Solving classification problems with HD computing/VSA

Neuroscience 299: Module 9
October 27, 2021

Laura I. Galindez Olascoaga

Berkeley UNIVERSITY OF CALIFORNIA

Berkeley Wireless Research Center

About me

- BS Mechatronics Engineering: Tecnológico de Monterrey, Mexico, 2012.
- MS Systems and Control: Eindhoven University of Technology, The Netherlands, 2015.
- PhD Electrical Engineering: KU Leuven, Belgium, 2020. Dissertation: Hardware Aware Probabilistic Machine Learning Models.
- Postdoc at Berkeley Wireless Research Center working with Prof. Jan Rabaey since February 2021.

Outline

- Motivation and goals
- Background
- Classification with HD computing/VSAs
- Application Examples
- Useful resources and questions

Motivation and goals

Why should we discuss classification problems?

- Classification is a widely used task in many application domains.
- Amenable to the practical implementation of many of the concepts studied so far.
- Variety of implementations and compelling results are the outcome of interdisciplinary interest in the topic of HD computing/VSAs.

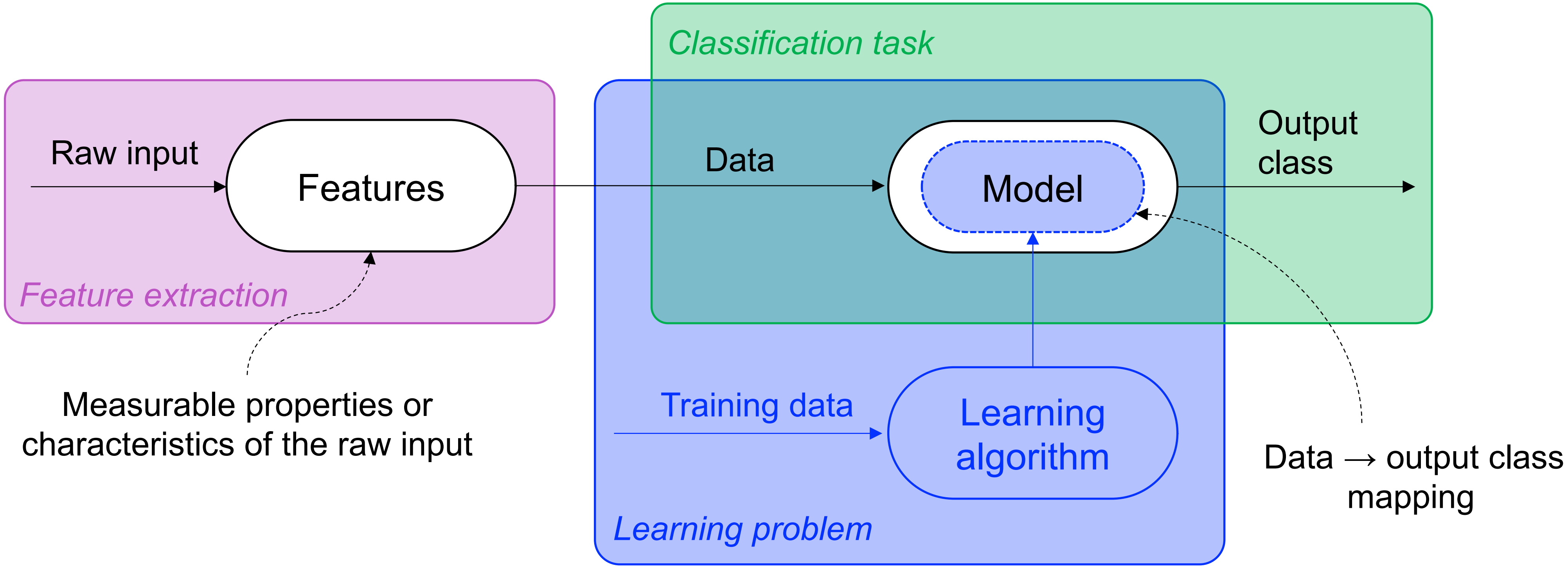
Goals for this lecture

- An overview of how HD computing/VSAs can be used to solve a classification problem.
- Highlight the different approaches available in the literature for each step of the "classification pipeline".
- Encourage you to think of how other properties or traits of HD computing/VSAs can be exploited in the classification realm and beyond.

Background

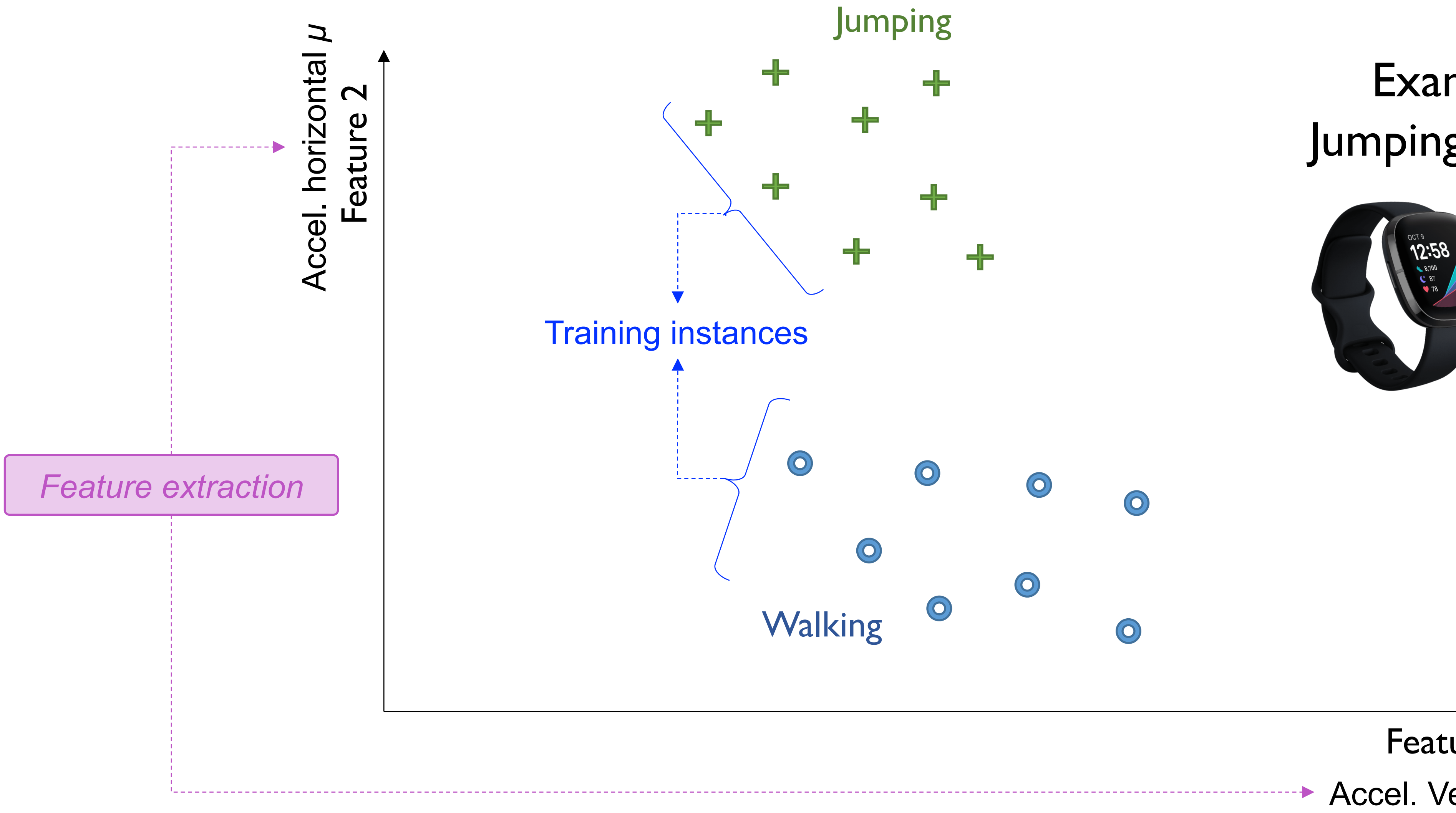
The “classification pipeline”

The classification pipeline in Machine Learning

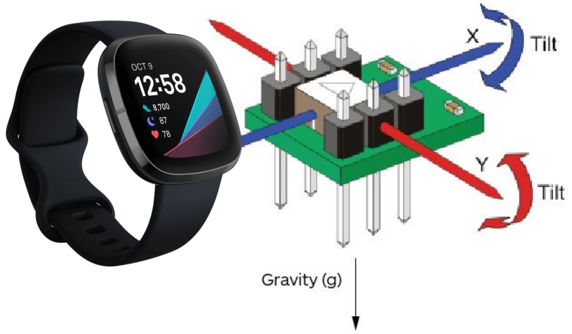


Based on Flach, P. (2012). *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press and Galindez Olascoaga, L. I.. Meert, W., & Verhelst, M (2021). Hardware-Aware Probabilistic Machine Learning Models. Springer, Cham.

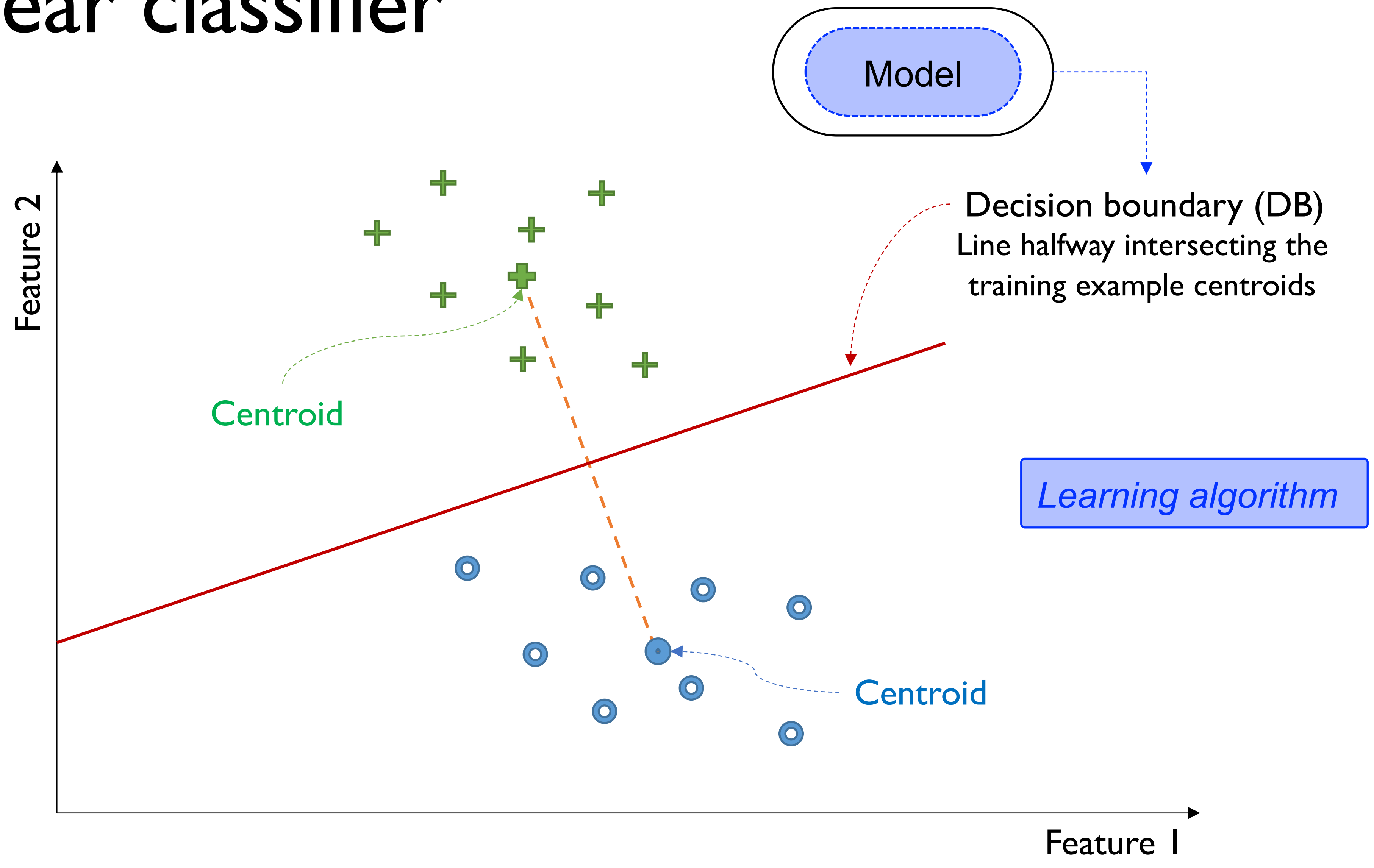
Basic linear classifier



Example task:
Jumping or walking?

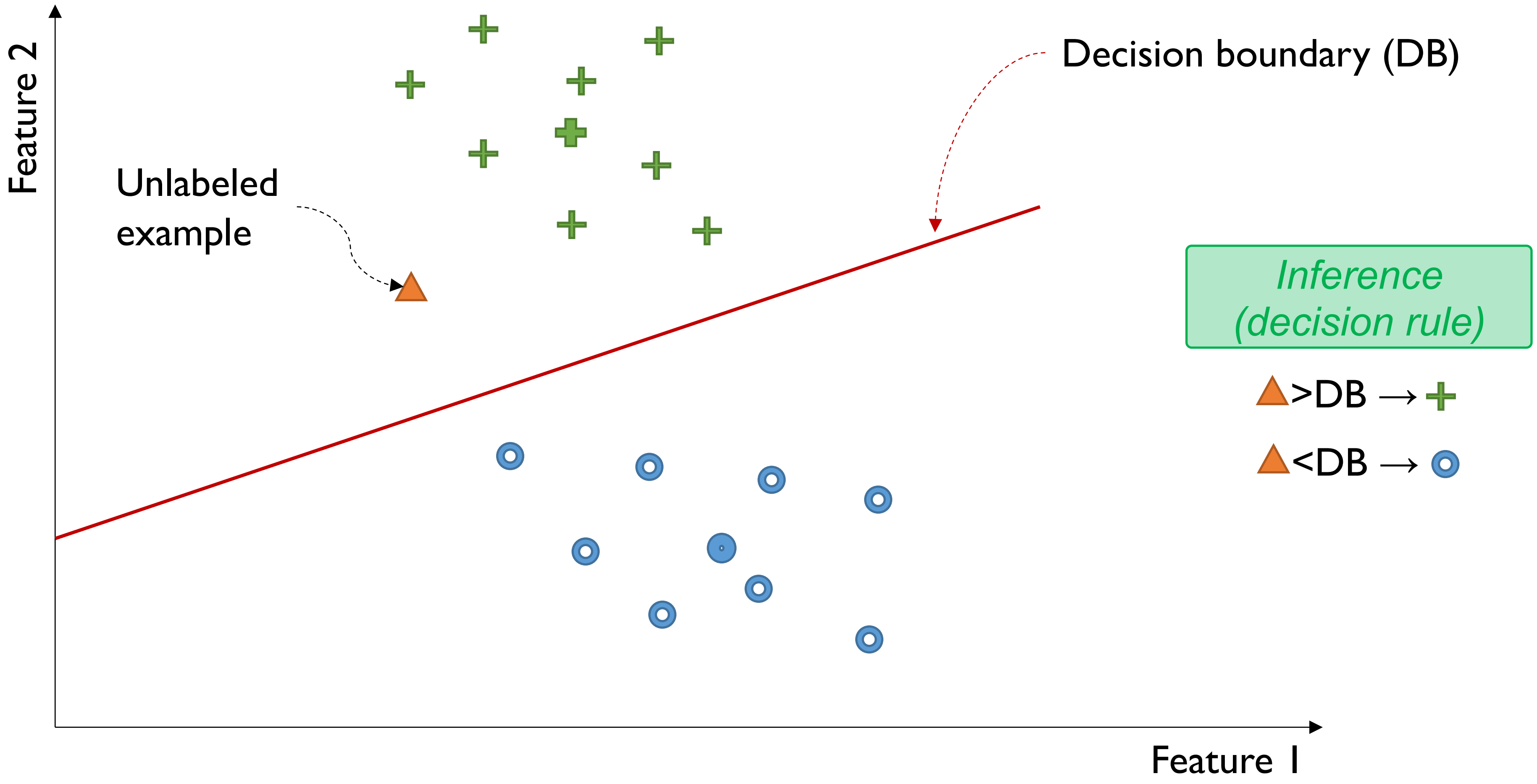


Basic linear classifier



Example from Flach, P. (2012). *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press.

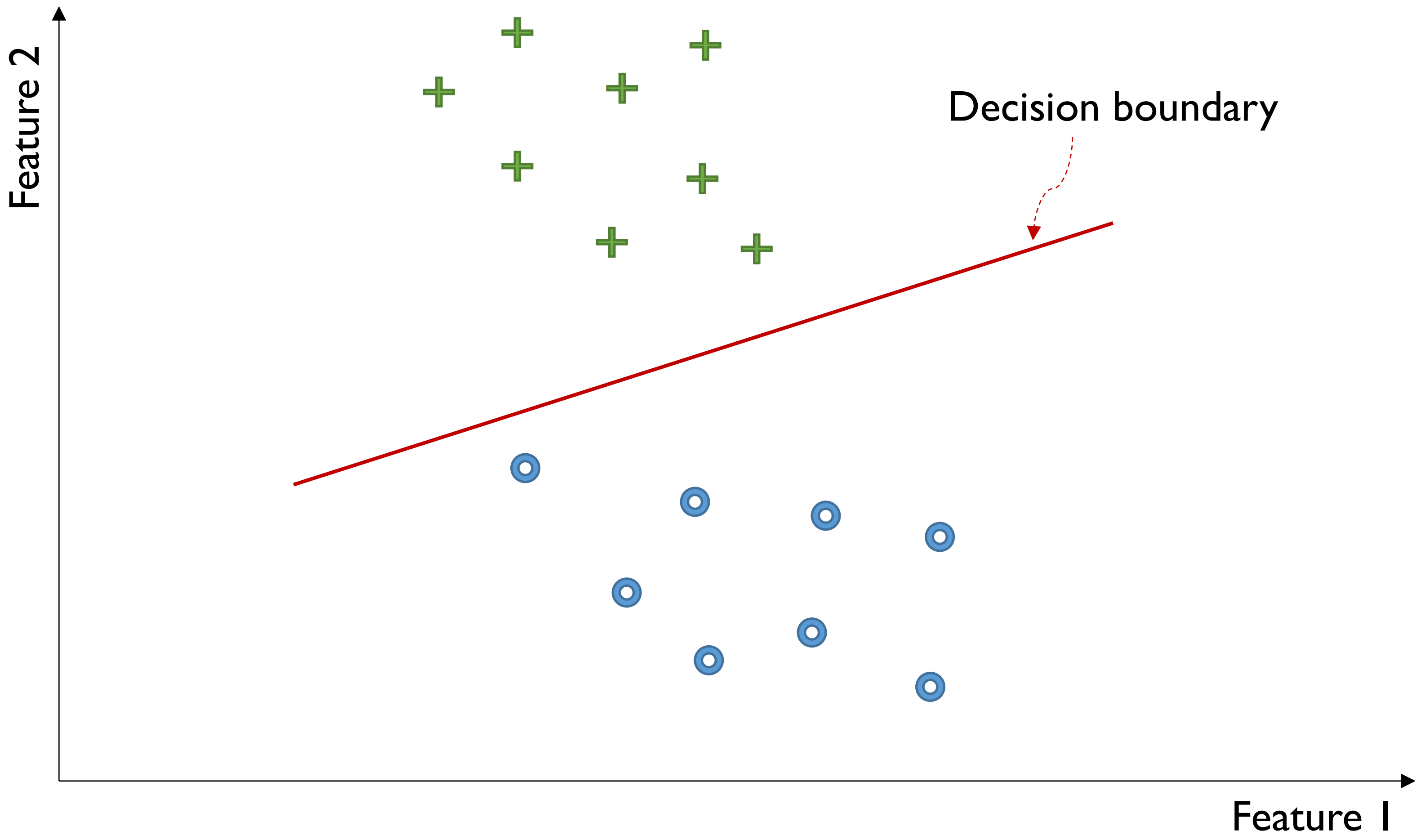
Basic linear classifier



Types of approaches/models*

- **Geometric**

Linear classifier

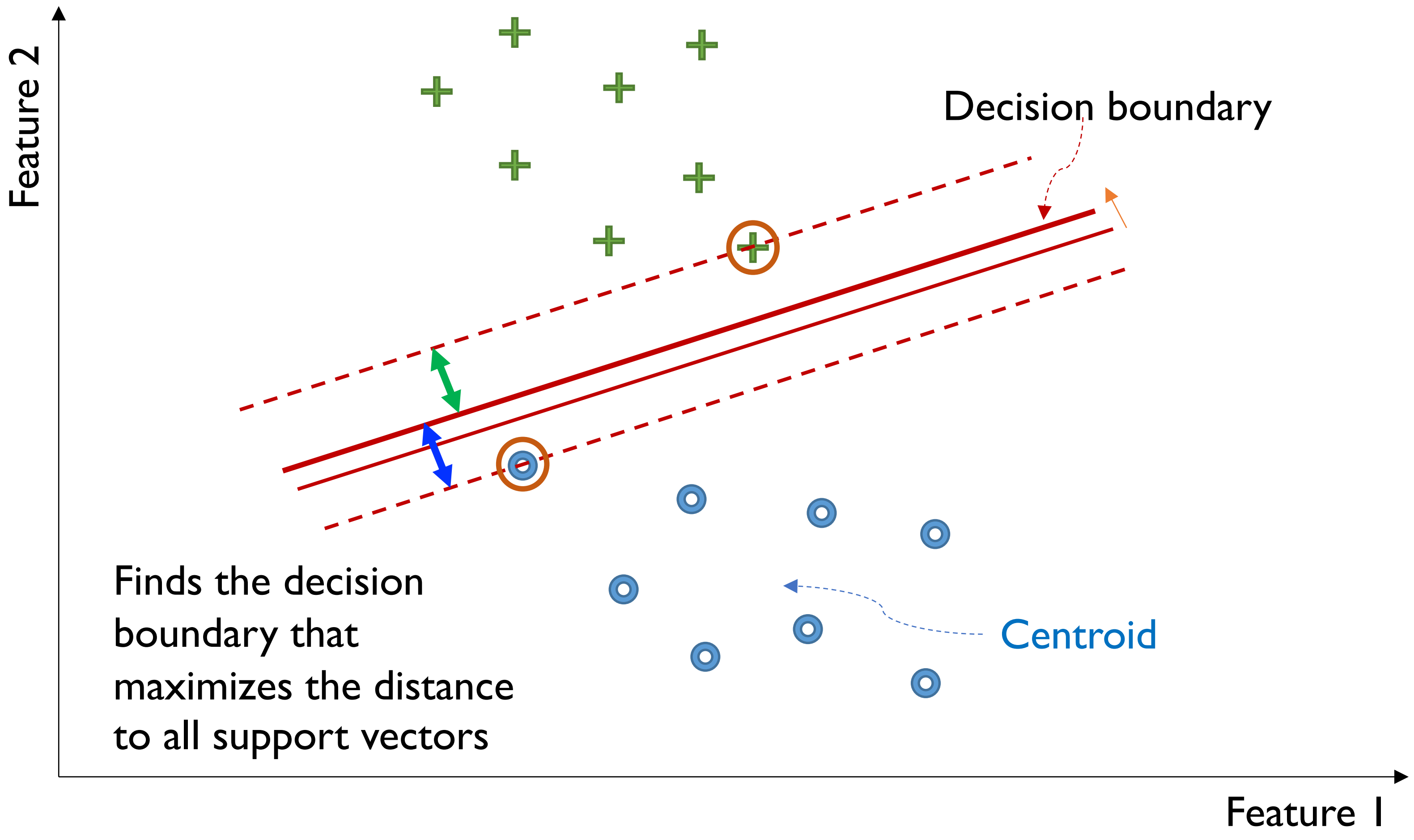


*Non exhaustive and non mutually exclusive list based on Flach, P. (2012). *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press.

Types of approaches/models*

- **Geometric**

Support Vector Machine

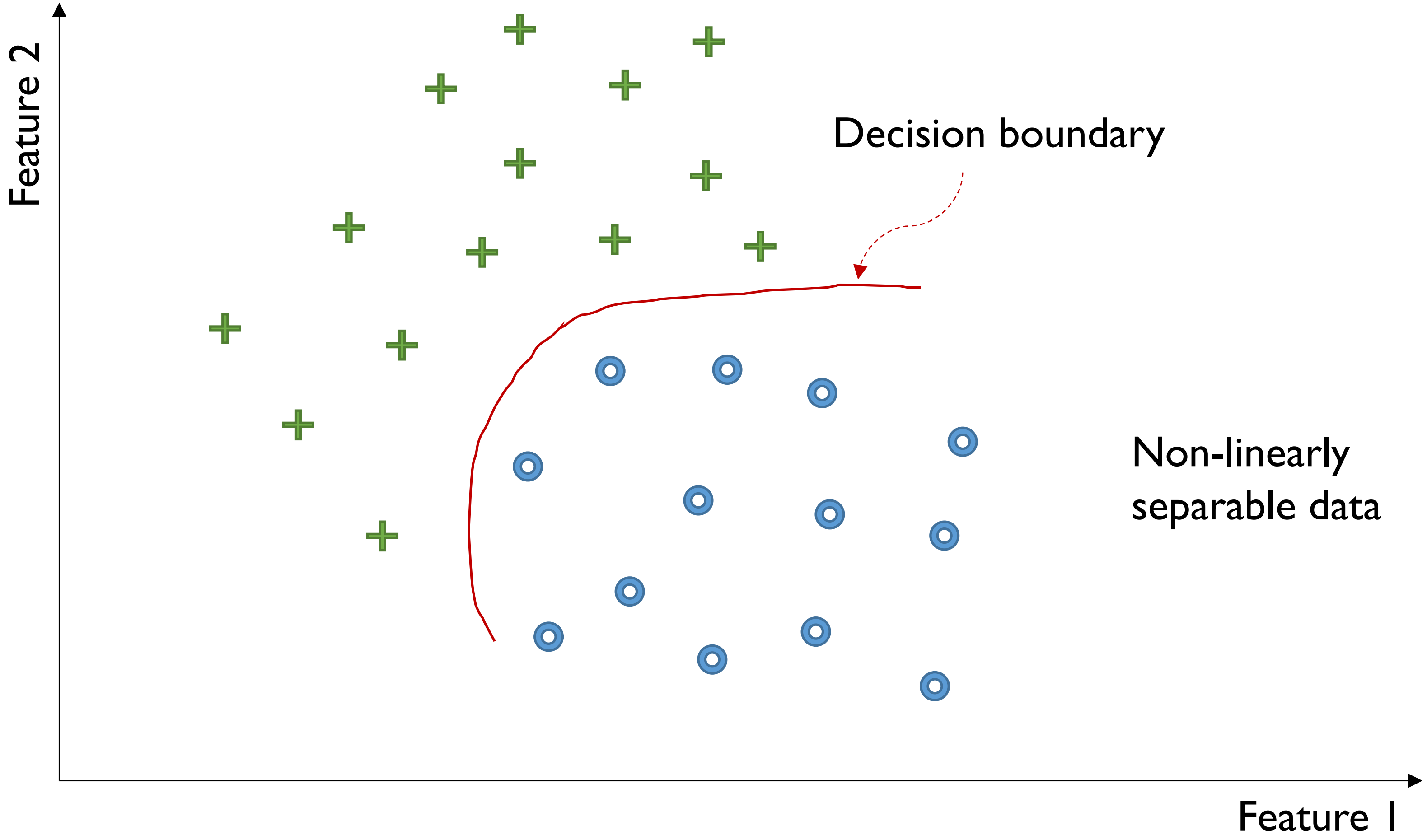


*Non exhaustive and non mutually exclusive list based on Flach, P. (2012). *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press.

Types of approaches/models*

- **Geometric**

(Multilayer) perceptron,
Deep neural networks

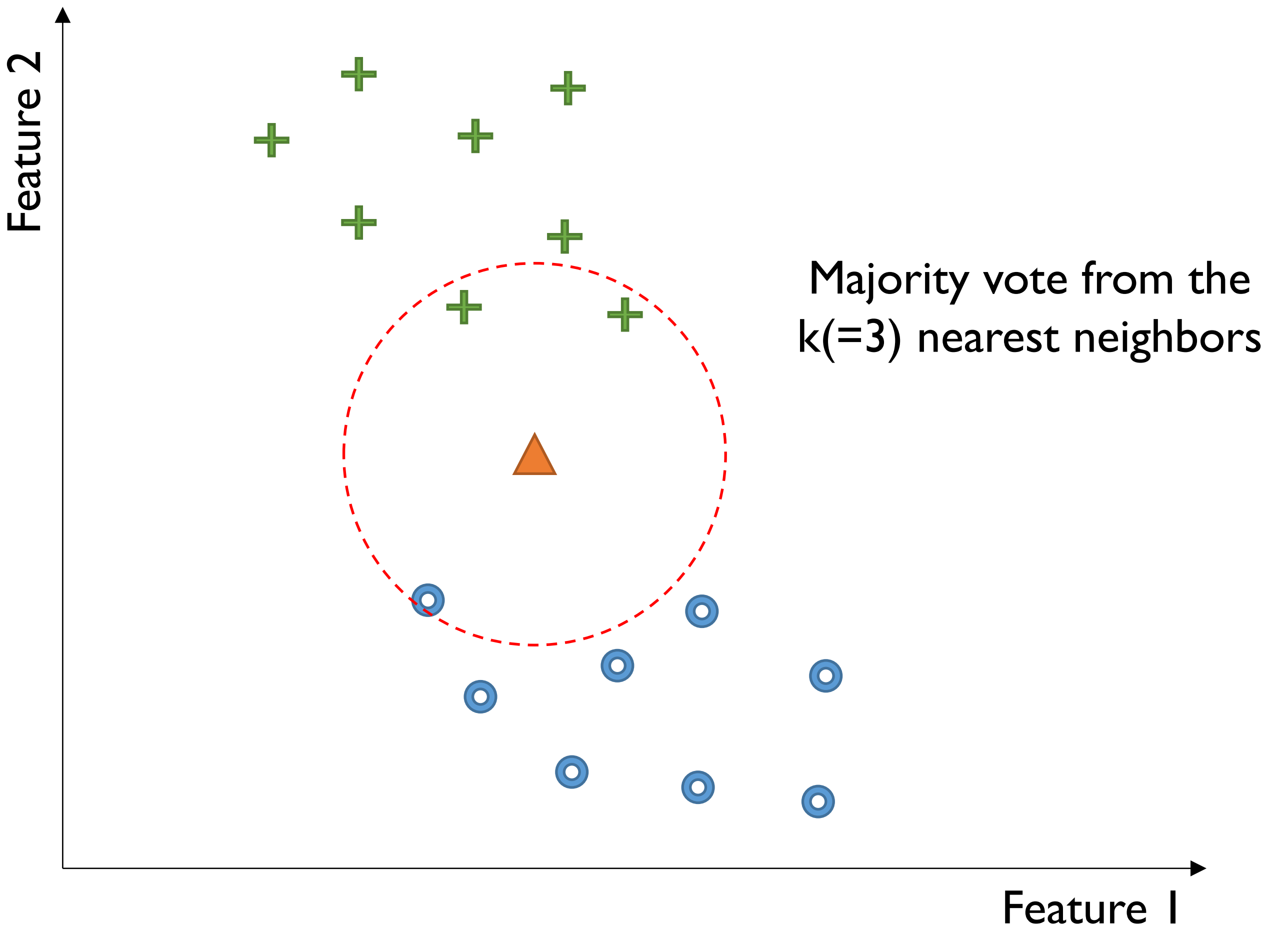


*Non exhaustive and non mutually exclusive list based on Flach, P. (2012). *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press.

Types of approaches/models*

- Geometric
- **Distance based (geometric)**

Nearest neighbor classifiers
(e.g. K nearest neighbors)

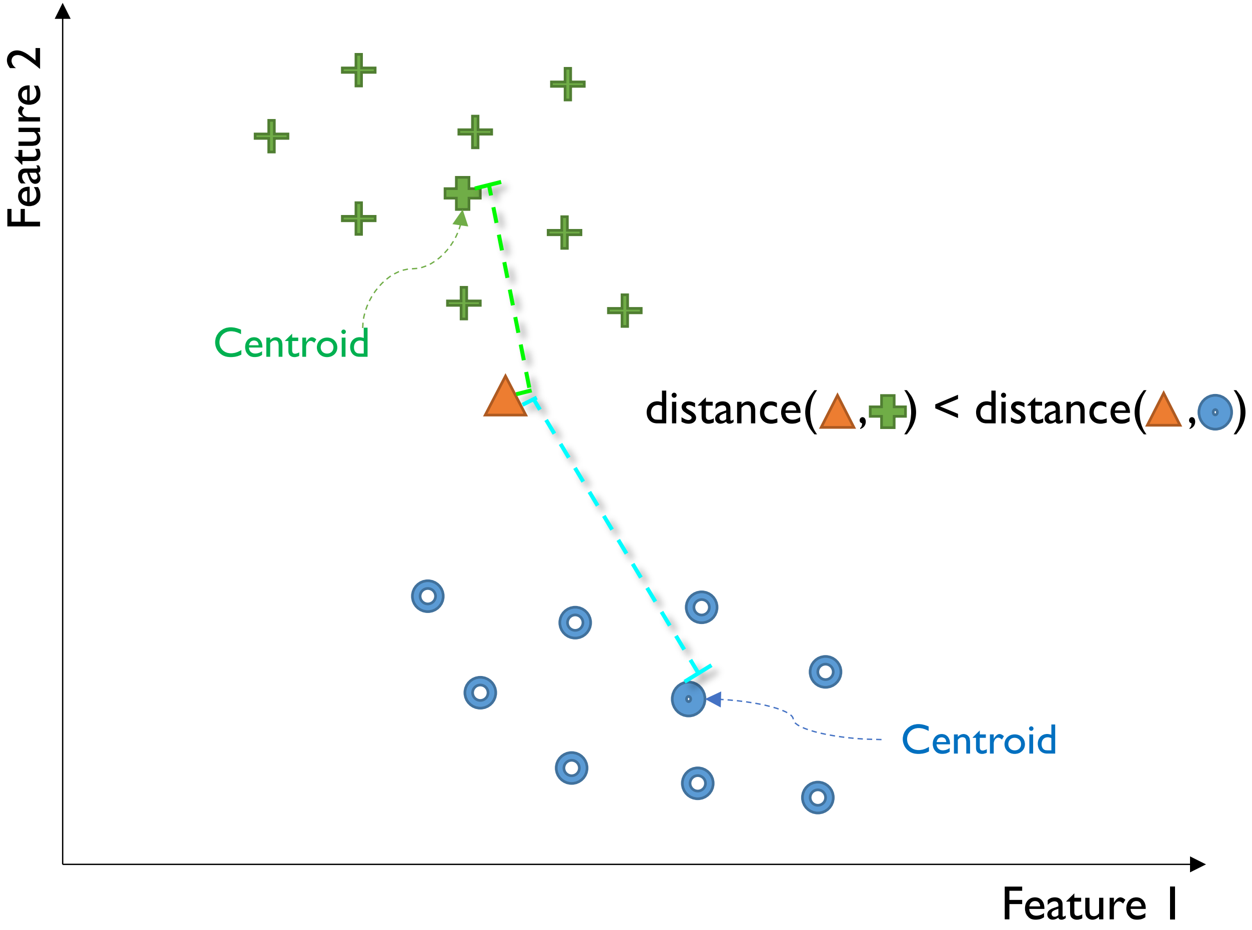


*Non exhaustive and non mutually exclusive list based on Flach, P. (2012). *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press.

Types of approaches/models*

- Geometric
- **Distance based (geometric)**

Nearest neighbor classifiers
(e.g. nearest centroid)

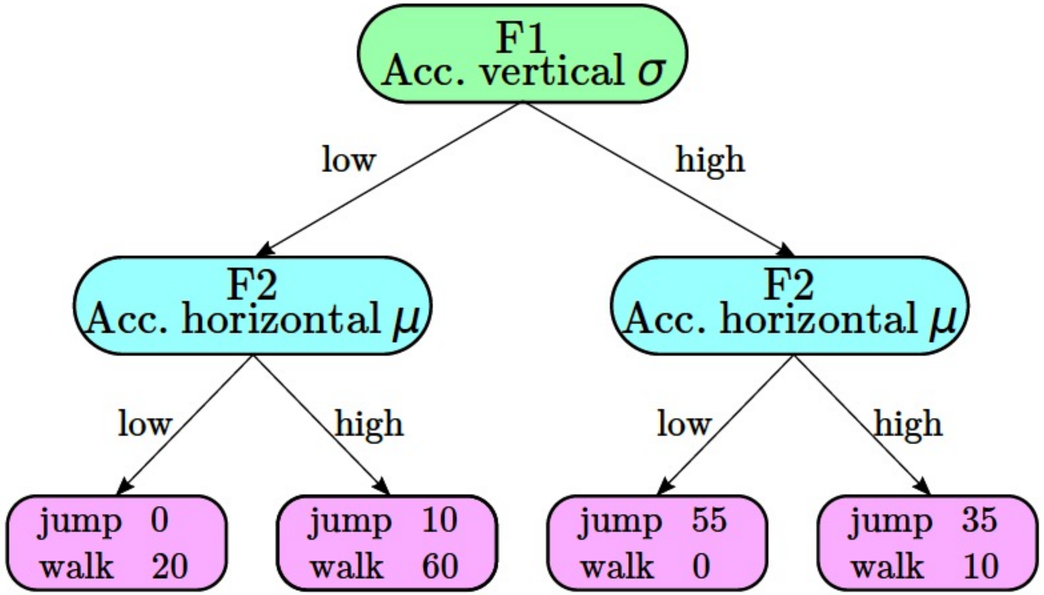


*Non exhaustive and non mutually exclusive list based on Flach, P. (2012). *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press.

Types of approaches/models*

- Geometric
- Distance based (geometric)
- **Logical**

Decision tree



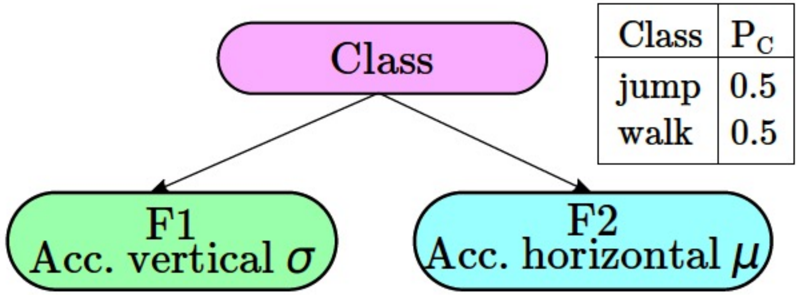
Taken from Galindez Olascoaga, L. I.. Meert, W., & Verhelst, M (2021). Hardware-Aware Probabilistic Machine Learning Models. Springer, Cham.

*Non exhaustive and non mutually exclusive list based on Flach, P. (2012). *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press.

Types of approaches/models*

- Geometric
- Distance based (geometric)
- Logical
- **Probabilistic**, etc.

Bayesian network classifiers



Class	P_C
jump	0.5
walk	0.5

Class	F1	P_{F1}
jump	low	0.1
jump	high	0.9
walk	low	0.7
walk	high	0.3

Class	F2	P_{F2}
jump	low	0.6
jump	high	0.4
walk	low	0.2
walk	high	0.8

$$\Pr(C{=}jump|F1{=}high,F2{=}low)$$
vs
$$\Pr(C{=}walk|F1{=}high,F2{=}low)$$

Taken from Galindez Olascoaga, L. I.. Meert, W., & Verhelst, M (2021). Hardware-Aware Probabilistic Machine Learning Models. Springer, Cham.

*Non exhaustive and non mutually exclusive list based on Flach, P. (2012). *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press.

Types of approaches/models*

- Geometric
- Distance based (geometric)
- Logical
- Probabilistic, etc.

- What kind of classification can HD computing/VSAs enable?

*Non exhaustive and non mutually exclusive list based on Flach, P. (2012). *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press.

Classification with HD computing/VSAs

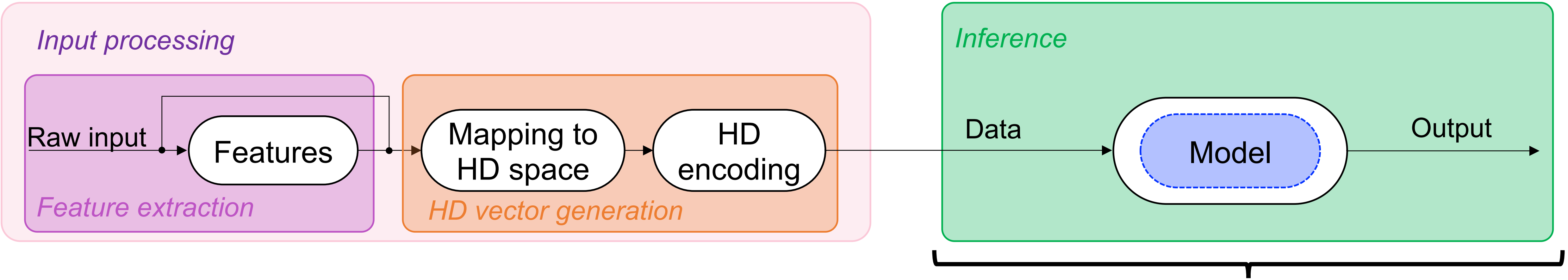
Overview

Main components of HD computing/VSAs

- High-dimensional vectors (HD vectors, hypervectors, HV) are approximately orthogonal to each other.
- Hyperdimensional arithmetic comprises three simple operators:
  - Bundling or superposition $+$
  - Binding: typically, a multiplicative operation $\otimes$
  - Permutation ρ
- Similarity metrics: dot product, Hamming distance, etc. distance(.)
- Associative memory and item memory (codebook).

Role of HD computing/VSAs in classification pipeline

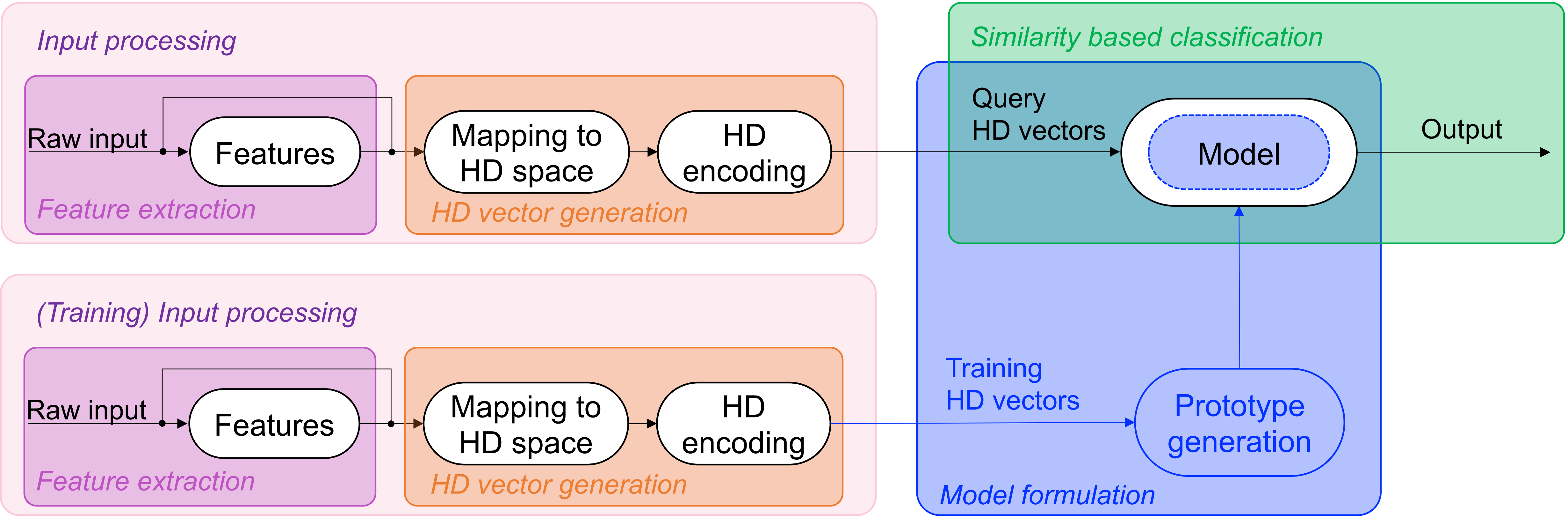
- Input data representation and encoding.



Can be done with HD computing/VSAs or within a hybrid scheme with other models. E.g. Module 10 on Relations to Neural Networks by Denis.

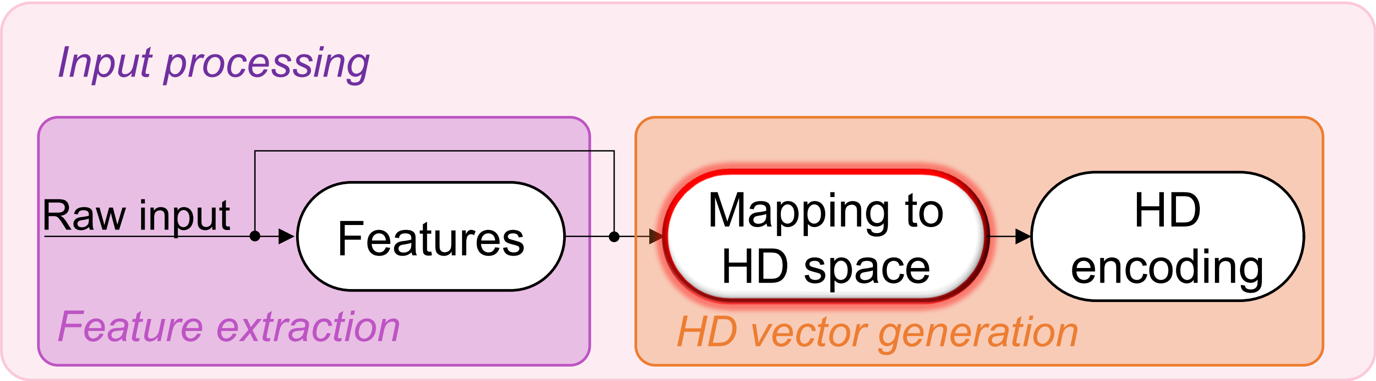
Role of HD computing/VSAs in classification pipeline

- Model formulation with HD computing/VSAs.



Classification with HD computing/VSAs

1. Mapping inputs to HD space



Mapping inputs to HD space

Determined by the type of input:

- **Symbolic or categorical**: Can be described by a finite alphabet of independent values or symbols. For example, letters in the alphabet (examples from Modules 1 by Pentti and Module 3 by Ryan).

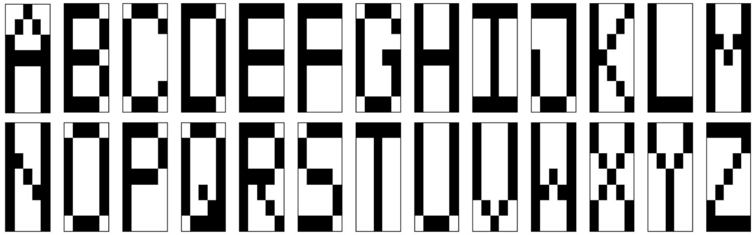
- **Real-valued**: Continuous input data or discrete variables with correlated values. For example, accelerometer signals (see also Module 8 by Chris).

Mapping symbolic inputs to HD space

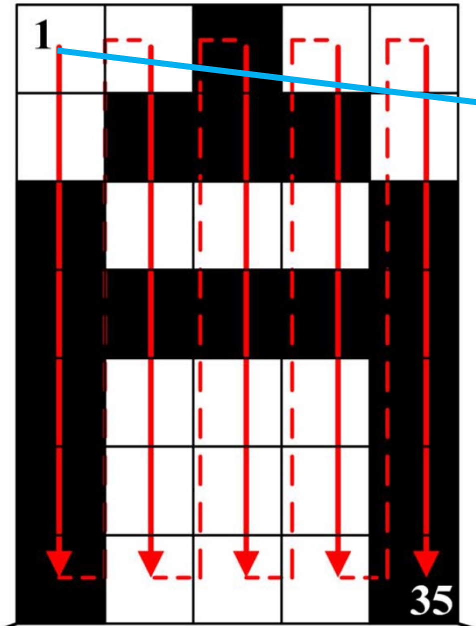
- Orthogonal mapping: a unique HD vector, randomly chosen, is assigned to each symbol.
- These unique HD vectors are saved in an *item memory* (I.M.) and remain fixed for the rest of the system execution.

Mapping symbolic inputs to HD space

Example 1: character recognition



Each pixel is treated as a feature. Features are assumed to be uncorrelated.



Each feature is assigned a randomly generated binary HD vector, these are saved in *I.M.*

Pix1	1	0	0	1	0	0	0	1	1	1

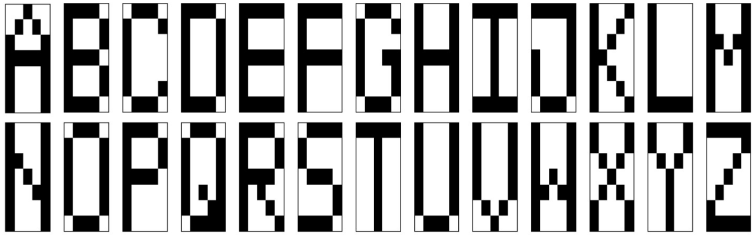
Black	1	0	0	1	0	0	0	1	1	1
White	1	1	0	0	1	0	0	0	1	1

Values can be represented by shifting the pixel ID HD vector: 0 bits for black and 1 bit for white.

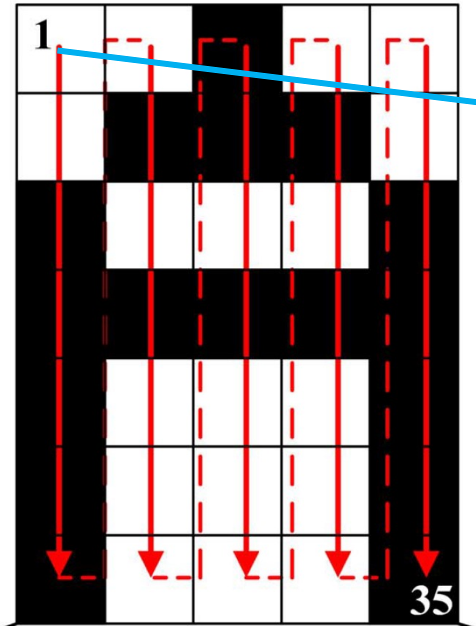
Kleyko, D., Osipov, E., Senior, A., Khan, A. I., & Şekercioğlu, Y. A. (2016). Holographic graph neuron: A bioinspired architecture for pattern processing. *IEEE transactions on neural networks and learning systems*, *28*(6), 1250-1262.

Mapping symbolic inputs to HD space

Example 1: character recognition
(key-value pair approach)



Each pixel is treated as a feature. Features are assumed to be uncorrelated.



Each feature is assigned a randomly generated binary HD vector, these are saved in *PixID I.M.*

Pix1	1	0	0	1	0	0	0	1	1	1

Each value is assigned a randomly generated binary HD vector, these are saved in the *Color I.M.*

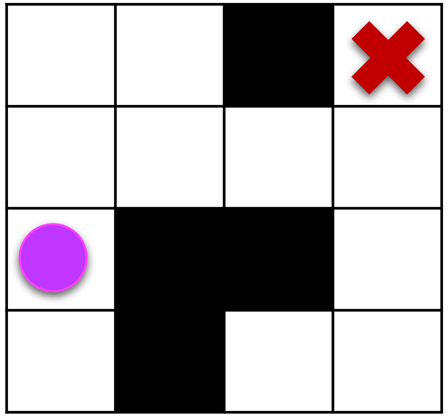
Black	1	1	0	0	0	1	1	0	1	0
White	0	1	1	0	0	1	0	1	0	1

Encoding as {Pixel ID: value} pair: **Pix1** $\otimes$ **White**

Kleyko, D., Osipov, E., Senior, A., Khan, A. I., & Şekercioğlu, Y. A. (2016). Holographic graph neuron: A bioinspired architecture for pattern processing. *IEEE transactions on neural networks and learning systems*, *28*(6), 1250-1262.

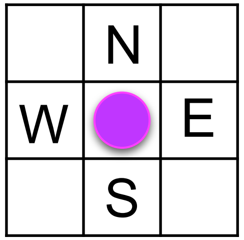
Mapping symbolic inputs to HD space

Example 2: 2D robot navigation



Agent navigating in 2D grid must reach goal while avoiding obstacles.

Sensors: presence or absence of obstacles in each cardinal direction



Sensor ID I.M.

N	1	0	0	1	0	0	0	1	1	1

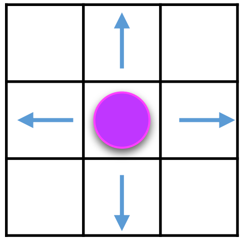
…

W	1	0	1	0	0	1	0	0	1	0

Sensor value I.M.

true	0	1	1	1	0	1	0	1	0	0
false	1	1	0	0	0	1	0	1	0	1

Actuation: the robot can move up, down, right or left



Actuation I.M.

UP	1	1	1	0	0	0	0	1	1	0

…

LEFT	0	1	1	0	1	0	1	0	1	0

Related works:
Levy, Simon D., Suraj Bajracharya, and Ross W. Gayler. "Learning Behavior Hierarchies via High-Dimensional Sensor Projection." *AAAI Workshop: Learning Rich Representations from Low-Level Sensors*. 2013.
P. Neubert, S. Schubert, and P. Protzel, "Learning vector symbolic architectures for reactive robot behaviours," IROS Workshop: *Machine Learning Methods for High-Level Cognitive Capabilities in Robotics*, 2016.

Mapping real-valued inputs to HD space

Real-valued inputs call for locality preserving encodings (LPEs). Introduced in module 8 by Chris.

Approaches I will discuss today:

- Discrete mapping: discretize the real-valued input and assign HD vectors such that they preserve relevant correlations.
- Random projection: directly project real-valued inputs to HD space by means of scalar multiplication. Often used as an intermediate representation.

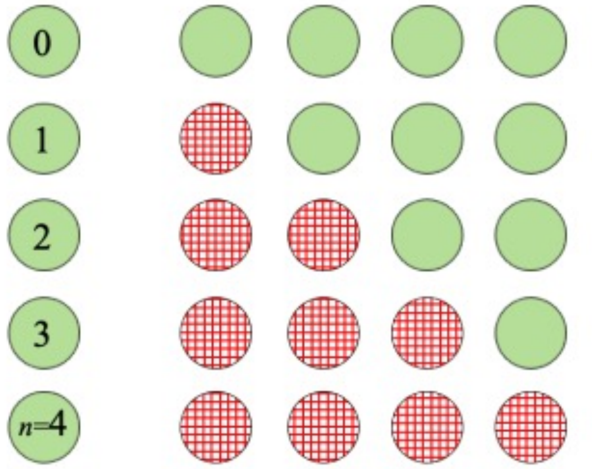
Other powerful approaches, still largely unexplored for practical applications:

- Vector Function Architectures (VFAs), kernel LPEs (in the context of HD computing/VSAs).

Similarity preserving discrete mapping

*Linear similarity** preserving mapping:

1. Discretize real-valued input into m+1 levels.
2. Randomly assign HD vector to level 0 (HV_0).
3. Gradually flip a predefined number of bits starting from HV_0 such that HV_0 and HV_{m+1} are uncorrelated.



*Linear similarity also preserved by e.g. thermometer code. (Taken from Chris's slides, module 8)

Considerations:

- Sample without replacement for the bit flipping.
- Need to store the HD vectors for all levels.

Similarity preserving discrete mapping

Linear similarity preserving mapping:

Example: accelerometer data

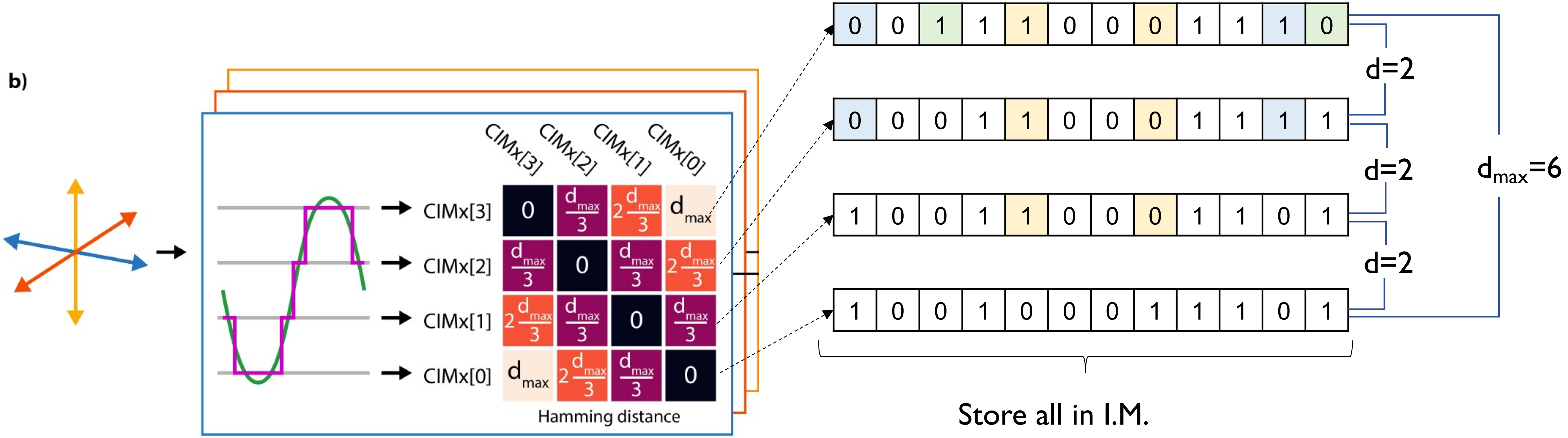


Figure taken from Zhou, A., Muller, R., & Rabaey, J. (2021, March). Memory-Efficient, Limb Position-Aware Hand Gesture Recognition using Hyperdimensional Computing. In *Research Symposium on Tiny Machine Learning*.

Similarity preserving discrete mapping

Approximately linear similarity preserving mapping*:

1. Discretize real-valued input into m+1 levels.
2. Randomly assign HD vectors to the first and last levels HV_0 and HV_{m+1}
3. Construct intermediate levels by concatenating sections of HV_0 and HV_{m+1}.

Considerations:

- Might only need to store first and last level.
- *Linear similarity* preservation not guaranteed.

*Based on the "encoding by concatenation" from Rachkovskiy, Dmitriy A., et al. "Sparse binary distributed encoding of scalars." *Journal of Automation and Information Sciences* 37.6 (2005).

Similarity preserving discrete mapping

Approximately linear similarity preserving mapping:

Example: accelerometer data

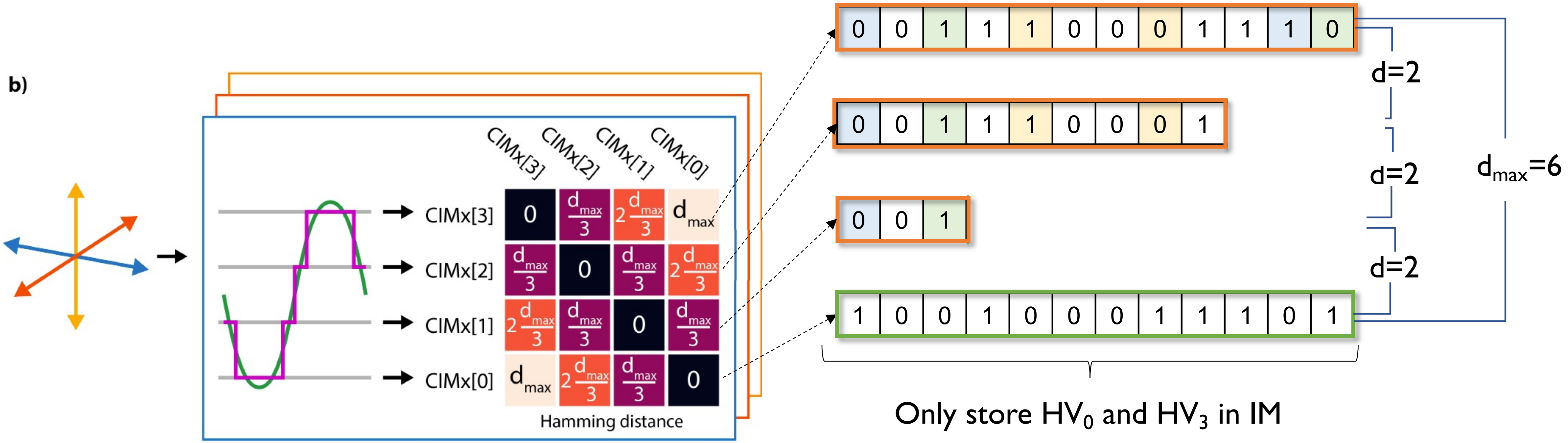


Figure taken from Zhou, A., Muller, R., & Rabaey, J. (2021, March). Memory-Efficient, Limb Position-Aware Hand Gesture Recognition using Hyperdimensional Computing. In *Research Symposium on Tiny Machine Learning*.

Similarity preserving discrete mapping

Approximately linear similarity preserving mapping:

Example: accelerometer data

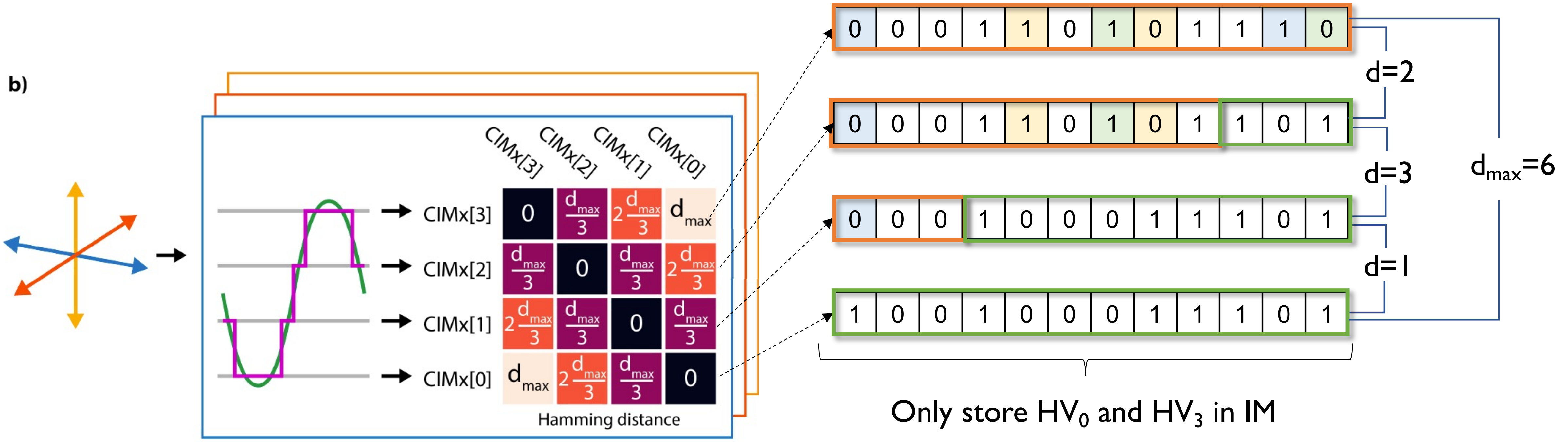
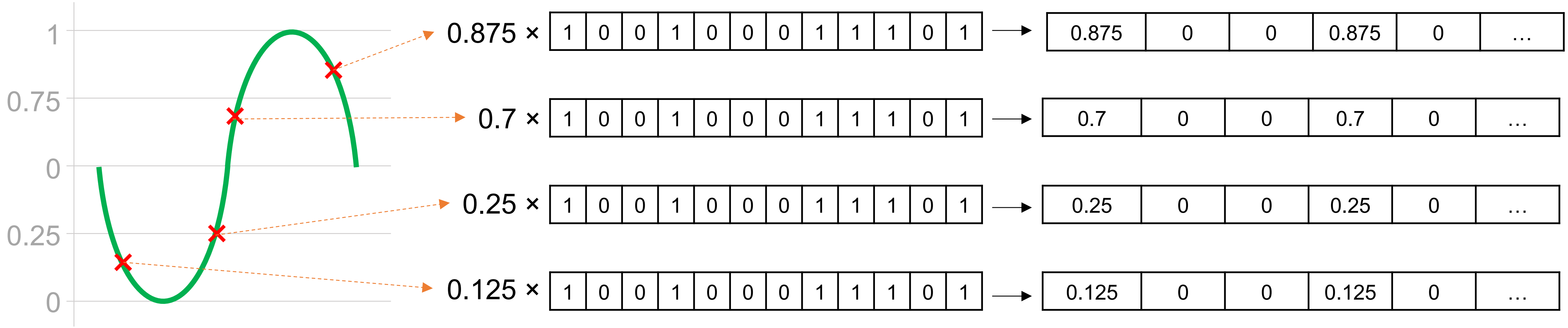


Figure taken from Zhou, A., Muller, R., & Rabaey, J. (2021, March). Memory-Efficient, Limb Position-Aware Hand Gesture Recognition using Hyperdimensional Computing. In *Research Symposium on Tiny Machine Learning*.

Random projection

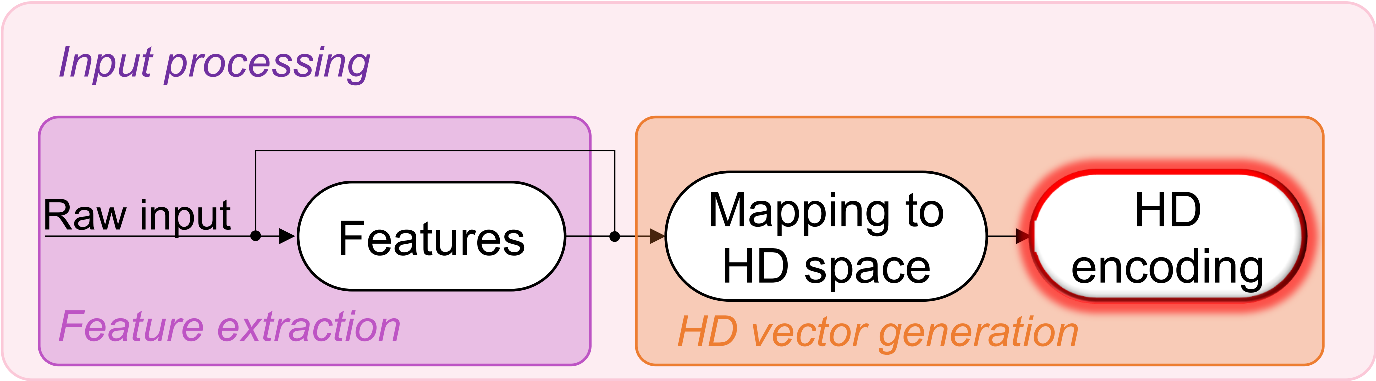
- Scalar multiplication of real value and HD vector.
- Useful when most appropriate quantization is unknown.
- Can be used as an intermediate representation, and normalized back to {-1,1} or {0,1} at a later step*.



* See also Rachkovskij, D.A., I. S. Misuno, and S.V. Slipchenko. "Randomized projective methods for the construction of binary sparse vector representations." Cybernetics and Systems Analysis 48.1 (2012): 146-156.

Classification with HD computing/VSAs

2. Encoding



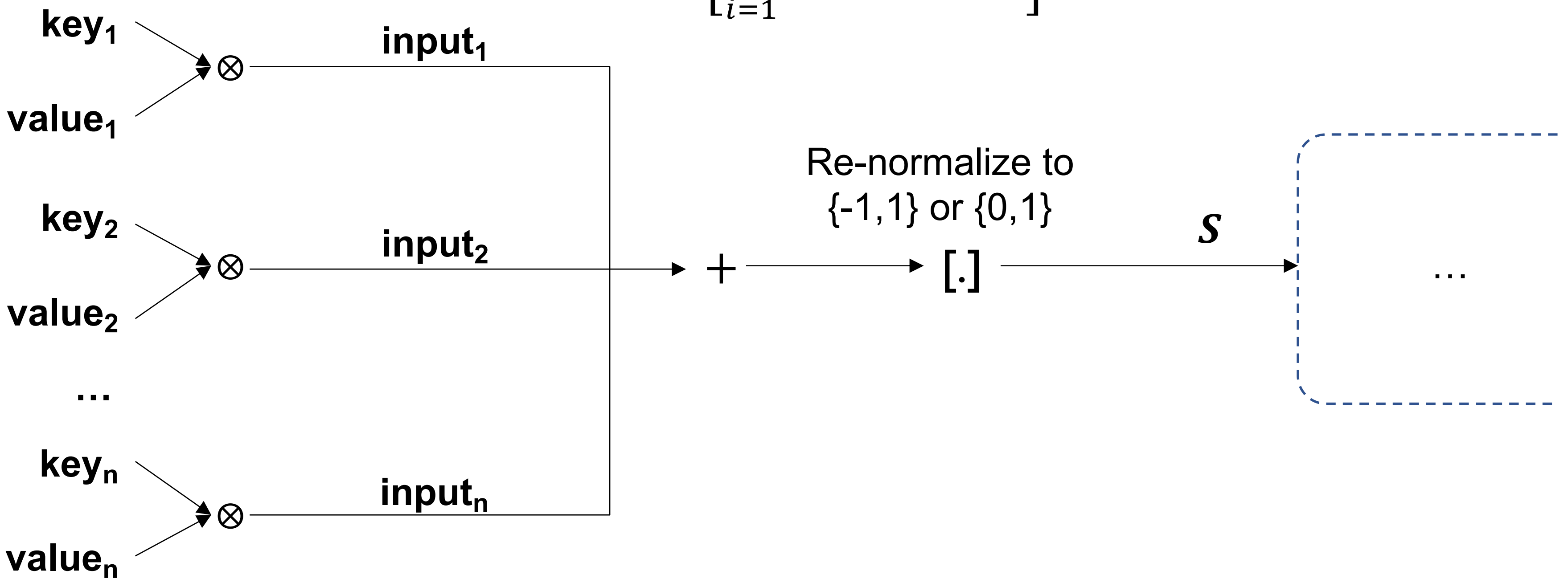
Encoding

- Exploit the properties of HD computing/VSAs, for example:
  - High capacity of HD vectors.
  - Binding: {key:value} / (variable:value) pairs.
  - Construction of data structures: sets, sequences, histograms, etc. Refer to Module 4: Representation and Manipulation of data structures by Denis.

Spatial encoding

- Combine inputs available at a given time-aligned sample into a single HD vector.
- Set of key-value pairs (refer to Module 4 on data structures by Denis).

$$S = \left[\sum_{i=1}^{n} key_i \otimes value_i \right]$$



P. Neubert and P. Protzel, "Towards HypervectorRepresentations for Learning and Planning with Schemas," Joint German/Austrian Conference on Artificial Intelligence, 2018

Spatial encoding

- Example: 2D robot navigation

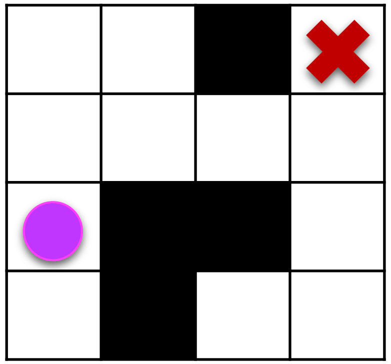
Sensor ID I.M.

N	1	0	0	1	0	0	0	1	1	1
S	0	1	1	0	0	1	0	1	0	0
E	0	1	0	0	1	1	1	1	0	1
W	1	1	0	1	0	0	1	1	0	0

Sensor value I.M.

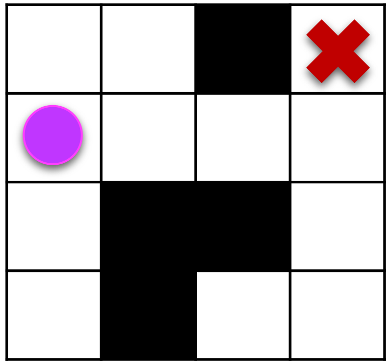
True	0	1	0	0	1	1	1	1	0	1
False	1	1	0	1	0	0	1	1	0	0

t=1



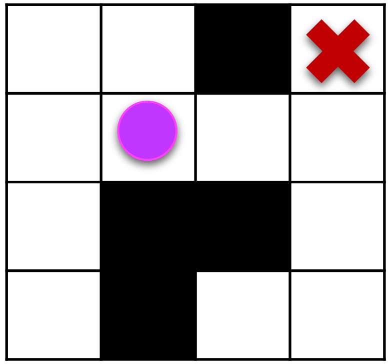
$S_{t=1}$=[N⊗F + S⊗F+E⊗T +W⊗T]

t=2



$S_{t=2}$=[N⊗F + S⊗F+E⊗F +W⊗T]

t=3

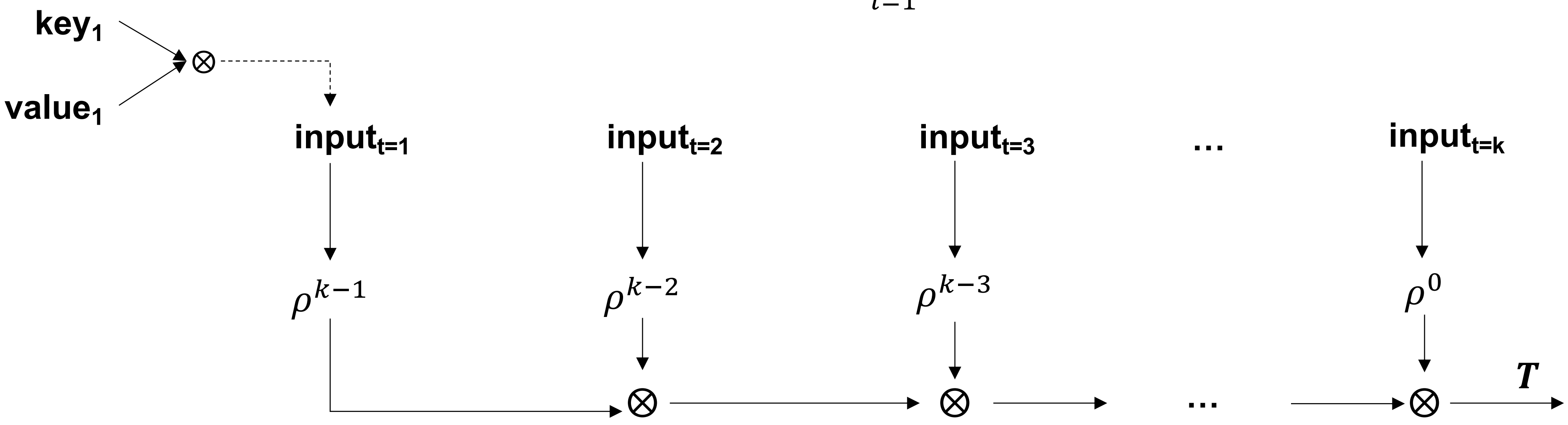


$S_{t=3}$=[N⊗F + S⊗T+E⊗F +W⊗F]

Temporal encoding

- Represent time dependencies of the inputs through sequences or n-grams over a specific window of length k.
- Sequence (of key-value pairs). Refer to Module 4.

$$T = \prod_{i=1}^{k} \rho^{k-i}(\,\mathrm{input}_{t=i})$$



Temporal encoding

- Example: 2D robot navigation

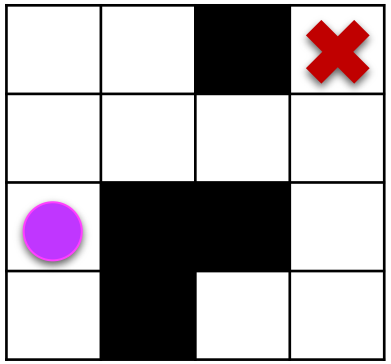
Sensor ID I.M.

N	1	0	0	1	0	0	0	1	1	1
S	0	1	1	0	0	1	0	1	0	0
E	0	1	0	0	1	1	1	1	0	1
W	1	1	0	1	0	0	1	1	0	0

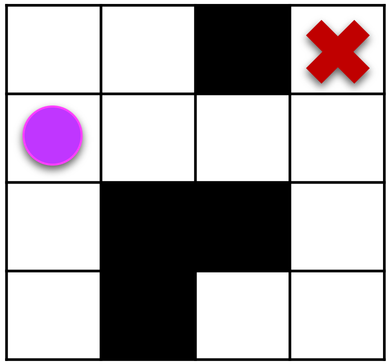
Sensor value I.M.

True	0	1	0	0	1	1	1	1	0	1
False	1	1	0	1	0	0	1	1	0	0

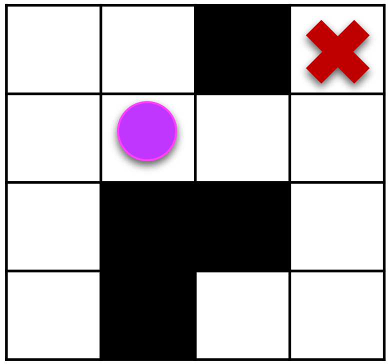
t=1



t=2



t=3



$$\underbrace{S_{t=1}=[N\otimes F + S\otimes F+E\otimes T +W\otimes T] \quad S_{t=2}=[N\otimes F + S\otimes F+E\otimes F +W\otimes T] \quad S_{t=3}=[N\otimes F + S\otimes T+E\otimes F +W\otimes F]}_{T=\, \rho^2(S_{t=1}) \otimes \rho(S_{t=2}) \otimes S_{t=3}}$$

Histogram encoding

- Some applications may benefit from frequency distribution representations.

$$H = \sum_{i=1}^{n} input_i$$

E.g. Language identification (Lecture 1 by Pentti)

Trigrams from letter seed vectors

```
THE = r(r(T)) * r(H) * E
```

Accumulate to form language profile (histogram of trigrams)

```
Engl += THE + HE# + E#Q + #QU + QUI + UIC + ...
```

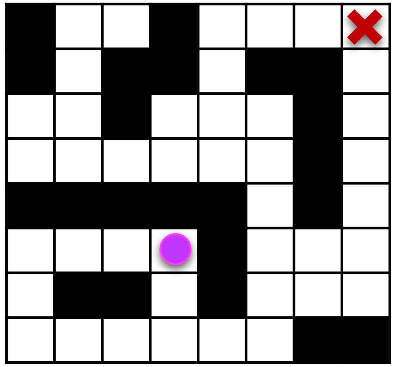
Compare to other language profiles

Query most likely letter after e.g. TH

Joshi, Aditya, Johan T. Halseth, and Pentti Kanerva. "Language geometry using random indexing." *International Symposium on Quantum Interaction*. Springer, Cham, 2016.

Histogram encoding

- Example: 2D robot navigation, anomaly detection



Actuation I.M.

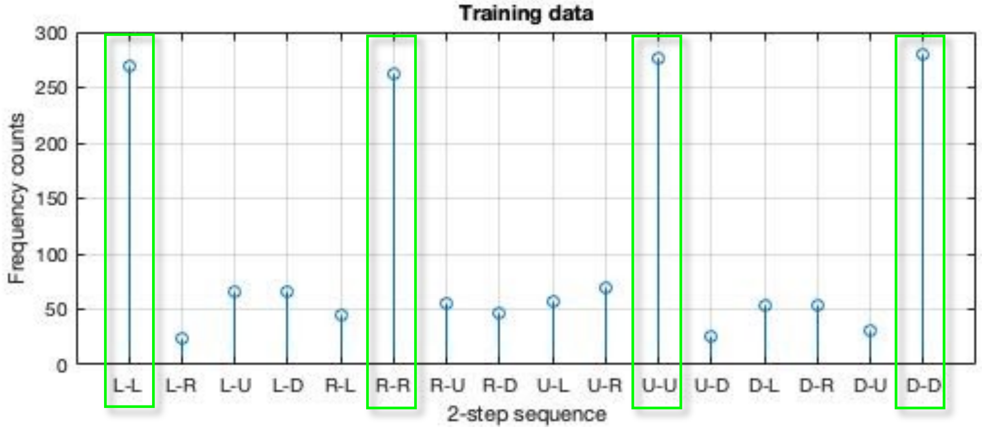
Up	1	0	0	1	0	0	0	1	1	1
Down	0	1	1	0	0	1	0	1	0	0
Right	0	1	0	0	1	1	1	1	0	1
Left	1	1	0	1	0	0	1	1	0	0

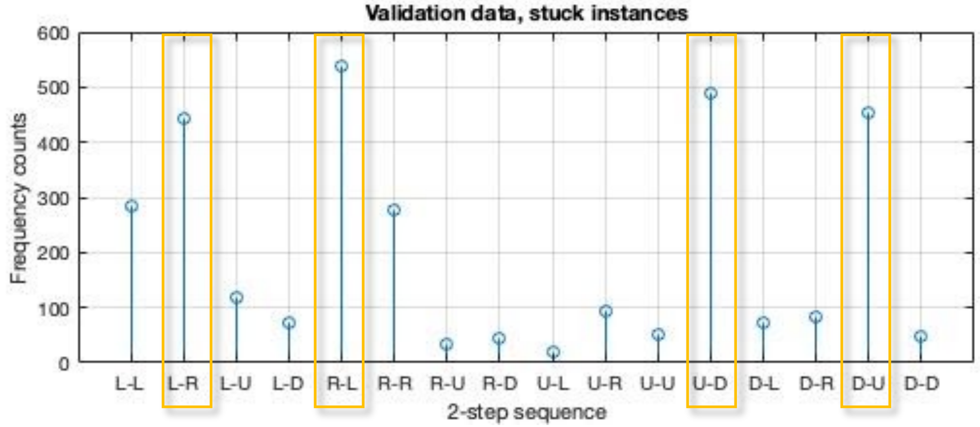
Histogram over x time steps

$$H = \sum_{i=1}^{x} S_{t_i}$$

2-step sequences (bi-grams)

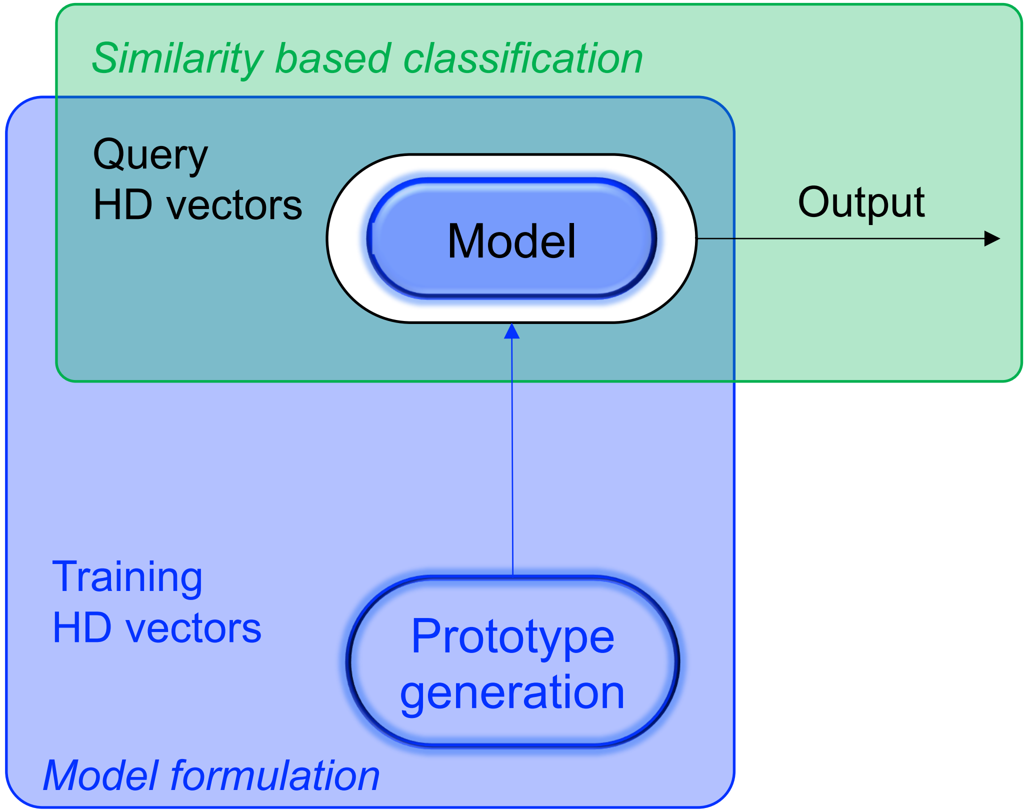
$$S_1 = \rho(L) \otimes L$$
$$S_2 = \rho(L) \otimes R$$
$$S_3 = \rho(L) \otimes U$$
$$S_4 = \rho(L) \otimes D$$
…
$$S_{16} = \rho(D) \otimes D$$





Classification with HD computing/VSAs

3. Model formulation and classification

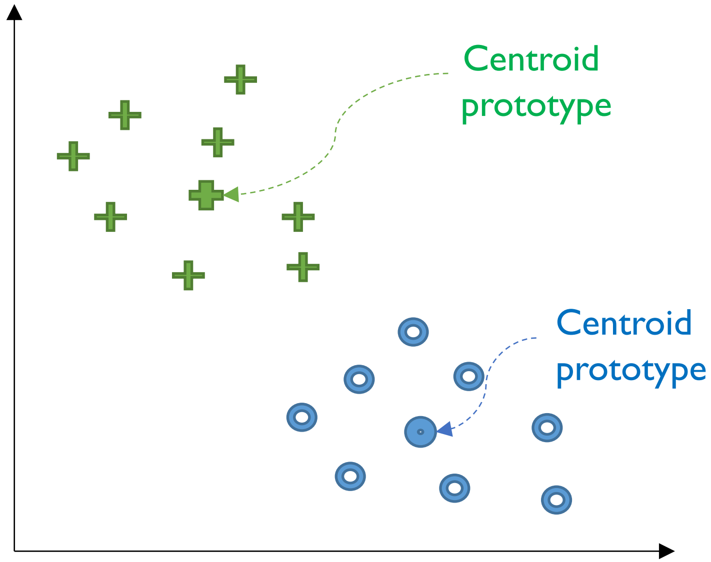


Model formulation

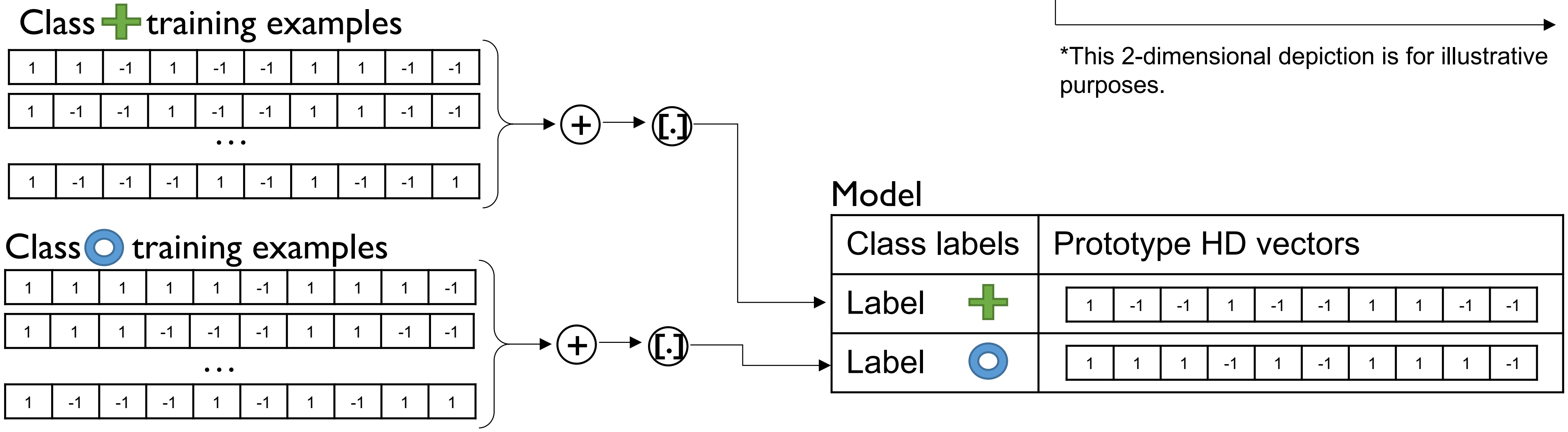
- Class prototype HD vector generation.
- Multiple approaches:
  - **Centroid based prototype generation**
  - **One-shot prototype generation**
  - **Learning Vector Quantization (LVQ) family of representations**
  - Multiple prototypes per class
  - Storing models in superposition

Class prototype generation (centroid computation)

- A prototype HD vector for each class is formed by computing the class centroid.
- For binary and bipolar HD vectors, this amounts to finding the majority of each element across all training examples.

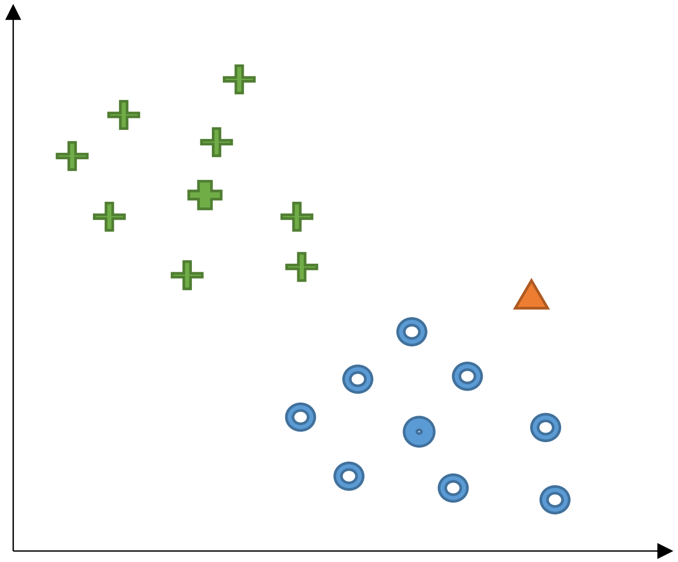


*This 2-dimensional depiction is for illustrative purposes.

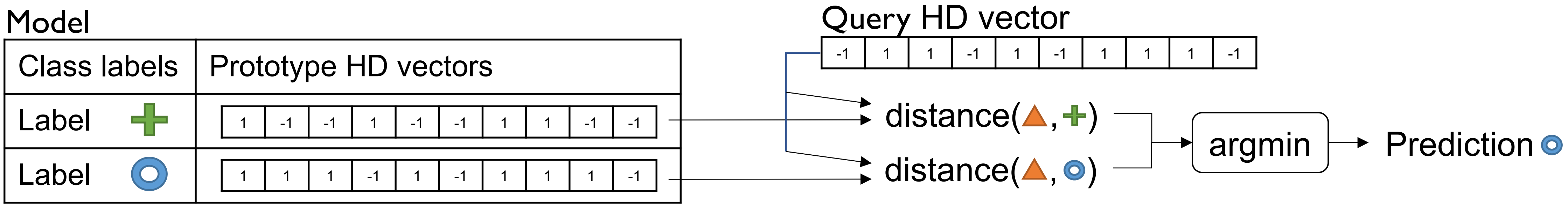


Classification

- Query HD vectors are formed using the same encoding as the training ones.
- Classification takes place by finding the nearest neighboring prototype to this query HD vector.
- Akin to a nearest centroid classifier in ML
(https://en.wikipedia.org/wiki/Nearest_centroid_classifier)

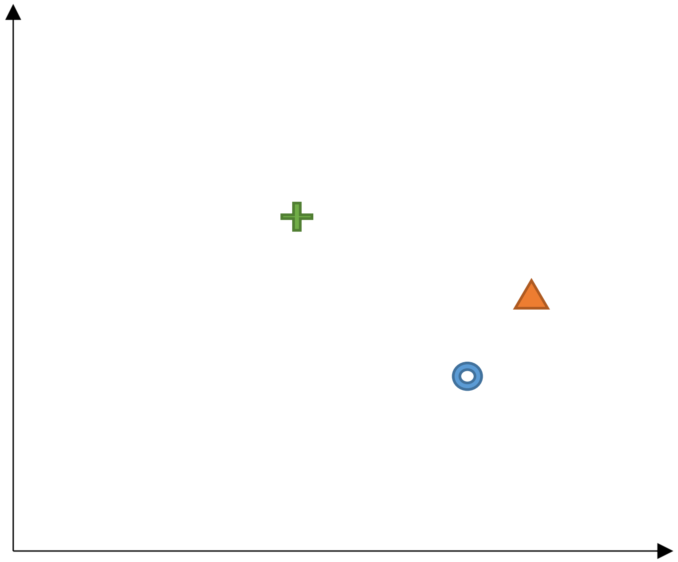


*This 2-dimensional depiction is for illustrative purposes.

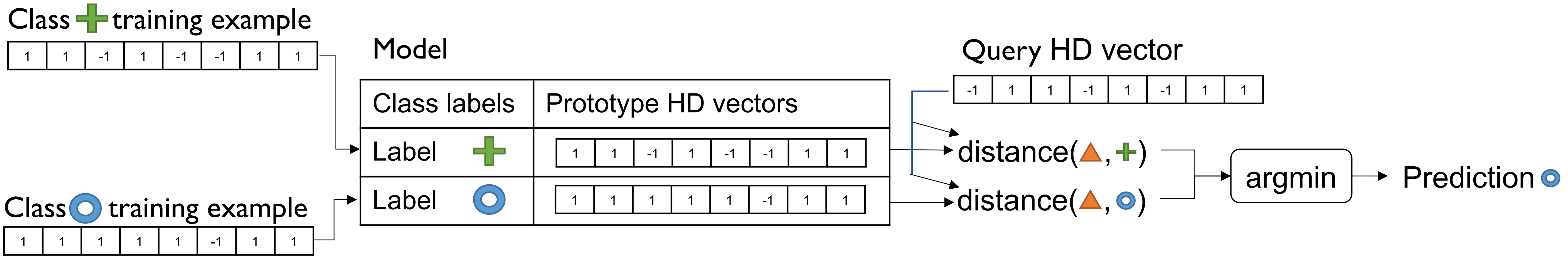


One-shot classification

- Single training sample constitutes the class prototype.
- Classification proceeds as usual.
- Works well with HD computing/VSAs because of information-rich representations enabled by HD vectors + encoding.



*This 2-dimensional depiction is for illustrative purposes.



Example: Rahimi, A., Tchouprina, A., Kanerva, P., Millán, J. D. R. & Rabaey, J. M. Hyperdimensional Computing for Blind and One-Shot Classification of EEG Error-Related Potentials. Mob. Networks Appl. 1–12 (2017)

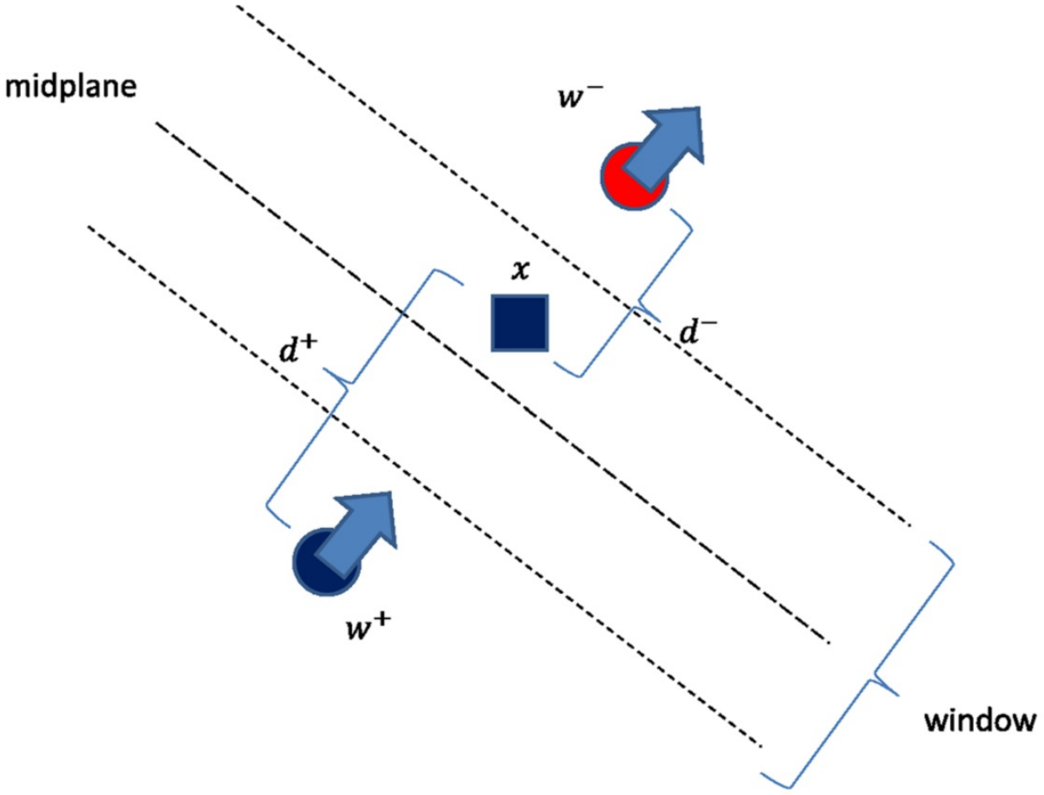
Learning Vector Quantization approaches

- Prototypes represent class regions.
- These prototypes are learned iteratively.

Heuristic approach:

$$\mathrm{w}^{+} = \mathrm{w}^{+} - \alpha(x - \mathrm{w}^{+})$$
$$\mathrm{w}^{-} = \mathrm{w}^{-} + \alpha(x - \mathrm{w}^{-})$$

Learning rate



Diao, Cameron, et al. "Generalized Learning Vector Quantization for Classification in Randomized Neural Networks and Hyperdimensional Computing." arXiv preprint arXiv:2106.09821 (2021).
Nova, David, and Pablo A. Estévez. "A review of learning vector quantization classifiers." *Neural Computing and Applications* 25.3 (2014): 511-524.

Learning Vector Quantization approaches

Margin maximalization approaches (e.g. Generalized LVQ):

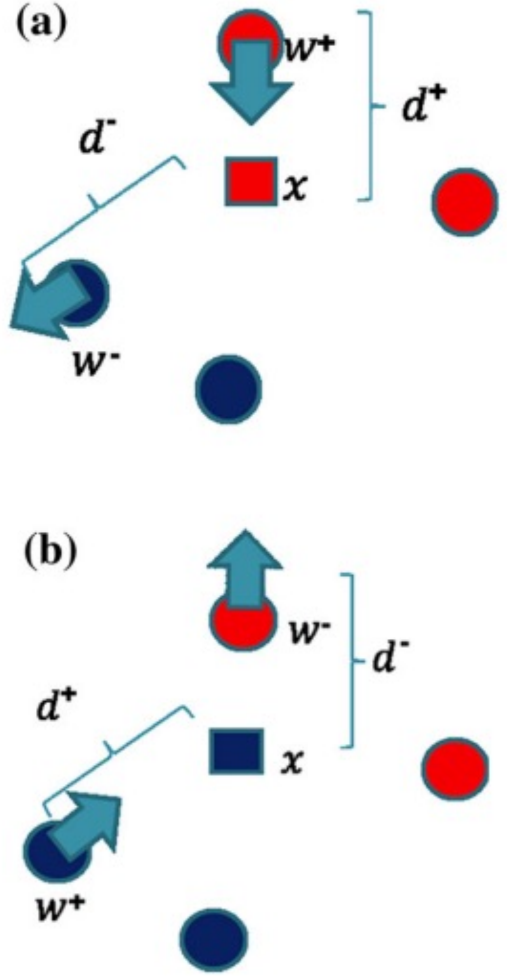
Define a cost function from which the learning rule is derived via gradient descent.

Cost function

Relative distance difference

$$\mathrm{w}^{+} = \mathrm{w}^{+} - 2\alpha \frac{\delta g}{\delta \mu} \mu^{-} (x - \mathrm{w}^{+})$$

$$\mathrm{w}^{-} = \mathrm{w}^{-} + 2\alpha \frac{\delta g}{\delta \mu} \mu^{+} (x - \mathrm{w}^{-})$$



Diao, Cameron, et al. "Generalized Learning Vector Quantization for Classification in Randomized Neural Networks and Hyperdimensional Computing." arXiv preprint arXiv:2106.09821 (2021).

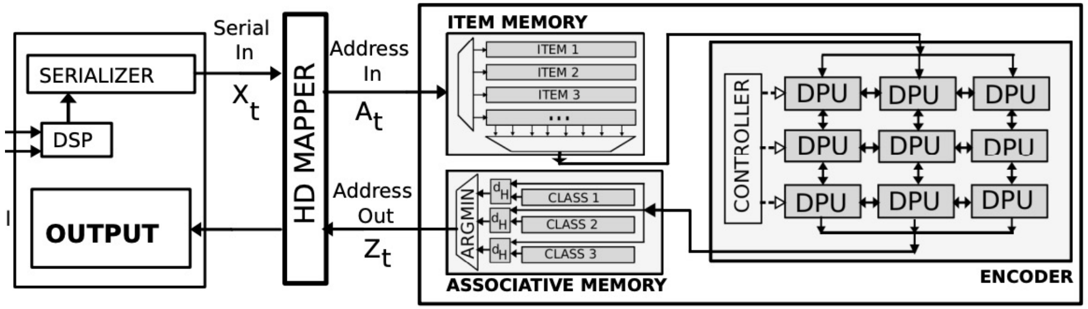
Nova, David, and Pablo A. Estévez. "A review of learning vector quantization classifiers." *Neural Computing and Applications* 25.3 (2014): 511-524.

Classification with HD computing/VSAs

4. Performance trade-offs

Energy efficiency

- Departure from traditional von Neumann architectures with inefficient memory exchanges.
- Enables bit-wise operations when using Binary Spatter Codes model.
- Robustness under low SNR and sparsity can enable resource efficiency.
- Will be discussed in detail by Mohamed in Module 11: Hardware implementations.



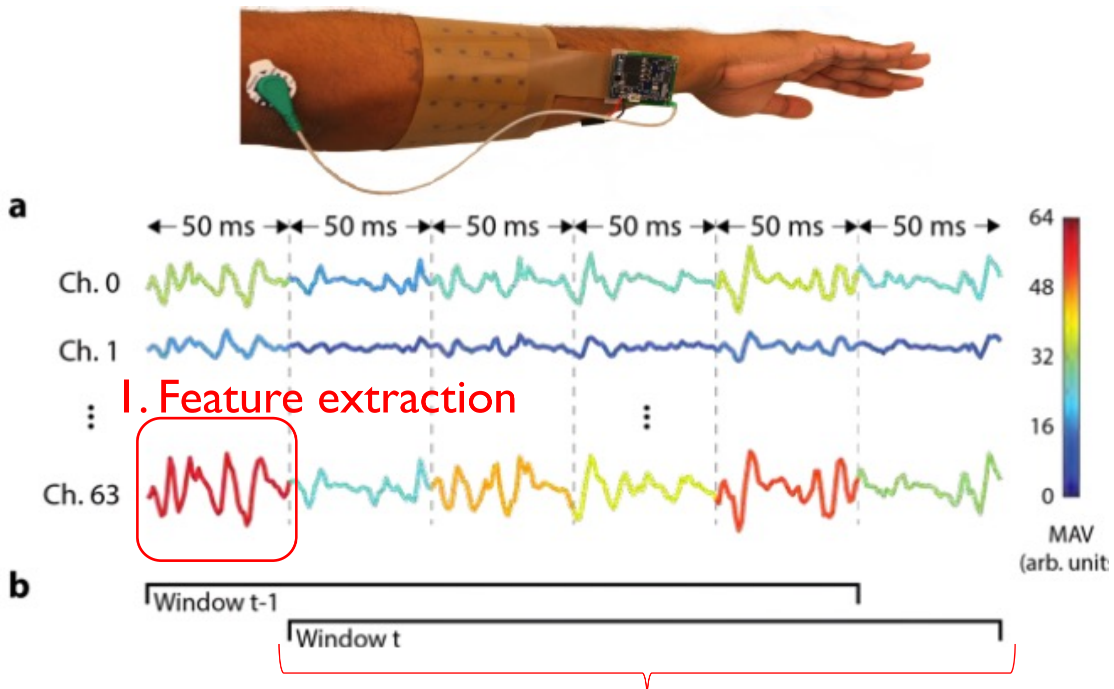
Datta, Sohum, et al. "A programmable hyper-dimensional processor architecture for human-centric IoT." *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9.3 (2019): 439-452.

Application examples

ExG signal classification, Robotics

EMG-based hand gesture recognition

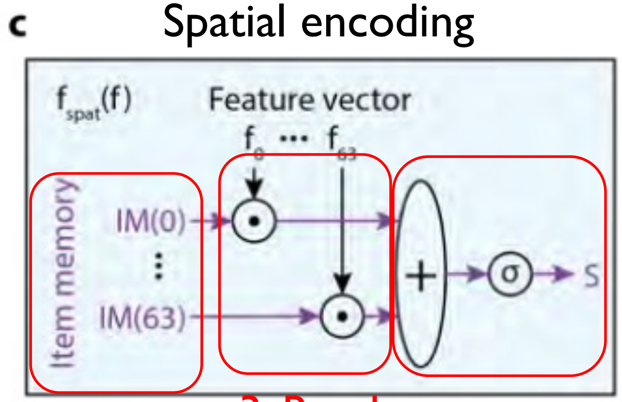
Array of 64 electrodes



1. Feature extraction

5. Sequence of spatial vectors over window

HD mapping and Spatial encoding

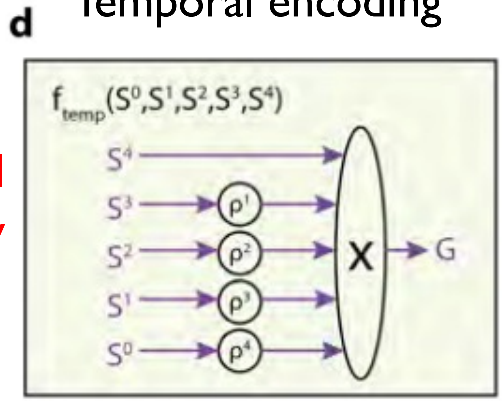


2. HD vectors for each electrode ID

3. Random projection

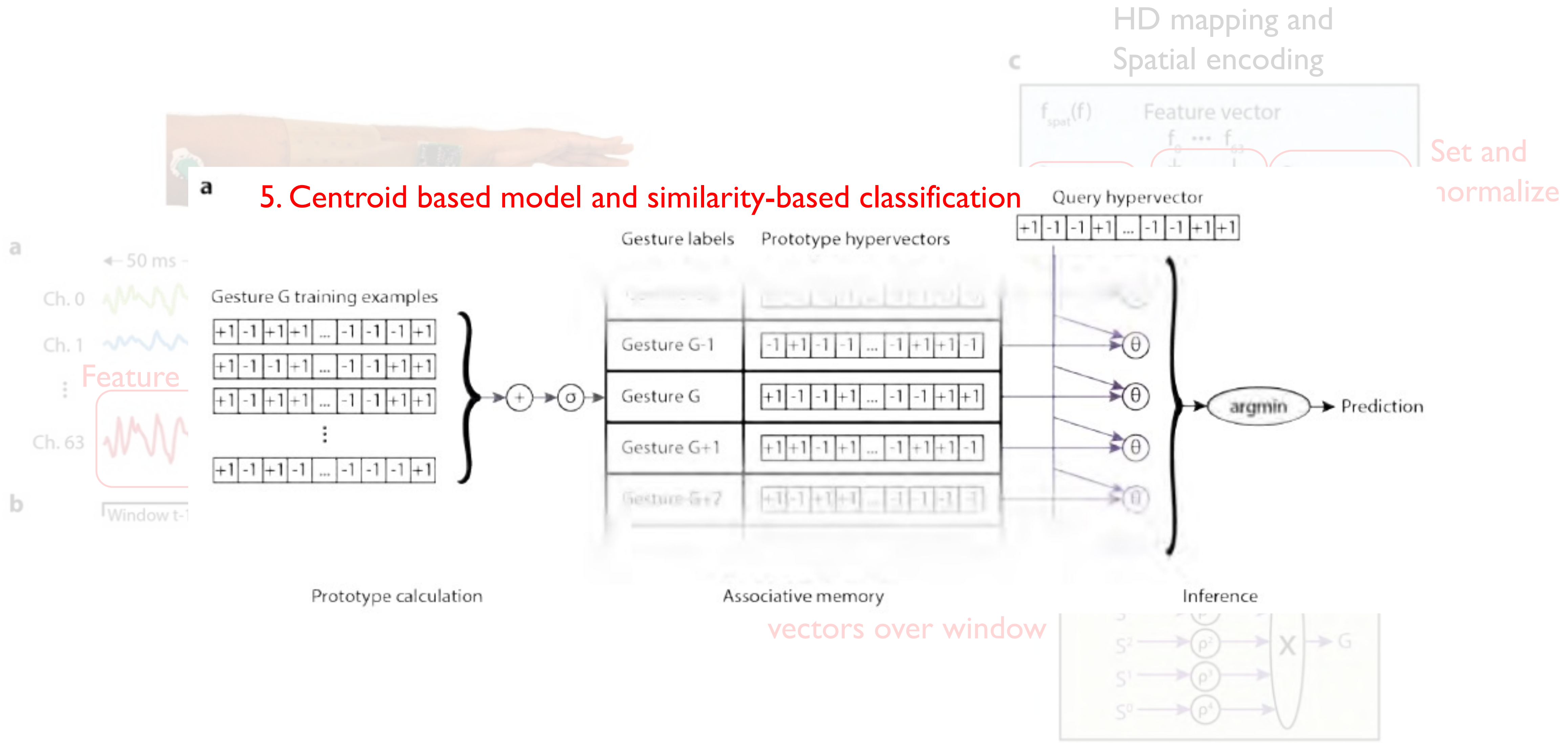
4. Spatial encoding (set of scaled vectors + normalization)

Temporal encoding



A. Moin, et al. "A wearable biosensing system with in-sensor adaptive machine learning for hand gesture recognition." *Nature Electronics* 4.1 (2021): 54-63.

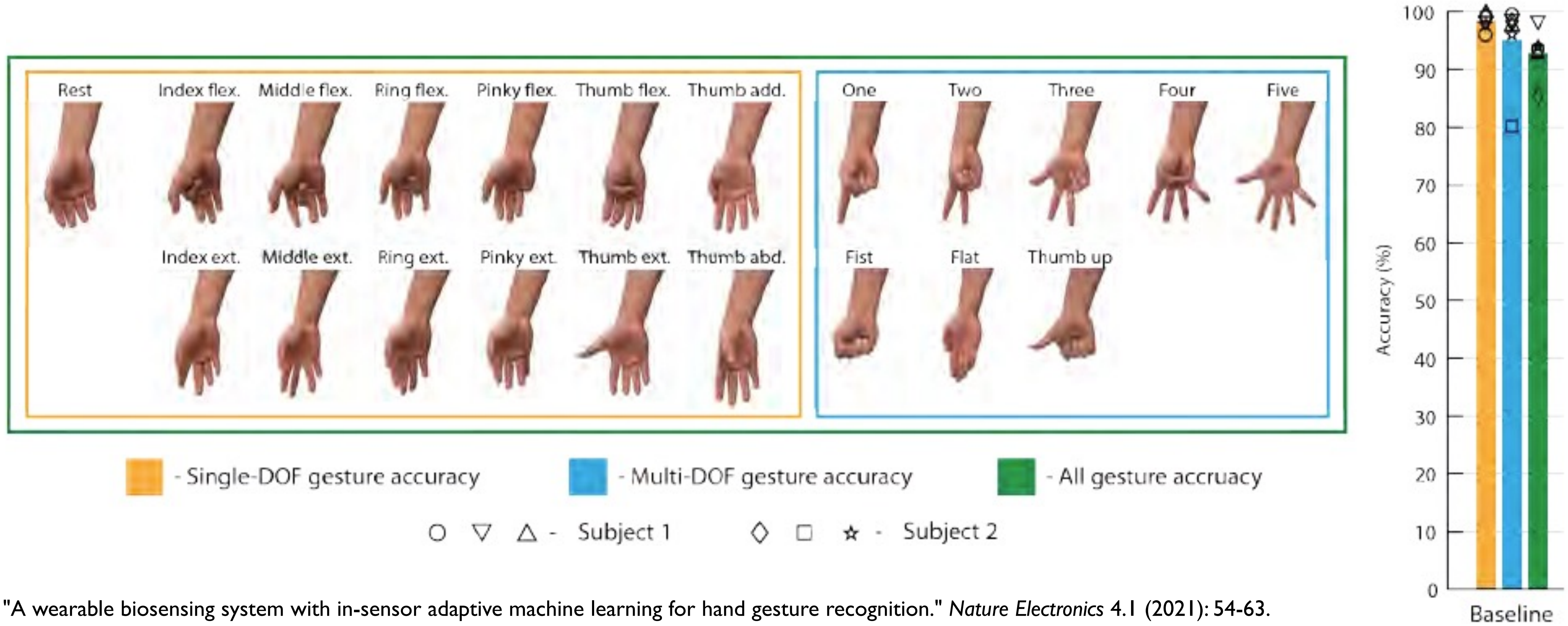
EMG-based hand gesture recognition



A. Moin, et al. "A wearable biosensing system with in-sensor adaptive machine learning for hand gesture recognition." *Nature Electronics* 4.1 (2021): 54-63.

EMG-based hand gesture recognition

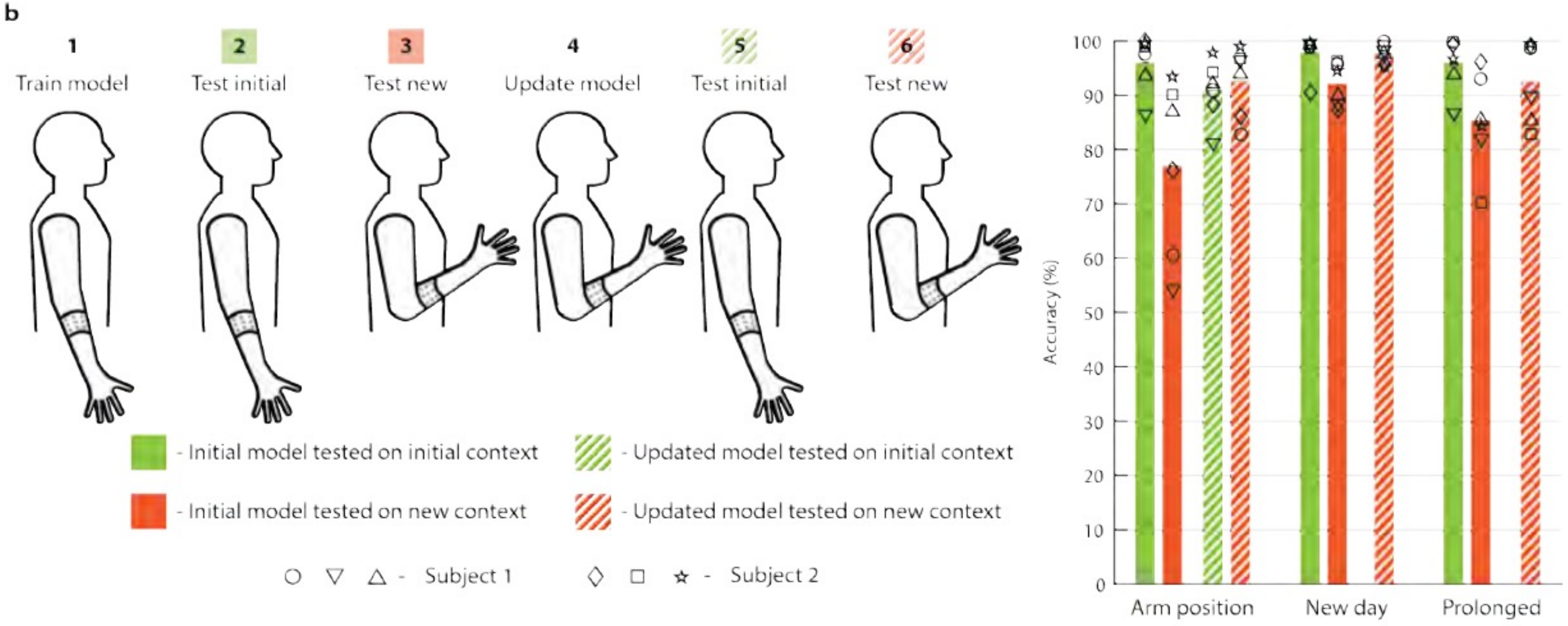
- Results: high accuracy and robustness to variations from different users.



A. Moin, et al. "A wearable biosensing system with in-sensor adaptive machine learning for hand gesture recognition." *Nature Electronics* 4.1 (2021): 54-63.

EMG-based hand gesture recognition

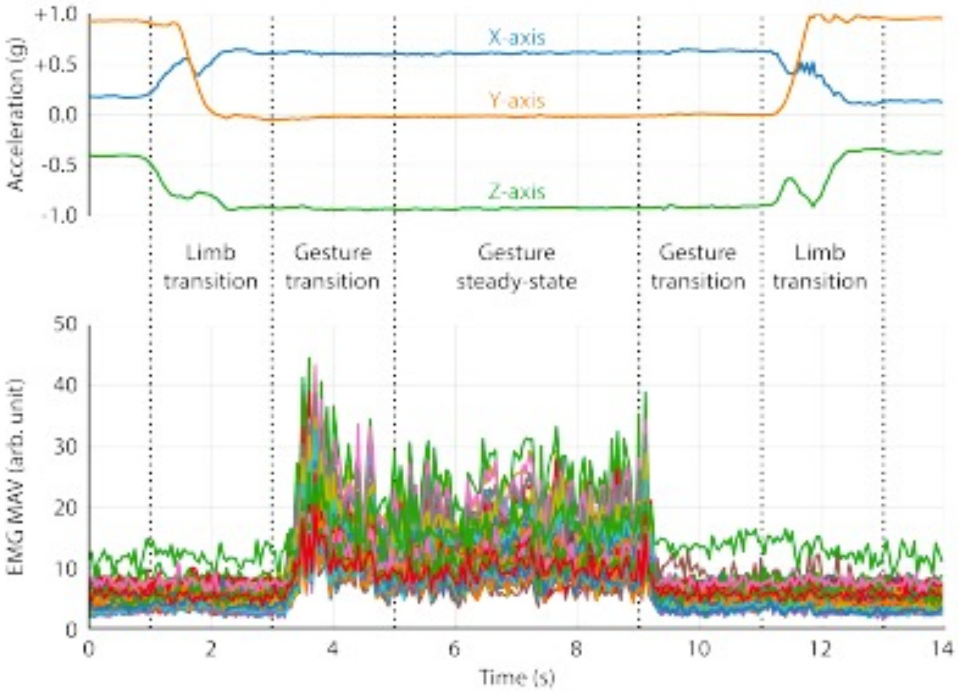
- Adaptive learning without significant overhead.

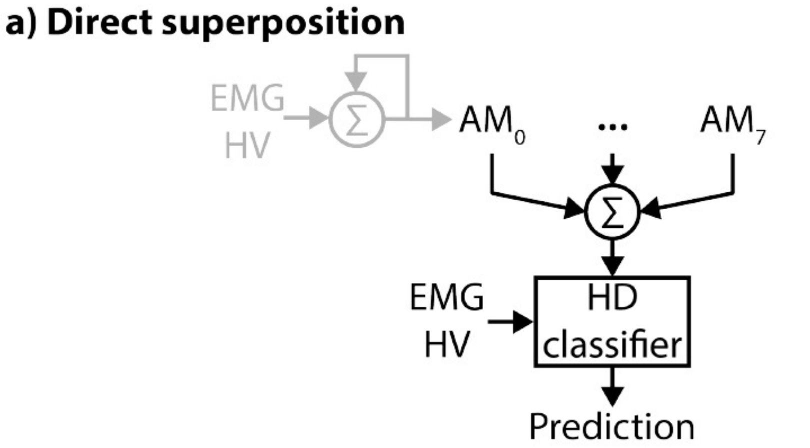


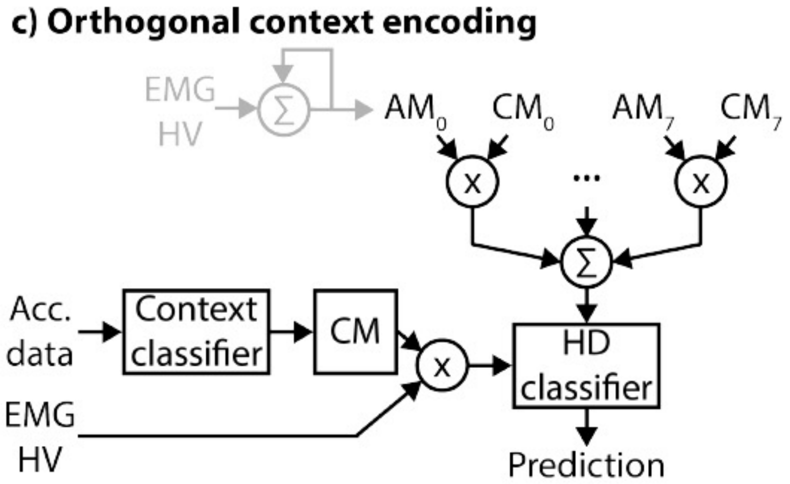
A. Moin, et al. "A wearable biosensing system with in-sensor adaptive machine learning for hand gesture recognition." *Nature Electronics* 4.1 (2021): 54-63.

EMG-based hand gesture recognition

- Sensor fusion and storing context-specific models in superposition.



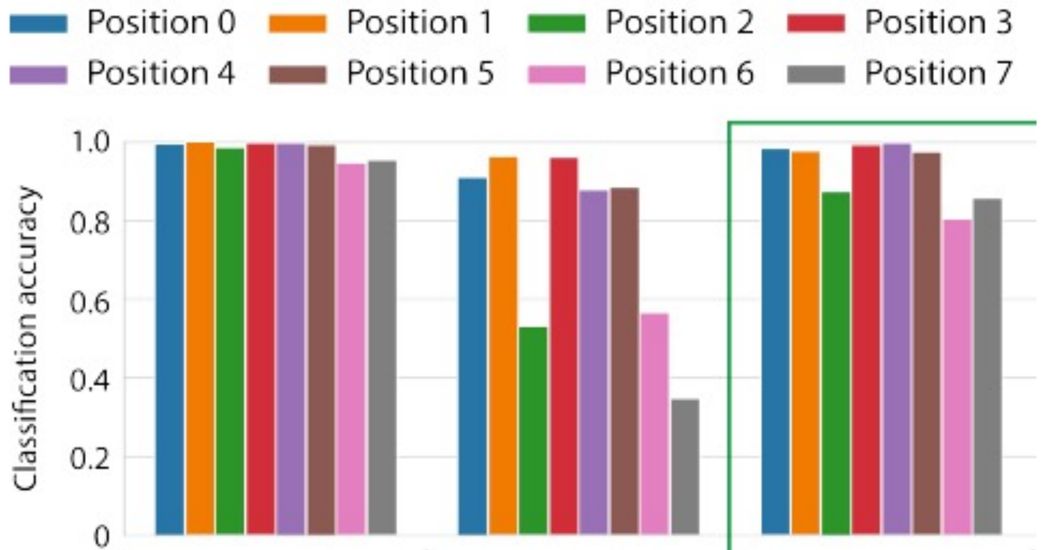




Zhou, A., Muller, R., & Rabaey, J. (2021, March). Memory-Efficient, Limb Position-Aware Hand Gesture Recognition using Hyperdimensional Computing. In *Research Symposium on Tiny Machine Learning*

EMG-based hand gesture recognition

- Results: the proposed orthogonal context encoding improves accuracy with respect to direct superposition and is significantly more efficient than dual-stage classification.

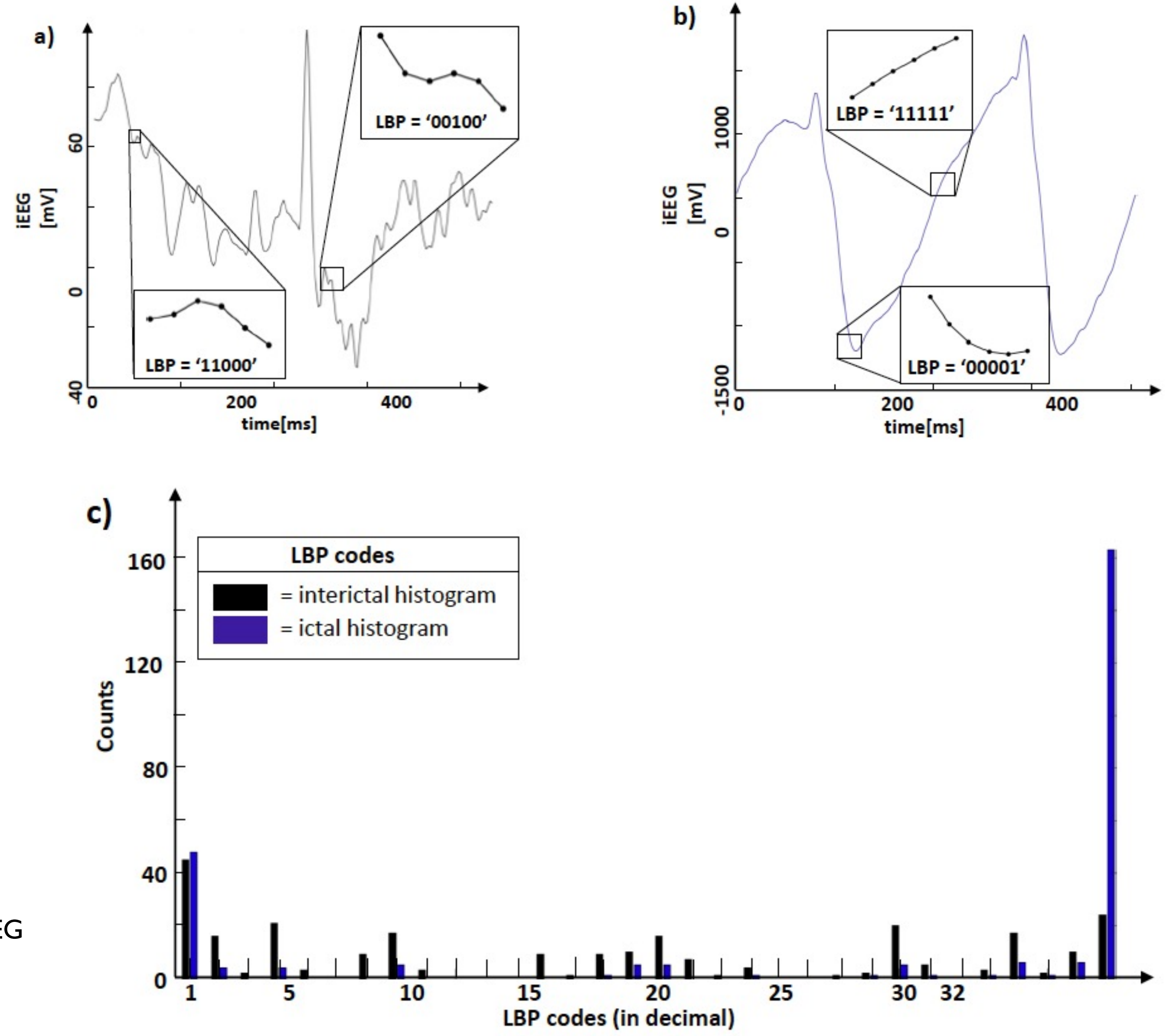


	Dual-stage classification	Direct superposition	Orthogonal context encoding
Average accuracy	98.46%	75.58%	93.37%
Parameter Count	HDC AM: 104 HVs SVM: 50 SVs	HDC AM: 13 HVs	HDC AM: 13 HVs SVM: 50 SVs
Parameter Memory	1044.8 kb	130 kb	134.8 kb
Implementation components	EMG HDC classifier + Limb position SVM	EMG HDC classifier	EMG HDC classifier + Limb position SVM

A. Moin, et al. "A wearable biosensing system with in-sensor adaptive machine learning for hand gesture recognition." *Nature Electronics* 4.1 (2021): 54-63.

iEEG-based Seizure Detection

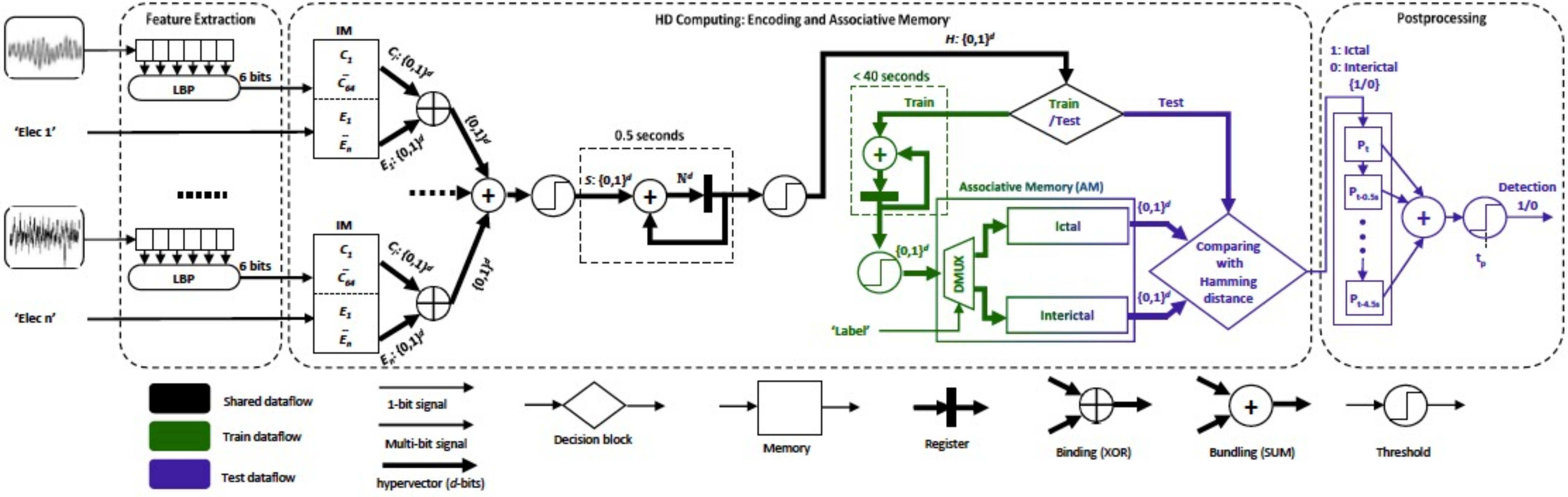
- Interictal (precede seizure) and ictal (during seizure) states of brain activity have distinct pattern frequency distributions.
- These patterns can be captured through short bit strings or local binary patterns (LBPs).
- The two brain states can be identified by comparing the frequency distributions.



Burrello, A., Schindler, K., Benini, L., & Rahimi, A. (2018, October). One-shot learning for iEEG seizure detection using end-to-end binary operations: Local binary patterns with hyperdimensional computing. In *2018 IEEE Biomedical Circuits and Systems Conference (BioCAS)* (pp. 1-4). IEEE..

iEEG-based Seizure Detection

- HD computing/VSAs can be used to represent the LBP and encode histograms over a pre-determined window. Similarity metrics are used to infer whether the current brain state is ictal or interictal.

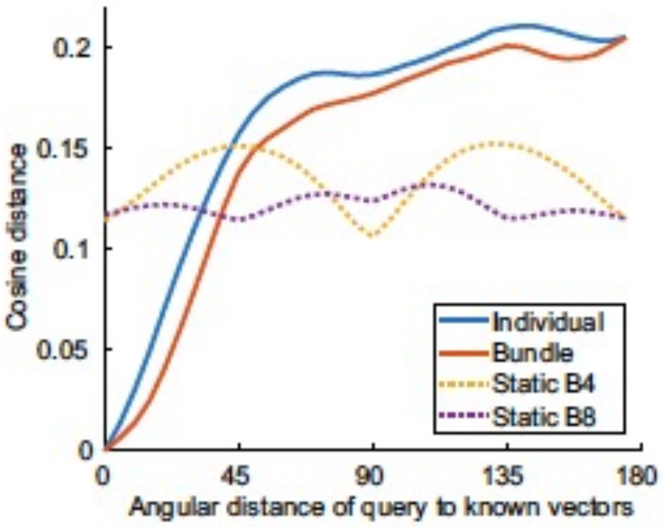


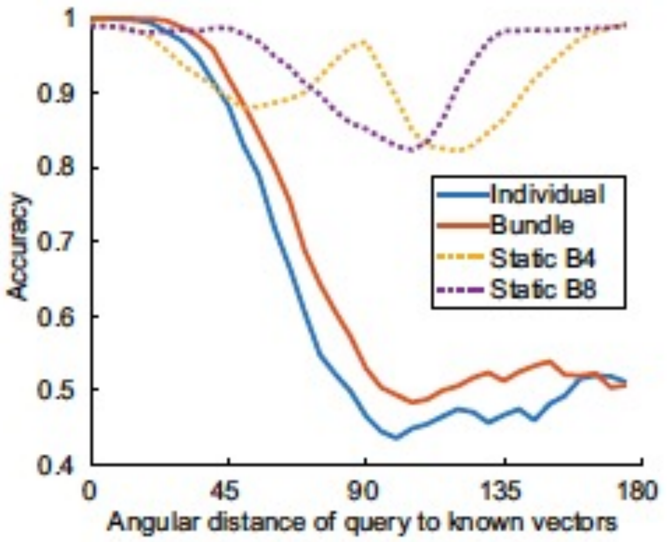
Burrello, A., Schindler, K., Benini, L., & Rahimi, A. (2018, October). One-shot learning for iEEG seizure detection using end-to-end binary operations: Local binary patterns with hyperdimensional computing. In *2018 IEEE Biomedical Circuits and Systems Conference (BioCAS)* (pp. 1-4). IEEE..

Robotics (vision)

Object recognition from multiple viewpoints

- Exploit bundling to combine multiple viewpoints in superposition.
- Then the comparison of a query vector to all views can be made with a single vector comparison.
- Straightforwardly update the representation of an object during online execution.



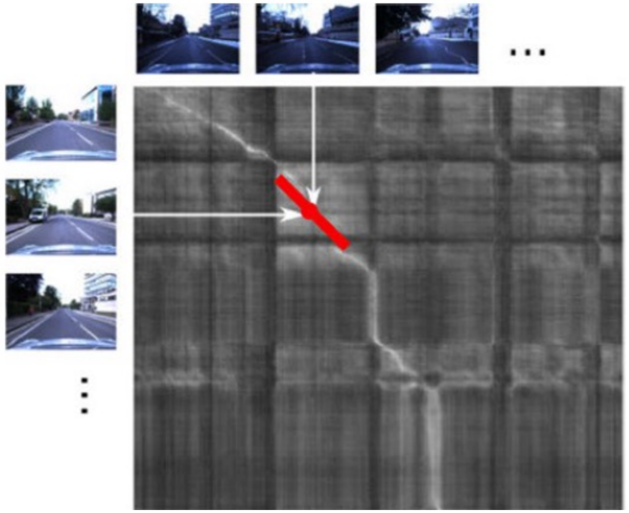


Neubert P., Schubert S., Protzel P. "An introduction to hyperdimensional computing for robotics." *KI-Künstliche Intelligenz* 33.4 (2019): 319-330.

Robotics (vision)

Sequence processing for place recognition

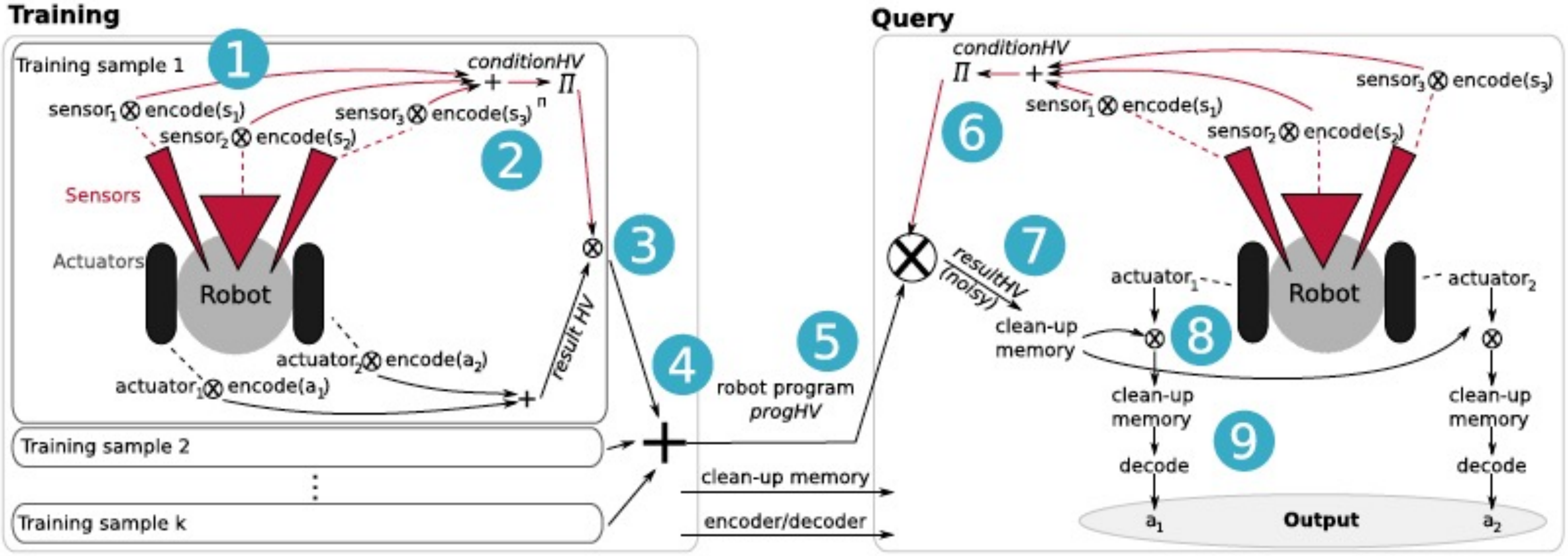
- Individual scenes are represented by binding their vector representation to their current position.
- Sequences of scenes are formed by bundling a set of the representations above.
- Prototype encodings are then compared to query ones to infer the scene.
- This representation is robust even through seasonal changes.



Neubert P., Schubert S., Protzel P. "An introduction to hyperdimensional computing for robotics." *KI-Künstliche Intelligenz* 33.4 (2019): 319-330.

Robotics

- Learning and recall of reactive behavior:
  - Learn from demonstrations a representation that encodes sensor-action pairs.
  - This behavior can be resembled at run-time by extracting the action corresponding to the current sensor value through an unbinding operation.



Neubert P., Schubert S., Protzel P. *Learning vector symbolic architectures for reactive robot behaviours*. Universitätsbibliothek Chemnitz, 2017.

Other applications of HD computing/VSAs

- Survey papers:

Module 9 (10/27): Solving classification problems

- Focus paper: Rahimi, Kanerva, Benini, Rabaey: Efficient Biosignal Processing Using Hyperdimensional Computing: Network Templates for Combined Learning and Classification of ExG Signals
- Further recommended reading:
  - Ge, Parhi: Classification using Hyperdimensional Computing: A Review
  - Neubert, Schubert, Protzel: An Introduction to Hyperdimensional Computing for Robotics
  - Joshi, Halseth, Kanerva. Language Geometry Using Random Indexing
  - Rahimi, Benatti, Kanerva, Benini, Rabaey: Hyperdimensional Biosignal Processing: A Case Study for EMG-based Hand Gesture Recognition
  - Kleyko, Rahimi, Rachkovskij, Osipov, Rabaey: Classification and Recall with Binary Hyperdimensional Computing: Tradeoffs in Choice of Density and Mapping Characteristic

- Comprehensive list of VSA/HD computing works

This page lists VSAs/Hyperdimensional Computing publications in the chronological order

We strive to keep this collection up-to-date but please let us know if some publications are currently missing and should be added to this list. We will take care of updating it.

Search | pairwise 1988 2021 | Type | Author(s) | Venue | Keywords

1 - 442 / 442

pairwise	Author(s)	Title	Venue	Pages	Keywords
2021	Poduval P., Zou Z., Najafi H., Homayoun H., Imani M.	StocHD: Stochastic Hyperdimensional System for Efficient and Robust Learning from Raw Data	ACM/ESDA/IEEE Design Automation Conference (DAC)	pp. 1-6	Machine Learning, Hardware
2021	Halawani Y., Kilani D., Hassan E., Tesfai H., Saleh H., Mohammad B.	RRAM-Based CAM Combined With Time-Domain Circuits for Hyperdimensional Computing	Scientific Reports	vol. 11, pp. 1-11	Hardware
2021	Thomas A., Dasgupta S., Rosing T.	A Theoretical Perspective on Hyperdimensional Computing	Journal of Artificial Intelligence Research	vol. 72, pp. 215-249	
2021	Eggimann M., Rahimi A., Benini L.	A 5 μW Standard Cell Memory-based Configurable Hyperdimensional Computing Accelerator for Always-on Smart Sensing	IEEE Transactions on Circuits and Systems I: Regular Papers	vol. 68, no. 10, pp. 4116-4128	

Questions?

Thank you!