FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

FACULTY OF ENGINEERING

Intent detection and somatosensory feedback

#08: Machine learning for intent detection

Claudio CASTELLINI, Sabine THÜRAUF

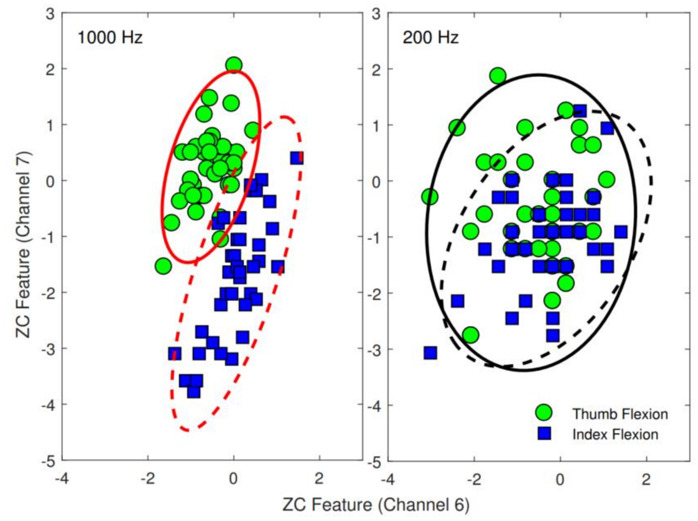


Figure 2. Differences in EMG patterns between using: (**left**) a 1000 Hz sampling rate; and (**right**) a 200 Hz sampling rate. ZC features are extracted from two different EMG channels (6 and 7) during thumb flexion (green circle markers and solid lines) and index flexion (blue square markers and dashed lines). Samples are from Subject 1 of Database 3.

EMG patterns related to two actions. Reproduced from Angkoon Phinyomark, Rami N. Khushaba and Erik Scheme, *Feature Extraction and Selection for Myoelectric Control Based on Wearable EMG Sensors*, MDPI Sensors 2018, 18, 1615

The *rubber hand illusion.* See Botvinick M, Cohen J., *Rubber hands 'feel' touch that eyes see.* Nature. 1998 Feb 19;391(6669):756. doi: 10.1038/35784. PMID: 9486643.

Lecture #08:

Machine learning for intent detection

- interaction and incrementality; „knowledge composition“
- classification

ML, specifically for I.D.

- …so here we are at the *core* of intent detection…

- machine learning!
    - isn‘t it so?

- no it is not. sorry to disappoint you ☺

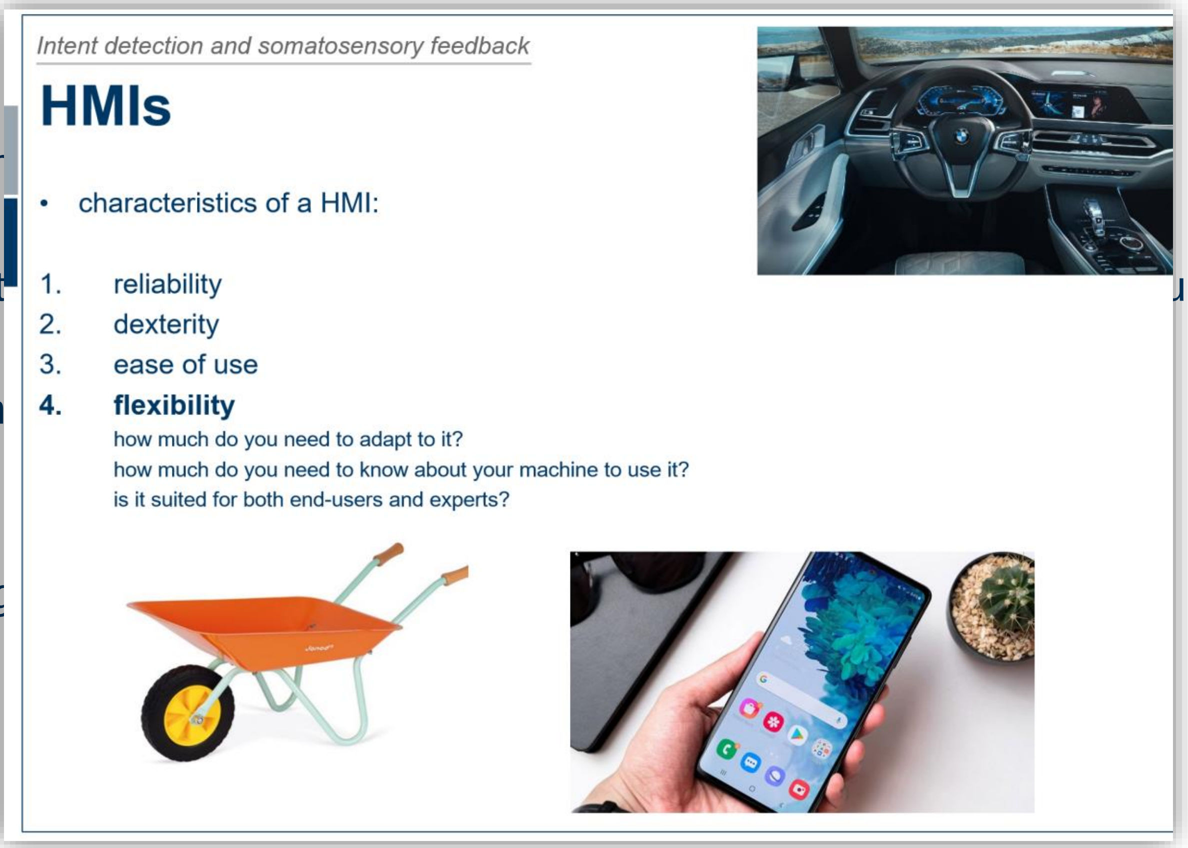
- ML is just „one of the tools in our belt“,
- and if you want to have my opinion,
- *it is a secondary one.*

ML, specifically for I.D.

- intent detection is more about *interaction.*
- together with somatosensory feedback, it builds up a closed-loop system with a human in it
- the user is the main player.
- *so machine learning must be at the service of interaction!*

ML, specifically for I.D.

- intent detection is m
- together with somat   uman in it
- the user is the main
- *so machine learning*



Intent detection and somatosensory feedback

HMIs

- characteristics of a HMI:

1. reliability
2. dexterity
3. ease of use
4. **flexibility**
   how much do you need to adapt to it?
   how much do you need to know about your machine to use it?
   is it suited for both end-users and experts?

ML, specifically for I.D.

- how do we build a ML system which exploits and fosters interaction?
- the simple answer: we make the system able to
  - acquire (incorporate) and use new data,
  - forget (exclude) and stop using past data,
  - ask the user what to do if unsure.
- so our system needs be incremental (and decremental)

ML, specifically for I.D.

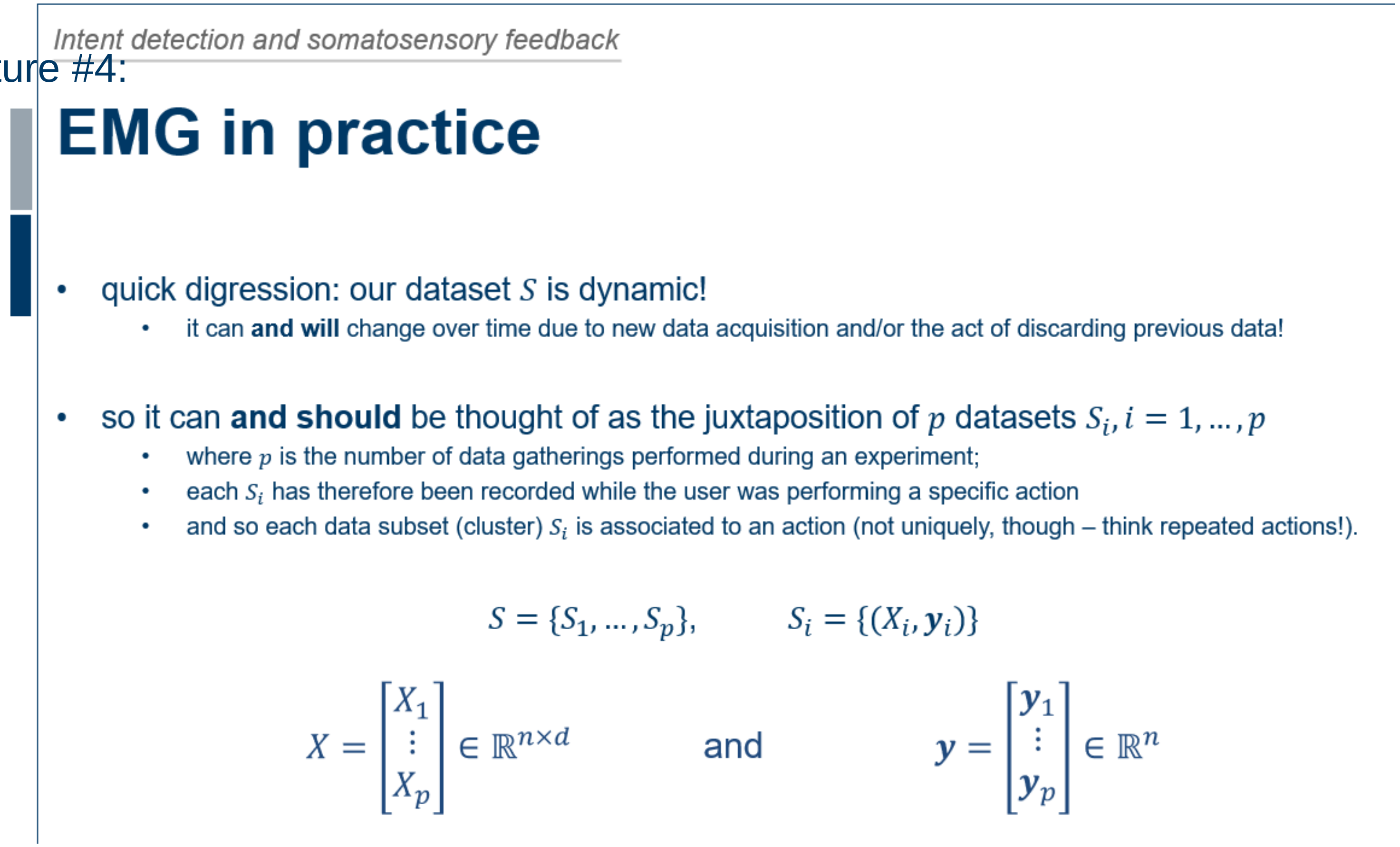
- a simple way to make any ML system (de-)incremental:

- store the data in a structured way,
- rebuild your model when needed.

- that sounds like our way of storing the (observation,target_value) pairs, innit?

ML, specifically for I.D.

- recall Lecture #4:



Intent detection and somatosensory feedback

EMG in practice

- quick digression: our dataset S is dynamic!
  - it can **and will** change over time due to new data acquisition and/or the act of discarding previous data!
- so it can **and should** be thought of as the juxtaposition of p datasets $S_i, i = 1, \dots, p$
  - where p is the number of data gatherings performed during an experiment;
  - each S_i has therefore been recorded while the user was performing a specific action
  - and so each data subset (cluster) S_i is associated to an action (not uniquely, though – think repeated actions!).

$$S = \{S_1, \dots, S_p\}, \qquad S_i = \{(X_i, \boldsymbol{y}_i)\}$$

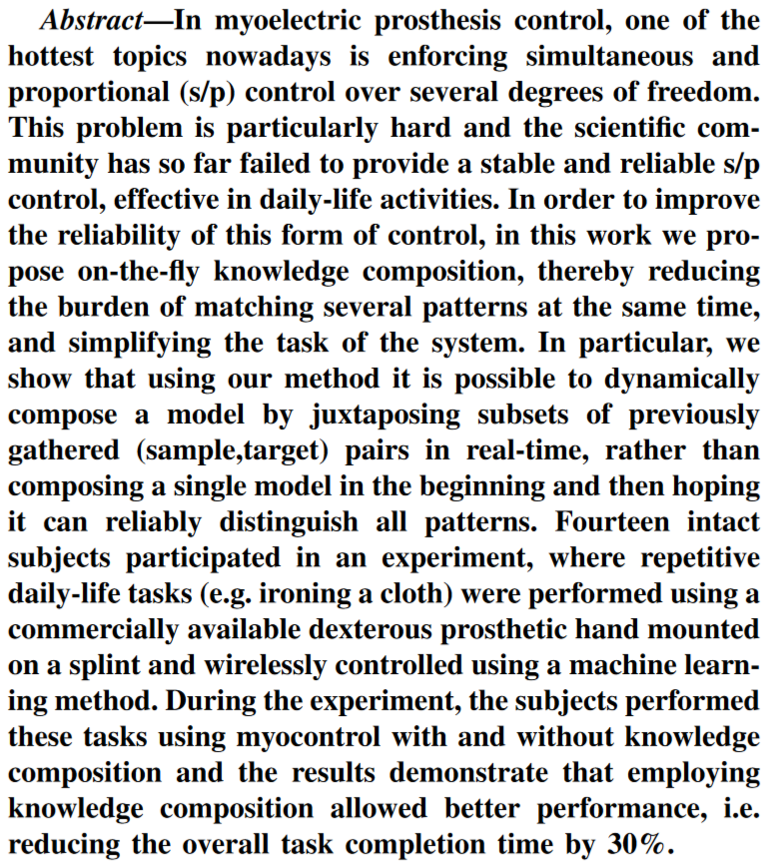
$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_p \end{bmatrix} \in \mathbb{R}^{n \times d} \qquad \text{and} \qquad \boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_p \end{bmatrix} \in \mathbb{R}^{n}$$

ML, specifically for I.D.

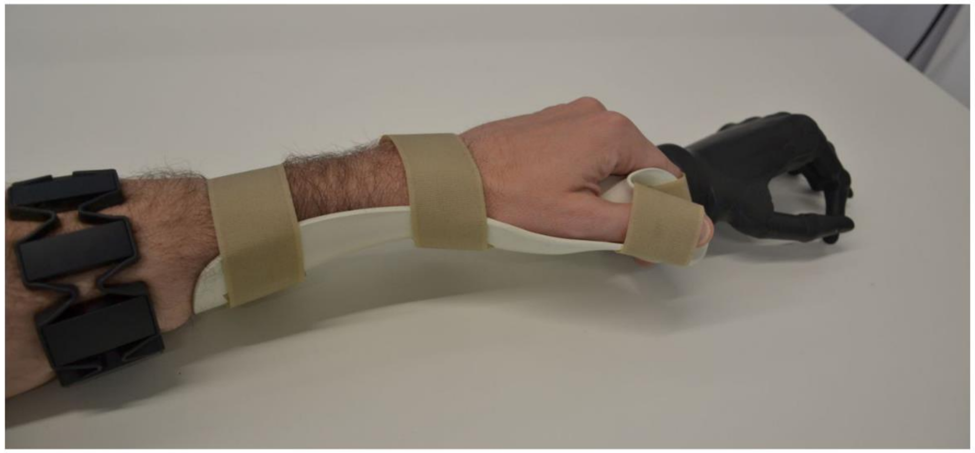
- literally, you can augment the data set S to you liking, $S' = S \cup S_{p+1}$, then build a new model
- or you can remove a cluster from the data set S, $S' = S \setminus S_3$, then build a new model
- or you can (cherry-)pick clusters from S, $S' = \{S_{i_1}, \dots, S_{i_k}\}$, then build a new model
- conditions for this to be possible:
  - just one, a reasonably fast way of building your model
  - …but in principle, *none.*
- *see demo.*

Patel, G., Nowak, M., & Castellini, C. (2017). Exploiting knowledge composition to improve real-life hand prosthetic control. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *25*(7), 967–975.

Knowledge composition



***Abstract*—In myoelectric prosthesis control, one of the hottest topics nowadays is enforcing simultaneous and proportional (s/p) control over several degrees of freedom. This problem is particularly hard and the scientific community has so far failed to provide a stable and reliable s/p control, effective in daily-life activities. In order to improve the reliability of this form of control, in this work we propose on-the-fly knowledge composition, thereby reducing the burden of matching several patterns at the same time, and simplifying the task of the system. In particular, we show that using our method it is possible to dynamically compose a model by juxtaposing subsets of previously gathered (sample,target) pairs in real-time, rather than composing a single model in the beginning and then hoping it can reliably distinguish all patterns. Fourteen intact subjects participated in an experiment, where repetitive daily-life tasks (e.g. ironing a cloth) were performed using a commercially available dexterous prosthetic hand mounted on a splint and wirelessly controlled using a machine learning method. During the experiment, the subjects performed these tasks using myocontrol with and without knowledge composition and the results demonstrate that employing knowledge composition allowed better performance, i.e. reducing the overall task completion time by 30%.**



Intent detection and somatosensory feedback

1. Knowledge composition is a method in myoelectric prosthesis control where control models are dynamically created in real-time by combining previously gathered data pairs (sample, target). This approach reduces the complexity of matching multiple patterns simultaneously and aims to improve the reliability and performance of prosthetic control systems.

•**Experiment Setup:** In an experiment, subjects use a prosthetic hand mounted on a splint, controlled via a machine learning algorithm. They perform tasks like ironing a cloth.
•**Without Knowledge Composition:** The prosthesis operates based on a static, pre-composed model, which might struggle with accurately distinguishing all control patterns.
•**With Knowledge Composition:** The prosthesis dynamically composes the control model from the gathered data pairs, leading to improved performance and a reduction in task completion time by 30%.

1. **Simultaneous and Proportional (s/p) Control refers to the ability of a myoelectric prosthesis to perform multiple movements at the same time and to control the extent or speed of these movements in a proportional manner based on the strength of the user's muscle signals.**
2. **Simultaneous Control**

- **Multiple Degrees of Freedom (DOF): The prosthesis can control several joints or axes of movement simultaneously. For example, a hand prosthesis might simultaneously perform wrist rotation, hand opening/closing, and finger movements.**
- **Complex Actions: This allows the user to perform complex, coordinated actions that involve multiple parts of the prosthetic limb working together, much like a natural hand.**

1. **Proportional Control**

- **Muscle Signal Intensity: The movement of the prosthesis is proportional to the strength of the muscle signals detected. Stronger muscle contractions result in faster or larger movements, while weaker contractions produce slower or smaller movements.**
- **Fine Motor Control: Proportional control enables users to have fine motor control over the prosthetic limb, allowing for delicate and precise actions. For example, gently picking up an object versus firmly grasping it.**

Patel, G., Nowak, M., & Castellini, C. (2017). Exploiting knowledge composition to improve real-life hand prosthetic control. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *25*(7), 967–975.

Knowledge composition

Action	$\mathbf{y}_{action}$	Description
Power	[1 0 1 1 1 1]	all fingers flexed, thumb abducted and flexed
Pinch	[1 0 1 1 0 0]	Index and Middle finger flexed, thumb abducted and flexed
Point	[1 0 0 1 1 1]	all fingers flexed but Index, thumb abducted and flexed
Pre-Flat	[0 1 1 1 1 1]	all fingers flexed, thumb adducted but not flexed
Flat	[1 1 1 1 1 1]	all fingers flexed, thumb adducted and flexed
Rest	[0 0 0 0 0 0]	all fingers at rest, thumb abducted but not flexed

b) Library: A book was placed on the top compartment of a shelf and a starting point was marked approximately 4 meters away from the shelf; additionally, a desk with a keyboard was placed near the shelf, alongside the path between the shelf and the starting point. After walking towards the shelf from the starting point, the subjects grabbed the book using a Power grasp and placed it on the desk near the keyboard. Next, the subjects used a Point grip to enter the title of the book via the keyboard; lastly, before returning back to the starting point, the subjects placed the book back to its original location using a Power grasp.

Patel, G., Nowak, M., & Castellini, C. (2017). Exploiting knowledge composition to improve real-life hand prosthetic control. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *25*(7), 967–975.

Knowledge composition

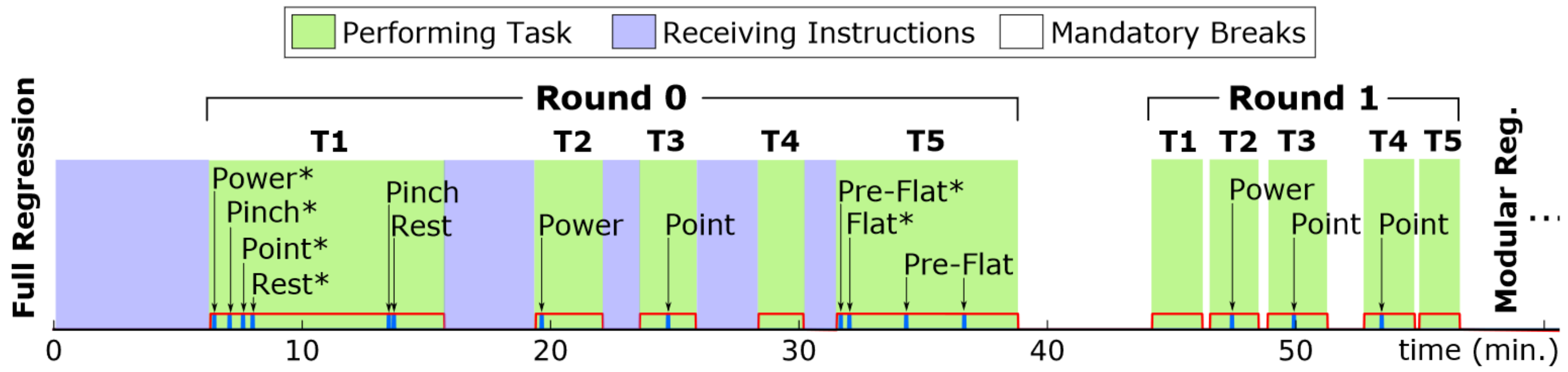


Fig. 3. A typical experimental timeline. The subject starts in the FR modality and performs five tasks (T1-Shopping, T2-Ironing, T3-Library, T4-Jenga and T5-Arranging); occasionally, model updates are requested (a star denotes mandatory updates, that is, updates for actions that had never been trained in the past and the event of model update has been indicated using a blue marker along the timeline).

Full Regression (FR): A single, comprehensive model that handles all tasks and variations. Simpler in deployment but may struggle with complexity.
Modular Regression (MR): Multiple, task-specific models that handle subsets of the overall task. More flexible and potentially more efficient but requires careful integration.

Patel, G., Nowak, M., & Castellini, C. (2017). Exploiting knowledge composition to improve real-life hand prosthetic control. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *25*(7), 967–975.

Knowledge composition

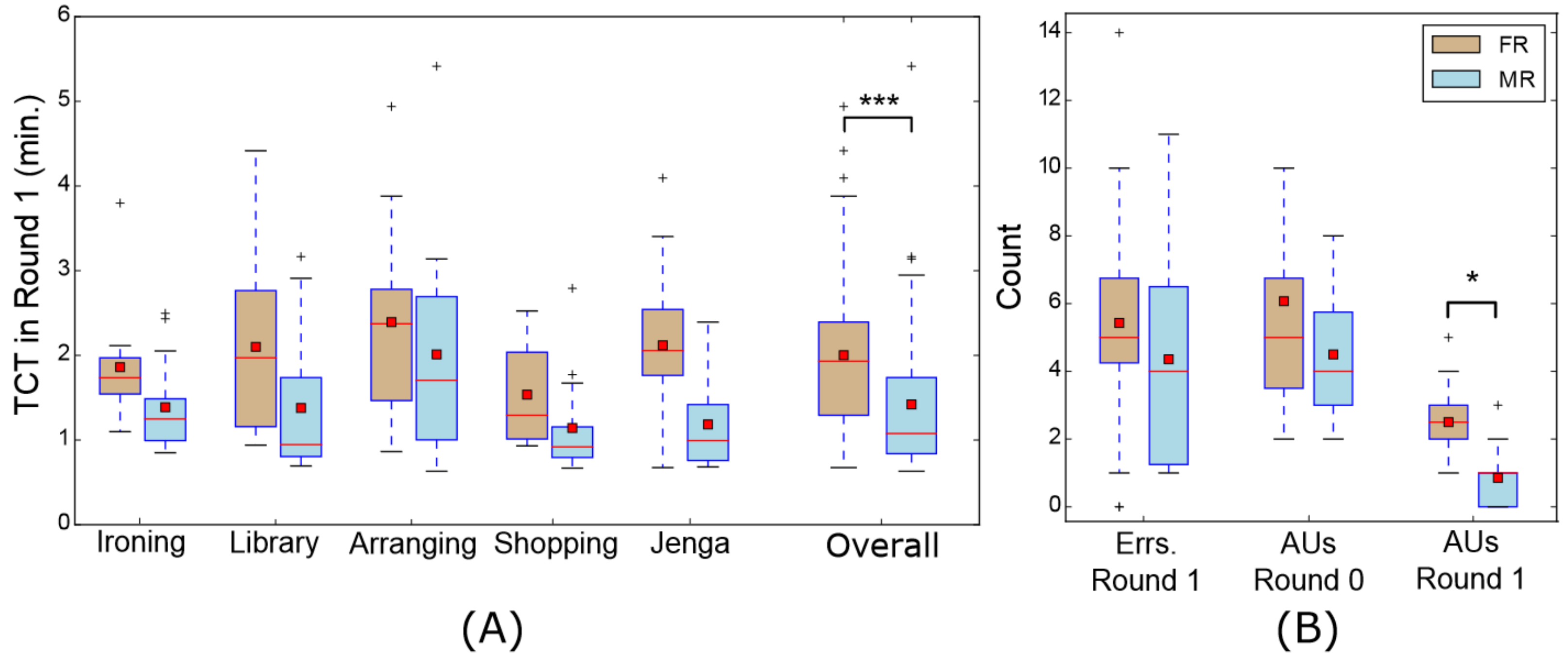


Fig. 4. Summary of the comparison between FR and MR. (A) median Time to Complete a Task (TCT) in Round 1 in minutes. (B) median count of the errors (Err.) during Round 1 and median count of action updates (AUs) in Round 0 and Round 1; one outlier with 18 AUs in Round 0 with FR has not been shown. (***,$p < 0.001$; **,$p < 0.01$; *,$p < 0.05$; '+', represents outliers; 'red square', population mean; 'red line', population median)

Patel, G., Nowak, M., & Castellini, C. (2017). Exploiting knowledge composition to improve real-life hand prosthetic control. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *25*(7), 967–975.

Knowledge composition

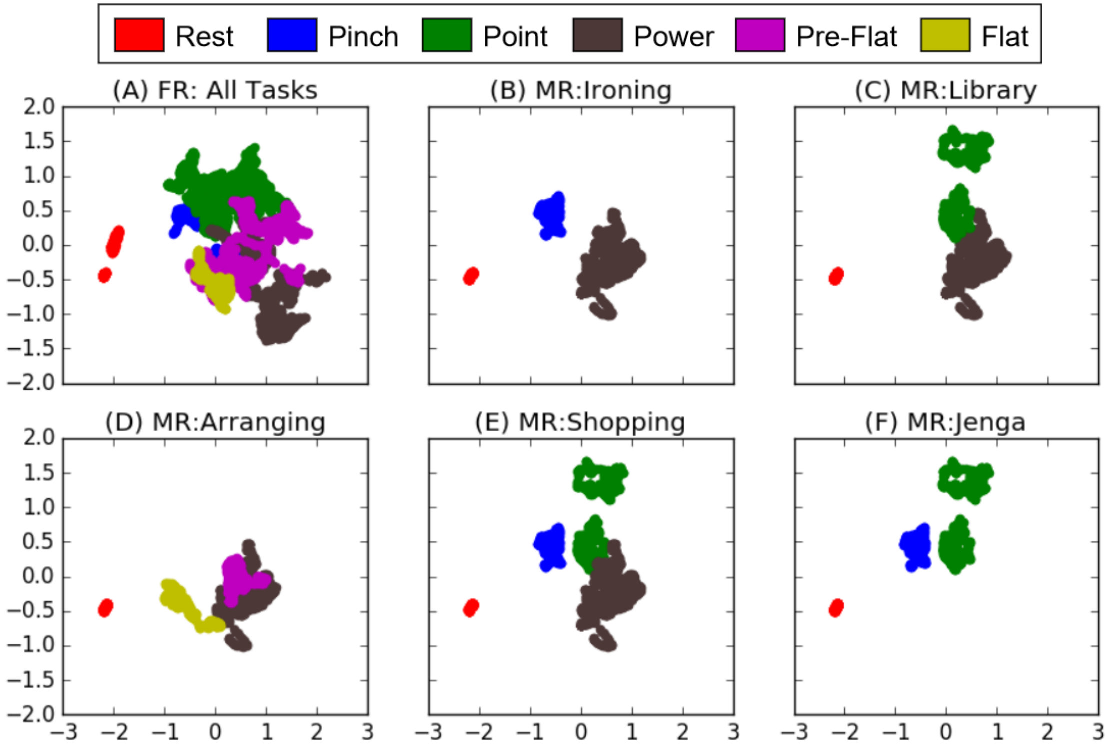


Fig. 6. Visualisation of training data used to generate the regression model when performing different tasks with FR (A) and MR (B-F). Principle Component Analysis (PCA) was used to reduce the dimensions of training data and the explained variance for the plotted data is 72%. (The plotted training data was collected on a single subject, whose experimental timeline is shown in Figure 3.)

ML, specifically for I.D.

- …but really, a (de-)incremental ML system is one which
  - given a new (observation,target_value) pair $(\boldsymbol{x}', y')$ (or a whole set of them)
  - can update its model without rebuilding it from scratch

$$f' \triangleq \mathcal{U}(f, \boldsymbol{x}', y')$$

- advantages:
  - no need to store any such pair!
  - potentially faster than the „simpler“ idea (storing and rebuilding)
- disadvantages:
  - how to „downdate“, i.e., how to forget past knowledge?
  - no chance to consider the quality of past data

ML, specifically for I.D.

- so, which one is better? it depends on
  - speed of rebuilding the model
  - need to store past pairs vs. memory requirements
  - …

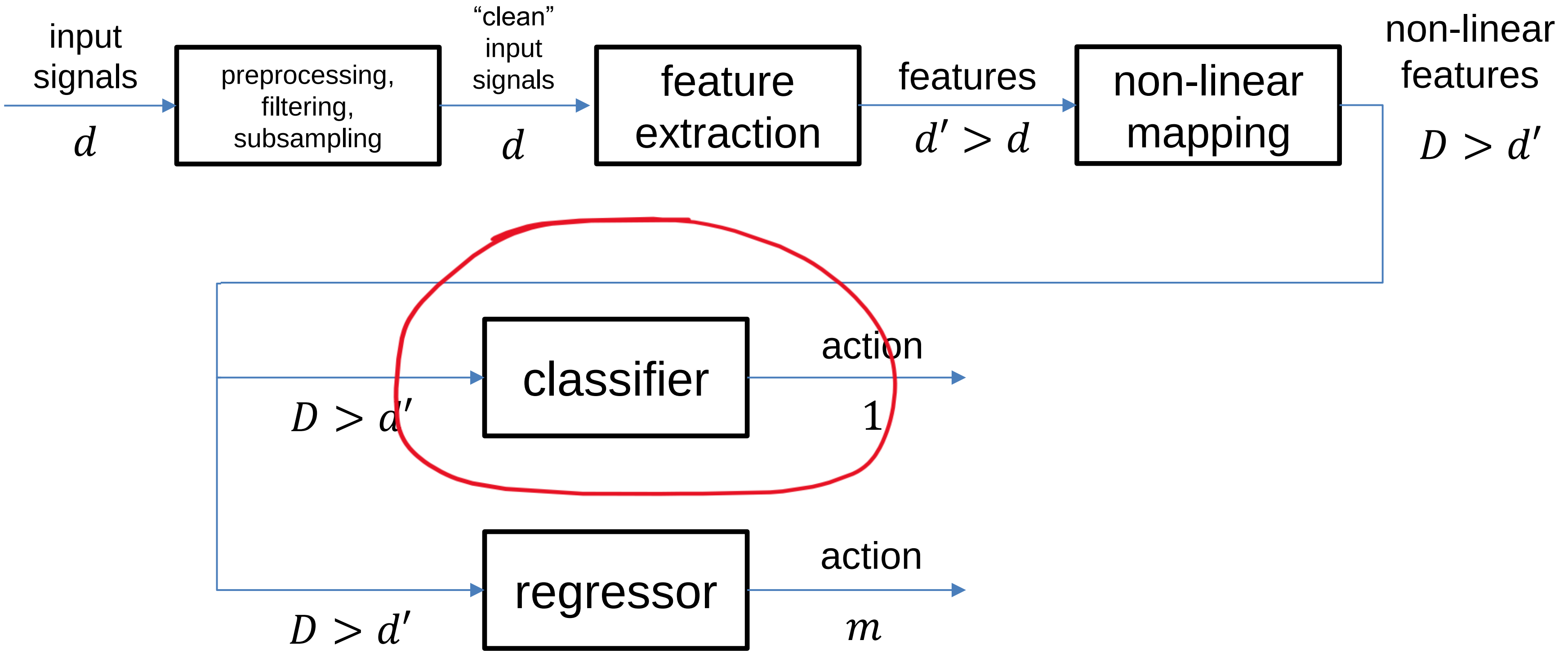
- in general, our ideal ML method for I.D.
  - is incremental / decremental (how? see above)
  - is numerically stable and fast to be rebuilt / updated
  - can tackle non-linear problems
  - yields a measure of confidence in its own prediction
  - can be either a classifier or a regressor

ML, specifically for I.D.

- so, which one is better? it depends on
  - speed of rebuilding the model
  - need to store past pairs vs. memory requirements
  - …

- in general, our ideal ML method for I.D.
  - is **incremental** / decremental (how? see above)
  - is numerically stable and **fast** to be rebuilt / updated
  - can tackle **non-linear** problems
  - yields a **measure of confidence** in its own prediction
  - can be either a **classifier** or a **regressor**

The Intent Detection pipeline



Classification

- mapping D-dimensional signals to an action.

$$y = f(\boldsymbol{x}), \boldsymbol{x} \in \mathbb{R}^D, y \in \{a_1, \dots, a_K\}$$

- this is typical of all classification procedures.
- classification: a function mapping an input space to an element of a discrete set of *labels / classes*
- alternatively: a function implicitly defining a *separating sub-manifold* of the input space
    - two-class case, easily generalised to K classes

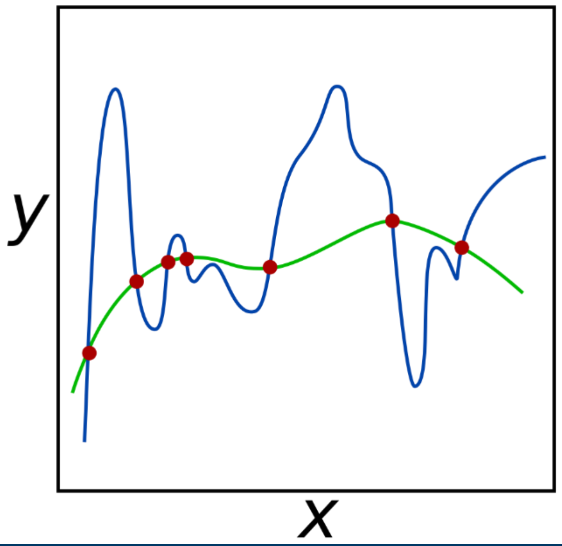
Classification

- how do we find a good f? first of all decide the set $\mathcal{H}$ of candidate fs (*hypothesis space*)
  - linear functions?
  - polynomial?
  - Gaussians?

- then search for the best $f \in \mathcal{H}$, where *best* means the one which:
  1. incurs the smallest error *(loss)* on the data we have, and
  2. does not behave oddly on whatever else (it is *regular*)

- item 1: *avoid **under**fitting!*
  - let my model be complex enough to accommodate for what I already know

- item 2: *avoid **over**fitting!*
  - let my model be regular enough to accommodate for what I do not know yet



Classification

- the appropriate f^* is found by minimising a *cost functional* generally looking like this:

$$f^* = \arg\min_f [\mathcal{L}(f, X, \boldsymbol{y}) + \mathcal{R}(f)]$$

- where $\mathcal{L}$ is the loss and $\mathcal{R}$ is a regulariser.

Classification

- the appropriate f^* is found by minimising a *cost functional* generally looking like this:

$$f^* = \arg\min_f [\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y}) + \mathcal{R}(f)]$$

- where $\mathcal{L}$ is the loss and $\mathcal{R}$ is a regulariser.

Classification

- a very simple idea: let f be linear, $y = \boldsymbol{w}^T\boldsymbol{x}$ with $\boldsymbol{x}, \boldsymbol{w} \in \mathbb{R}^D$, and find the optimal one by
  - *(loss)* minimising the mean squared error between $f(\boldsymbol{x})$ and y:
    $$\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{n}\|X\boldsymbol{w} - \boldsymbol{y}\|^2$$
  - *(regulariser)* minimising the magnitude of $\boldsymbol{w}$:
    $$\mathcal{R}(f) = \|\boldsymbol{w}\|^2$$
  - that is,
    $$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \left[\frac{1}{n}\sum_{i=1}^{N}\|\boldsymbol{w}^T\boldsymbol{x}_i - y_i\|^2 + \lambda\|\boldsymbol{w}\|^2\right]$$
- you then classify your new $\boldsymbol{x}$s according to the sign of f, that is, $y = \operatorname{sign}(\boldsymbol{w}^{*T}\boldsymbol{x})$
- f ends up being a linear manifold embedded in $\mathbb{R}^{D+1}$
  - implicitly defining, e.g., a straight line (when $D = 2$),
  - a plane (when $D = 3$),
  - …
- and this manifold is your separating (hyper)surface.

Classification

- notice that in this case we don‘t need minimise anything, because the solution to the problem exists in closed form!
- the problem is to find $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \left[\frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{w}^T\boldsymbol{x}_i - y_i\|^2 + \lambda\|\boldsymbol{w}\|^2\right]$
- solve by zero-ing the differential over $\boldsymbol{w}$:
$$\frac{\partial}{\partial\boldsymbol{w}}\left(\sum_{i=1}^{N}\|\boldsymbol{w}^T\boldsymbol{x}_i - y_i\|^2 + \lambda\|\boldsymbol{w}\|^2\right) = 0$$
- in the end we obtain that – using linear regression for classification
$$\boldsymbol{w}^* = (X^TX + \lambda I)^{-1}\,X^T\boldsymbol{y}$$
- which is exactly the optimal $\boldsymbol{w}$, yielding the optimal f (given λ): $y = \mathrm{sign}(\boldsymbol{w}^{*T}\boldsymbol{x})$
- Linear classification is discussed above y {-1,1}, used when we have a linearly separable problem

Classification

- that the solution can be *computed* and not *evaluated* by optimisation is good news.

- moreover, this classifier works even in the non-linear case (non-linear in the $\boldsymbol{x}$s, but still linear in the $\boldsymbol{w}$s of course!)

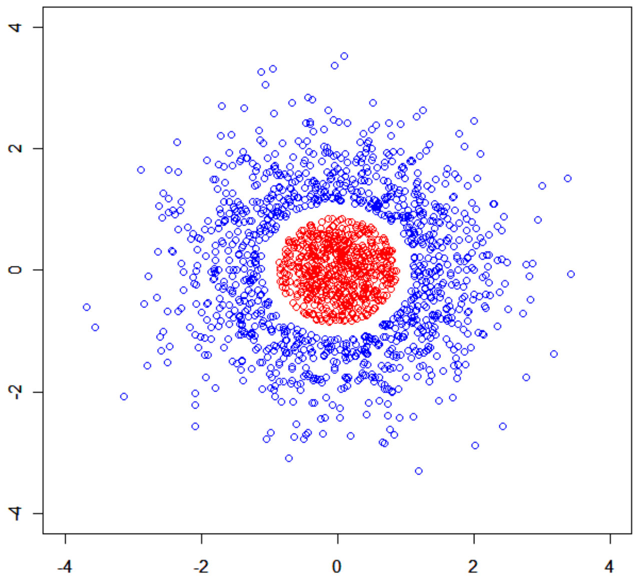
$$\textbf{find } \boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \left[\frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{w}^T\boldsymbol{x}_i - y_i\|^2 + \lambda\|\boldsymbol{w}\|^2\right]$$
$$\textbf{obtain } \boldsymbol{w}^* = (X^T X + \lambda I)^{-1} X^T \boldsymbol{y}$$
$$\textbf{use } y = \operatorname{sign}(\boldsymbol{w}^{*T}\boldsymbol{x})$$

- can be directly rewritten like this: let $\Phi \triangleq \phi(X)$, where ϕ can be any non-linear mapping then

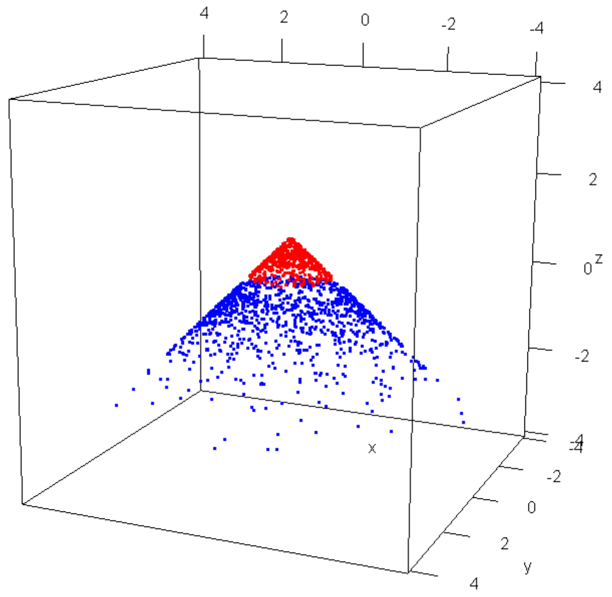
$$\textbf{find } \boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \left[\frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{w}^T\phi(\boldsymbol{x}_i) - y_i\|^2 + \lambda\|\boldsymbol{w}\|^2\right]$$
$$\textbf{obtain } \boldsymbol{w}^* = (\Phi^T \Phi + \lambda I)^{-1} \Phi^T \boldsymbol{y}$$
$$\textbf{use } y = \operatorname{sign}(\boldsymbol{w}^{*T}\phi(\boldsymbol{x}))$$

Classification

- this can be directly applied to the example in lecture 8!
- rather than solving a complex problem in d dimensions, use ϕ then solve in D dimensions using the same machinery for linear classification!

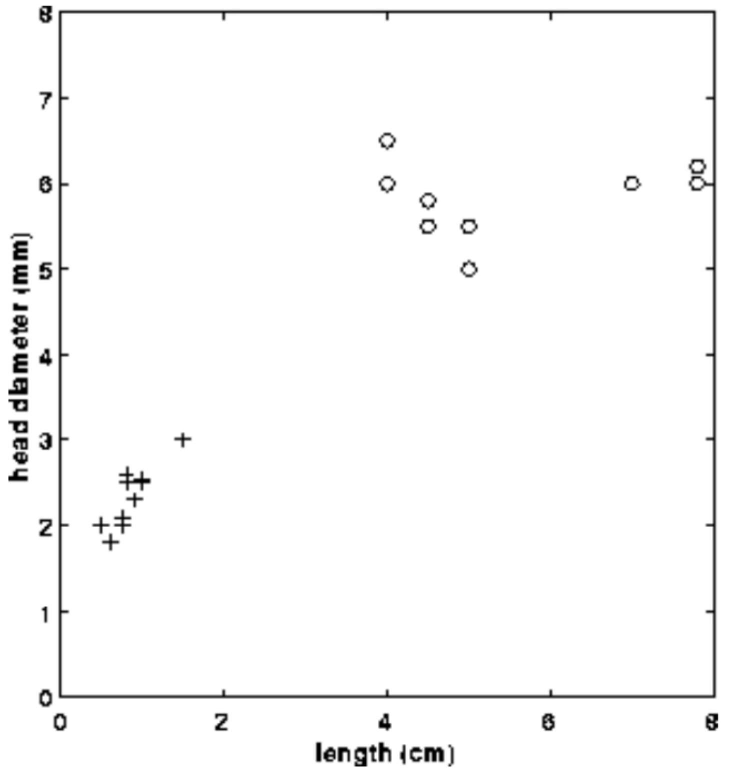


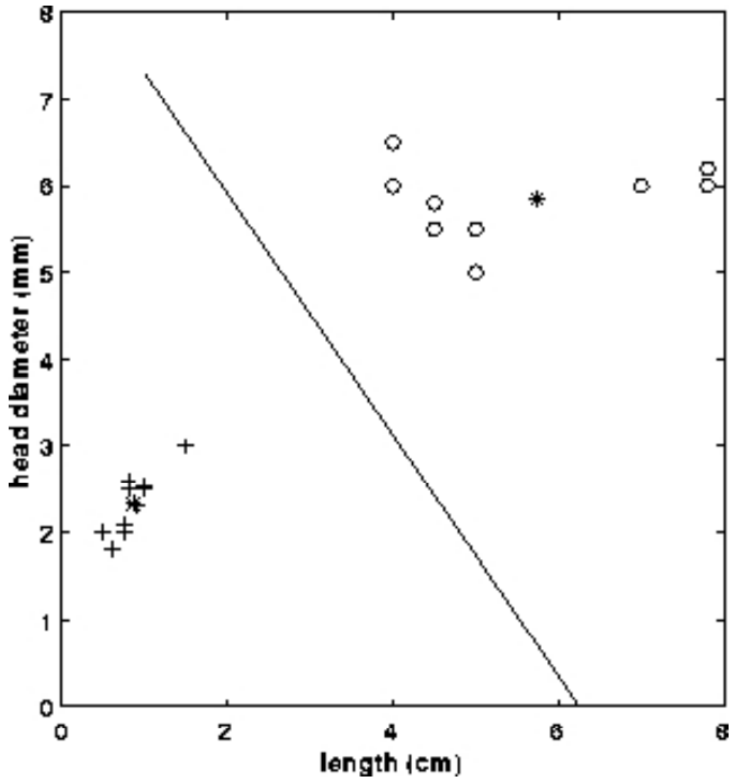
$$\phi(x_1, x_2) = (x_1, x_2, -\|x\|) \Longrightarrow$$



Classification

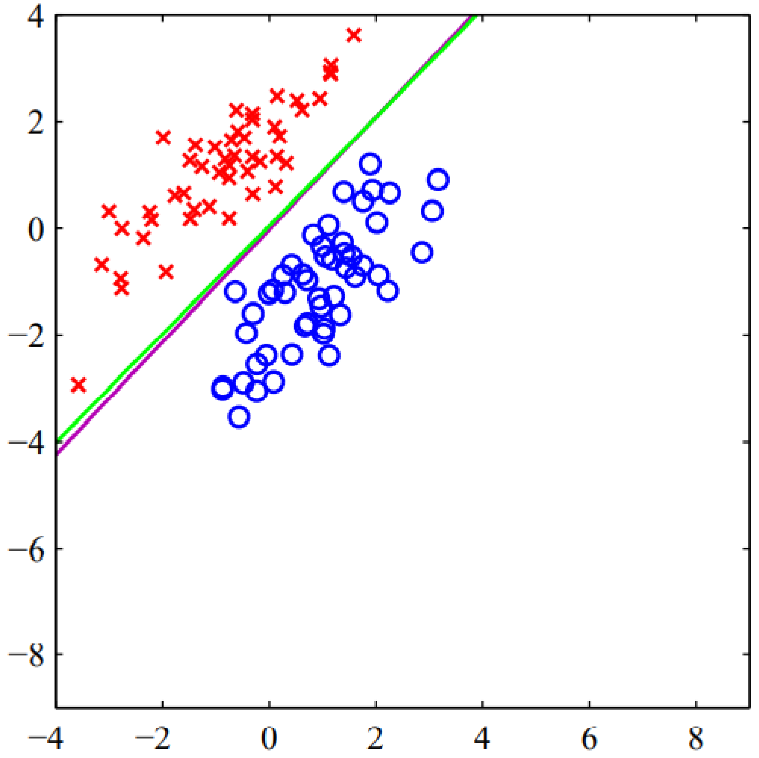
- now for something simpler.
- here‘s an *extra simple* idea: use the centroids of the classes (Prototype classifier)

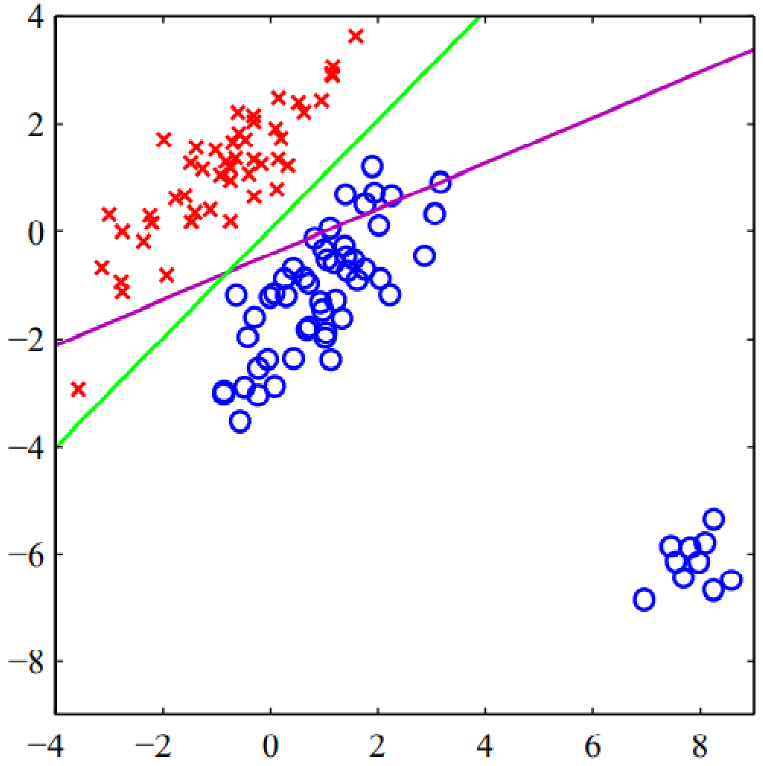




Classification

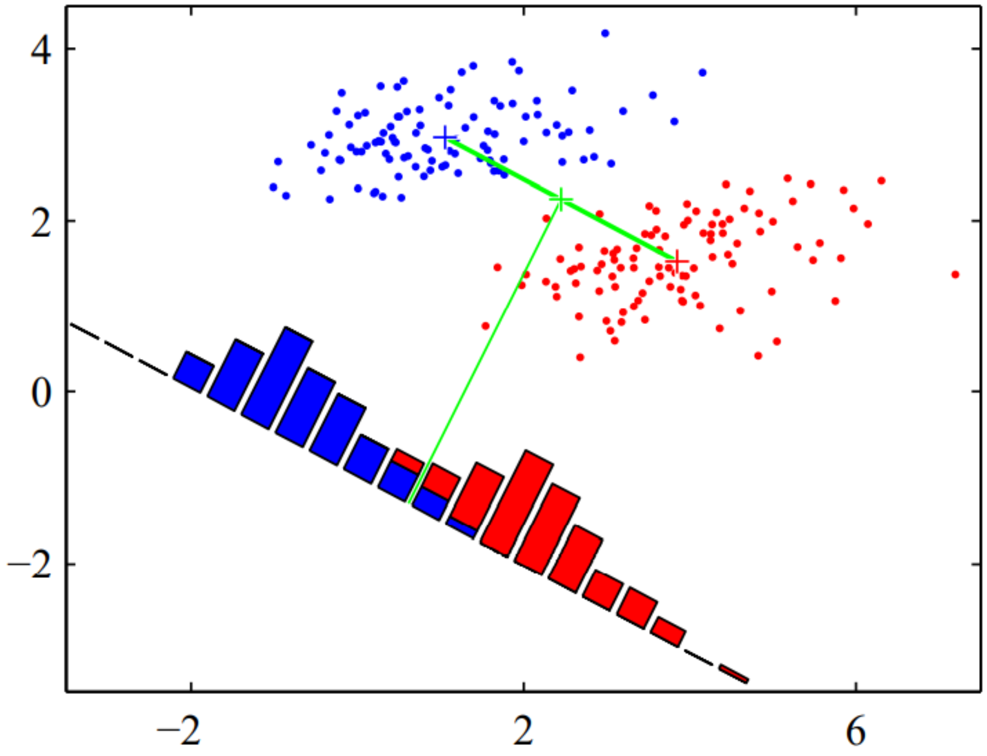
- problem: prone to outliers, or non-normally distributed datasets





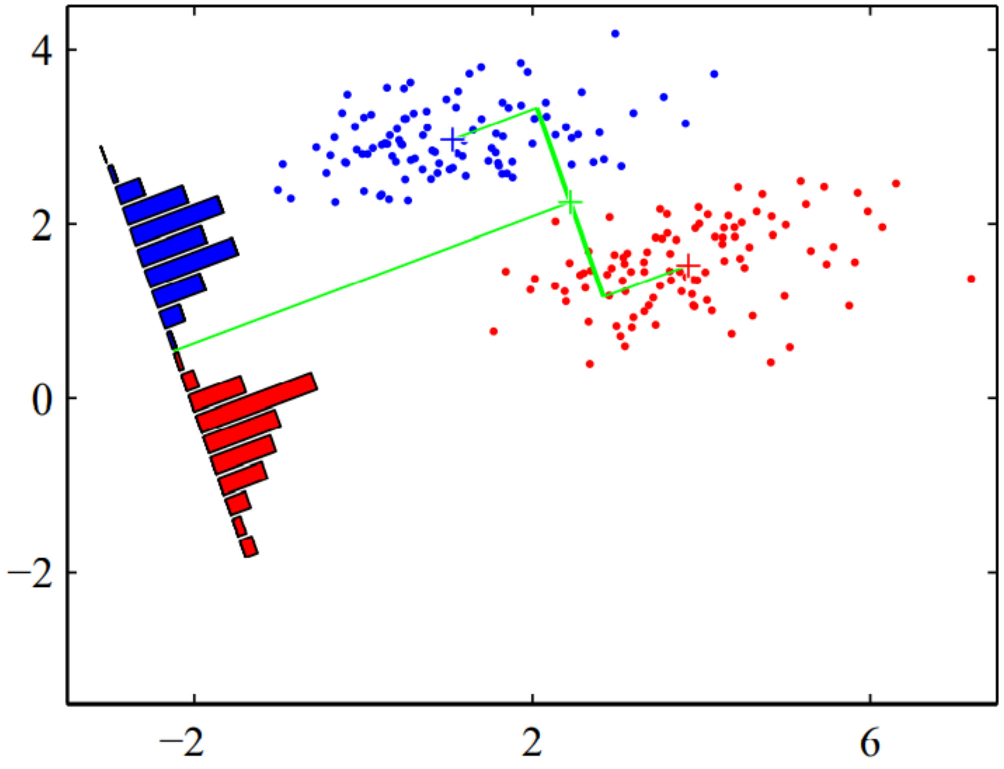
Classification

- a better idea: Fisher‘s classifier a.k.a. LDA (Linear Discriminant Analysis):



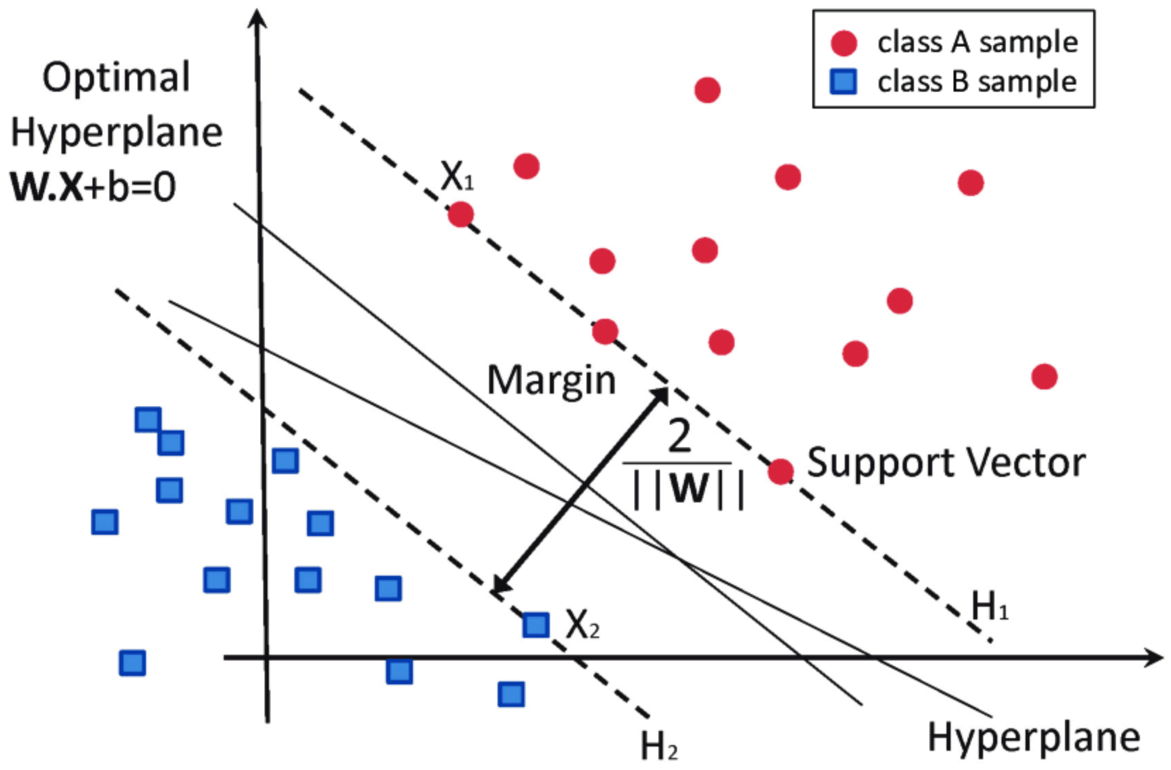
Classification

- a better idea: Fisher‘s classifier a.k.a. LDA (Linear Discriminant Analysis):



Classification

- another idea: draw a line according to the ambiguous samples (support vectors):



Classification

- recall Lecture #08:
- literally dozens of classification approaches have been tried on EMG

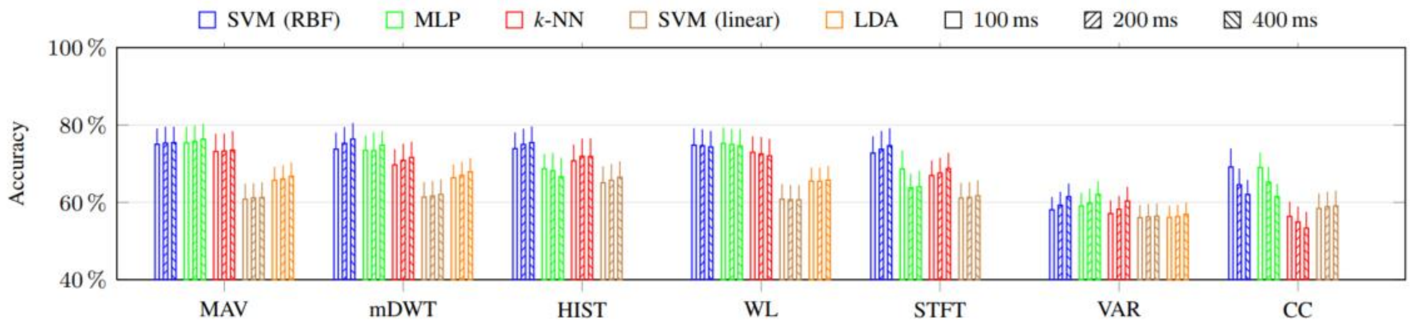


Fig. 2. Classification accuracies. Each bar represents method classification accuracy with respect to feature representation and window length, while line atop the bar is one standard deviation of accuracy. Classifiers are grouped by feature representations and labeled by different colors. Window lengths are represented in increasing order, namely 100ms, 200ms and 400ms and are tagged with different textures. Note, that LDA results are missing in case of STFT, CC and HIST due to reasons described in Section IV.

- has this brought us any closer to a solution?

Ilja Kuzborskij, Arjan Gijsberts, and Barbara Caputo, On the Challenge of Classifying 52 Hand Movements from Surface Electromyography, 2012

Classification

- what good is classification?

- **it is easy to enforce**
  - ground truth very simple to represent
- **it gives a definite answer**
  - either action 1 or 2 or 3 or …

- **it can become unstable**
- **it constrains the user to a set of a few discrete choices**

Summary

- today:
- on interaction, and incrementality
  - two „types“ of incrementality, a „fake“ one and a „true“ one
  - Fake – rebuilds from scratch
  - True – updates the model
  - take-home lesson: any ML method is incremental in principle!
- classification

References

- Patel, G., Nowak, M., & Castellini, C. (2017). Exploiting knowledge composition to improve real-life hand prosthetic control. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 25(7), 967–975.
- Ilja Kuzborskij, Arjan Gijsberts, and Barbara Caputo, On the Challenge of Classifying 52 Hand Movements from Surface Electromyography, 2012