Intro to ML for Input Recognition

CHI Course Computational Interaction 2016

Otmar Hilliges

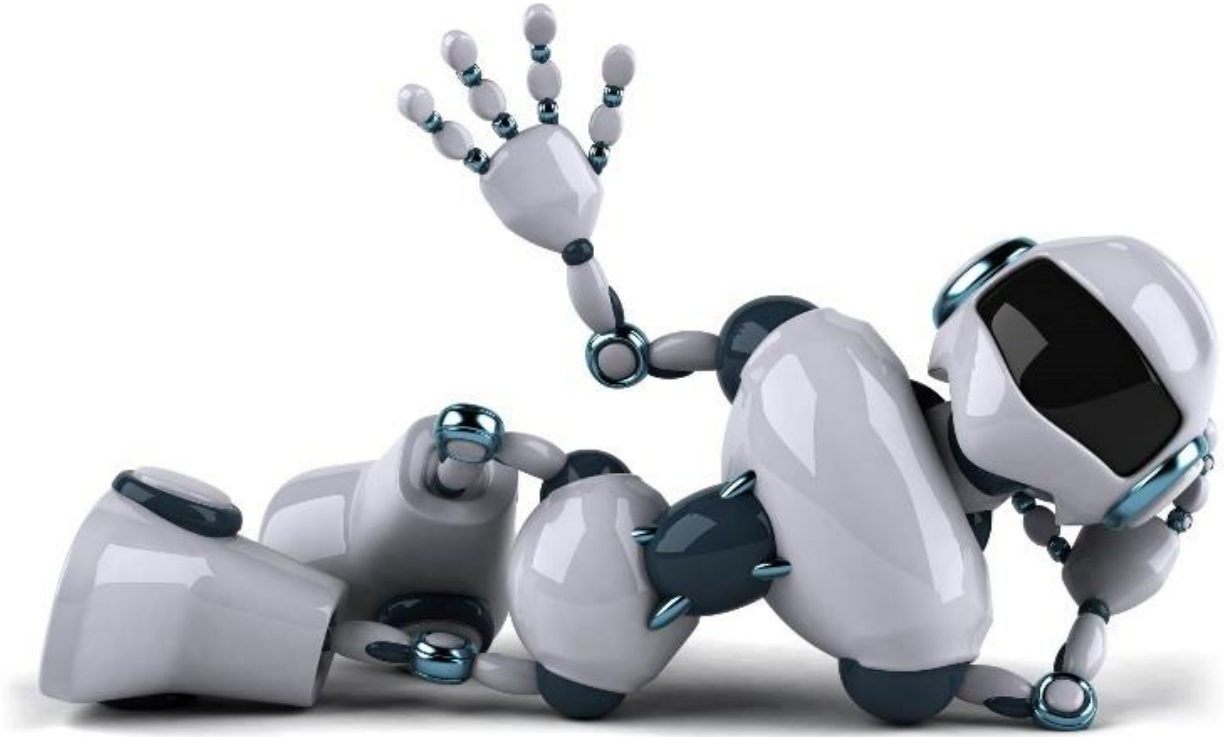
10 April 2017

Advanced Interactive Technologies

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich


Oculus
Development Kit


nest
72
75

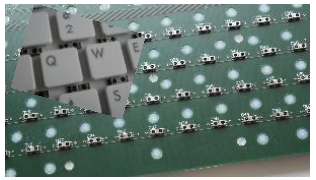


Designing Interactive Systems

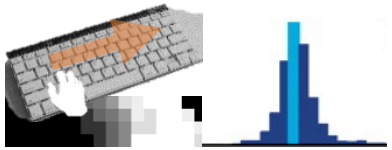
Interaction Task

Sensing Technology

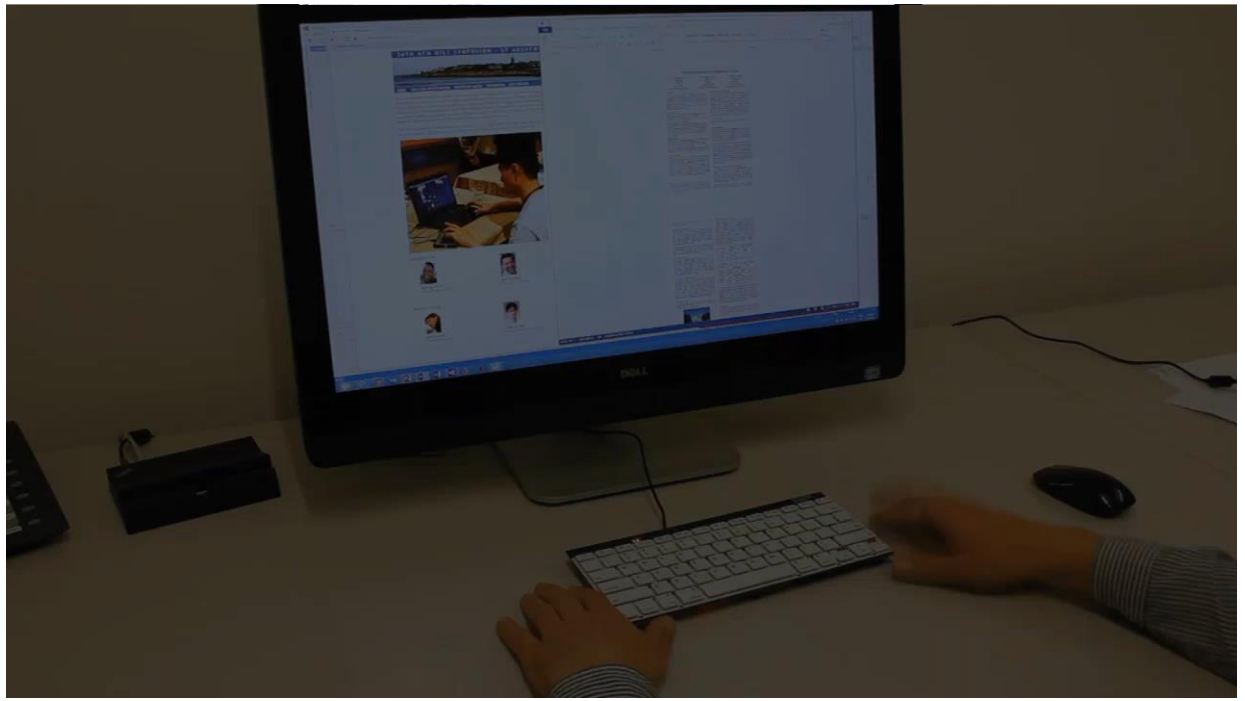
Sensor Placement



Input Recognition



System Output



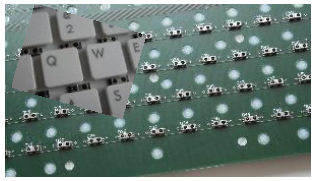
[Motion Sensing Keyboard. ACM CHI'14]

Today's talk – Part I

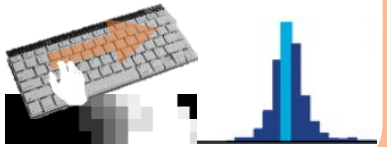
Interaction Task

Sensing Technology

Sensor Configuration



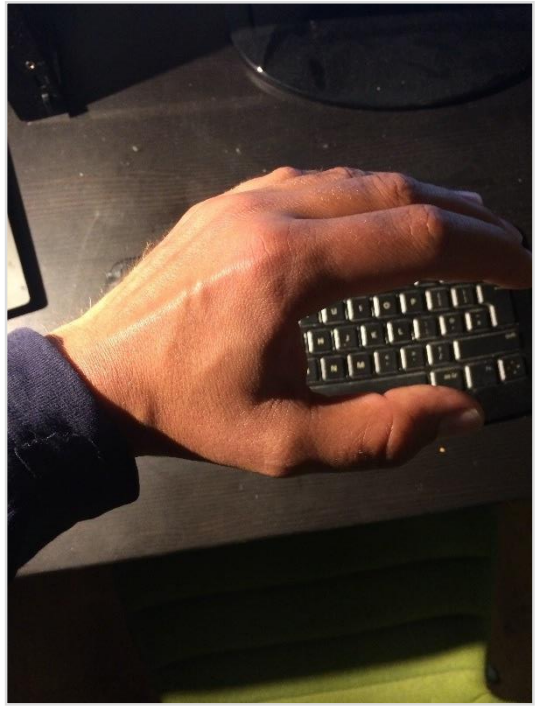
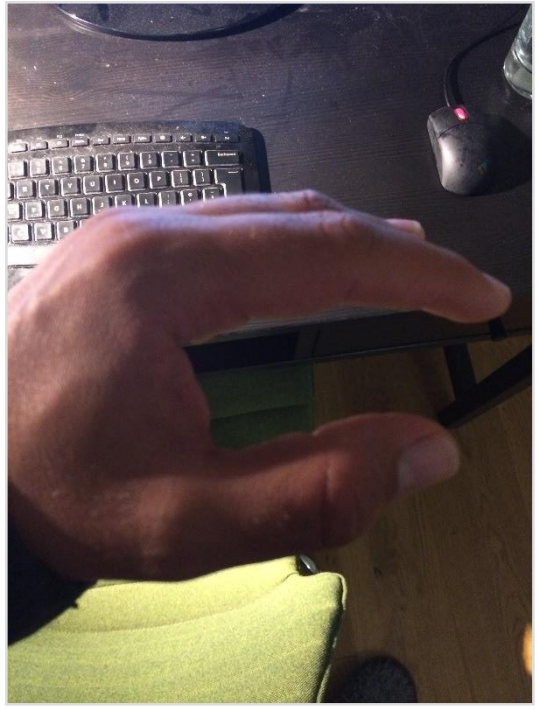
Input Recognition



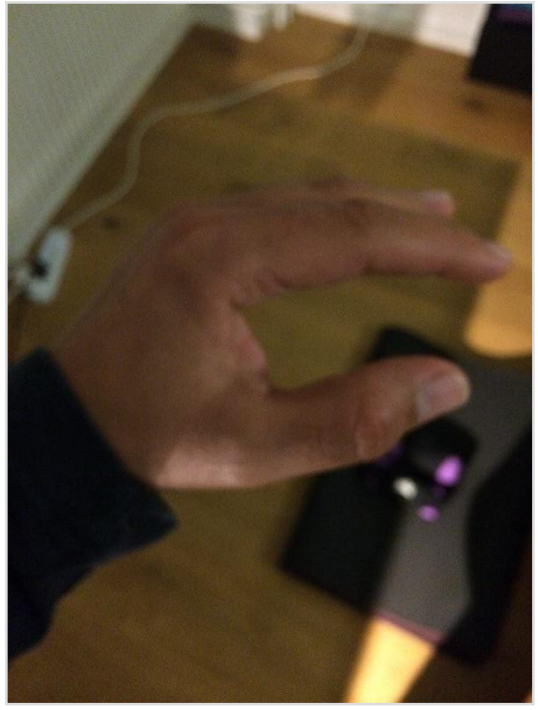
System Output

Today: Machine Learning for Gesture Recognition







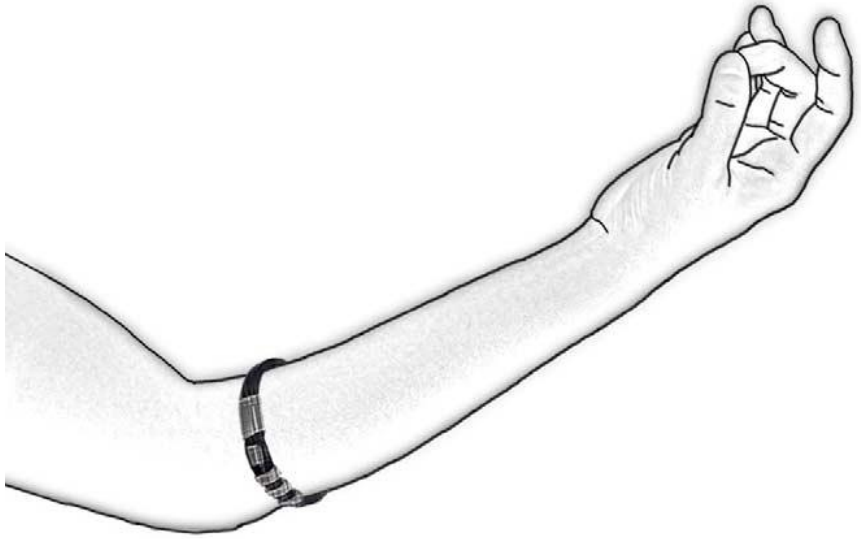
Muscle-Computer Interfaces

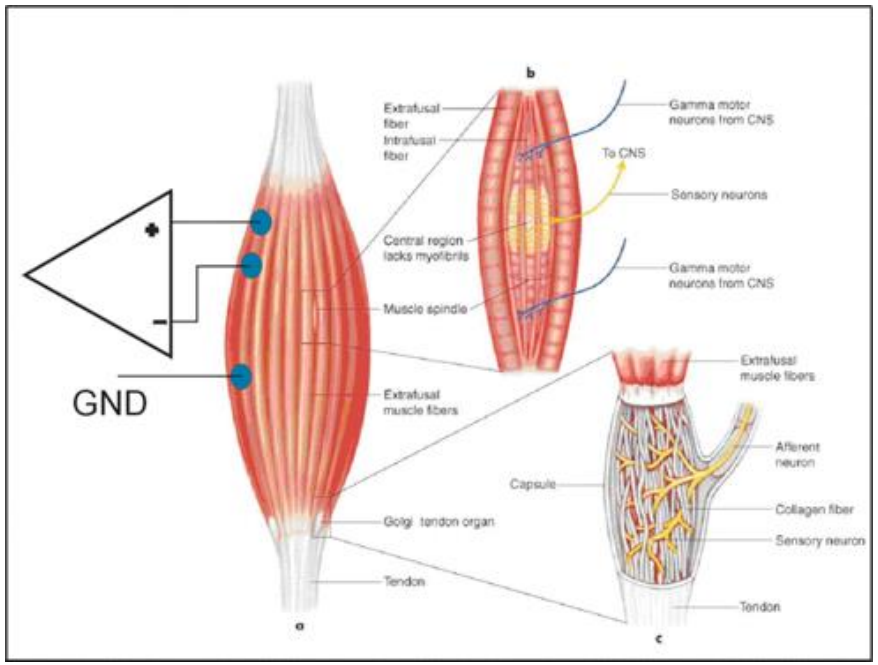
Hands free interaction – even when holding an object

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*]

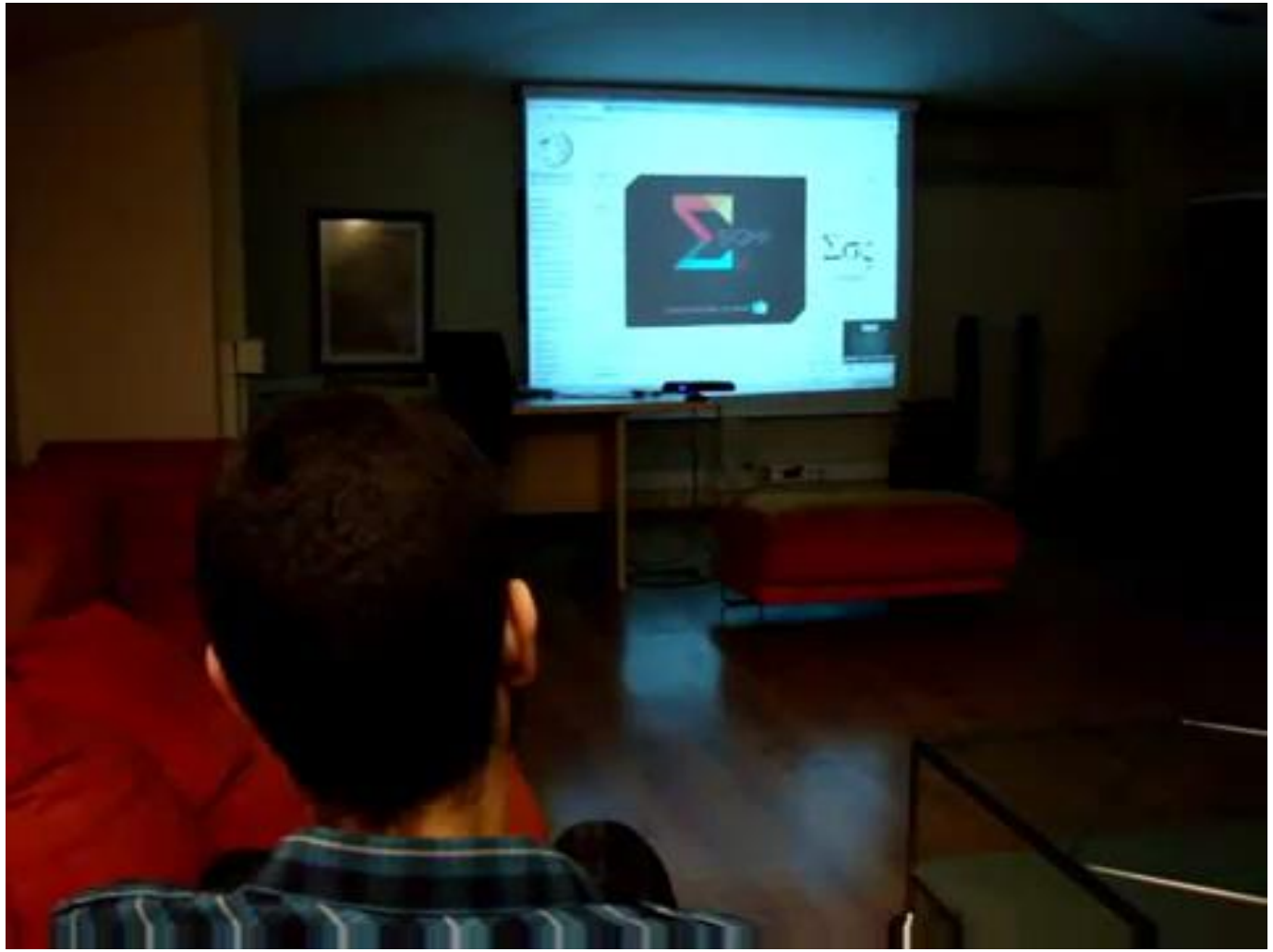
Video

[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

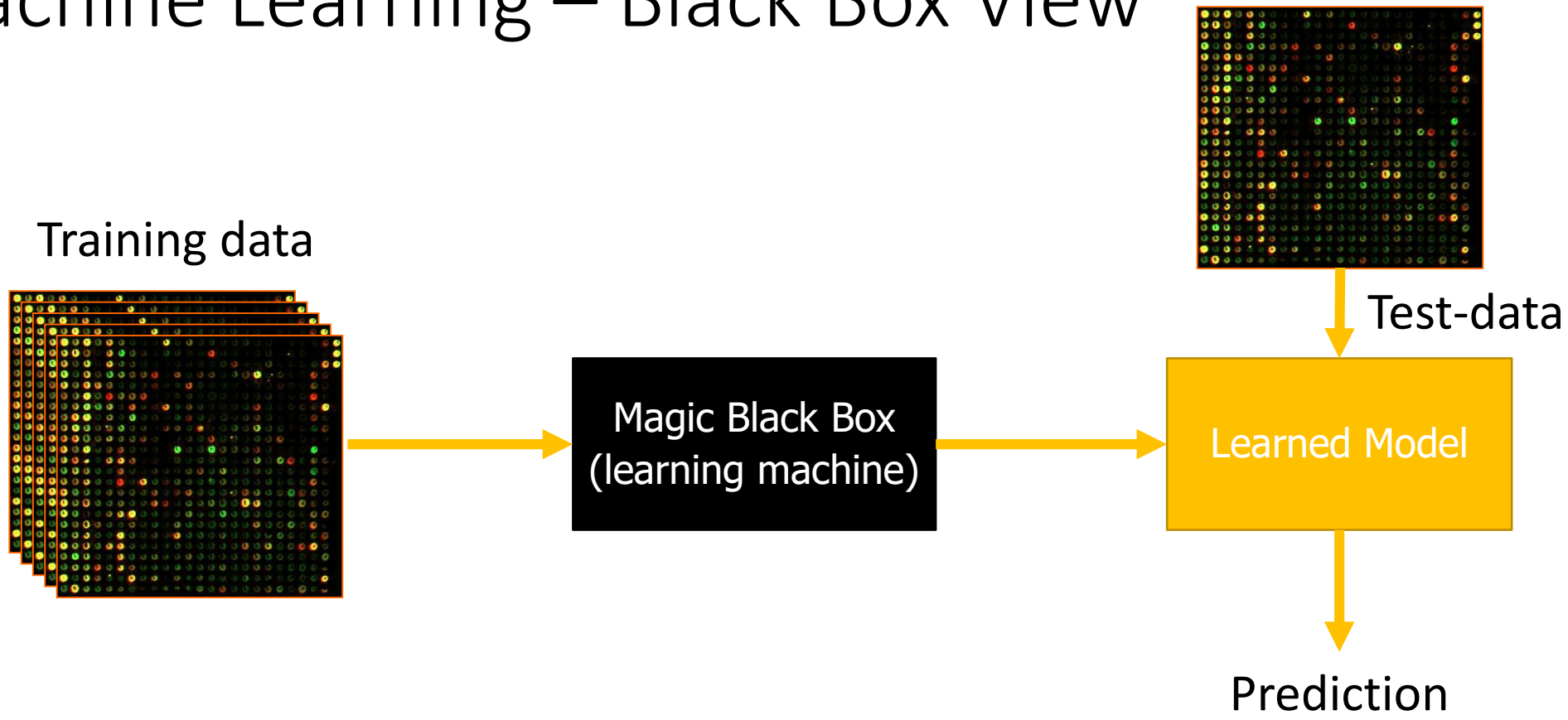
Humans use a variety of channels to interact & communicate

- 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



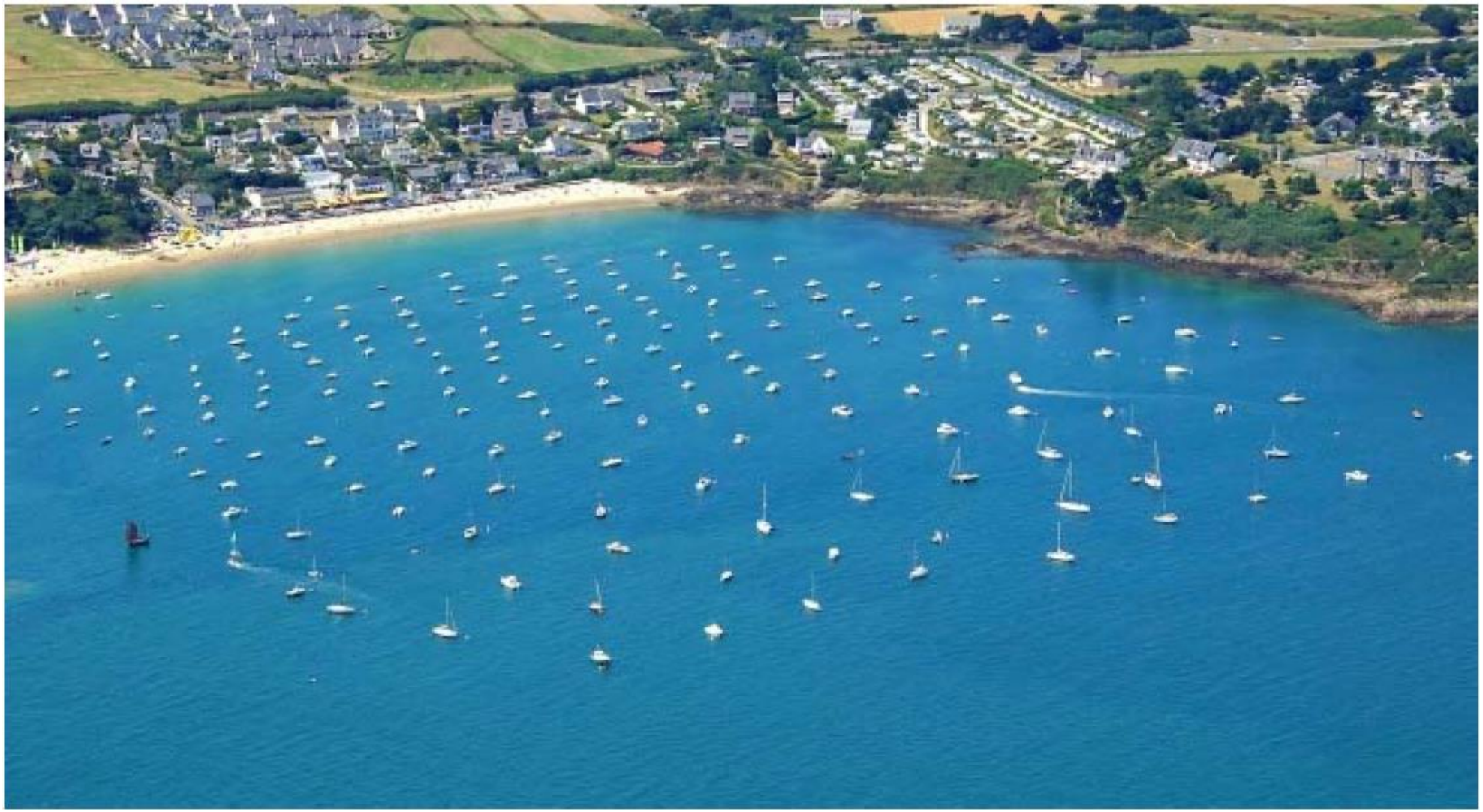
<u>Training data:</u>

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

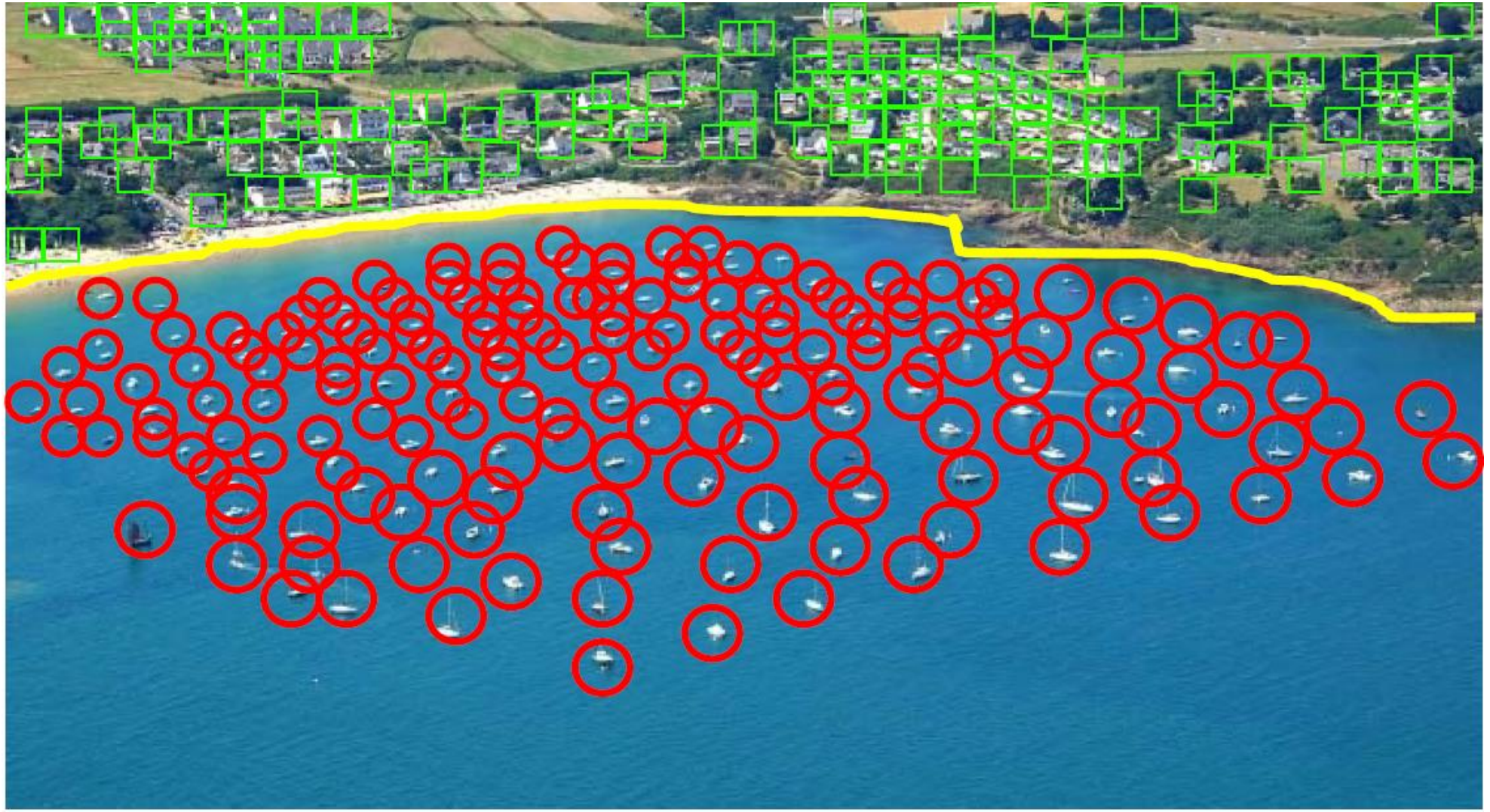
<u>Model:</u>

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

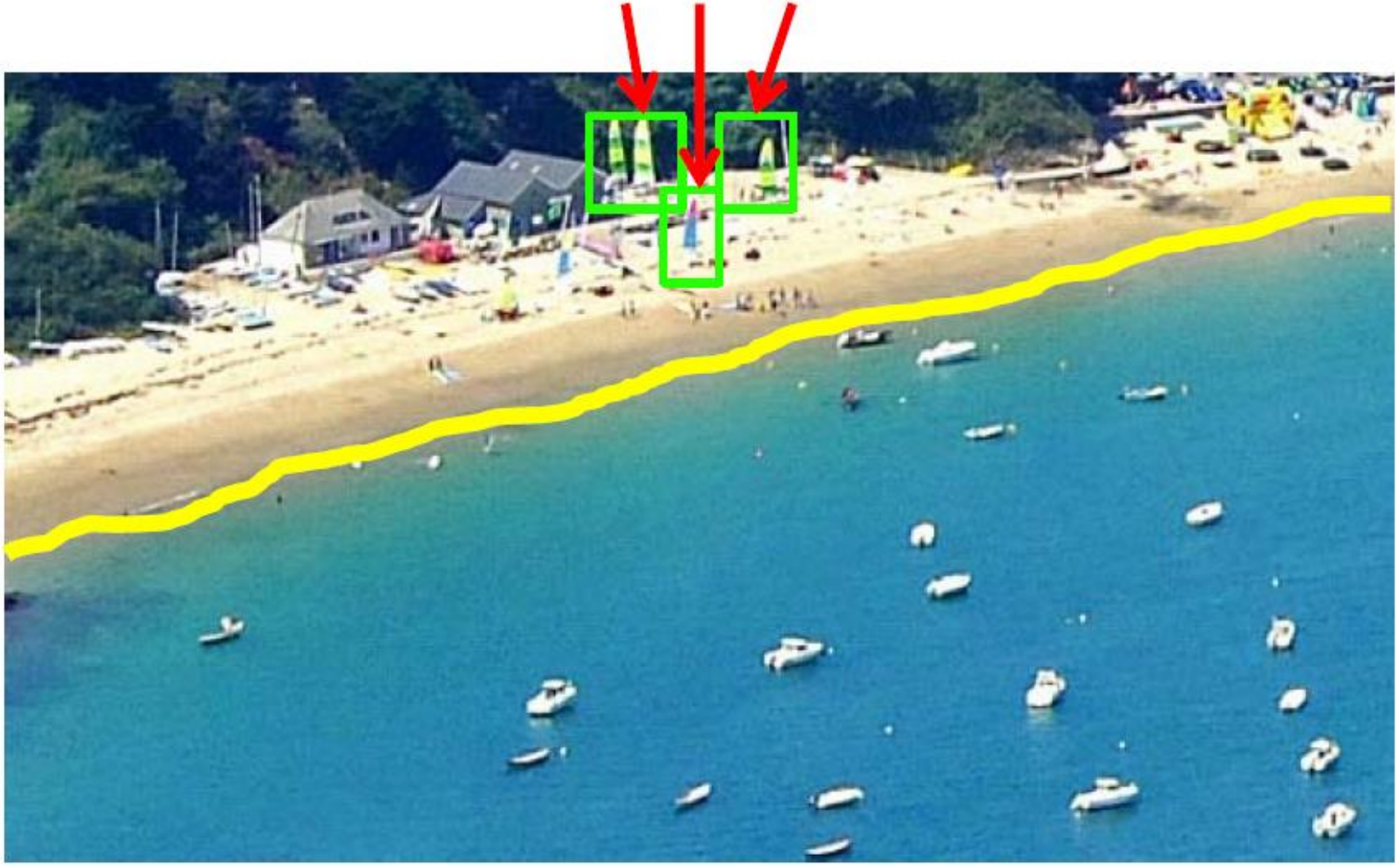
Classification: Basic Principles



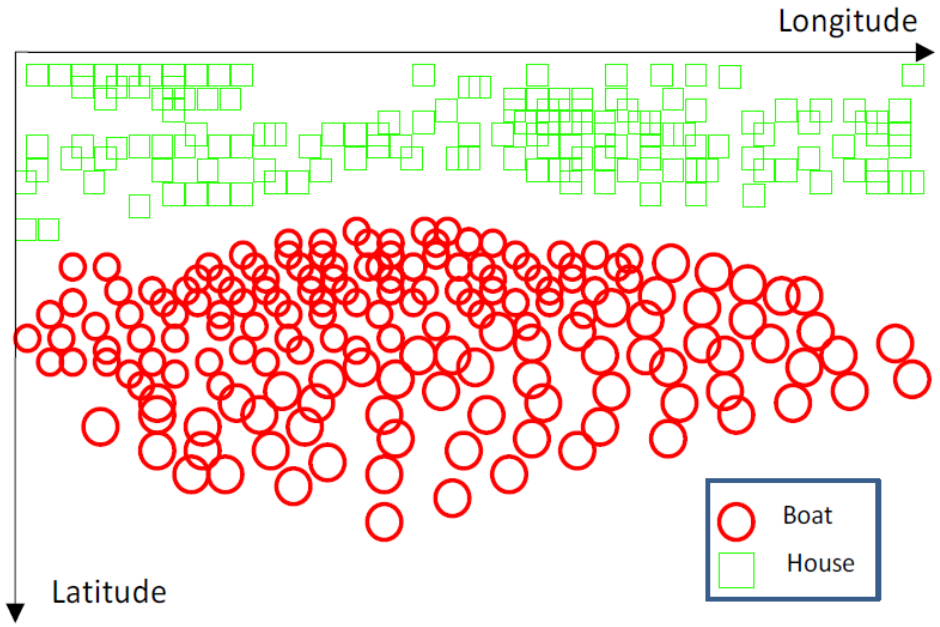
Classification: Decision Surface

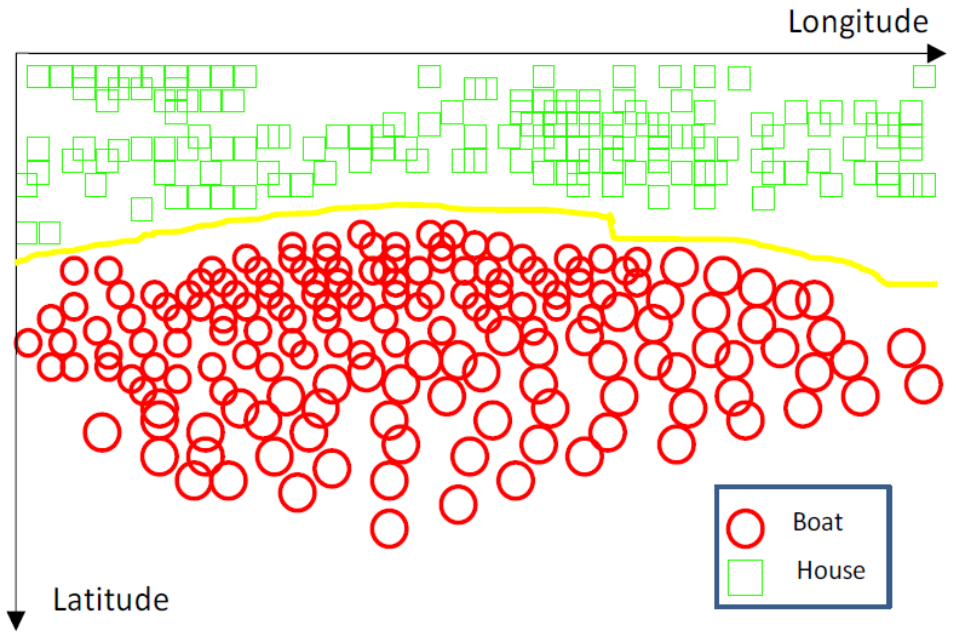


Classification: Classification Errors

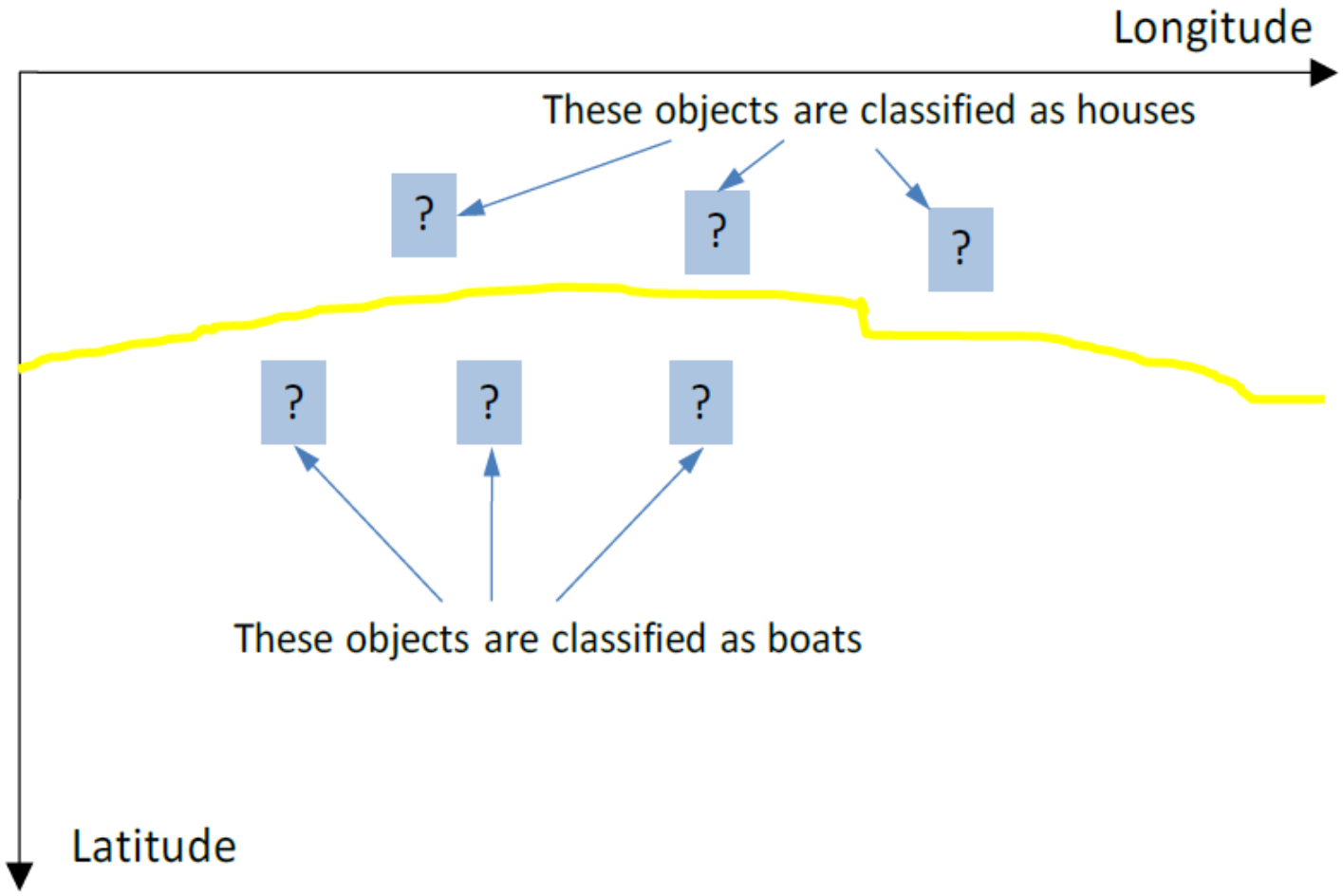


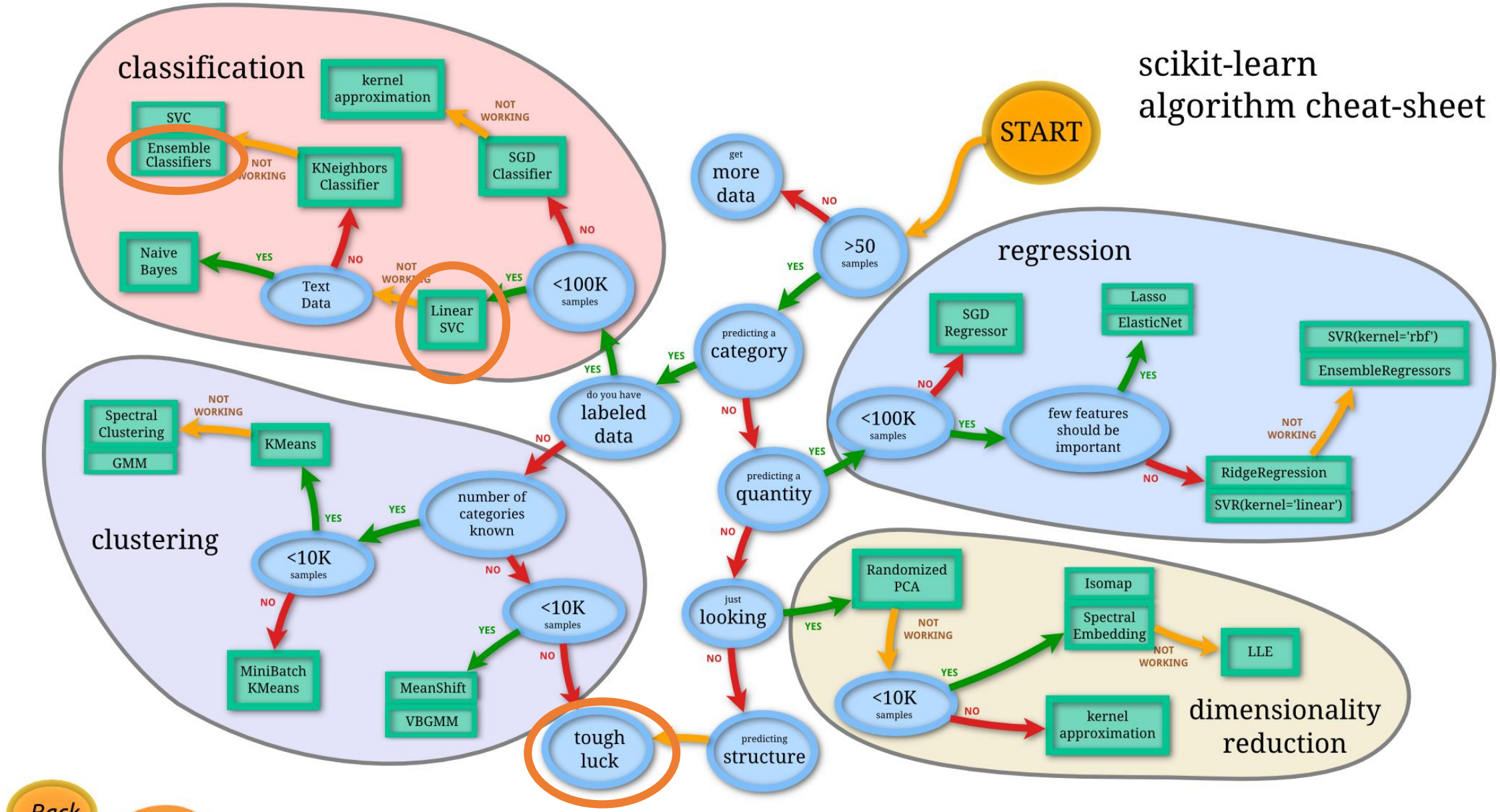
Classification Algorithms – Training time





Classification Algorithms – Test time





Back

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

Support Vector Machine (SVM)

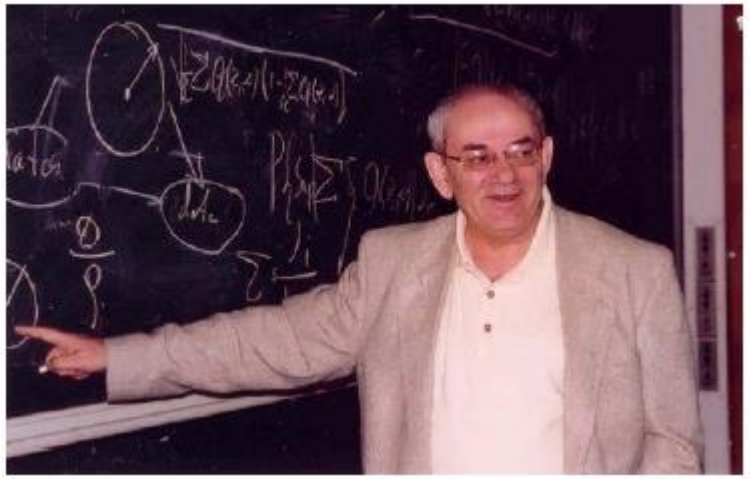
Derived from statistical learning theory by Vapnik, et al. in 1992

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

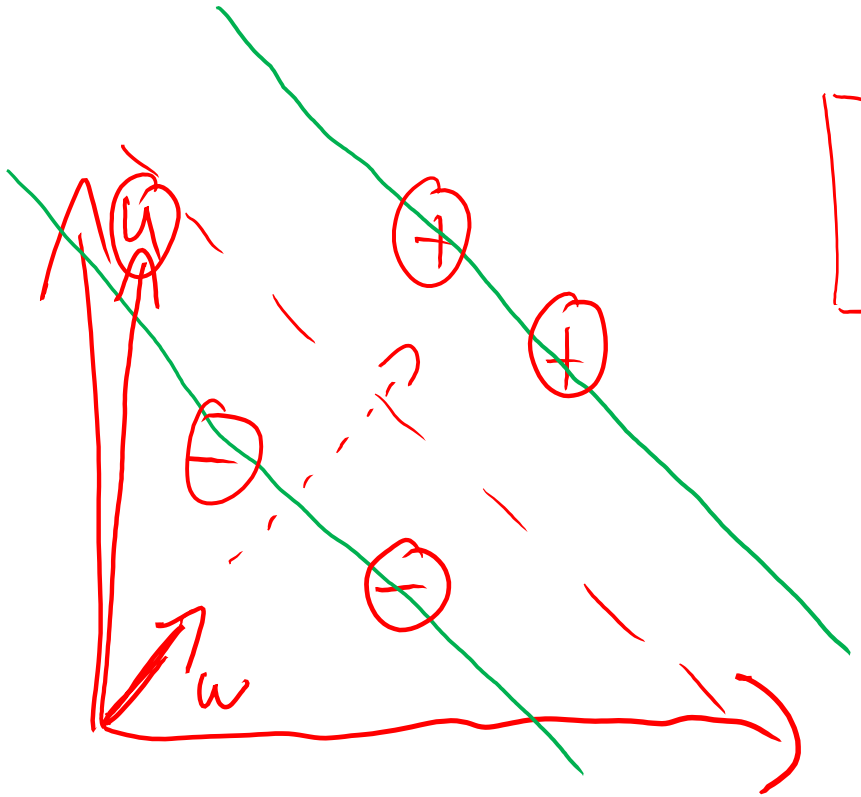
SVM widely used in object detection & recognition, content-based image retrieval, text recognition, biometrics, speech recognition, etc.

Also used for regression (will not cover today)

Vladimir Vapnik



SVM Derivation (Intuition with some slight of hand)



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

$b = -c$

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

what are $\bar{w}$ & b

are the unknows

SVM Derivation (Intuition with some slight of hand)

$\bar{\omega} \cdot \bar{x}_+ + b \geq 1$

$\bar{\omega} \cdot \bar{x}_- + b \leq -1$

for convenience we introduce y s.t. $+1$ for $\oplus$, -1 for $\ominus$

$y_i(\bar{\omega} \cdot \bar{x}_i + b) \geq 1$ for SV

$\Rightarrow y_i(\bar{\omega} \cdot \bar{x}_i + b) \geq 1 \Rightarrow$

$y_i(\bar{\omega} \cdot \bar{x}_i + b) = 1$

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

SVM Derivation (Intuition with some slight of hand)

we need to constrain $\bar{w}$ & b

positive sample

$\bar{w} \cdot \bar{x}_+ + b \geq 1$

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

↑ negative sample

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

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

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

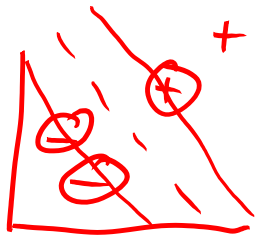
(2)

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

FOR x_i ON THE GUTTER

y_i such that y_i + 1 for + samples

- 1 for - samples



SVM Derivation (Intuition with some slight of hand)

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

width $= (x_+ - x_-) \cdot \frac{w}{\|w\|} = \frac{2}{\|w\|}$ * The primal problem

$1 + b + 1 - b = 2$

x_+: $g_i(\bar{\omega} \cdot \bar{x}_+ + b) - 1 = 0$

$\bar{\omega} \cdot x_+ = 1 - b$

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

SVM Derivation (Intuition with some slight of hand)

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

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

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

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

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

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

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

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

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

The Optimization Problem Solution

The solution has the form:

$$\mathbf{w} = \Sigma \alpha_i y_i \mathbf{x_i} \qquad b = y_k - \mathbf{w}^T \mathbf{x_k} \text{ for any } \mathbf{x_k} \text{ such that } \alpha_k \neq 0$$

Each non-zero α_i indicates that corresponding $\mathbf{x_i}$ is a support vector.

Then the classifying function will have the form:

$$f(\mathbf{x}) = \Sigma \alpha_i y_i \mathbf{x_i}^T \mathbf{x} + b$$

Optimization is provably convex (for linearly separable sets)

Notice that it relies only on the *dot product* between the test point **x** and the support vectors $\mathbf{x_i}$

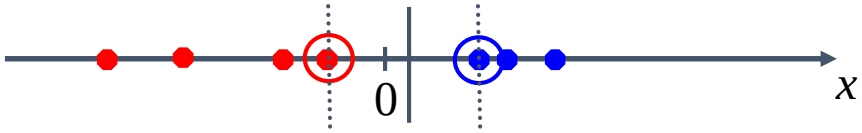
Also keep in mind that solving the optimization problem involved computing the inner products $\mathbf{x_i}^T \mathbf{x_j}$ between all pairs of training points.

Now Some Practical Issues (iPython Notebook)

Non-linear SVMs

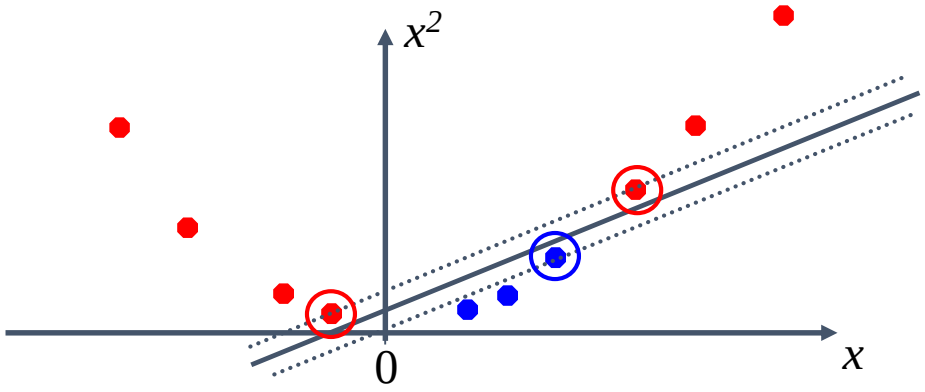
Non-linear SVMs

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



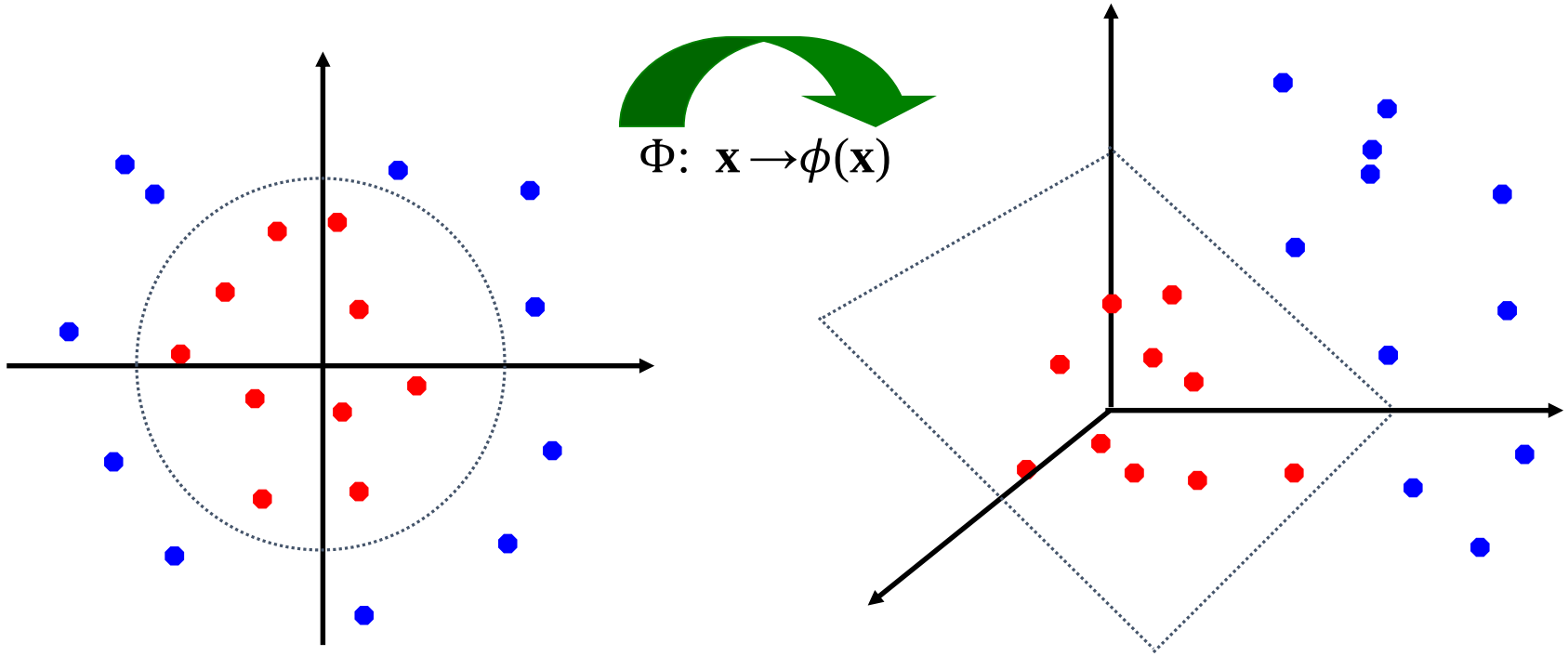
But what are we going to do if the dataset is just too hard?

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:

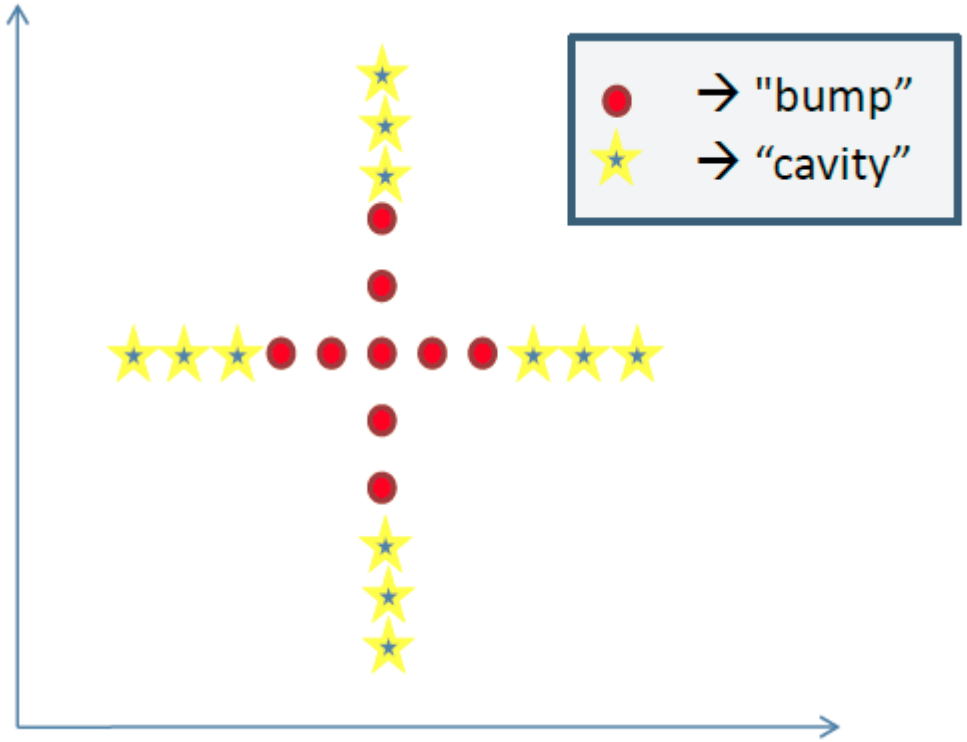
$$K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$$

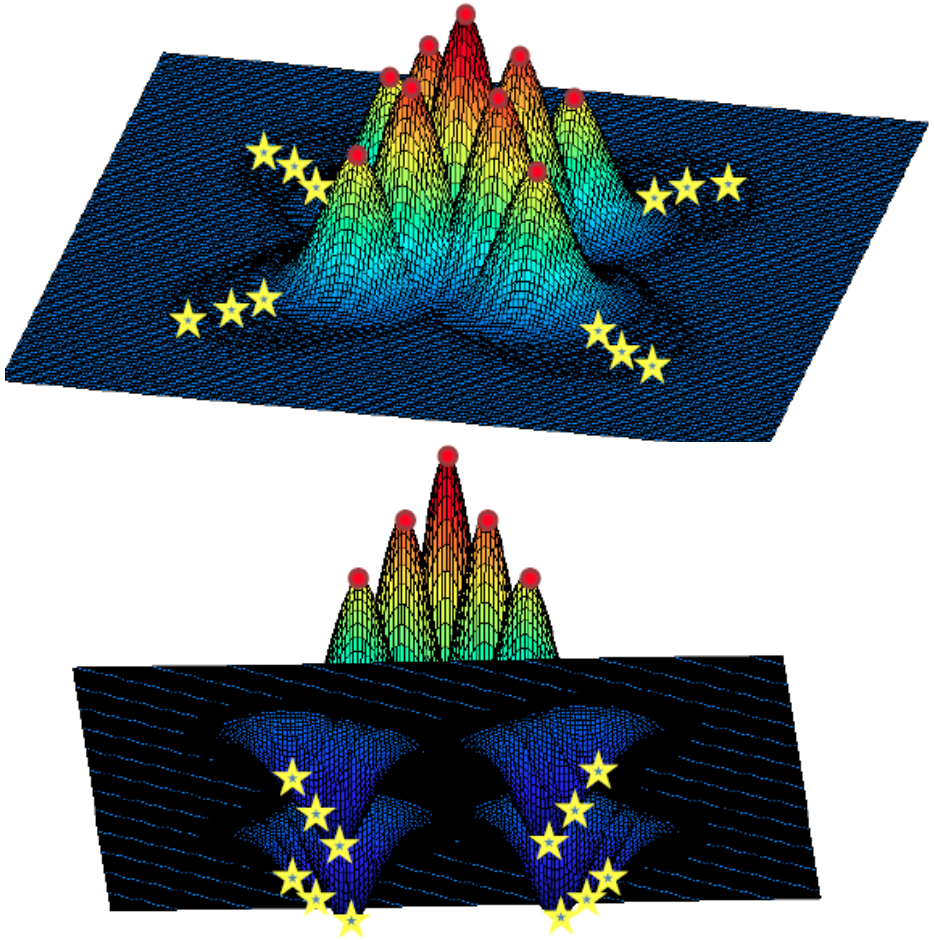
- Sigmoid: $K(x_i, x_j) = \tanh(\beta_0 x_i^T + \beta_1)$

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

Example: Gaussian Kernel

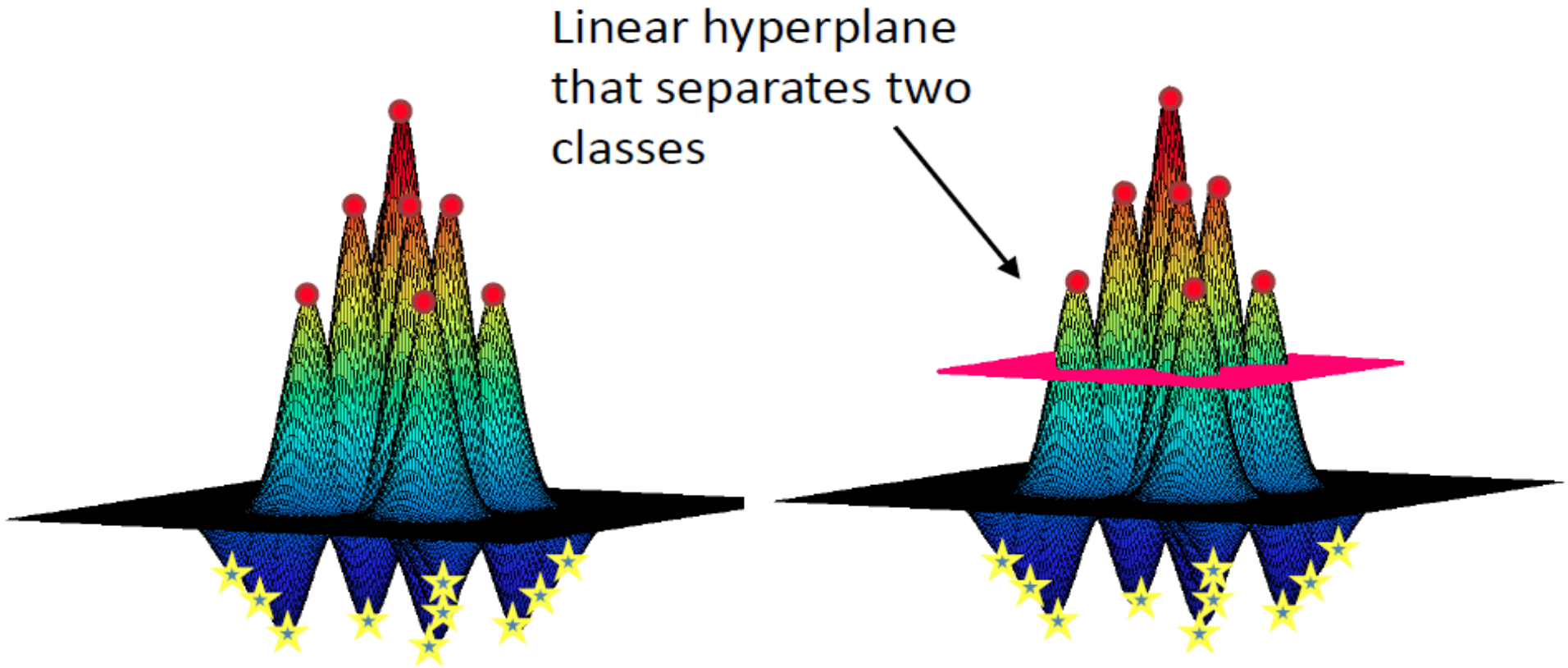
Geometrically, this is a “bump” or “cavity” centered at the training data point x:





Example: Gaussian Kernel - Classification

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



SVMs in HCI

Many software packages exist, implementing linear SVMs, kernel SVMs and more

- LibSVM (http://www.csie.ntu.edu.tw/~cjlin/libsvm/)
- Matlab SVM toolbox (http://www.isis.ecs.soton.ac.uk/resources/svminfo/)
- Weka Data Mining toolbox (http://www.cs.waikato.ac.nz/ml/weka/index.html)

- More resources:
  - http://www.kernel-machines.org

SVMs probably *the most frequently* used ML technique in HCI

Reading Suggestions

V. Vapnik, Statistical Learning Theory. Wiley-Interscience; 1998

Christopher M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer-Verlag New York, Inc., Secaucus, NJ, 2006

C.J.C. Burges. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2):955-974, 1998.

http://www.kernel-machines.org