Intro to ML for Input Recognition

CHI Course Computational Interaction 2017

Otmar Hilliges

8 May 2017

Advanced Interactive Technologies

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Explicit interaction

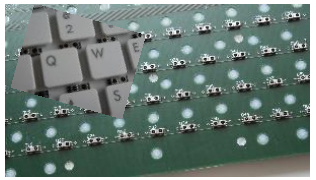
Implicit interaction

Designing Interactive Systems

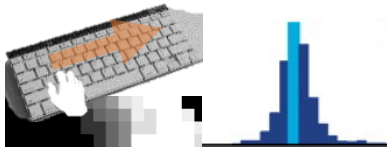
Interaction Task

Sensing Technology

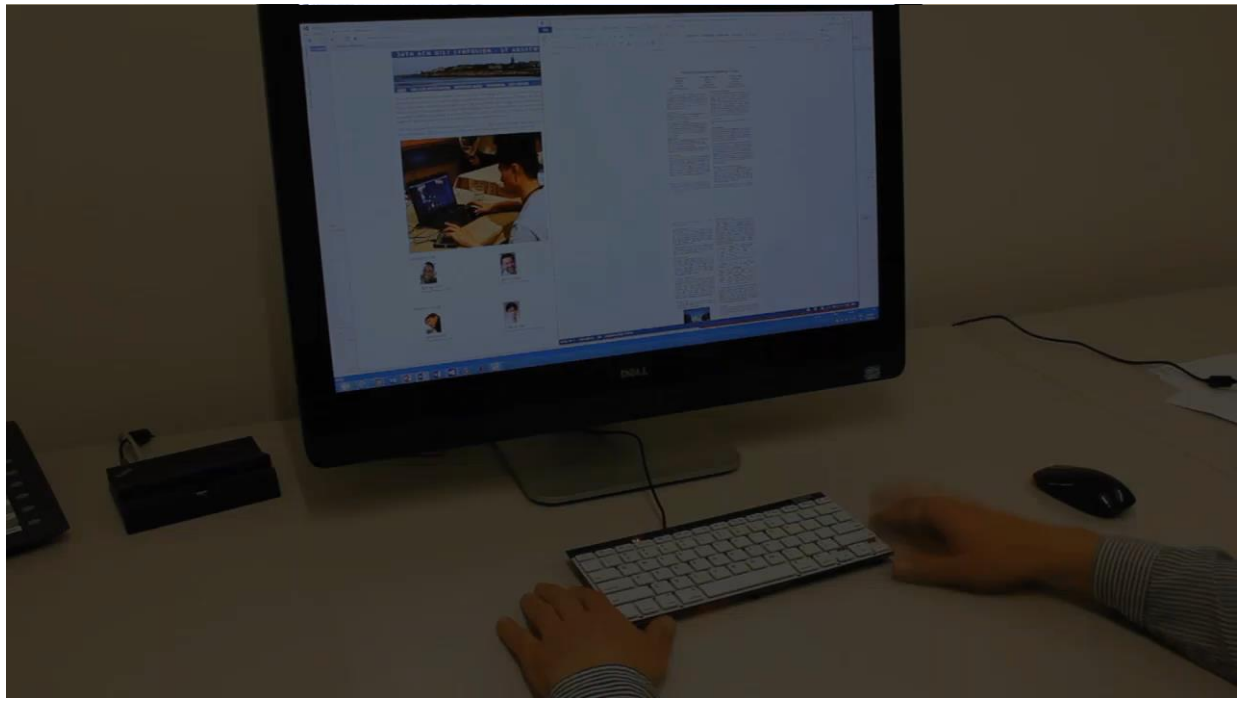
Sensor Placement



Input Recognition



System Output



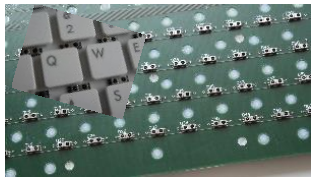
[Motion Sensing Keyboard. ACM CHI'14]

Today's topic

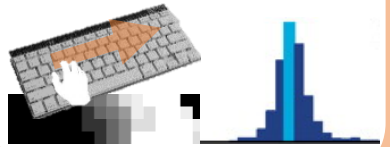
Interaction Task

Sensing Technology

Sensor Configuration



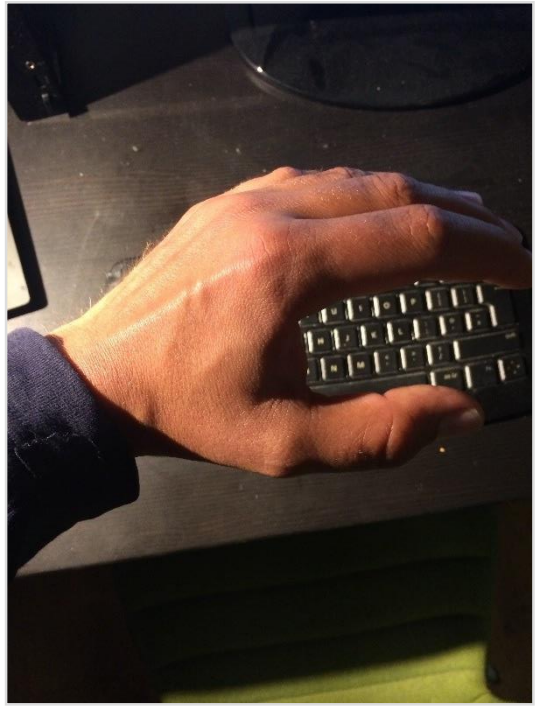
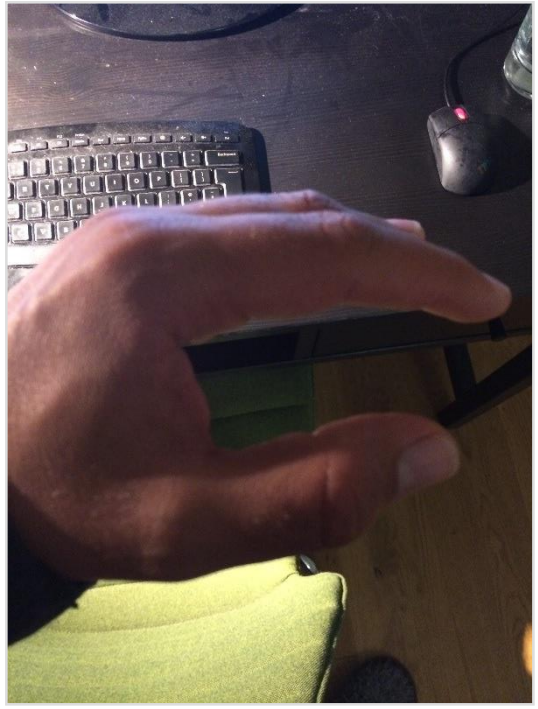
Input Recognition



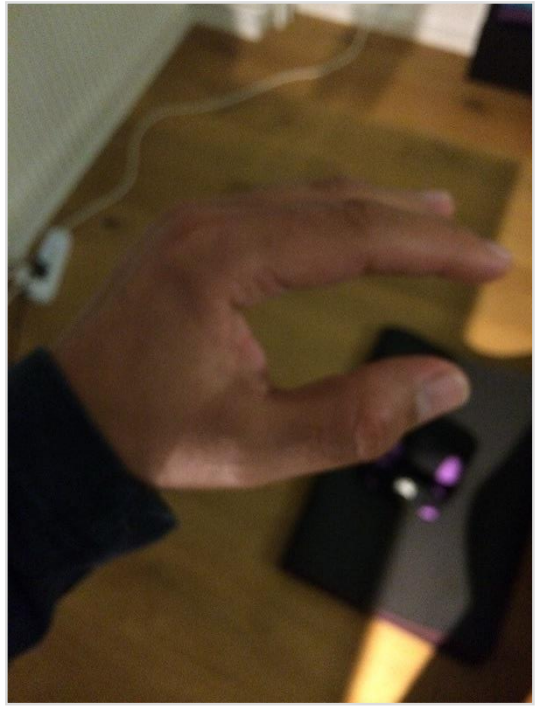
System Output

Today: Machine Learning for Input Recognition







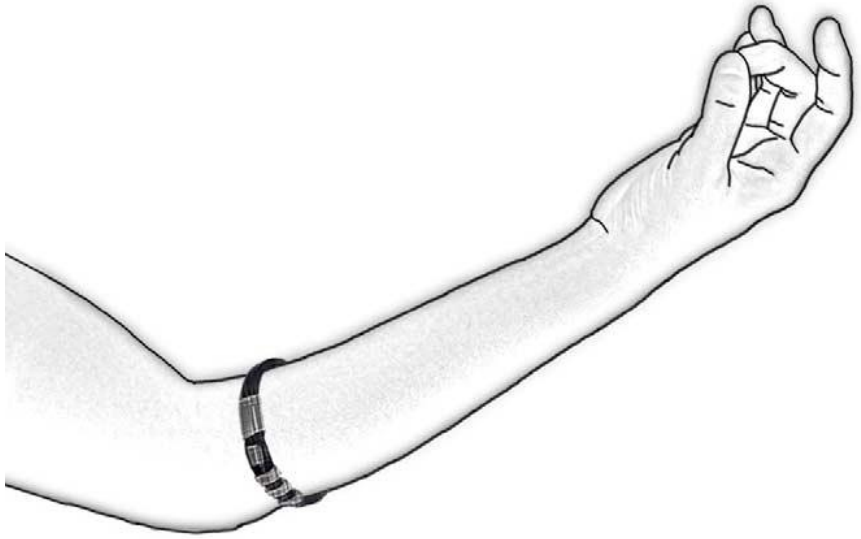
Muscle-Computer Interfaces

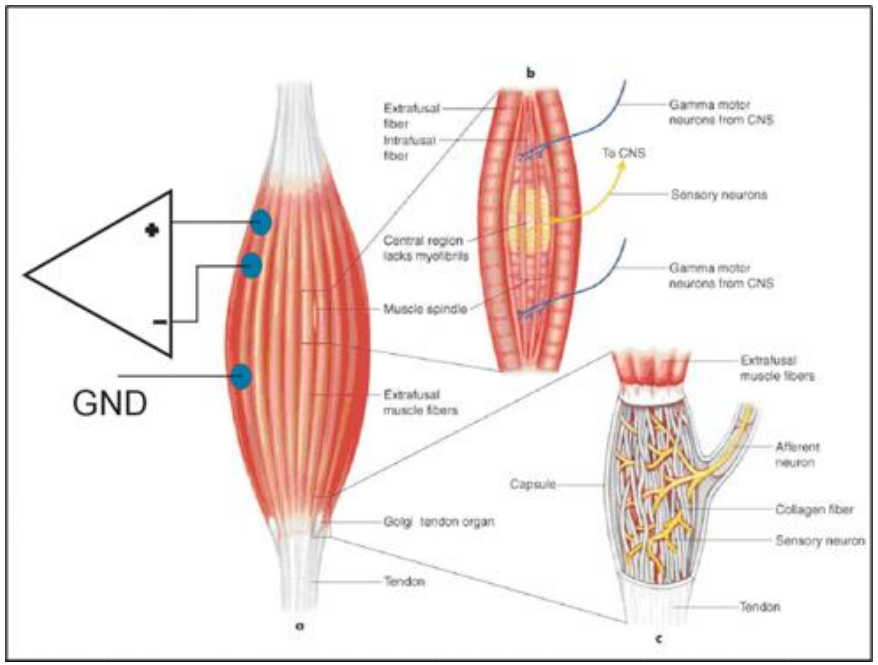
Hands free interaction – even when holding an object

EMG or Electromyography (primarily used in medical therapy)

Action potential generated by muscle when signal arrives from motor neuron

Can be sensed non-invasively by placing electrodes on the skin





[Saponas et al., *Proceedings of ACM UIST '09*]

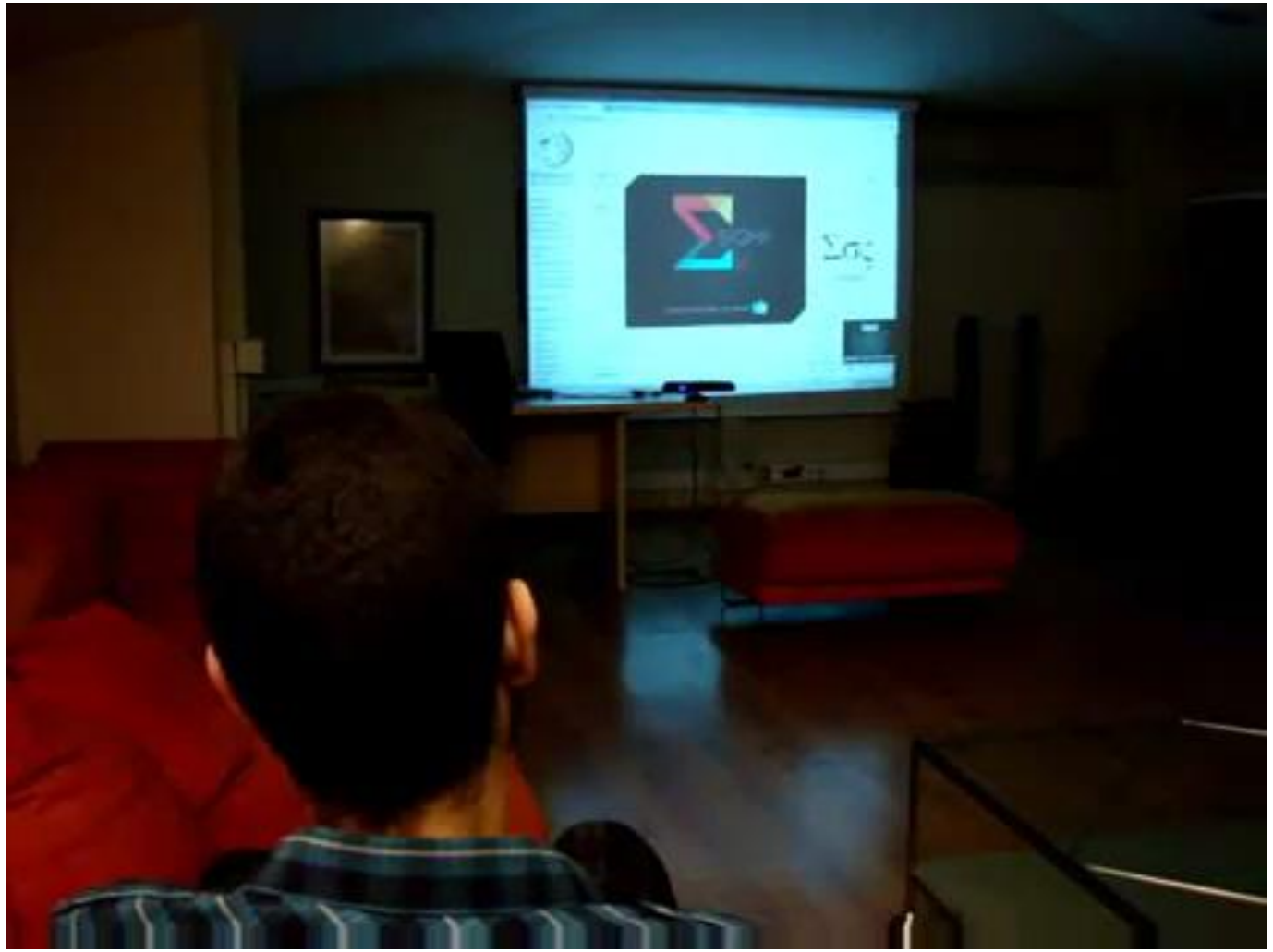
Video

[Saponas et al., *Proceedings of ACM UIST '09*]

Touch & Activate: Video

[Ono et al., *Proceedings of ACM UIST '13*]

Gesture Recognition



http://www.sigmard.com/portfolio/sigmanil-framework/

Take Home Message

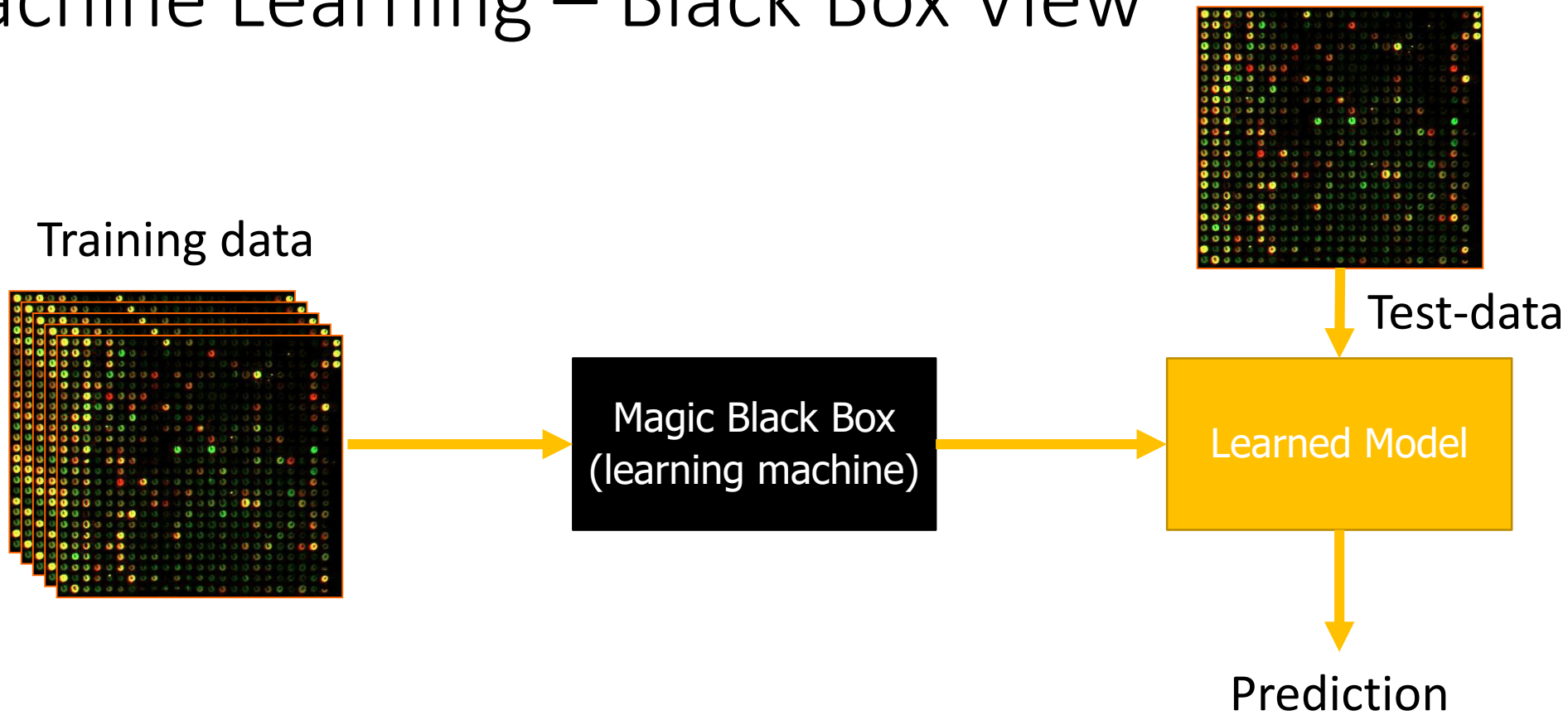
Humans use a variety of channels to interact & communicate

- Human behavior is highly complex
- Human behavior is highly ambiguous
- Too create we intuitive interfaces we want to use:
  - Small, low-effort natural motion
  - Avoid heavy user instrumentation

In consequence,

- Recognizing human input is challenging
- Most heuristic / traditional approaches are not powerful enough to model the variance in (gesture) execution
- Many different sensor modalities, often noisy data

Machine Learning – Black Box View



Training data:

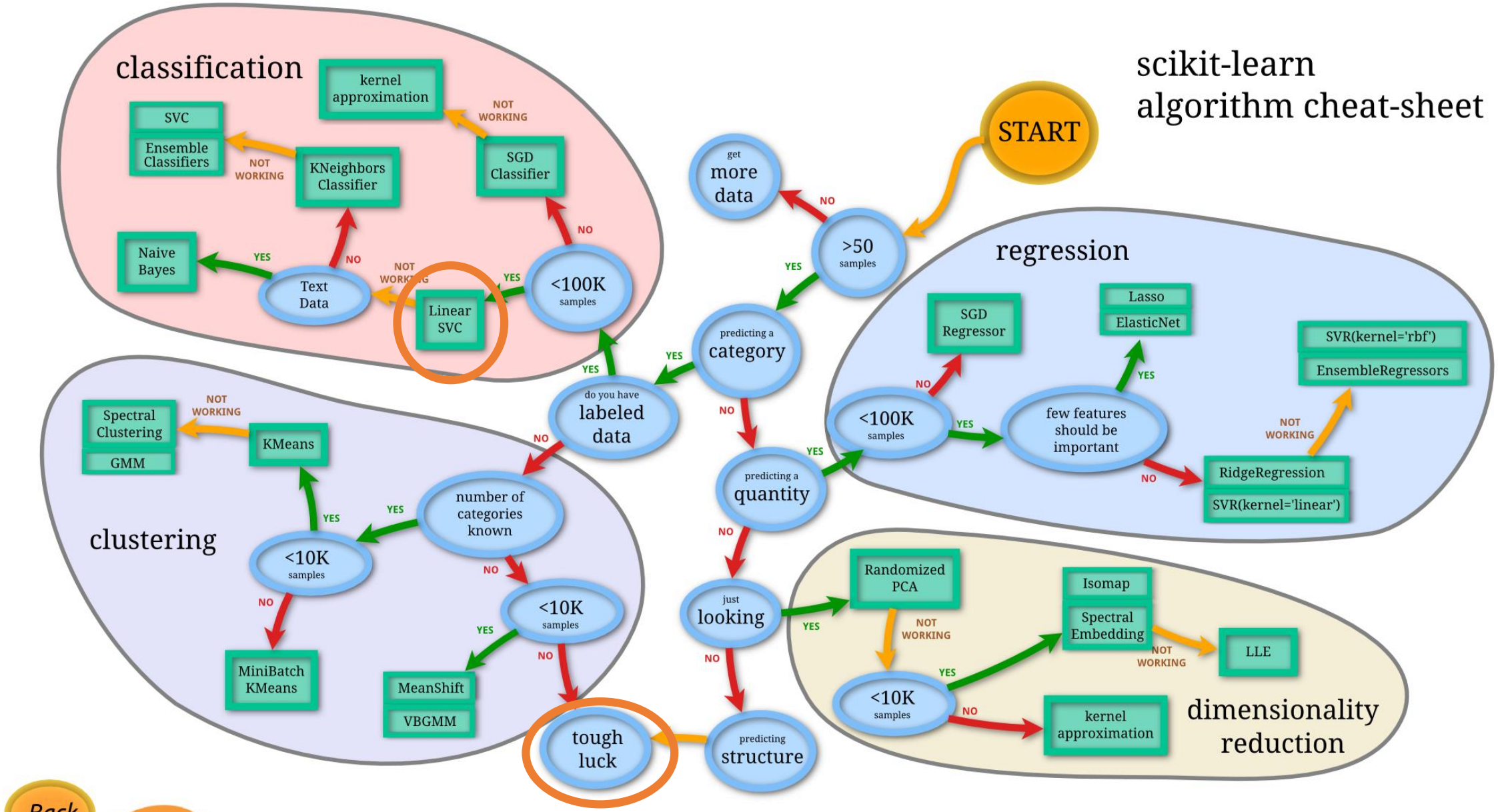
-Positive *and* negative examples. Data has to be labelled (ground truth).

Model:

- The model can distinguish between classes or estimate real valued outcome.

Categorization of Machine Learning Tasks (+HCI examples)

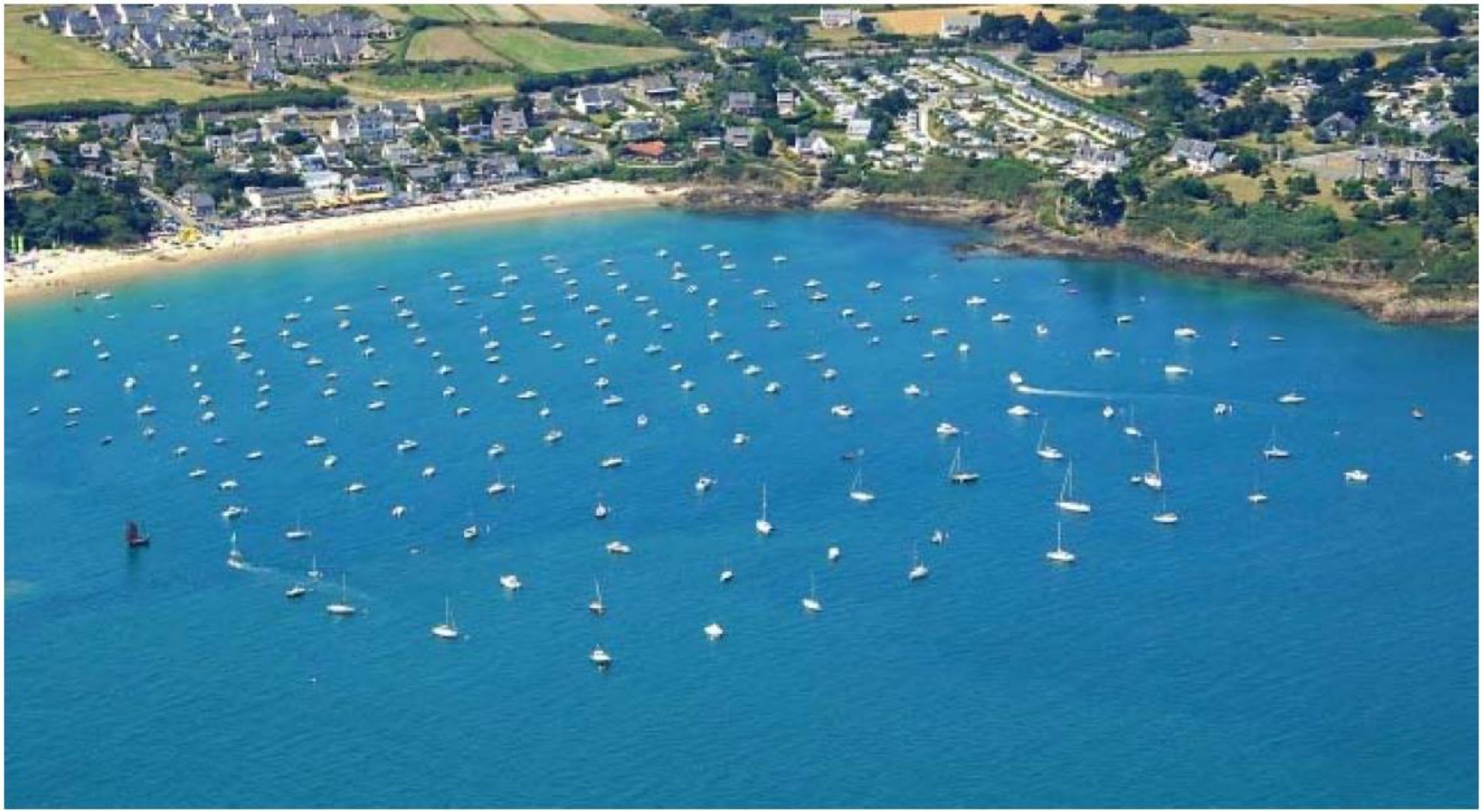
- ***Classification:*** inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more of these classes **(recognize gestures)**.
- ***Regression:*** outputs are continuous rather than discrete **(regress x,y,z positions from electro-magnetic field)**.
- ***Clustering:*** divide a set of inputs into (a priori unknown groups). **(Find similarity in users in some data domain)**
- ***Dimensionality reduction:*** simplifies inputs by mapping them into a lower-dimensional space. **(Visualize high-dimensional data for human consumption)**



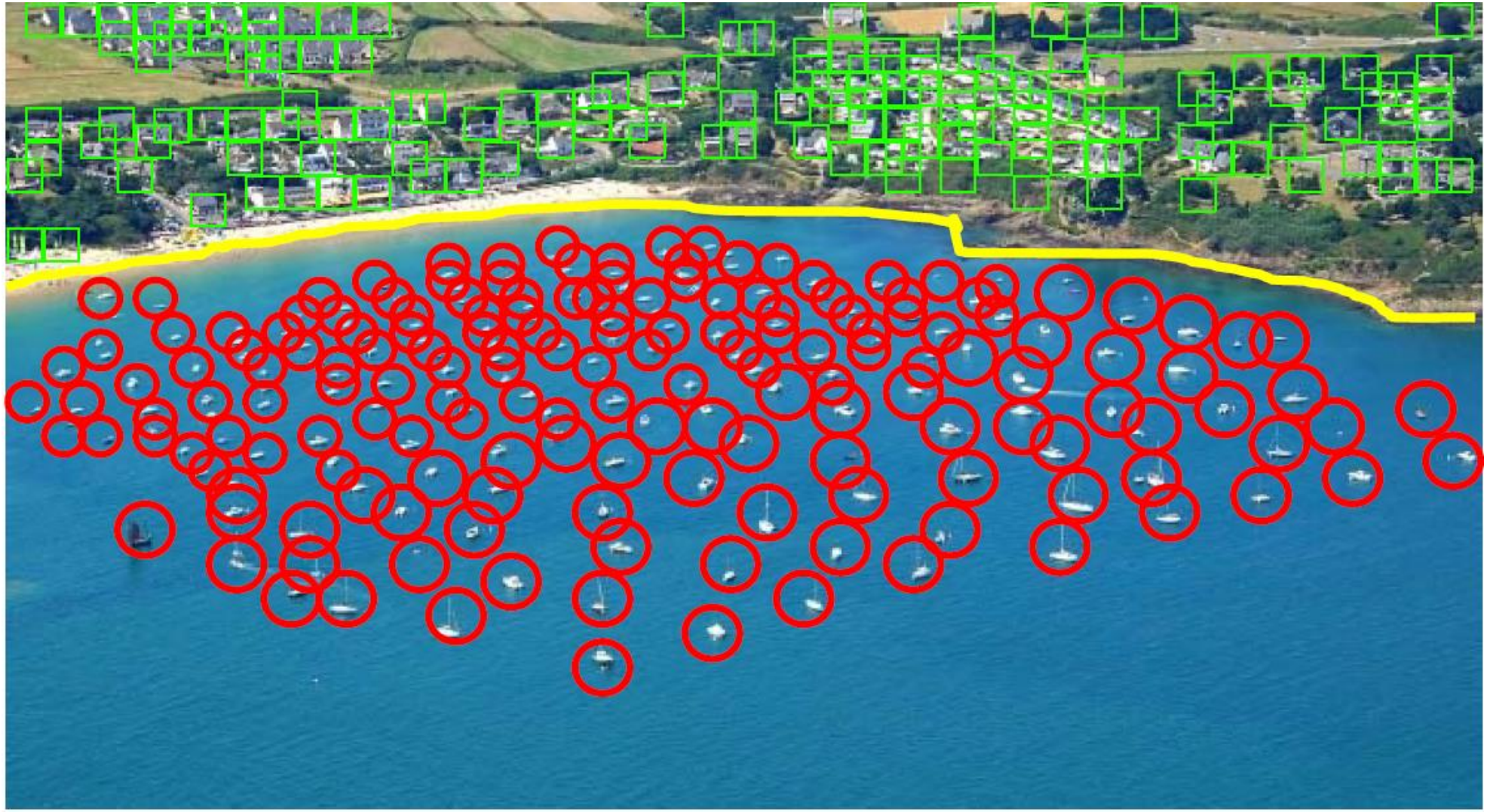
[http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html]

Let’s get our hands dirty!

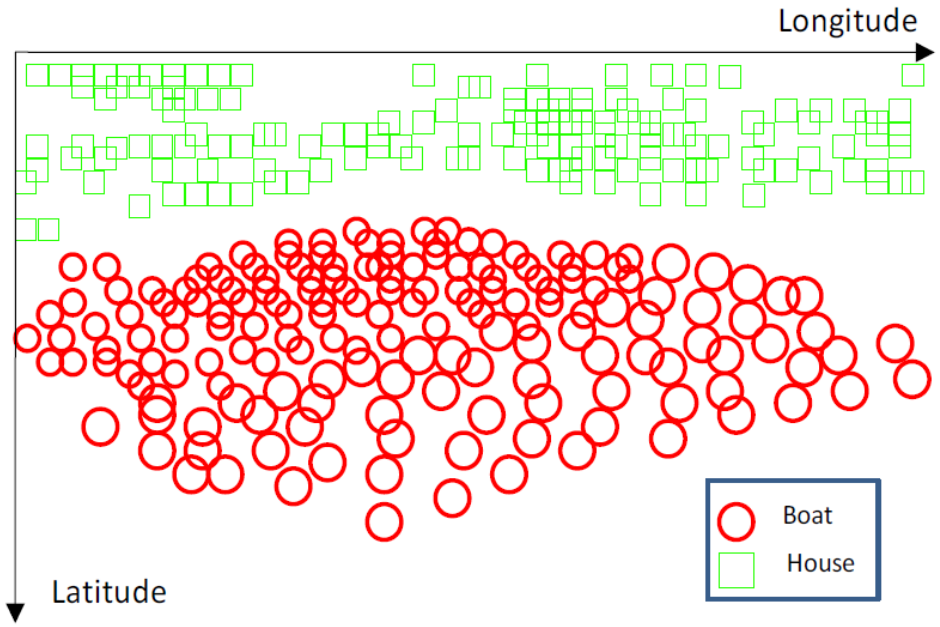
Classification: The Task

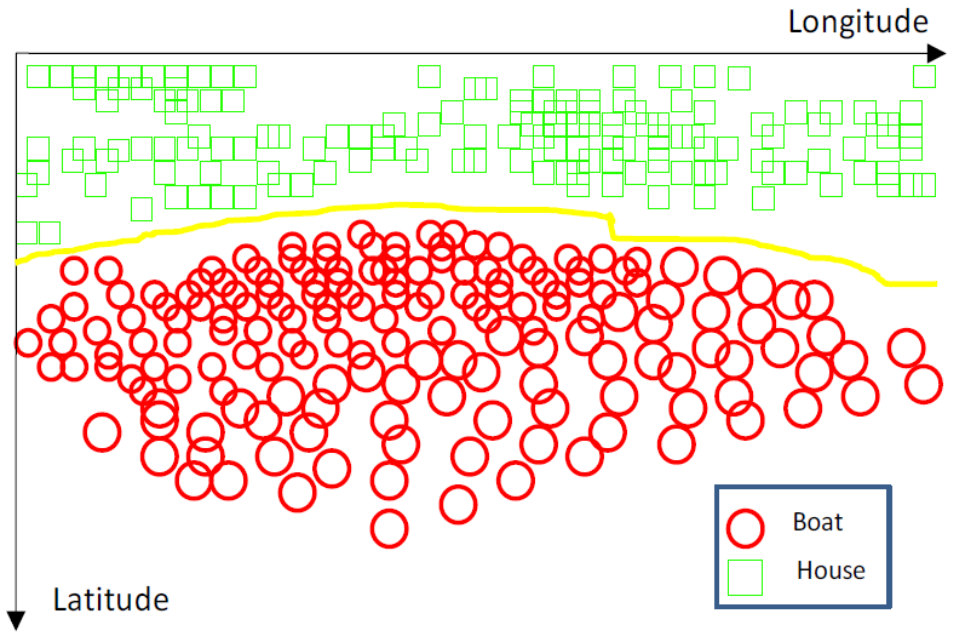


Classification: Decision Surface

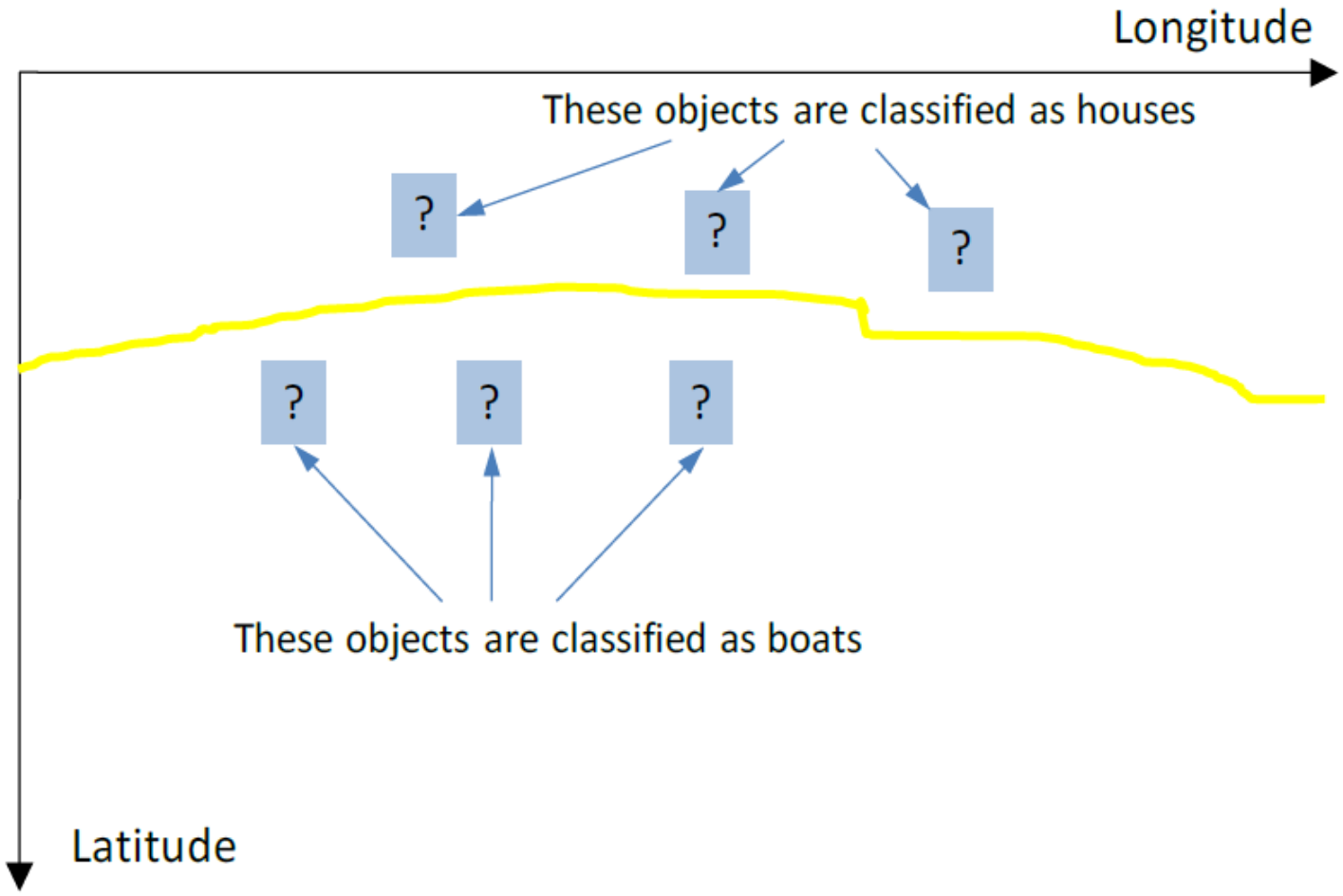


Classification Algorithms – Training time

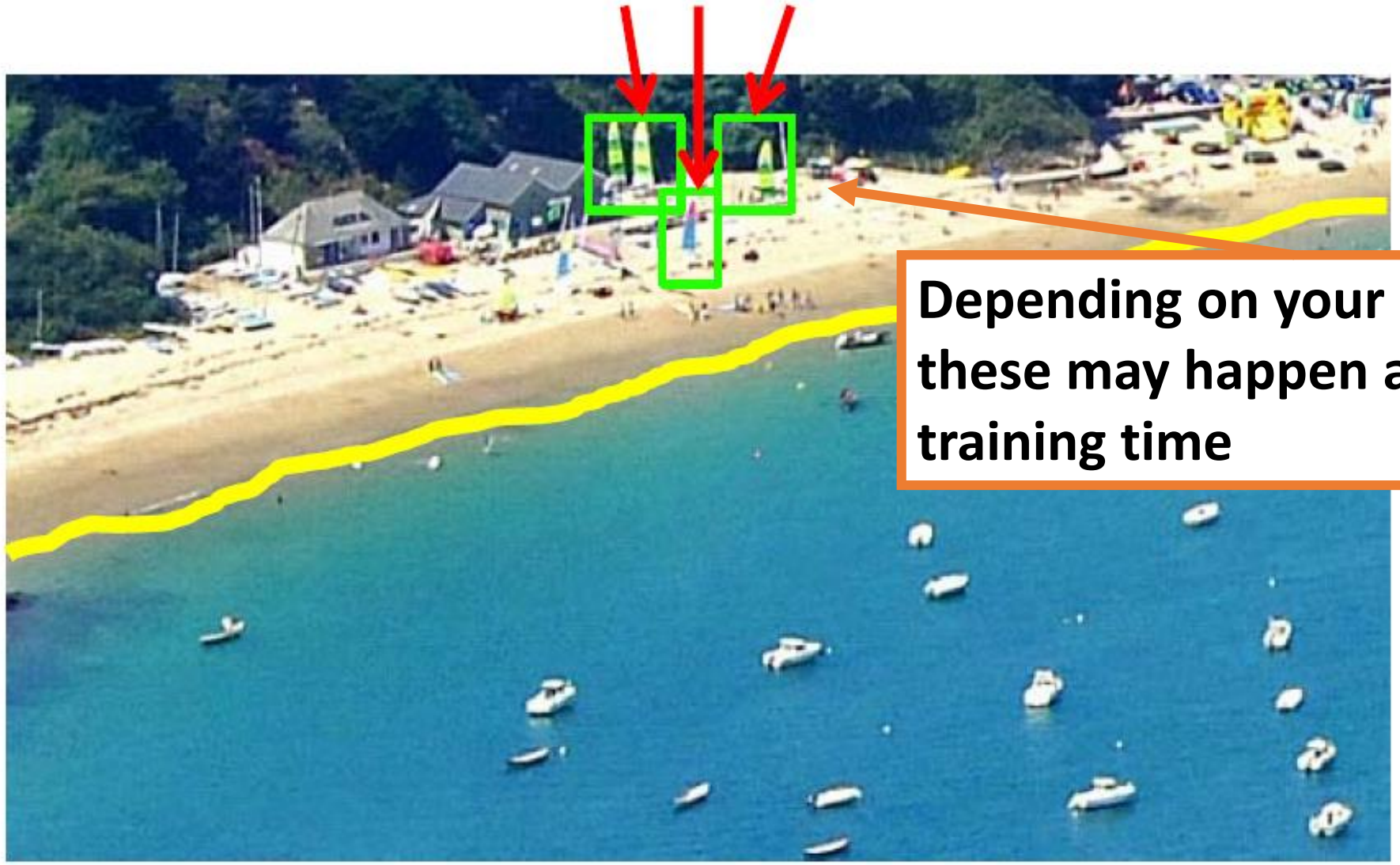




Classification Algorithms – Test time



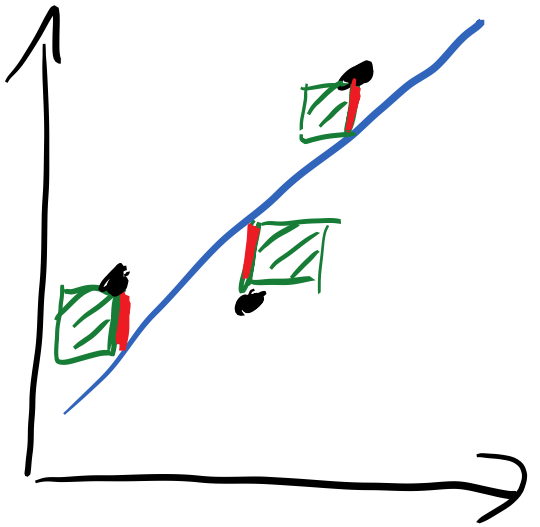
Classification: Errors



Depending on your algorithm, these may happen at test *and* training time

Supervised learning - Ingredients

- **Training data** $D = \{x_1, y_1, \ldots, x_n y_n\}$, where $\mathrm{y_i} = \begin{cases} 1 & \text{if } x \text{ positive} \\ -1 & \text{otherwise} \end{cases}$
- **Model** $g: X \rightarrow Y$ -> g is element of the hypothesis space G and has parameters Θ. We write its predictions as $g(x|\Theta)$
- **Loss function** $E(\Theta|X) = \sum_i L(y_i, g(x_i|\Theta))$
- **Optimization procedure** $\Theta^* = \underset{\Theta}{\operatorname{argmin}}\, E(\Theta|X)$



Gradient Descent in One Picture

Update rule:

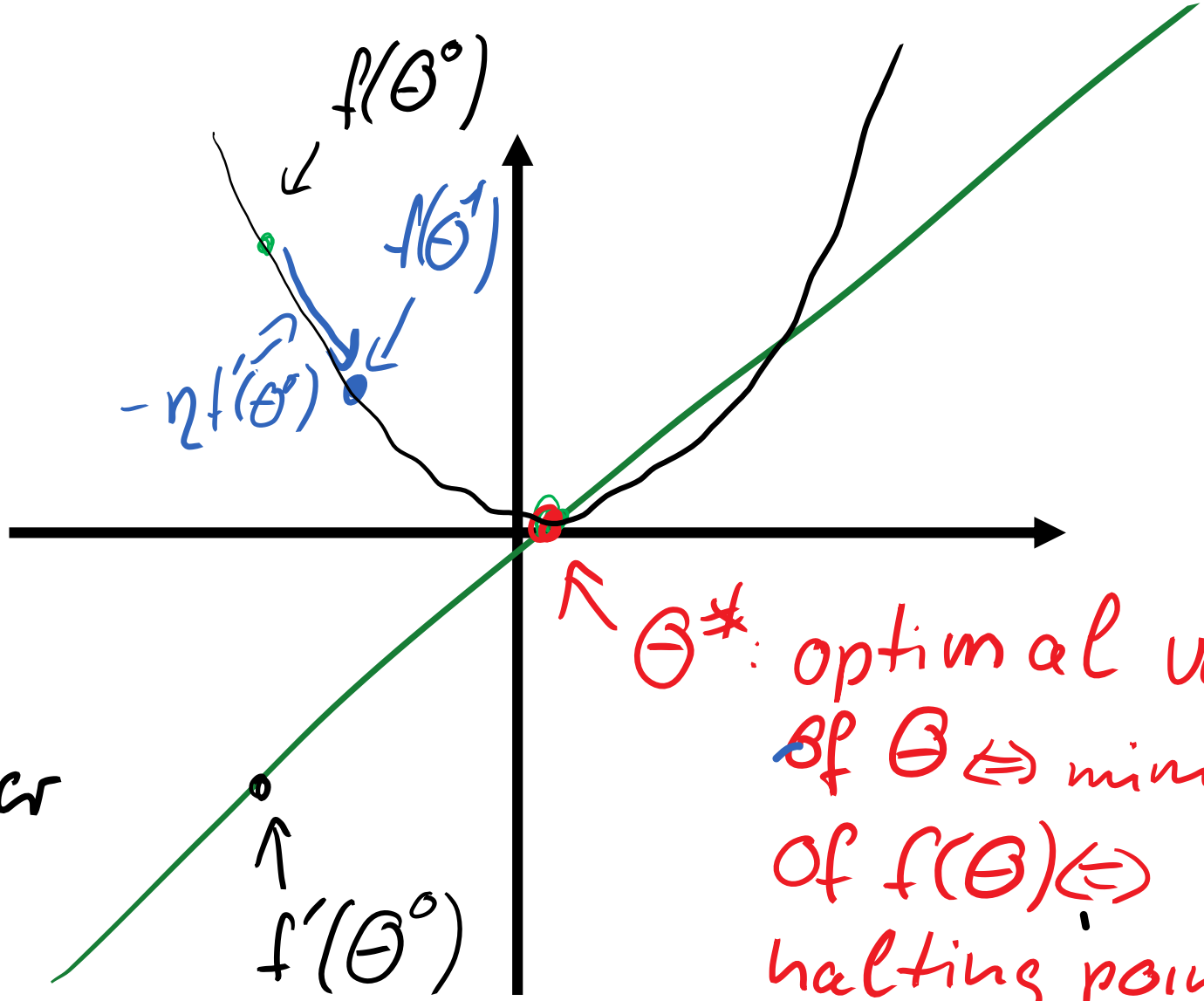
- randomly pick start value for parameter θ
- compute negative gradient at θ^0:

$$-f'(\theta^0)$$

- update parameter

$$\theta^1 = \theta^0 - \eta f'(\theta^0)$$

η: learning rate



Support Vector Machine (SVM)

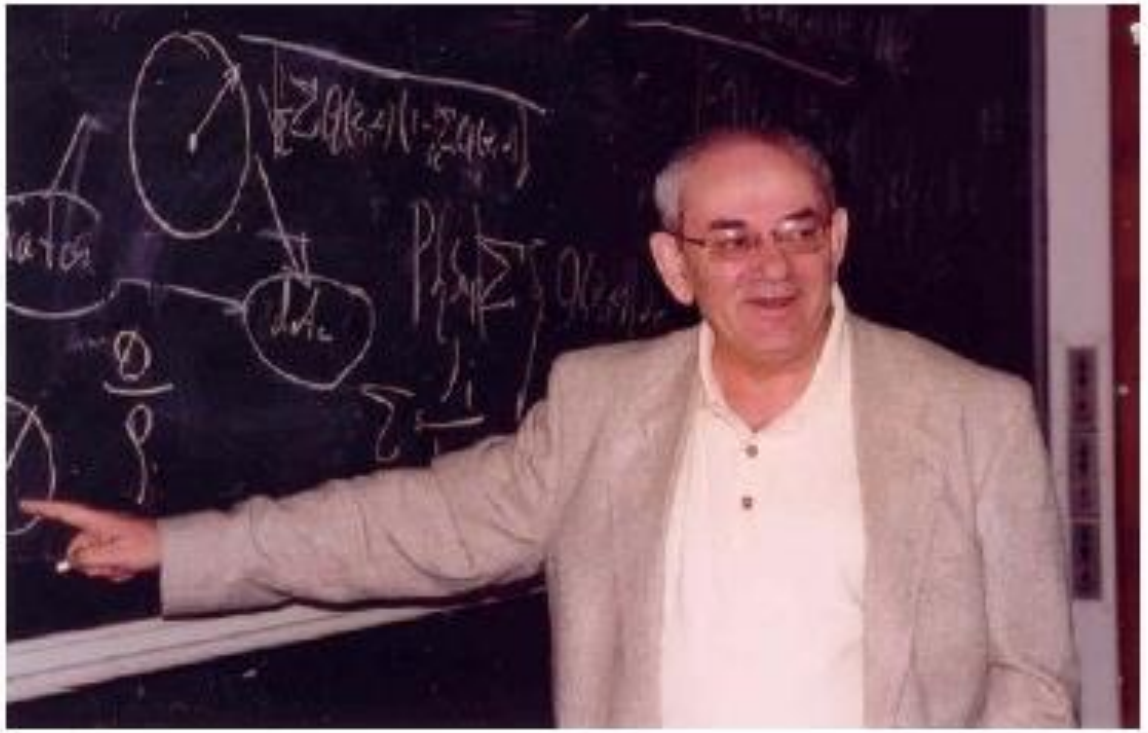
Derived from statistical learning theory by Vapnik, et al. in 1992

Simple SVM achieved accuracy comparable to neural-network with hand-designed features in a handwriting recognition task

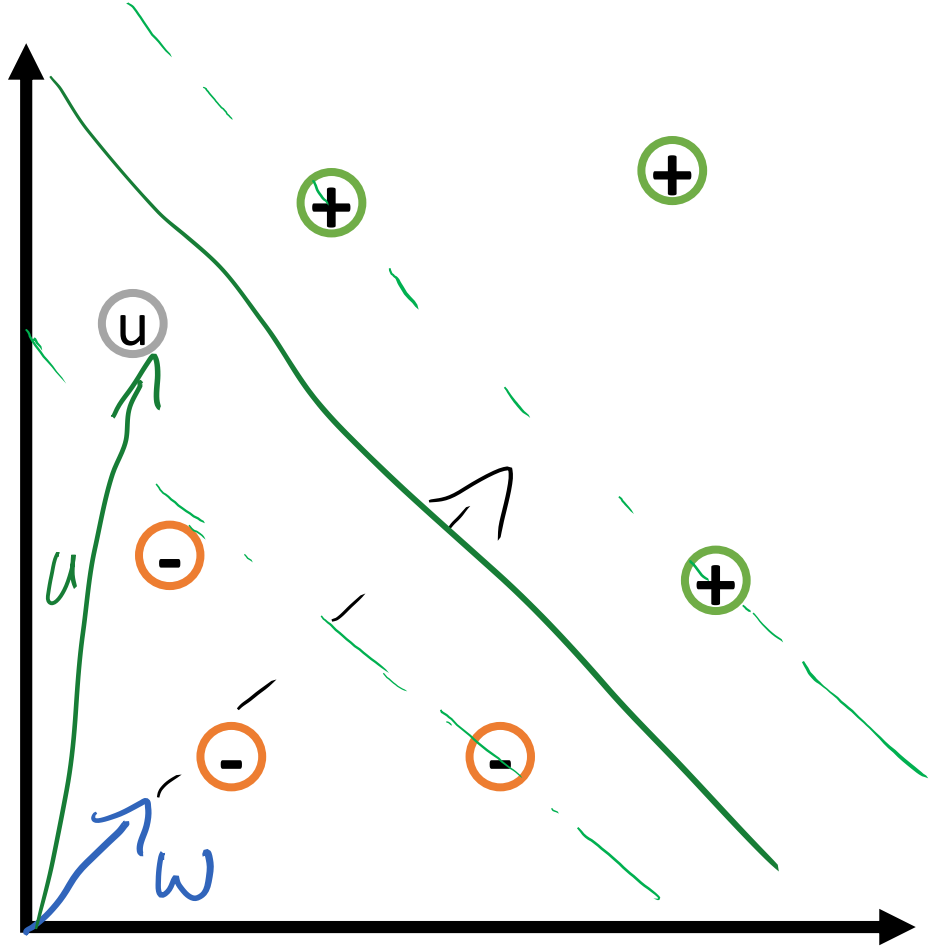
SVM widely used in object detection & recognition, content-based image retrieval, text recognition, biometrics, speech recognition, etc.

Also used for regression (will not cover today)

Vladimir Vapnik



SVM Derivation (Intuition with some slight of hand)



$\bar{w} \cdot u \geq c$

with $b = -c$

$$\bar{w} \cdot \bar{u} + b \geq 0$$

then (+)

how to find w & b?

SVM Derivation (Intuition with some slight of hand)

We need to introduce constraints:

$(\bar{w} \cdot \overline{x_+} + b) \geq 1$

$(\bar{w} \cdot \overline{x_-} + b) \leq -1$

$$y_i = \begin{cases} 1 & \text{if } x \text{ positive} \\ -1 & \text{otherwise} \end{cases}$$

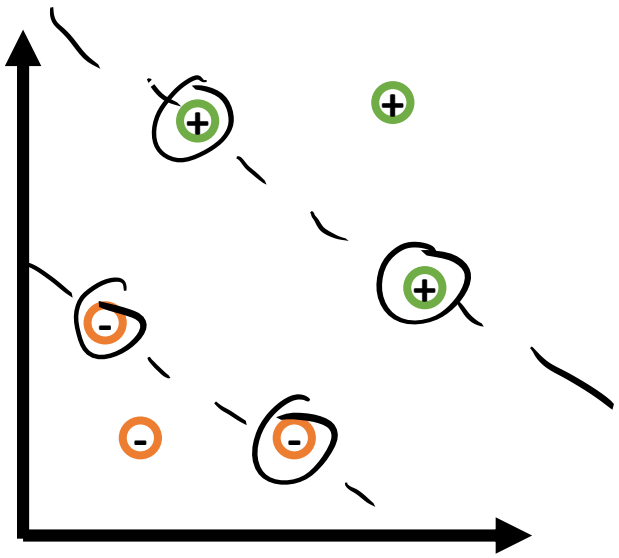
Multiplying with labels y_i gives:

$y_i(\bar{w} \cdot \tilde{x}_+ + b) \geq 1$

$y_i(\bar{w} \cdot \tilde{x}_- + b) \geq 1$

now we are going to ask that for points on the margin:

$y_i(\bar{w} \cdot \tilde{x}_i + b) - 1 \approx 0$

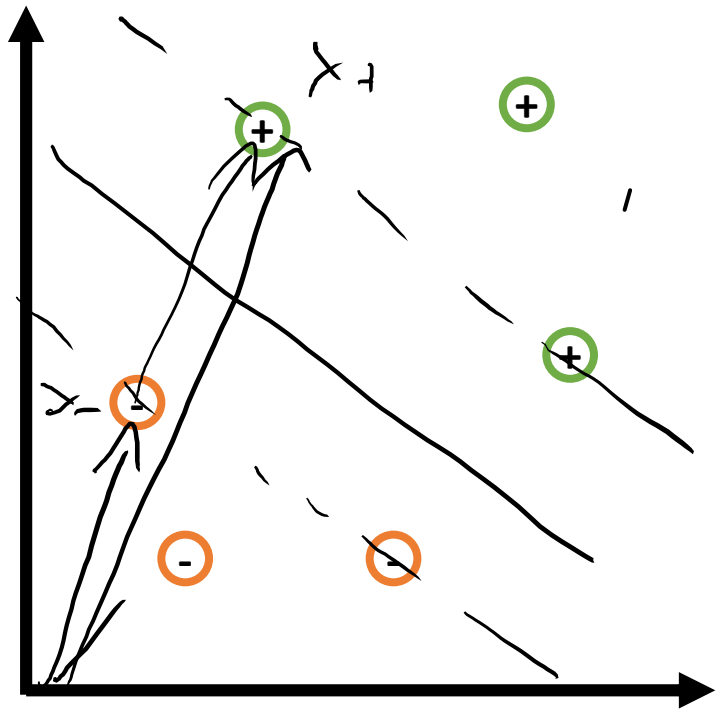


SVM Derivation (Intuition with some slight of hand)

$y_i(\bar{w} \cdot \bar{x}_i + b) - 1 = 0$

$$\text{width} = (x_+ - x_-) \cdot \frac{w}{\|w\|} = \frac{2}{\|w\|} \quad *$$

$$1 - b + 1 + b = 2$$



x_+: $1(w \cdot x_+ + b) - 1 = 0$

$$w \cdot x_+ = 1 - b$$

x_-: $-1(w * x_- + b) - 1 = 0$

$$w * x_- = -1 - b$$

$$\max \frac{2}{\|w\|} = \max \frac{1}{\|w\|} = \min \|w\|$$

$$\approx \boxed{\frac{1}{2}\|w\|^2}$$ primal problem

Now Some Practical Issues (iPython Notebook)

SVM Derivation (Intuition with some slight of hand)

$y_i(\bar{w} \cdot \bar{x_i} + b) - 1 = 0$

$\min \frac{1}{2}\|w\|$

// dual problem

$$L = \frac{1}{2}\|\bar{\omega}\|^2 - \sum \alpha_i \left[y_i(\bar{\omega} \cdot \bar{x}_i + b) - 1 \right]$$

$$\frac{\partial L}{\partial \omega} = \bar{\omega} - \sum \alpha_i y_i \bar{x}_i = 0 \Rightarrow \boxed{\bar{\omega} = \sum_i \alpha_i y_i \bar{x}_i}$$

linear sum of samples $\bar{x}_i$

$$\frac{\partial L}{\partial b} = -\sum \alpha_i y_i = 0 \Rightarrow \boxed{\sum \alpha_i y_i = 0}$$

$$L = \frac{1}{2}\left(\sum \alpha_i y_i \bar{x}_i\right)\left(\sum \alpha_j y_j \bar{x}_j\right) - \left(\sum \alpha_i y_i \bar{x}_i\right)\left(\sum \alpha_j y_j \bar{x}_j\right) - \underbrace{\sum \alpha_i y_i}_{=0} b + \sum \alpha_i$$

$$= \sum \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \bar{x}_i \cdot \bar{x}_j$$

SVM Derivation (Intuition with some slight of hand)

$$L = \sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \boxed{\bar{x}_i \cdot \bar{x}_j}$$

$\alpha_i \alpha_j$: output

$y_i y_j$: coeffs from training data

$\bar{x}_i \cdot \bar{x}_j$: optimization only depends on dot product of sample vecs

DECISION RULE:

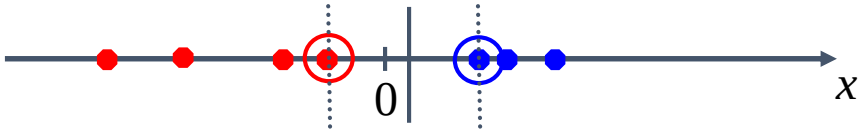
$$\sum \alpha_i y_i \boxed{\bar{x}_i \cdot \bar{u}} + b \geq 0 \quad \text{THEN } +$$

decision also only depends on dot product

Non-linear SVMs

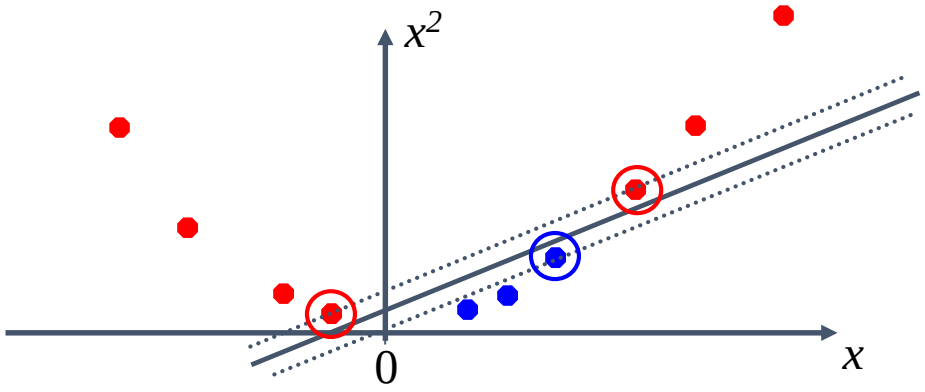
Non-linear SVMs

Datasets that are linearly separable (with some noise) work out great:



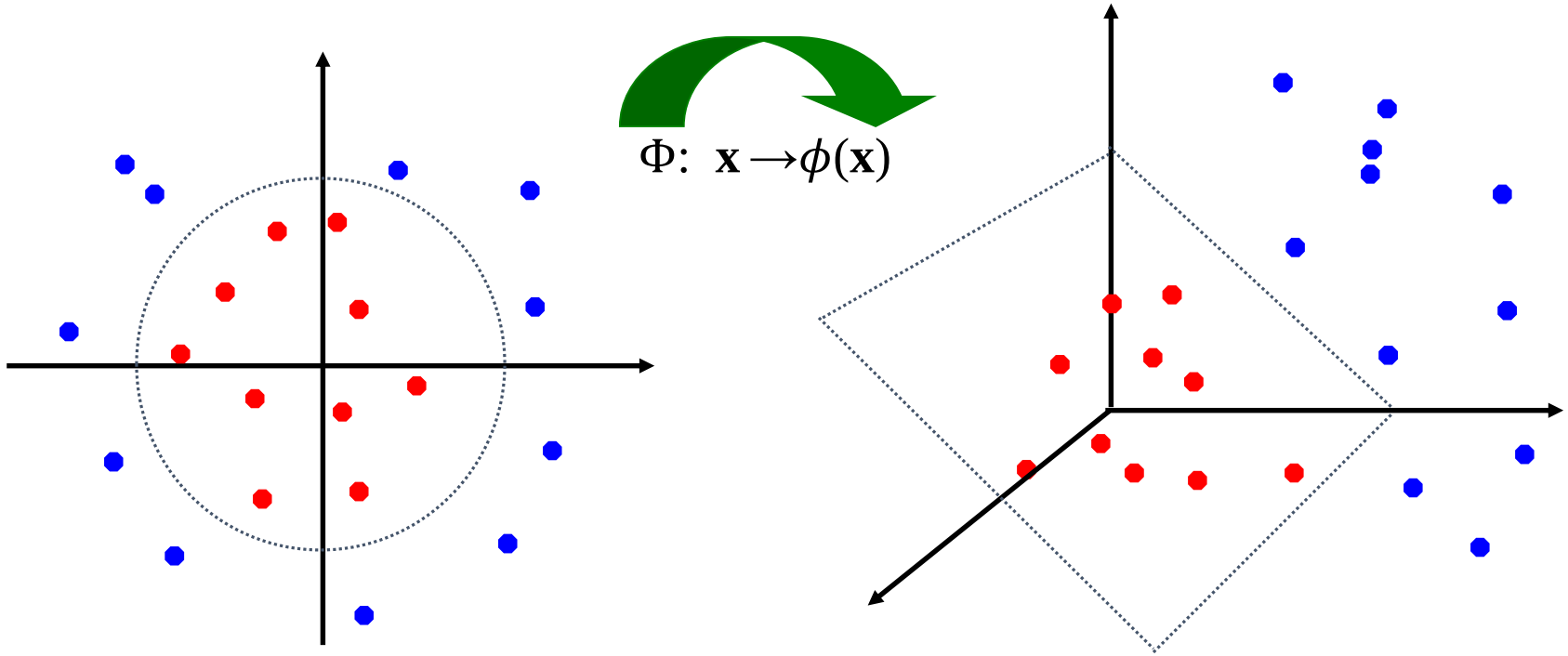
But what are we going to do if the dataset is just too hard?

How about … mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is linearly separable:



Non-linear SVMs: The Kernel Trick

With this mapping, our discriminant function is now:

$$y(x) = w^T \phi(x) + b = \sum_{i \in SV} \alpha_i \phi(x_i)^T \phi(x) + b$$

No need to know mapping function ϕ explicitly, because we only use the dot product of feature vectors in both training and test.

A *kernel function* is defined as a function that corresponds to a inner product of two feature vectors in some expanded feature space:

$$K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

Non-linear SVMs: The Kernel Trick

Commonly used kernel functions:

- Linear kernel: $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \boldsymbol{x_i}^T \boldsymbol{x_j}$
- Polynomial kernel: $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = (1 + \boldsymbol{x_i}^T \boldsymbol{x_j})^p$
- Gaussian (Radial-Basis Function Kernel (RBF)) kernel:

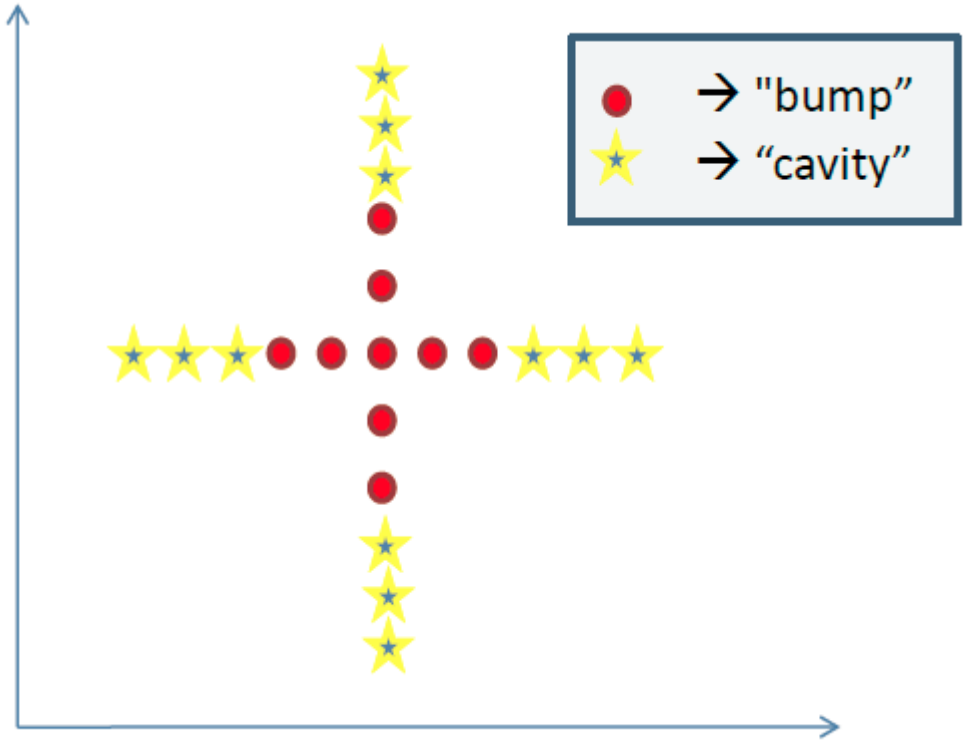
$$K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$$

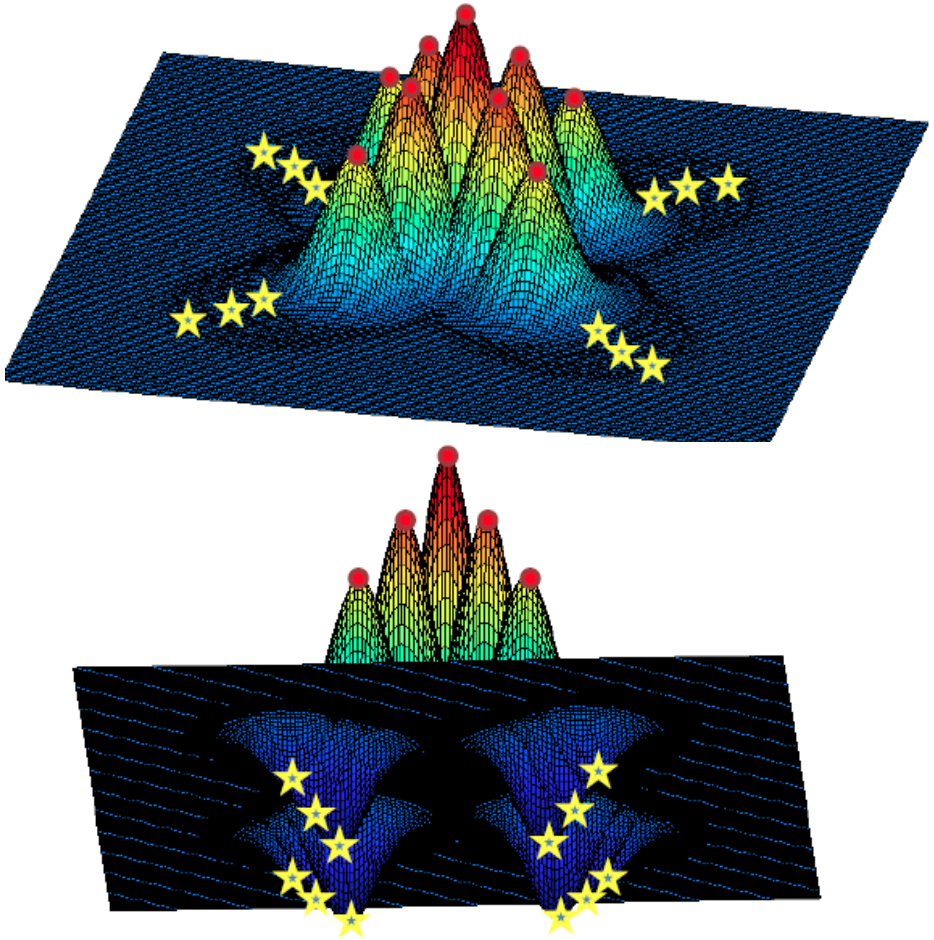
- Sigmoid: $K(x_i, x_j) = \tanh(\beta_0 x_i^T + \beta_1)$

In general, functions that satisfy *Mercer's condition* can be kernel functions.

Example: Gaussian Kernel

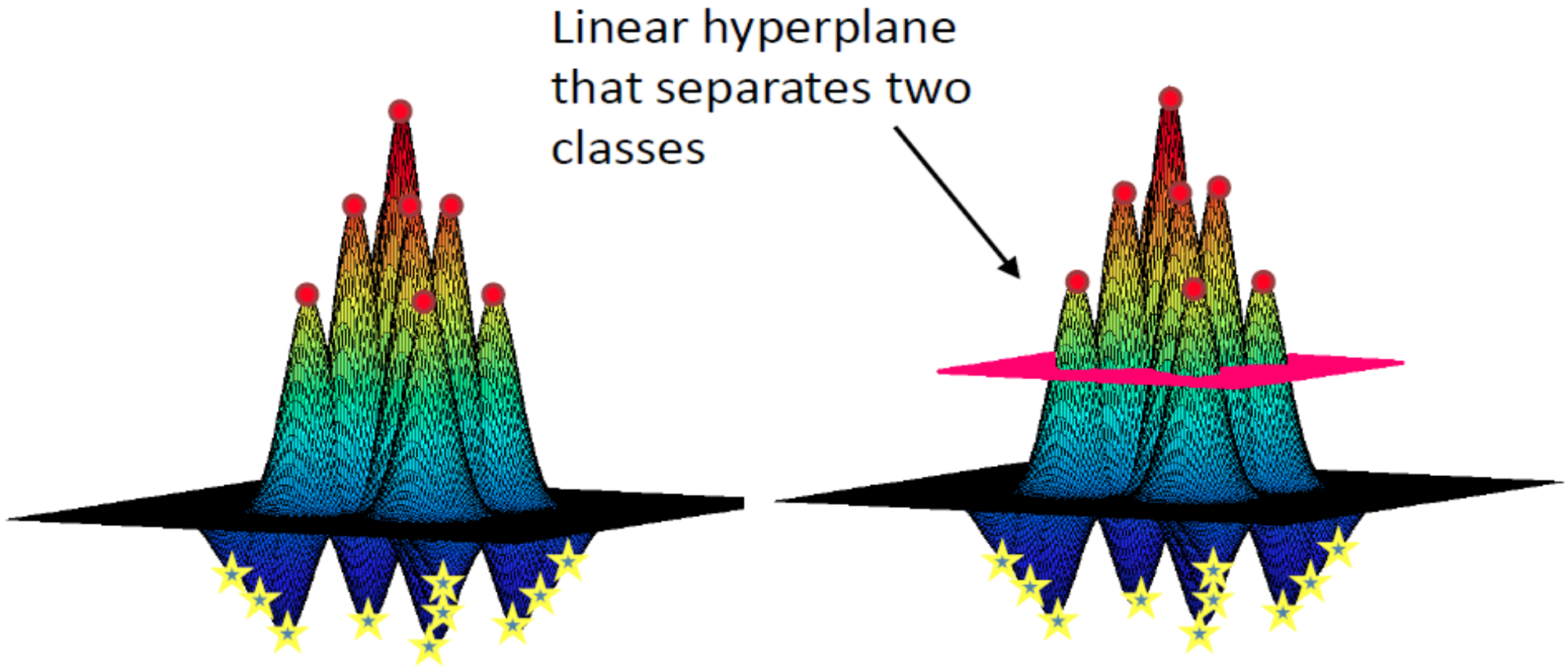
Geometrically, this is a “bump” or “cavity” centered at the training data point x:





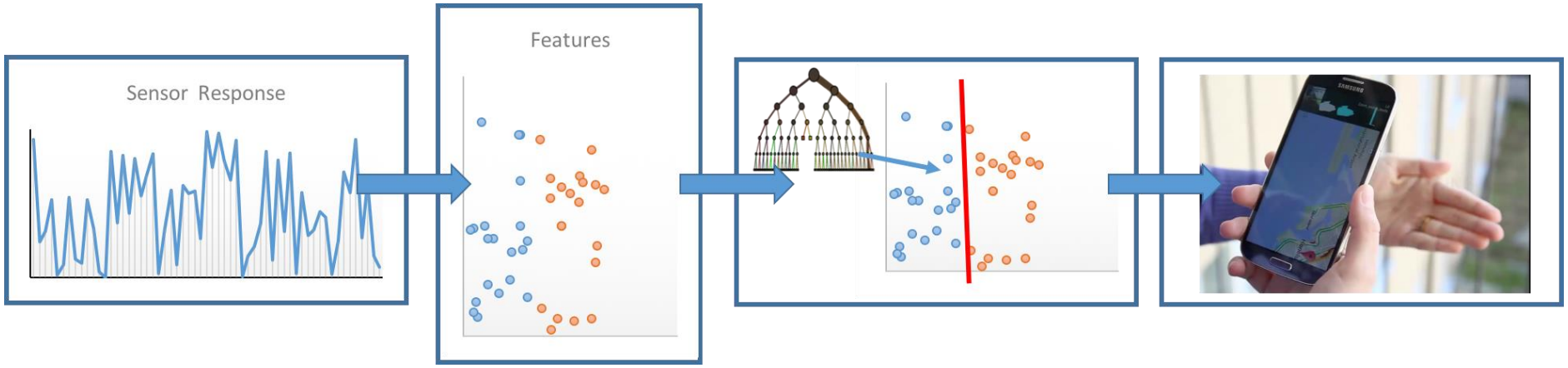
Example: Gaussian Kernel - Classification

Geometrically, this is a “bump” or “cavity” centered at the training data point x:



General Approach to supervised learning

1) Collect and label data.
2) Extract features.
3) Load the dataset.
4) Summarize the dataset.
5) Visualize the dataset.
6) Evaluate some ML algorithms.
7) Make predictions.
8) Integrate best performing model into your pipeline



Typical Issues in Computational Input Recognition

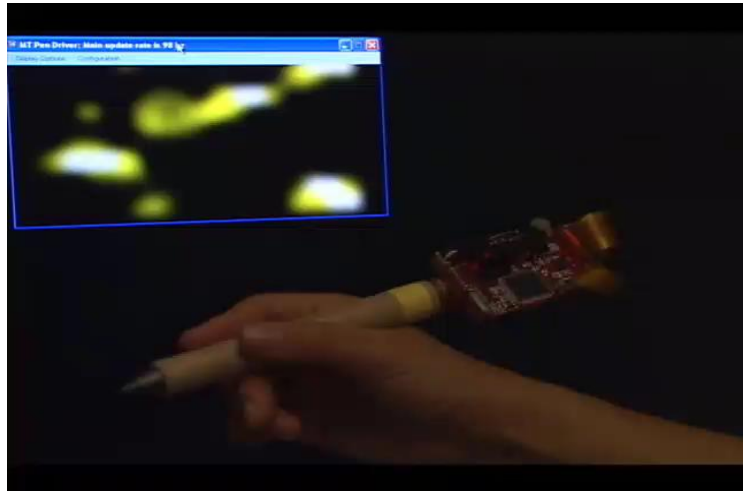
- Data is sometimes extremely **noisy**
- A lot of **variance** between subjects
- Collecting **data** is **expensive**
- Need models that can be evaluated at **interactive rates**
- **Feature engineering** can be an “art”
- ***Spotting*** versus ***recognition***

Outlook

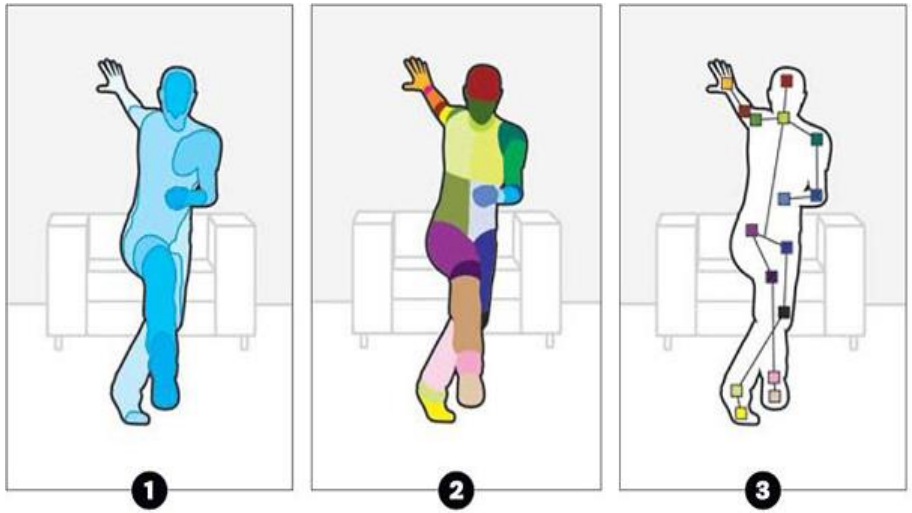
Traditional Machine Learning Pipeline

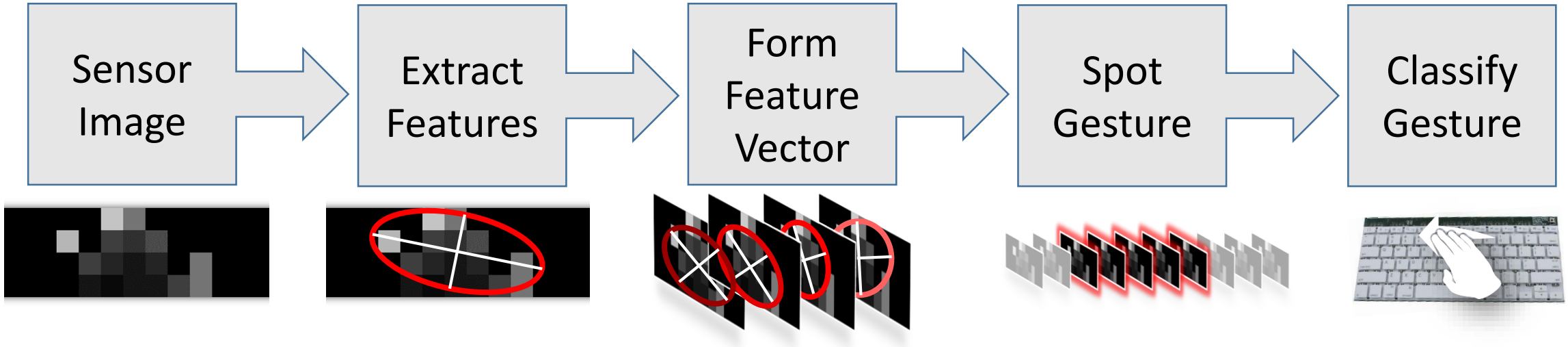
[Zhang *et al. CHI'14*]

[Song *et al. CHI'11*]

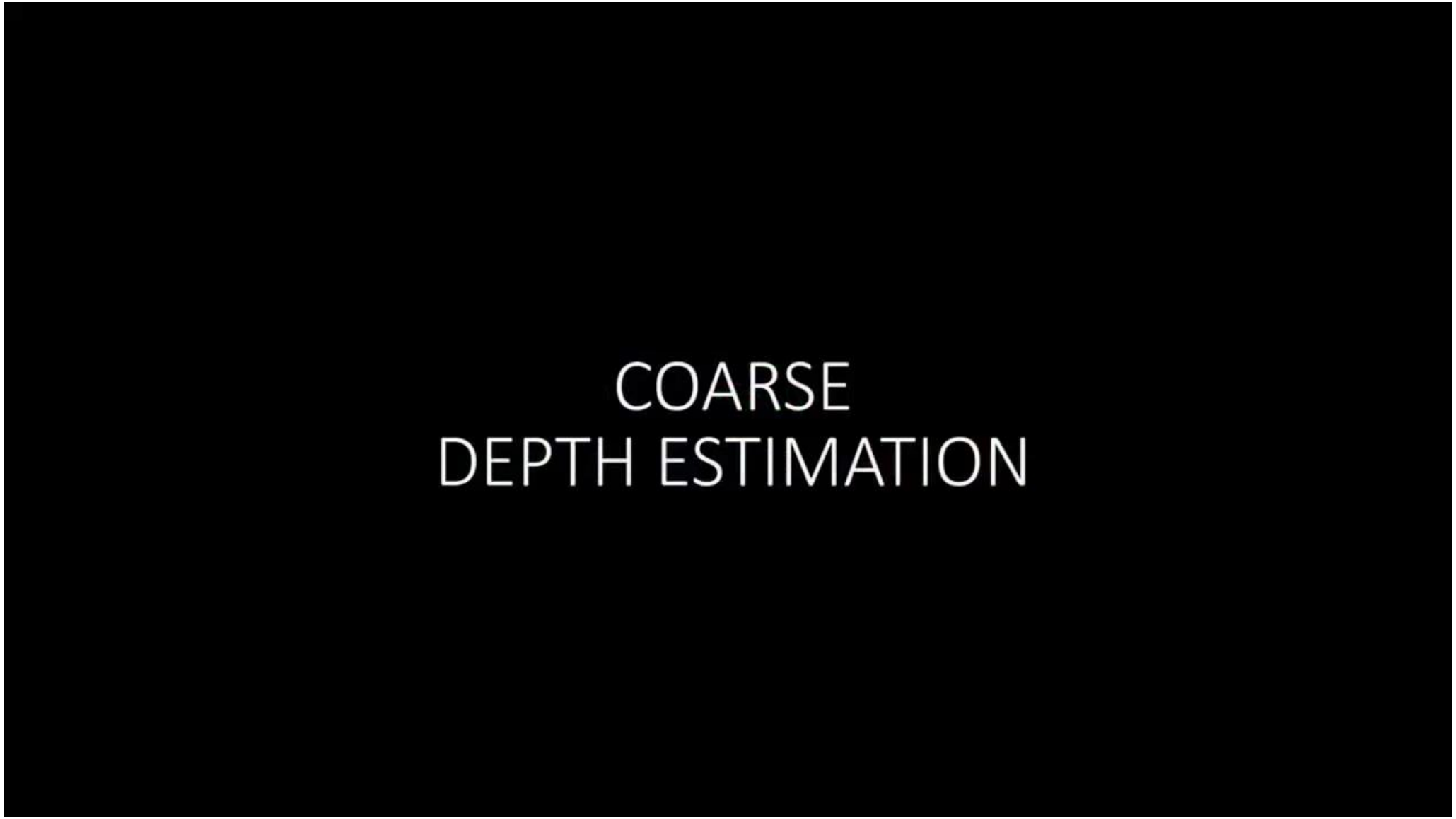


[Nowozin *et al.*, *MSR TR '11*]





Feature (parameter) learning: Random Forests



Representation learning

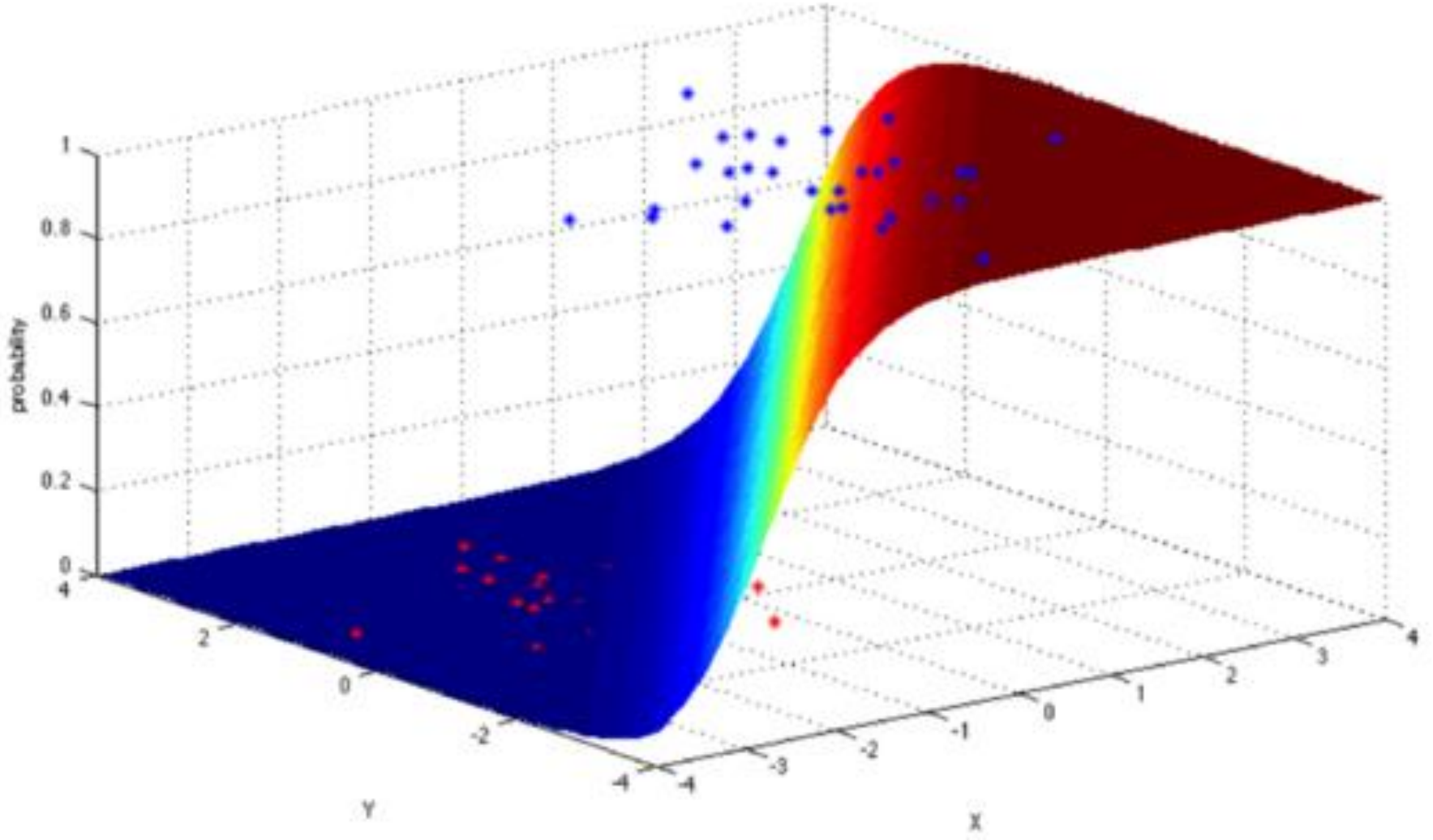
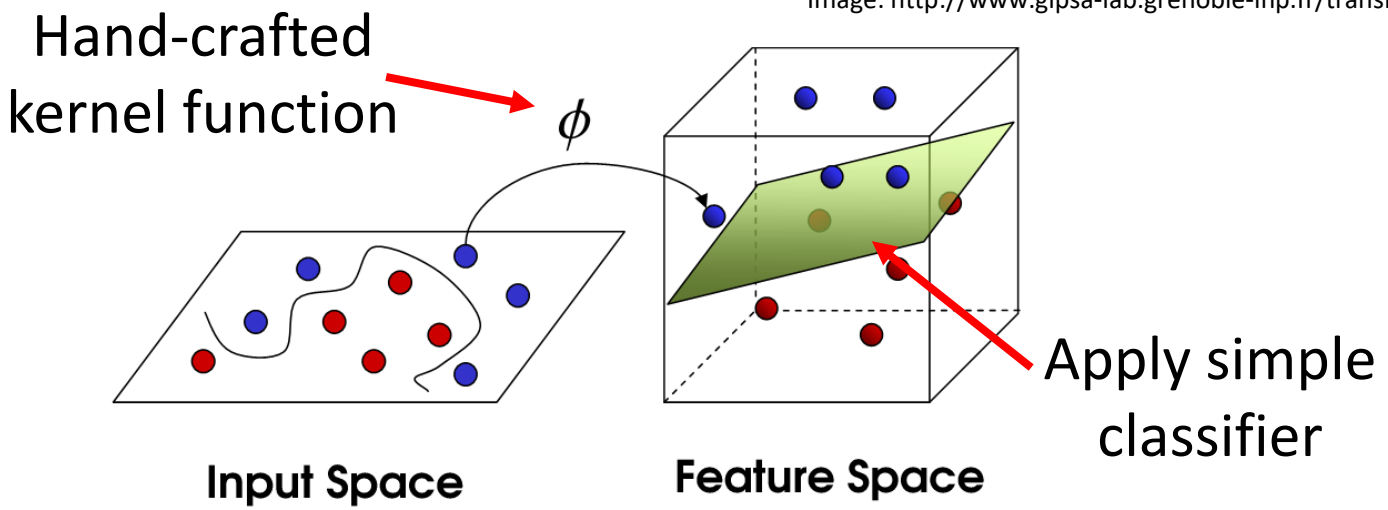
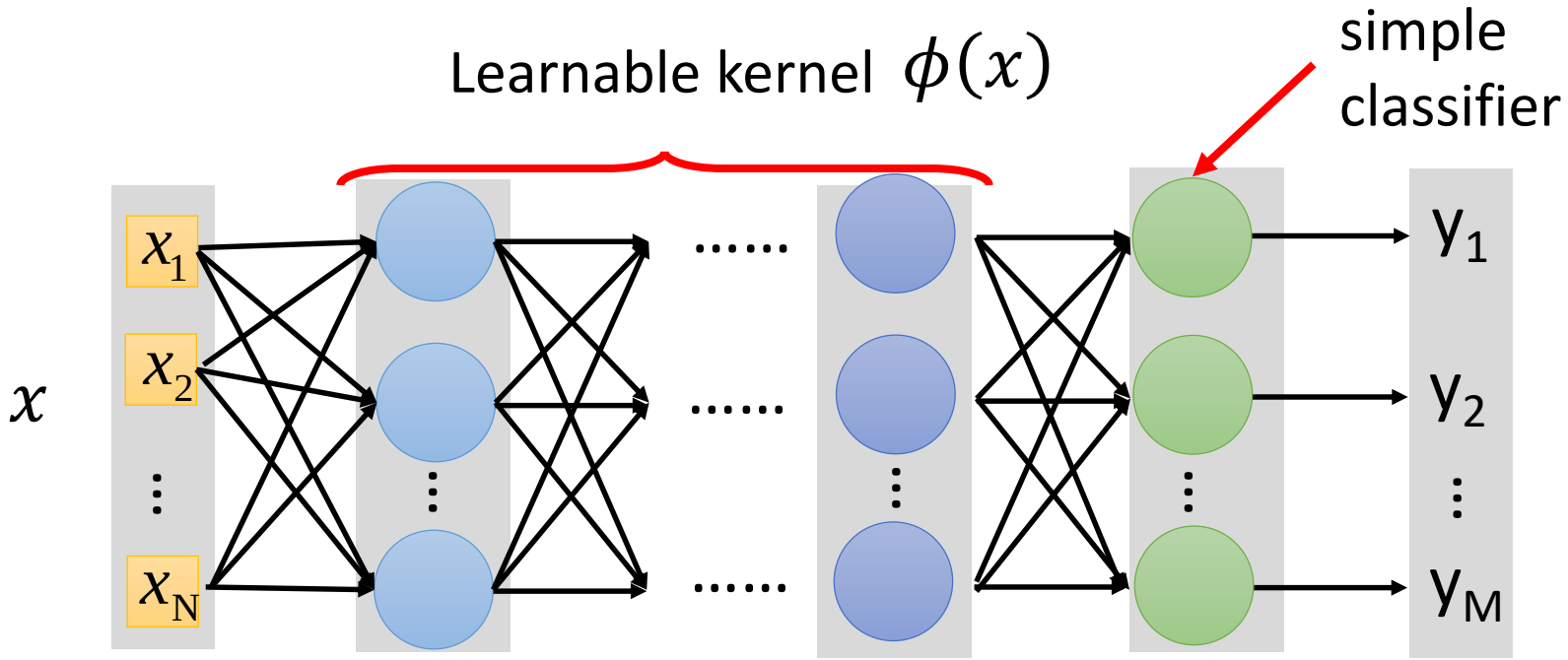


Image: http://www.gipsa-lab.grenoble-inp.fr/transfert/seminaire/455_Kadri2013Gipsa-lab.pdf

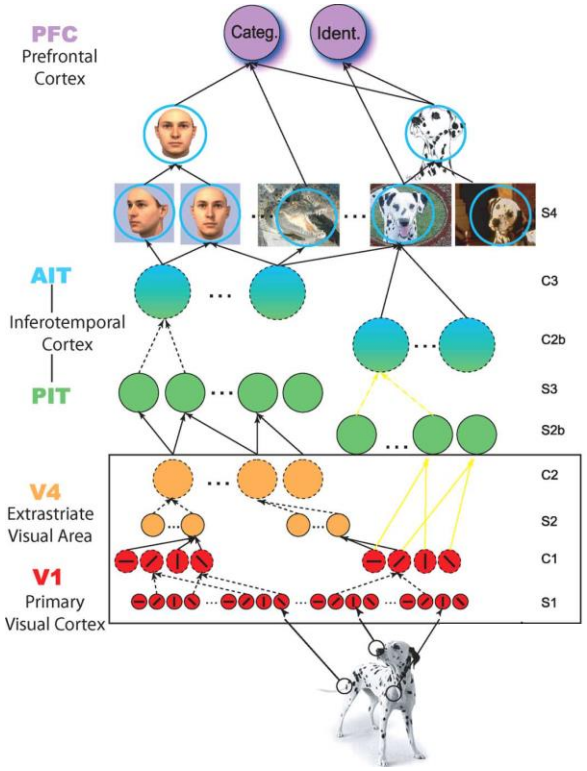
Traditional ML



Deep Learning

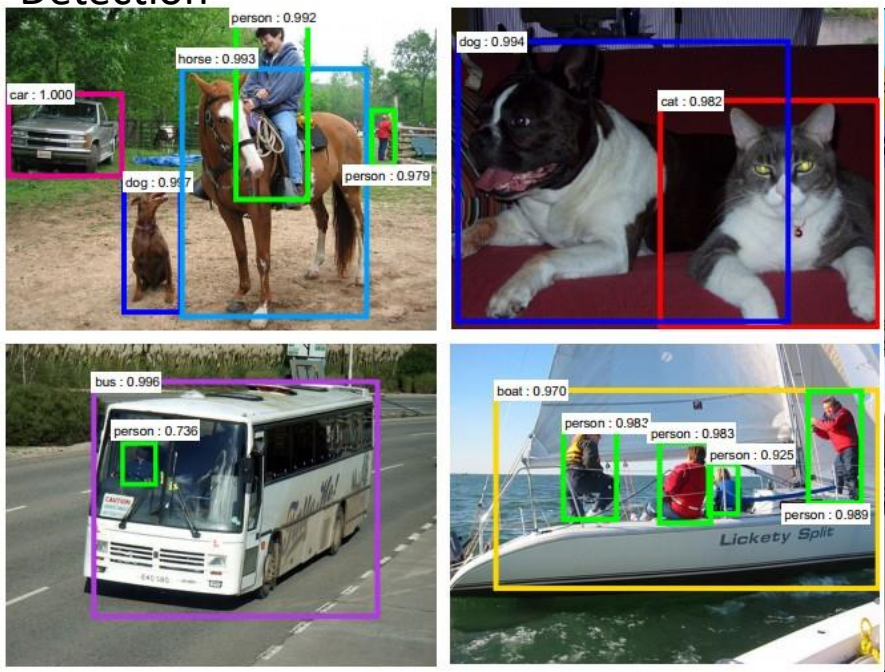


Deep-learning



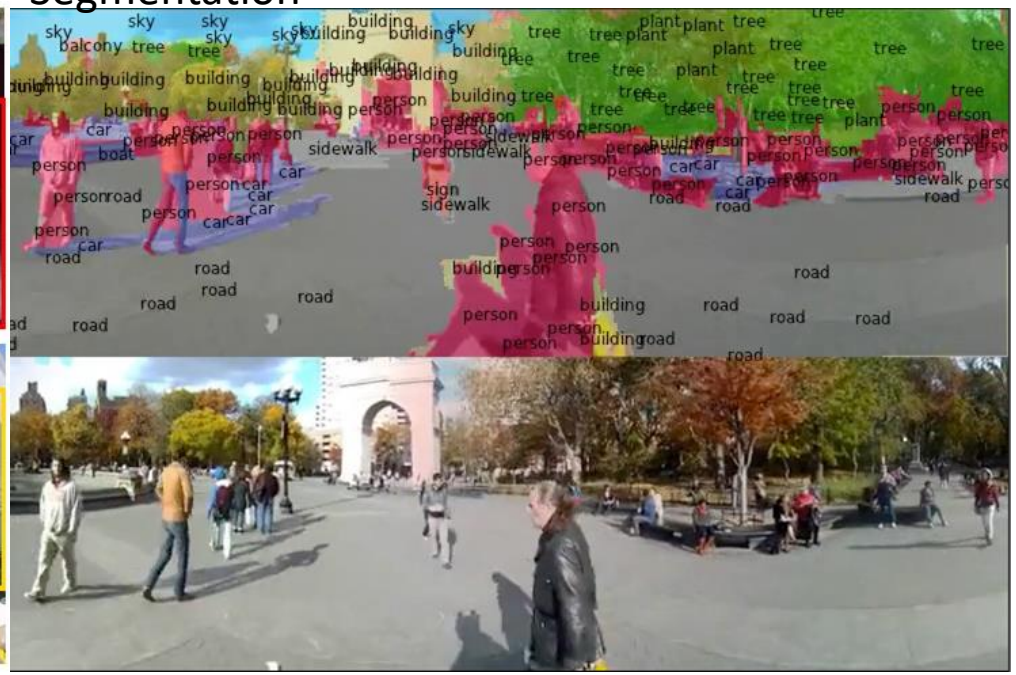
[Serre et al. PNAS 2007]

Detection



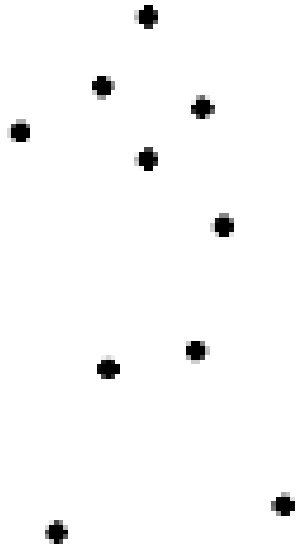
[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation



[Farabet et al., 2012]

What about motion?



G. Johansson, “Visual Perception of Biological Motion and a Model For Its Analysis", *Perception and Psychophysics 14, 201-211, 1973.*