predict(inputs)
print(predictions)

>>>  [2.  1.  1.]
```

CHOICE OF *K* AND WEIGHTS









K-NEAREST NEIGHBOURS: SUMMARY

- *k*-nearest Neighbours usually does really well on a large number of classification problem, but can underperform near the classification boundary.

K-NEAREST NEIGHBOURS: SUMMARY

- *k*-nearest Neighbours usually does really well on a large number of classification problem, but can underperform near the classification boundary.
- might struggle with very large datasets, as it needs to calculate the distance to all training data every time you classify a new observation (some work around are available, relying on approximate distances and optimisation of the algorithmic code)

K-NEAREST NEIGHBOURS: SUMMARY

- *k*-nearest Neighbours usually does really well on a large number of classification problem, but can underperform near the classification boundary.
- might struggle with very large datasets, as it needs to calculate the distance to all training data every time you classify a new observation (some work around are available, relying on approximate distances and optimisation of the algorithmic code)
- commonly, the neighbours are weighted with the inverse of the distance $\frac{1}{d}$ to the observation (i.e. closer neighbours have stronger influence). With `sklearn`:

```
neighbors.KNeighborsClassifier(k, weights='distance')
```

SUPPORT VECTOR MACHINES

A very popular classification algorithm introduced by Vapnik in 1992.

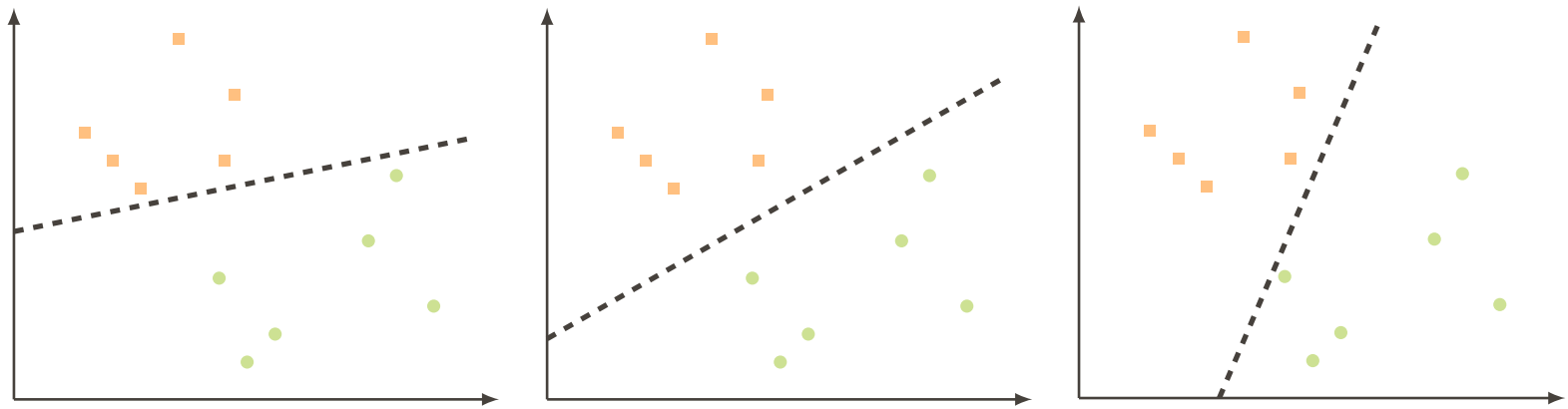
SUPPORT VECTOR MACHINES

A very popular classification algorithm introduced by Vapnik in 1992.

Often (but not always) provides very impressive classification performance on reasonably sized datasets.

SVM PRINCIPLE

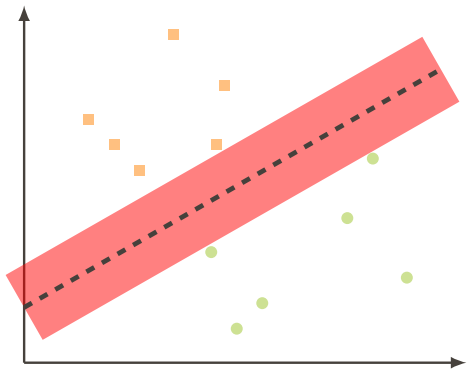
Three different classification lines. All are correct, but is there any reason why one is better than the others?



Intuitively, the line that is most distant to the training data is the best division.

SVM PRINCIPLE

Three different classification lines. All are correct, but is there any reason why one is better than the others?



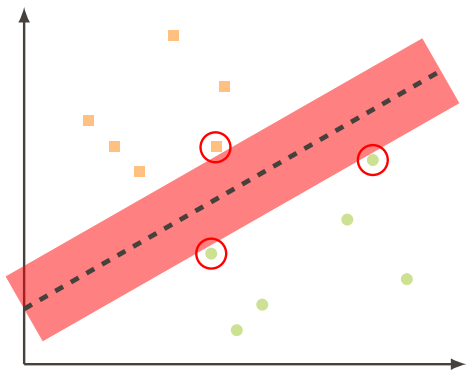
Intuitively, the line that is most distant to the training data is the best division.

The empty area near the division line is symmetric. It forms a bar in 2D space, a cylinder in 3D space, and a hyper-cylinder in n-D space.

SVM PRINCIPLE

The classifier in the middle is called the **maximum margin classifier**.

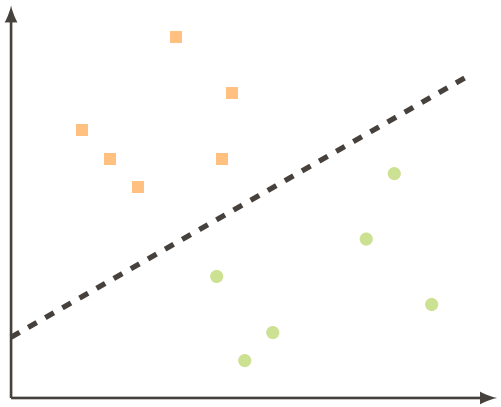
The data points nearest the classifier are called **support vectors**.



Two observations

- The margins should be as large as possible.
- The support vectors are the most useful datapoints because they are the ones that we might get wrong.

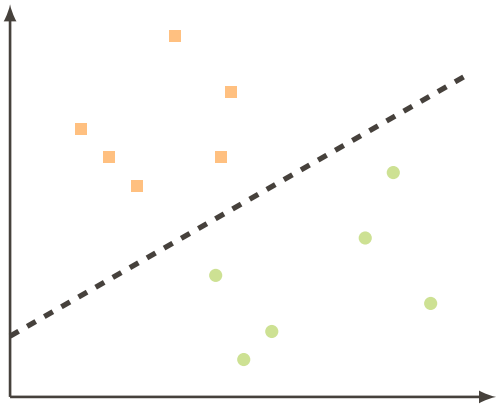
LINEAR CLASSIFIER



A linear classifier for two categories (**binary classifier**) can be written as $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$.

- $\mathbf{w} \cdot \mathbf{x} = \sum_i w_i x_i$. This is called the scalar product or inner product. Can also be written as a matrix multiplication $\mathbf{w}^T \mathbf{x}$.
- $\mathbf{w}$ is a **weight vector**, tilting the line
- $\mathbf{x}$ is a data point (of dimension n)
- b is a **bias**, lifting the line up along the y-axis

LINEAR CLASSIFIER

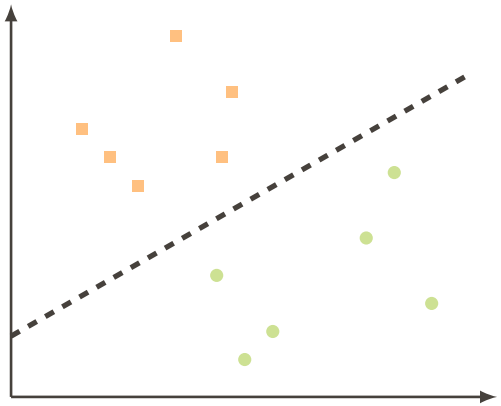


w has to be calculated such that:

$$f(\mathbf{x}) < 0 \Rightarrow \mathbf{x} \in \{\blacksquare\}$$

$$f(\mathbf{x}) > 0 \Rightarrow \mathbf{x} \in \{\circ\}$$

LINEAR CLASSIFIER



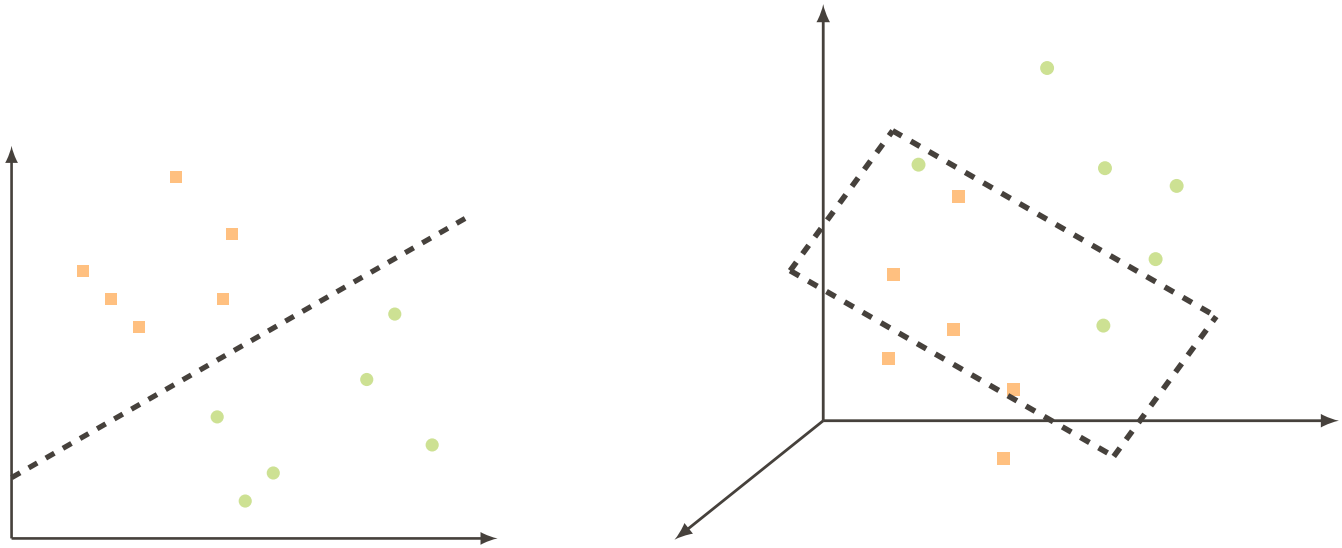
w has to be calculated such that:

$$f(\mathbf{x}) < 0 \Rightarrow \mathbf{x} \in \{\blacksquare\}$$

$$f(\mathbf{x}) > 0 \Rightarrow \mathbf{x} \in \{\circ\}$$

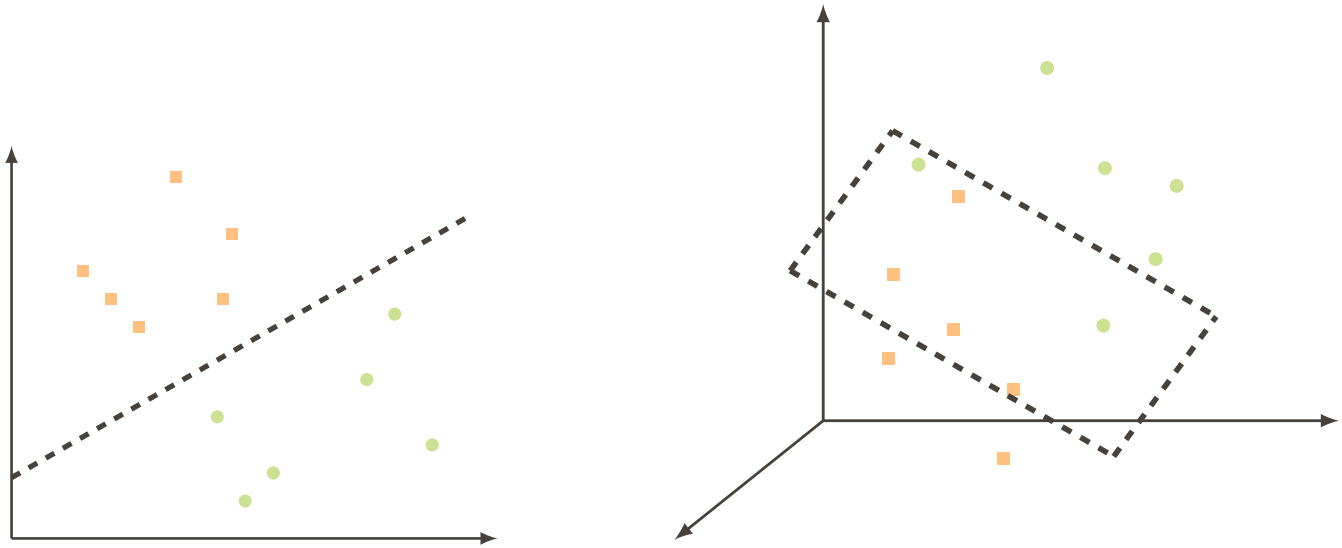
for kNNs, we have to carry along the training data; with a linear classifier, once **w** is found, we can discard the training data

LINEAR CLASSIFIER



- In 2D, the **discriminant** is a line
- In 3D, the discriminant is a plane
- In n-D , the discriminant is a hyperplane

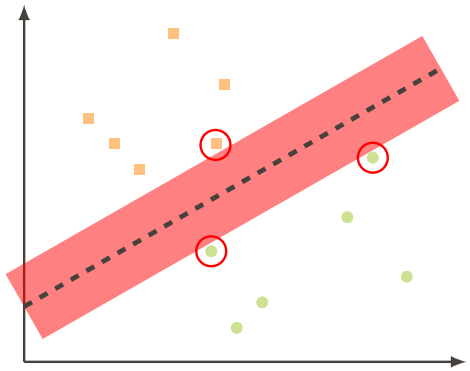
LINEAR CLASSIFIER



- In 2D, the **discriminant** is a line
- In 3D, the discriminant is a plane
- In n-D , the discriminant is a hyperplane

the 'dimensions' are the **features** of our inputs,
e.g. for a human, the size, age, skin colour, gender...

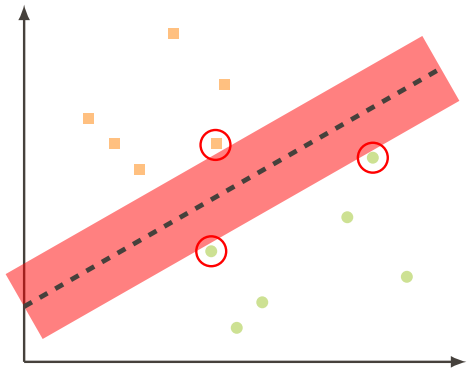
SUPPORT VECTOR CLASSIFIER



In a SVM, we want to **maximise the margin** M. We can rewrite the linear classifier.

If $\mathbf{w}^T\mathbf{x} + b \geq M$, then an observation belongs to ■, for $\mathbf{w}^T\mathbf{x} + b \leq -M$, it belongs to ○.

SUPPORT VECTOR CLASSIFIER



In a SVM, we want to **maximise the margin** M. We can rewrite the linear classifier.

If $\mathbf{w}^T\mathbf{x} + b \geq M$, then an observation belongs to ■, for $\mathbf{w}^T\mathbf{x} + b \leq -M$, it belongs to ○.

A point $\mathbf{x}^{■}$ that lies on the ■ class boundary line, i.e. $\mathbf{w}^T\mathbf{x}^{■} + b = M$, is a **support vector**.

FINDING THE OPTIMAL CLASSIFIER

The problem is to find the **optimal** values for **w** and *b*.

The classifier needs to satisfy two conditions

- It needs to correctly classify the training data,
- and needs the margin from the classifier to be as large as possible.

FINDING THE OPTIMAL CLASSIFIER

The problem is to find the **optimal** values for **w** and *b*.

The classifier needs to satisfy two conditions

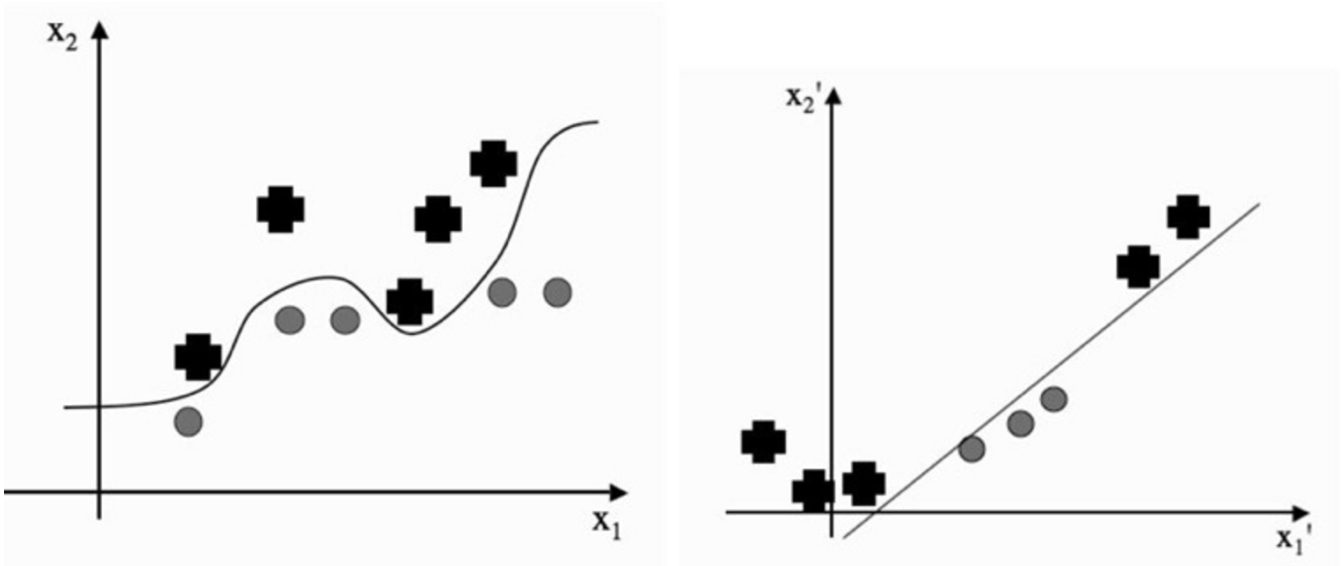
- It needs to correctly classify the training data,
- and needs the margin from the classifier to be as large as possible.

A **quadratic programming solver** is used to find the optimal values for **w** and *b*.

Oxford lecture on SVM to learn more about the mathematical formulation and the *perceptron algorithm*.

TRANSFORMATION OF DATA

- Unfortunately, ***all this still assumes that the data is linearly separable***. But what if it is not, and we are not prepared for a few misclassifications?
- The solution is to **transform** the data: move data points in the n-D space until the training data is linearly separable again



FEATURE MAPS

We can transform data points (= move them) or even add more dimensions using a function $\theta(\mathbf{x}_i)$ from input $\mathbf{x}_i$. θ is a **feature map**.

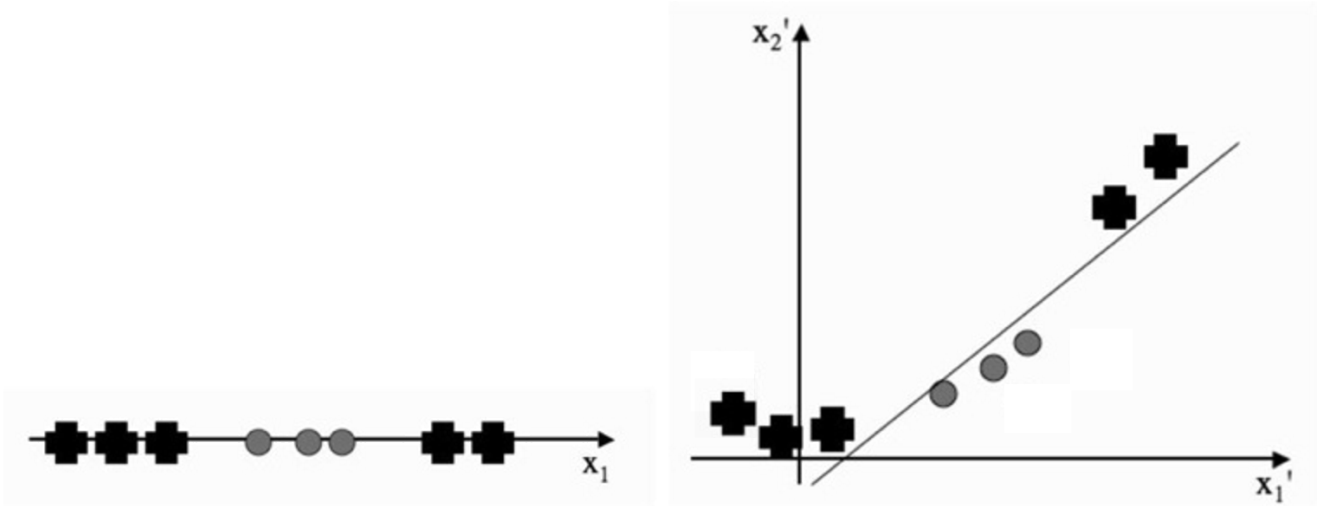
Example:

FEATURE MAPS

We can transform data points (= move them) or even add more dimensions using a function $\theta(\mathbf{x}_i)$ from input $\mathbf{x}_i$. θ is a **feature map**.

Example:

$$\theta(\mathbf{x}_i) = [\mathbf{x}_i, \mathbf{x}_i^2]$$

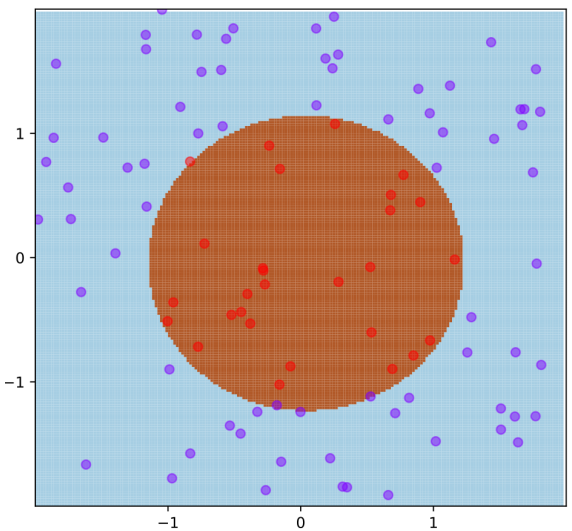


$\rightarrow$ now, linearly separable

FEATURE MAPS

We can transform data points (= move them) or even add more dimensions using a function $\theta(\mathbf{x}_i)$ from input $\mathbf{x}_i$. θ is a **feature map**.

Example:

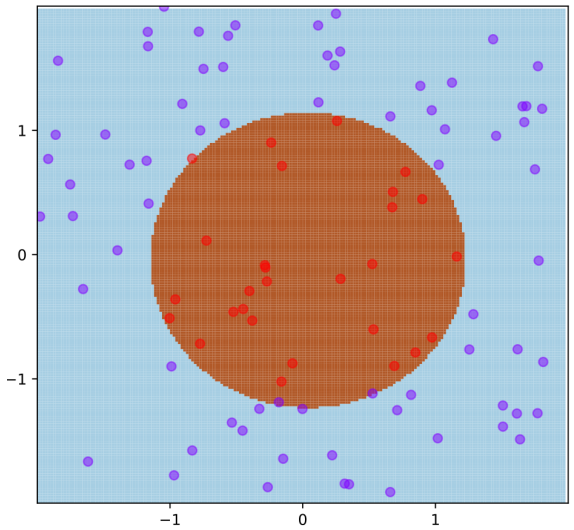


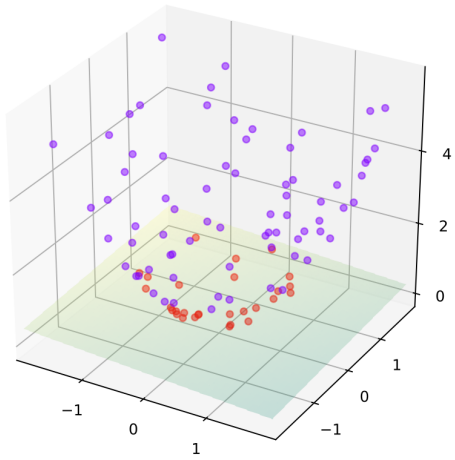
FEATURE MAPS

We can transform data points (= move them) or even add more dimensions using a function $\theta(\mathbf{x}_i)$ from input $\mathbf{x}_i$. θ is a **feature map**.

Example:

$$\theta(< x_i, y_i >) = [x_i, y_i, x_i^2 + y_i^2]$$





$\rightarrow$ now, linearly separable

FEATURE MAPS

If we know something about the structure of the data, then we might be able to identify feature maps that would be effective.

Often however we do not have such domain knowledge.

THE KERNEL TRICK

$$\theta : \mathbf{x} \rightarrow \theta(\mathbf{x}), \mathbb{R}^d \rightarrow \mathbb{R}^D$$

$$f(\mathbf{x}) = \mathbf{w}^T \theta(\mathbf{x}) + b$$

- Simply map $\mathbf{x}$ to $\theta(\mathbf{x})$ where data is separable
- Solve for $\mathbf{w}$ in high dimensional space $\mathbb{R}^D$
- If $D >> d$ then there are many more parameters to learn for $\mathbf{w}$. Can this be avoided?

THE KERNEL TRICK

$$\theta : \mathbf{x} \rightarrow \theta(\mathbf{x}), \mathbb{R}^d \rightarrow \mathbb{R}^D$$

$$f(\mathbf{x}) = \mathbf{w}^T \theta(\mathbf{x}) + b$$

It can be shown that you do not actually need to calculate explicitely $\theta(\mathbf{x})$. Instead, you only need $\theta(\mathbf{x}_i)^T \theta(\mathbf{x}_j)$ (with $\mathbf{x}_i$ the training data) to compute $\mathbf{w}$.

We call the **kernel function** $K(\mathbf{x}_i, \mathbf{x}_j) = \theta(\mathbf{x}_i)^T \theta(\mathbf{x}_j)$. The SVM classifier can be learnt and applied with only K, and the complexity depends only on N (size of training data), not on D.

STANDARD KERNELS

- **Polynomials** up to some degree s in the elements x_k of the input vector (e.g. x_3^3 or $x_1 x_4$). This can be written as:

$$K(x, y) = (1 + x^T y)^s$$

- **Sigmoid functions** of the x_k with parameters κ and δ, and kernel:

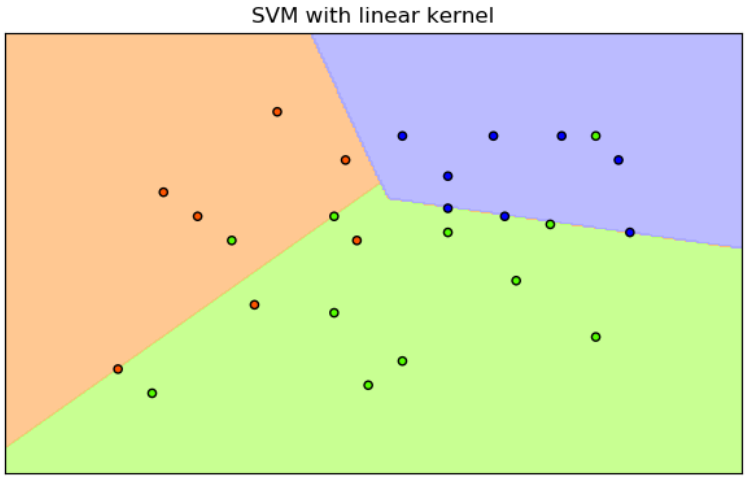
$$K(x, y) = tanh(\kappa x^T y - \delta)$$

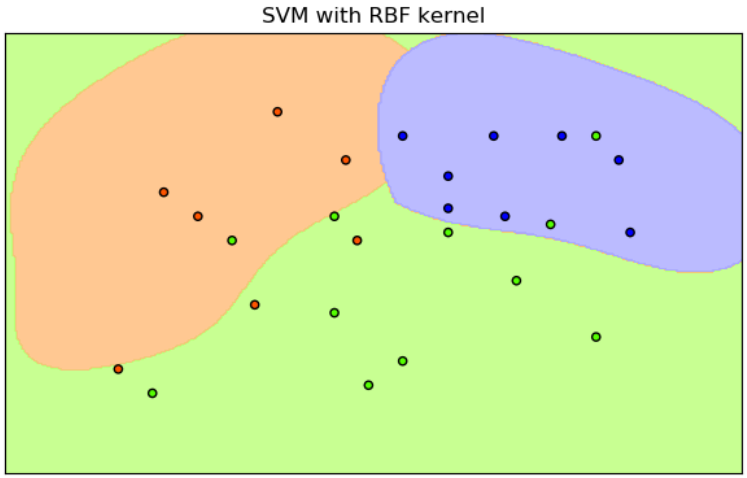
- **Radial basis function** expansions of the x_k with parameter σ and kernel:

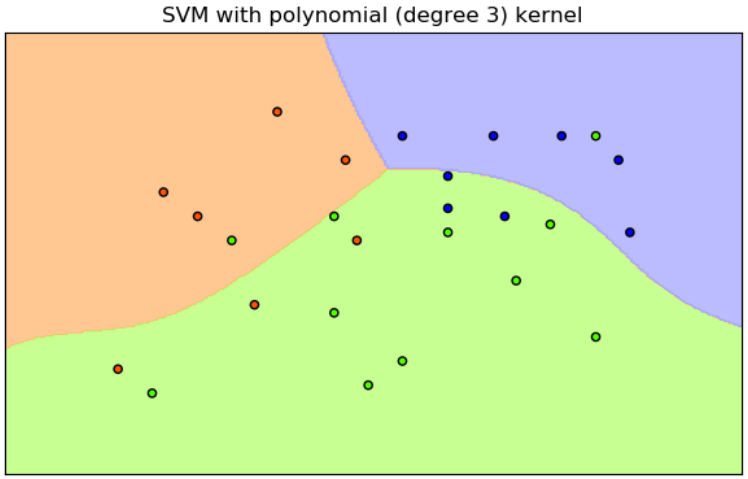
$$K(x, y) = exp(-\frac{(x - y)^2}{2\sigma^2})$$

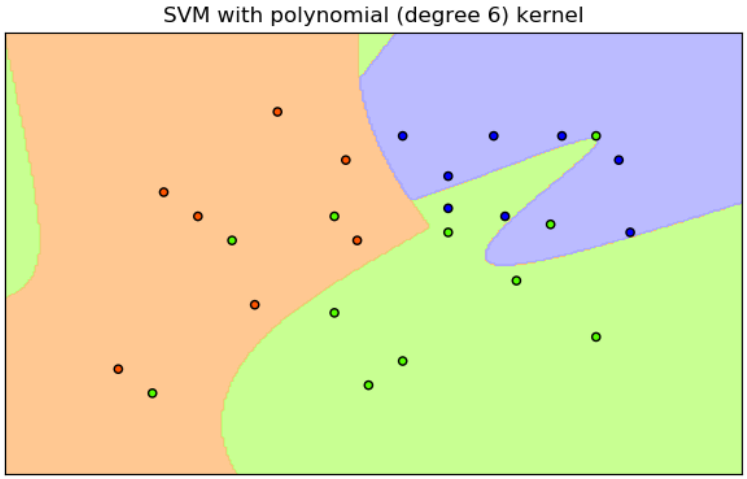
(a Gaussian kernel)

COMPARISON OF KERNEL FUNCTIONS









PYTHON IMPLEMENTATION OF SUPPORT VECTOR MACHINE

```
from sklearn import svm

C = 1.0  # SVM regularization parameter
         # +- 'how acceptable are misclassification'
clf = svm.SVC(kernel='linear', C=C) # kernel='poly', 'rbf', 'sigmoid'...

clf.fit(data, categories)

predictions = clf.predict(inputs)
```

PYTHON IMPLEMENTATION OF SUPPORT VECTOR MACHINE

Complete example, with our data:

```
""" data.csv:
2.2,1.3,0 -> circles
3.2,2.3,0
...

2.3,3.2,1 -> squares
0.3,0.6,1
...

2.8,3.5,2 -> stars
3.2,2.6,2
...
"""
```

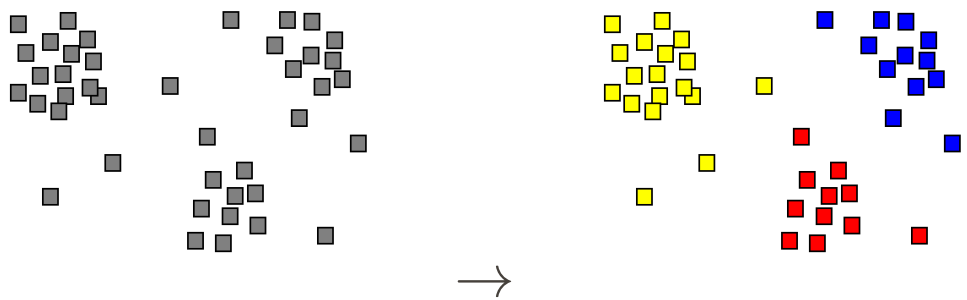
```
from numpy import genfromtxt
from sklearn import svm

csv = genfromtxt('data.csv', delimiter=',')
data = csv[:,:2]
categories = csv[:,2]


inputs = [ [3.5,3], [1.5,3], [1.8,1.9] ]

clf = svm.SVC(kernel='rbf',
              gamma = 0.7,
              C=1.0)
clf.fit(data, categories)

predictions = clf.predict(inputs)
print(predictions)

>>>  [2.  1.  0.]
```

CLUSTERING VS CLASSIFICATION

Clustering is sometimes confused with **classification**. (often because some algorithms have similar names,e.g. *k-means clustering* and *k-nearest neighbours*).

Clustering starts from unlabelled data points and tries to find *k* clusters in the data → **unsupervised learning**.

→ Used to find patterns in the data.



Classification starts from training data (→ **supervised learning**), and attempts to correctly classify an unknown observation.

LET'S BUILD OUR OWN EMOTION CLASSIFIER FROM SCRATCH

SHOPPING LIST

What do we need?

SHOPPING LIST

What do we need?

- a dataset of labelled faces

SHOPPING LIST

What do we need?

- a dataset of labelled faces
- some pre-processing to normalise the faces

SHOPPING LIST

What do we need?

- a dataset of labelled faces
- some pre-processing to normalise the faces
- features to extract

SHOPPING LIST

What do we need?

- a dataset of labelled faces
- some pre-processing to normalise the faces
- features to extract
- a classifier

Google

human face happiness

16

All | Images | Videos | Shopping | News | More | Settings | Tools | View saved | SafeSearch

smile | cartoon | male | boy | grandpa | black and white | real | happiness | joy | sad | gratitude | beauty | emotional | smiley











Disgust
Closed mouth

FEATURES: FACIAL ACTION UNITS

AU	Description	Example image
1	Inner Brow Raiser	
2	Outer Brow Raiser	
4	Brow Lowerer	
5	Upper Lid Raiser	
6	Cheek Raiser	

AU	Description	Example image
7	Lid Tightener	
9	Nose Wrinkler	
10	Upper Lip Raiser	
12	Lip Corner Puller	

FEATURES: FACIAL ACTION UNITS

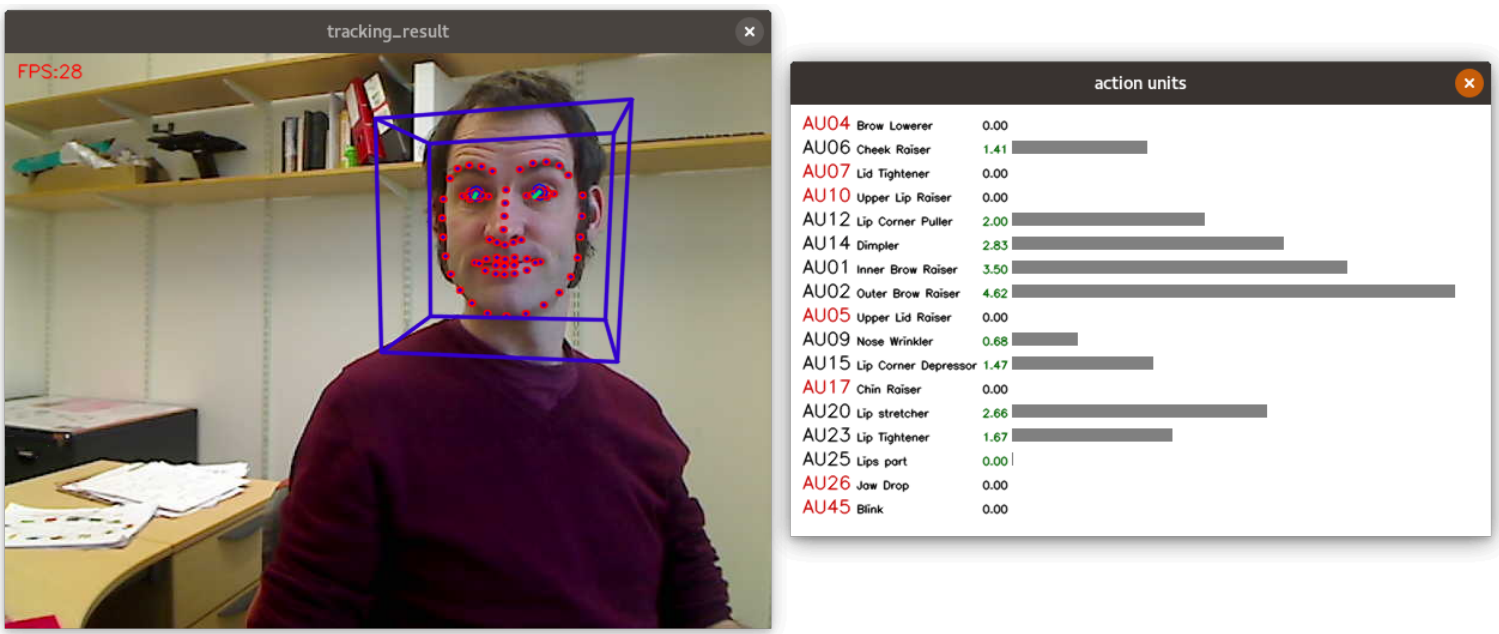
AU	Description	Example image
15	Lip Corner Depressor	
17	Chin Raiser	
20	Lip stretcher	
23	Lip Tightener	

AU	Description	Example image
14	Dimpler	
25	Lips part	
26	Jaw Drop	
45	Blink	

OPENFACE ACTION UNITS

OpenFace is an open-source library that recognises 17 action units (amongst many other things).

github.com/TadasBaltrusaitis/OpenFace



(not to be confused with this other CMU OpenFace)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	emotion	AU01	AU02	AU04	AU05	AU06	AU07	AU09	AU10	AU12	AU14	AU15	AU17	AU20	AU23	AU25	AU26	AU45	face
2	0	1.07	0	0.05	2.95	0	0	0	0	0	0	0	0	0	0	1.36	0	0	fear/processed/aligned/fear-041_aligned.bmp
3	0	1.57	0	0.63	0	1.02	1.39	0	0.45	0	0	0	0	0	0	0.45	1.35	0	fear/processed/aligned/fear-053_aligned.bmp
4	0	1.38	0	1.17	0	1.36	0	1.85	1.21	0	0	0	0	0.05	0	0	0	0	fear/processed/aligned/fear-035_aligned.bmp
5	0	0	0.97	0.87	2.48	0	0	0	0	0	0	0.31	0	0.71	0.32	1.41	0.24	0	fear/processed/aligned/fear-018_aligned.bmp
6	0	3.79	2.52	0	4.61	0	0	0	0	0	0	0	0	0	0	1.53	0.67	0	fear/processed/aligned/fear-043_aligned.bmp
7	0	2.55	0	0	1.2	1.58	0	0	1.27	0.64	0	0.93	1.14	0	0	0	0	0	fear/processed/aligned/fear-044_aligned.bmp
8	0	1.93	2.72	0	3.86	0	0	0	0	0	0	1.38	0.67	1.23	0	0.15	0.83	0	fear/processed/aligned/fear-006_aligned.bmp
75	1	0	1	0	0	1.79	3.94	0	2.96	2.62	0	0.79	0	0	0.36	0	0	1.08	happiness/processed/aligned/happiness-098_aligned.bmp
76	1	0	0	0	1.78	0	0	0	0	0	0	0	0	0	0	2.14	1.55	0	happiness/processed/aligned/happiness-093_aligned.bmp
77	1	0	0	0	0	2.17	2.74	1.8	1.83	1.91	1.23	0	0	0.61	0	3.24	0	0.04	happiness/processed/aligned/happiness-095_aligned.bmp
78	1	0	0	2.91	0	1.38	0	2.71	2.11	1.04	1.38	0.15	0.71	1.4	0.23	0.56	0	0.65	happiness/processed/aligned/happiness-046_aligned.bmp
79	1	0.51	0	0	0	1.23	2.38	0.82	0.33	2.21	0.99	0.03	0	0	0	1.5	0.39	0.07	happiness/processed/aligned/happiness-025_aligned.bmp
80	1	0.51	0.76	0	0	1.56	0	0	1.23	2.56	1.12	0	0	0	0	0	0	0	happiness/processed/aligned/happiness-024_aligned.bmp
81	1	0	0	0	0	3.22	3.45	0	2.84	3.37	0	0	0	0	0	0	0	0	happiness/processed/aligned/happiness-075_aligned.bmp
196	2	0	0.1	3.45	1.14	0	0	1.75	0.65	0	0	0.11	1.04	2.05	0	0.67	0	0.12	anger/processed/aligned/anger-051_aligned.bmp
197	2	0	0.92	0	0	0.43	1.79	0	0.24	0	0	0	0	0	0	0	0	0	anger/processed/aligned/anger-027_aligned.bmp
198	2	1.27	0	2.58	0	0	0	0	0.39	0	0	1.35	2.39	1.28	0.28	0	0	0	anger/processed/aligned/anger-057_aligned.bmp
199	2	4.5	1.72	0	1.24	1.54	0	0	1.63	0	0	0.22	0	0	0	2.29	0	0	anger/processed/aligned/anger-047_aligned.bmp
200	2	0	0	0	0.29	0	0	0.56	0	0	0	0	0	0	0	1.5	0	0	anger/processed/aligned/anger-008_aligned.bmp
201	2	0	0	0	0.93	0.86	0	0.8	1.35	0.85	0	0	0	0	0.5	0	0	0	anger/processed/aligned/anger-031_aligned.bmp
202	2	2.73	0	0	1.11	1.55	0.32	3.07	1.75	1.21	1.34	1.15	0	0	0.03	2.07	0	0	anger/processed/aligned/anger-019_aligned.bmp
203	2	0	1.1	3.59	0	2.33	3.16	2.56	3.38	0	1.34	0	0	1.92	0	3.03	0	0	anger/processed/aligned/anger-016_aligned.bmp
280	3	0	0	0	0.33	0	0	0	0	0	0	0	0	0	0.59	0	0	0	surprise/processed/aligned/surprise-077_aligned.bmp
281	3	0.34	0	0	1.38	1.24	0	0	1.48	2.05	0	0.57	0	0	0.12	3.27	0	0	surprise/processed/aligned/surprise-079_aligned.bmp
282	3	1.44	0	0	0.59	0	0	0	1.66	0.76	0	0.49	0	0	0.08	1.31	0	0	surprise/processed/aligned/surprise-040_aligned.bmp
283	3	0.24	0	0.9	1.43	0	0.89	0	1.23	0	0	1.29	0	0.56	0.11	0	0	0	surprise/processed/aligned/surprise-004_aligned.bmp
284	3	0.95	0	0	0	0	0	1.04	0	0	0.13	0	1.07	2.53	0.36	0	0	0	surprise/processed/aligned/surprise-062_aligned.bmp
285	3	0.69	0.91	0	1.87	0	0	0	1.34	0	0	1.09	0	0	0	2.94	1.45	0	surprise/processed/aligned/surprise-005_aligned.bmp
286	3	1.19	0.68	0	2.03	0.33	0	0	0.58	1.73	1.65	0.22	0	0	0	1.92	0.7	0	surprise/processed/aligned/surprise-029_aligned.bmp
287	3	3.88	2.83	0	4.25	0	0	0	0.52	1.14	0	1.09	0	0	0.1	1.49	1.64	0	surprise/processed/aligned/surprise-017_aligned.bmp
288	3	0	0	0	1.48	1.46	0.79	0	2.65	1.39	0	1.08	0.4	0	0.89	0.11	0	0	surprise/processed/aligned/surprise-080_aligned.bmp
289	3	0	0	0.79	1.62	0	2.49	0	0	0	0	0	0	0	0.01	1.62	1.87	0	surprise/processed/aligned/surprise-060_aligned.bmp
290	3	1.87	0.79	0	2.75	0	0	0	0	0	0	0	0	0	0.56	1.76	0.58	0	surprise/processed/aligned/surprise-002_aligned.bmp
368	4	0	0	1.58	0.84	0	0	0	0	0	0.59	0	0	0	0	0	0	0	sadness/processed/aligned/sadness-025_aligned.bmp

OUR TINY 'EMOTIONS' DATASET

- A couple of Google queries
- a bit of manual filtering
- 558 faces, between 70 and 130 per emotion
- split into 2 datasets: a **training** dataset (50 face per emotion) and a **testing** dataset
- for each face, OpenFace pre-processing to:
  - extract facial landmarks
  - normalise and mask out the faces
  - extract action units

The dataset is available for you to play with on the lecture's GitHub repo.

CLASSIFICATION RESULTS, ACTION UNITS

n = 17 dimensions (i.e., 17 action units)

kNN:

- k=1: 46.9%
- k=2: 42.2%
- k=4: 50.8%
- k=6: 46.1%
- k=9: 52.3%

SVM:

- kernel: rbf: 48.8%
- kernel: linear: 52.3%
- kernel: poly: 38.0%

```
from numpy import genfromtxt
from sklearn import neighbors, svm

csv_train = genfromtxt('training.csv')
training = csv_train[:,1:-1]
training_categories = csv_train[:,0]

csv_test = genfromtxt('testing.csv')
testing = csv_test[:,1:-1]
testing_categories = csv_test[:,0]

clf = neighbors.KNeighborsClassifier(k)
clf.fit(training, training_categories)
knns_predictions = clf.predict(testing)

clf = svm.SVC(kernel=kernel)
clf.fit(training, training_categories)
svm_predictions = clf.predict(testing)
```

CLASSIFICATION RESULTS, ACTION UNITS

n = 17 dimensions (i.e., 17 action units)

kNN:

- k=1: 46.9%
- k=2: 42.2%
- k=4: 50.8%
- k=6: 46.1%
- k=9: 52.3%

SVM:

- kernel: rbf: 48.8%
- kernel: linear: 52.3%
- kernel: poly: 38.0%

Chance: 16.7%

```
from numpy import genfromtxt
from sklearn import neighbors, svm

csv_train = genfromtxt('training.csv')
training = csv_train[:,1:-1]
training_categories = csv_train[:,0]

csv_test = genfromtxt('testing.csv')
testing = csv_test[:,1:-1]
testing_categories = csv_test[:,0]

clf = neighbors.KNeighborsClassifier(k)
clf.fit(training, training_categories)
knns_predictions = clf.predict(testing)

clf = svm.SVC(kernel=kernel)
clf.fit(training, training_categories)
svm_predictions = clf.predict(testing)
```


Natural emotions are much harder

MULTI-MODAL SOCIAL PROCESSING

We can usually improve social signal processing by **combining several modalities**.

Source: Berlin Database of Emotional Speech

MULTI-MODAL SOCIAL PROCESSING

We can usually improve social signal processing by **combining several modalities**.

Source: Berlin Database of Emotional Speech

„Sie haben es gerade hochgetragen und jetzt gehen sie wieder runter“ (They just carried it upstairs and now they are going down again).

Which emotion do you recognise?

Anger – Boredom – Disgust – Anxiety/Fear – Happiness – Sadness – Neutral

MULTI-MODAL SOCIAL PROCESSING

We can usually improve social signal processing by **combining several modalities**.

Source: Berlin Database of Emotional Speech

Prosody is an important *non-verbal* social signal.

Humans recognise emotions from prosody with approx. 80% accuracy (*caveat: this is for speech spoken by actors in a recording studio*)

WE'VE BARELY SCRATCHED THE SURFACE!

Social signal processing is extracting relevant information from the social environment.

Some work relatively well

- Face **detection**, voice activity detection, gender classification, ...

Some work, but need improvement

- Gaze detection, basic emotion recognition, face **recognition**, speech recognition, ...

But still many open problems remaining

- Complex real-word affect and emotion recognition (e.g. embarrassment, pride).
- Speech recognition for atypical speakers (children, elderly), multi-party interaction, ...

That's all for today, folks!

Questions:
severin.lemaignan@brl.ac.uk

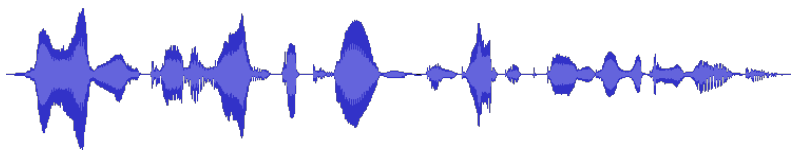
Slides:
github.com/severin-lemaignan/lecture-social-signal-processing

CLASSIFICATION OF AUDITORY SIGNALS

Raw signals will in most cases require pre-processing to extract features.

The raw social signal (audio or video) requires pre-processing to extract between 10 and over a 1000 **features**.

- A raw signals contains too much data, and cannot be fed to the classifier immediately.



- Pre-processing extracts feature data which is relevant for the information which we are after (pitch, volume/energy, duration, formant frequencies, …)
- These features then form the input for the classifier.

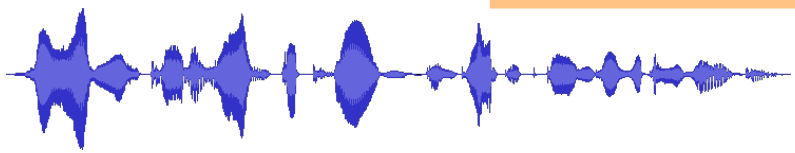
For more information see, for example, Liang et al. (2005) Feature analysis and extraction for audio automatic classification, Systems, Man and Cybernetics, 2005 IEEE International Conference on.

CLASSIFICATION OF AUDITORY SIGNALS

Raw signals will in most cases require pre-processing to extract features.

The raw social signal (audio or video) requires pre-processing to extract between 10 and over a 1000 **features**.

- A raw signals contains too much data and cannot be fed to the classifier immediately.



- Pre-processing extracts feature data which is relevant for the information which we are after (pitch, volume/energy, duration, formant frequencies, …)
- These features then form the input for the classifier.

An exception to this are Convolutional Neural Networks, which can deal with unprocessed data

For more information see, for example, Liang et al. (2005) Feature analysis and extraction for audio automatic classification, Systems, Man and Cybernetics, 2005 IEEE International Conference on.

EXAMPLE: RECOGNISING GENDER FROM SPEECH

Can we automatically recognise someone's gender from speech?

3,168 recorded voice samples, collected from male and female speakers.

- Examples from the database: male (US), female (US), male (Scotish)

The voice samples are pre-processed by acoustic analysis in R using the `seewave`, `warbleR` and `tuneR` packages, with an analysed frequency range of 0hz-280Hz (fundamental frequency of human speech).

Source: data and more information

THE 20 FEATURES USED FOR GENDER CLASSIFICATION

- **meanfreq**: mean frequency (in kHz)
- **sd**: standard deviation of frequency
- **median**: median frequency (in kHz)
- **Q25**: first quantile (in kHz)
- **Q75**: third quantile (in kHz)
- **IQR**: interquantile range (in kHz)
- **skew**: skewness (see note in specprop description)
- **kurt**: kurtosis (see note in specprop description)
- **sp.ent**: spectral entropy
- **sfm**: spectral flatness
- **mode**: mode frequency
- **centroid**: frequency centroid (see specprop)
- **peakf**: peak frequency (frequency with highest energy)
- **meanfun**: average of fundamental frequency measured across acoustic signal

THE 20 FEATURES USED FOR GENDER CLASSIFICATION

- **minfun**: minimum fundamental frequency measured across acoustic signal
- **maxfun**: maximum fundamental frequency measured across acoustic signal
- **meandom**: average of dominant frequency measured across acoustic signal
- **mindom**: minimum of dominant frequency measured across acoustic signal
- **maxdom**: maximum of dominant frequency measured across acoustic signal
- **dfrange**: range of dominant frequency measured across acoustic signal
- **modindx**: modulation index. Calculated as the accumulated absolute difference between adjacent measurements of fundamental frequencies divided by the frequency range

EXAMPLE: RECOGNISING GENDER FROM SPEECH

meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	mode	centroid	meanfun	minfun	maxfun	meandom	mindom	maxdom	dfrange	modindx	label
0.05978	0.06424	0.03203	0.01507	0.09019	0.07512	12.8635	274.403	0.89337	0.49192	0	0.05978	0.08428	0.0157	0.27586	0.00781	0.00781	0.00781	0	0	male
0.06601	0.06731	0.04023	0.01941	0.09267	0.07325	22.4233	634.614	0.89219	0.51372	0	0.06601	0.10794	0.01583	0.25	0.00901	0.00781	0.05469	0.04688	0.05263	male
0.07732	0.08383	0.03672	0.0087	0.13191	0.12321	30.7572	1024.93	0.84639	0.47891	0	0.07732	0.09871	0.01566	0.27119	0.00799	0.00781	0.01563	0.00781	0.04651	male
0.192275	0.060818	0.21913	0.130952	0.242491	0.111539	1.891999	6.680009	0.915249	0.461751	0.244465	0.192275	0.114544	0.016615	0.210526	0.518682	0.03125	4.164063	4.132813	0.118491	male
0.203083	0.058876	0.232857	0.134286	0.247143	0.112857	2.858921	12.34421	0.852594	0.320577	0.246429	0.203083	0.108871	0.023426	0.15534	0.4375	0.21875	0.734375	0.515625	0.296296	male
0.166858	0.076829	0.202062	0.112096	0.228522	0.116426	1.971154	7.121826	0.937182	0.624363	0.216014	0.166858	0.095246	0.016563	0.142857	0.310547	0.15625	0.734375	0.578125	0.3	male
0.187391	0.059659	0.202846	0.125692	0.235809	0.110116	1.722606	6.693799	0.923242	0.464031	0.232549	0.187391	0.098694	0.027972	0.141593	0.324405	0.164063	0.59375	0.429688	0.324545	male
0.194088	0.061379	0.216466	0.127631	0.246827	0.119197	1.490315	4.321145	0.88923	0.364435	0.249639	0.194088	0.10925	0.036782	0.231884	0.495793	0.117188	2.164063	2.046875	0.192748	male
0.185908	0.062359	0.198432	0.133178	0.242034	0.108856	1.396273	5.493992	0.934298	0.521624	0.260127	0.185908	0.11307	0.020434	0.195122	0.696514	0.140625	5.414063	5.273438	0.165689	male
0.178028	0.070548	0.19	0.127436	0.242821	0.115385	2.149795	9.1742	0.945836	0.637708	0.251795	0.178028	0.116113	0.020101	0.275862	0.983854	0.03125	5.109375	5.078125	0.225199	male
0.187952	0.065855	0.203059	0.132068	0.245099	0.113031	1.480657	5.018319	0.934454	0.582426	0.243909	0.187952	0.111375	0.019704	0.181818	0.519737	0.0625	2.84375	2.78125	0.17603	male
0.208232	0.033483	0.214075	0.186861	0.231538	0.044678	2.569042	10.79804	0.86938	0.16029	0.226299	0.208232	0.186654	0.023121	0.258065	0.79974	0.171875	3.242188	3.070313	0.229271	female
0.199387	0.03544	0.196046	0.180719	0.226471	0.045752	2.342196	9.294968	0.860738	0.210884	0.18415	0.199387	0.159956	0.027119	0.271186	0.898438	0.007813	5.976563	5.96875	0.189915	female
0.195679	0.031613	0.193891	0.181785	0.21166	0.029874	3.189926	15.15865	0.850763	0.201765	0.193501	0.195679	0.183362	0.017778	0.25	0.935397	0.1875	5.921875	5.734375	0.193079	female
0.195605	0.033526	0.197941	0.179728	0.210751	0.031022	3.079277	14.56234	0.861835	0.22135	0.197341	0.195605	0.182781	0.029685	0.266667	1.055226	0.015625	6.25	6.234375	0.196491	female
0.200325	0.033138	0.205588	0.179753	0.223236	0.043483	2.107451	7.372721	0.869482	0.179791	0.223418	0.200325	0.17806	0.037915	0.266667	1.13151	0.164063	5.609375	5.445313	0.202108	female
0.212681	0.042392	0.212152	0.180529	0.25138	0.070851	1.142382	3.271853	0.895565	0.198001	0.19654	0.212681	0.169677	0.017837	0.266667	1.740885	0.148438	7	6.851563	0.35467	female
0.198039	0.030396	0.198105	0.183464	0.212974	0.02951	2.118279	7.139244	0.857263	0.177914	0.189412	0.198039	0.188897	0.025932	0.242424	0.508878	0.109375	1.507813	1.398438	0.324904	female
0.218552	0.037574	0.220555	0.200416	0.246274	0.045858	2.4775	11.06604	0.877562	0.188399	0.220555	0.218552	0.182308	0.020725	0.275862	0.474609	0.007813	1.492188	1.484375	0.199624	female
0.198203	0.031488	0.195094	0.182032	0.218269	0.036238	2.643353	12.04094	0.862682	0.174607	0.183085	0.198203	0.186066	0.07619	0.238806	0.714154	0.171875	6.171875	6	0.113542	female
0.202647	0.031964	0.198973	0.184434	0.223804	0.03937	2.44728	10.35293	0.864479	0.165662	0.185904	0.202647	0.184609	0.021769	0.25	1.107759	0.070313	6.140625	6.070313	0.19701	female
0.217759	0.031261	0.223285	0.1991	0.237762	0.038662	2.038032	6.67446	0.861819	0.15492	0.226691	0.217759	0.193159	0.017335	0.271186	1.109086	0.007813	5.914063	5.90625	0.177407	female
0.191456	0.030422	0.19173	0.172434	0.212874	0.04044	2.109024	7.296761	0.85787	0.175168	0.172023	0.191456	0.179518	0.028269	0.271186	0.642188	0.171875	3.429688	3.257813	0.174889	female

EXAMPLE: RECOGNISING GENDER FROM SPEECH

Performance

- kNN (k = 7): 97.8% classified correctly.
- SVM: 97.5% classified correctly.

Recognising gender from speech is easy and robust.

All classification algorithms can deal with this problem.