Natural Learning

Unlocking a New Capacity in Machine Learning

Hadi Fanaee-T
Associate Professor of Machine Learning
School of Information Technology
Halmstad University, Sweden
Email: hadi.fanaee@hh.se

Natural Learning
(**Prototype-based** Categorization)

Deep Learning, SVM, K-NN
(**Exemplar** based Categorization)

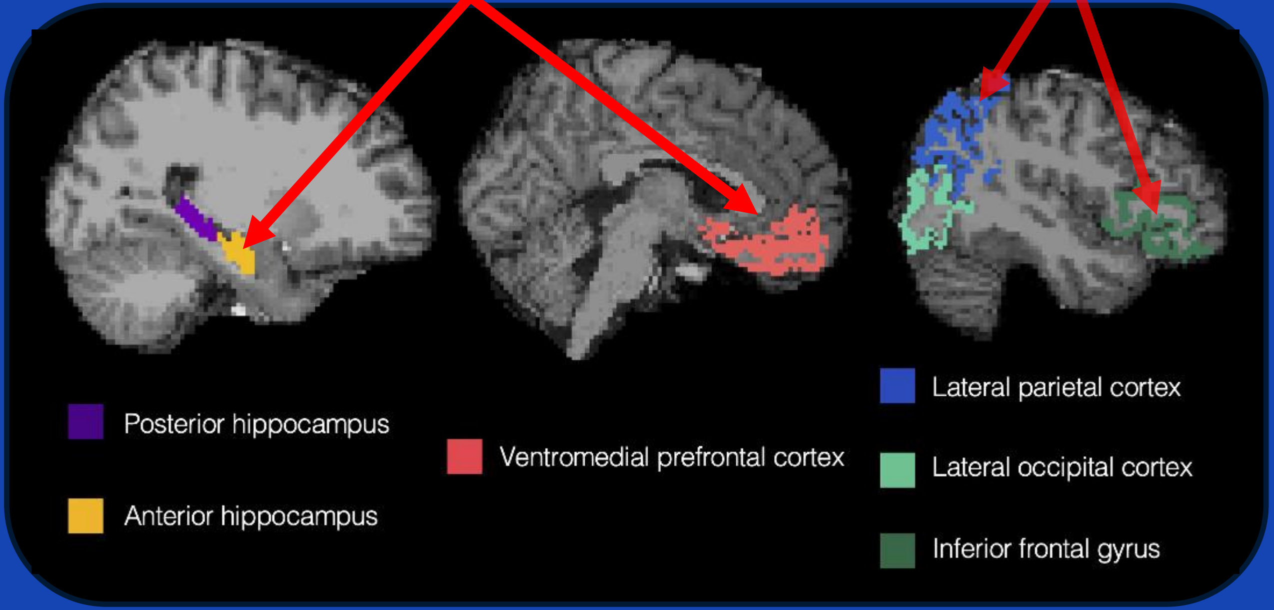


Image Source: Bowman, Caitlin R., Takako Iwashita, and Dagmar Zeithamova. "Tracking prototype and exemplar representations in the brain across learning." elife 9 (2020): e59360.

natural-learning.cc

Curse of "black box" Modelling

- Black-box models have achieved **state-of-the-art performance** on benchmark datasets and real-world applications.
  - Their internal mechanisms are not understandable by humans.(**Model's Transparency**)
  - Unable to explain their predictions comprehensibly to humans (**Decision's Explainability**)
  - Unable to provide justifications or reasoning for their decisions (**Decision's Interpretability**)

You cannot say, 'I'll do open-heart surgery because the neural network said so.' You have to have a very good reason."

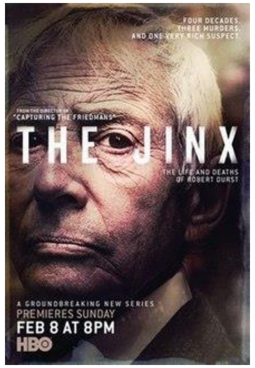
Christos Faloutsos, professor of computer science, CMU

Post-hoc Explanations

- Aim to **shed light** on **why** a particular prediction was made by a black-box model.
- There is a **narrow difference** between **explaining** something and **justifying** it.

Analogy of post hoc explanations in Real-life (1)

- On October 9, 2001, Robert Durst (**Black-box model**), **murdered** his neighbor Morris Black.
- During the trial, Durst's defense team **(Post-hoc explainer)** argued that he acted in **self-defense**.
- In November 2003, Durst was acquitted of murder charges.
- In the Netflix show "The Jinx," there's a scene where Durst talks to himself in a bathroom after an interview while still being recorded. He confessed that has killed his neighbor and two others.
- Both **murder** and **self-defense** are convincing stories.
- But does that mean that the explanation by the defense team reflected the truth?
- Only the killer (**Black-box model**) knows it.



Analogy of post hoc explanations in Real-life (2)

- A Black-box Model rejects a loan application solely based on the subject's **race**
- Post-hoc explanation can justify that **income** and **employment** status were the critical factors for the decision.
- The **connection** between Post-hoc explanation and the model's actual behavior can **never** be **proven**
  - **illusion of explainability**
- Post-hoc explanations can be even **more dangerous** than black-box models.
  - Sometimes, they can **cover up** the mistakes of black-box models by giving **believable reasons**.

Post-hoc Explanations are inherently wrong!

Cynthia Rudin
Professor of Computer Science, Duke University

- Another good argument: If the post-hoc explanation fully matched the original model, why would we need the original model after all?

nature machine intelligence

Explore content ⌄ **About the journal** ⌄ **Publish with us** ⌄ | **Subscribe**

nature › nature machine intelligence › perspectives › article

Perspective | Published: 13 May 2019

Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead

Cynthia Rudin

Nature Machine Intelligence **1**, 206–215 (2019) | Cite this article

73k Accesses | **2977** Citations | **502** Altmetric | Metrics

natural-learning.cc

New regulations may restrict use of black-box models

europarl.europa.eu/topics/en/article/20230601STO93804/eu-ai-act-first-regulation-on-artificial-int...

EN - English

News Topics MEPs About Parliament Plenary Committees Delegations Elections

Topics
European Parliament

Search

Digital Energy Gender equality Climate and environment Circular economy All topics

Topics › Digital › Artificial intelligence › EU AI Act: first regulation on artificial intelligence

EU AI Act: first regulation on artificial intelligence

The use of artificial intelligence in the EU will be regulated by the AI Act, the world's first comprehensive AI law. Find out how it will protect you.

Published: 08-06-2023 • Last updated: 19-12-2023 - 11:45

New regulations may restrict use of black-box models

- EU AI Act
  - Regulating high-risk AI applications used in critical infrastructure, such as transportation and **healthcare,** as well as those with potential risks to fundamental rights, safety, or other public interests.
  - Set of requirements to ensure their safety, **transparency**, and **accountability**.

An illusion of black-box models' universal superiority

- Real-world datasets typically contain between **8.0% and 38.5% label noise** (Semenova, et al., 2023)

- (Semenova, et al., 2023) provided **theoretical evidence** that in noisy datasets, such as datasets about humans like healthcare, criminal justice, and finance**, simple, interpretable classifiers should perform as well as black-box models.**

Semenova, et al. "A Path to Simpler Models Starts With Noise.", NeurIPS 2023

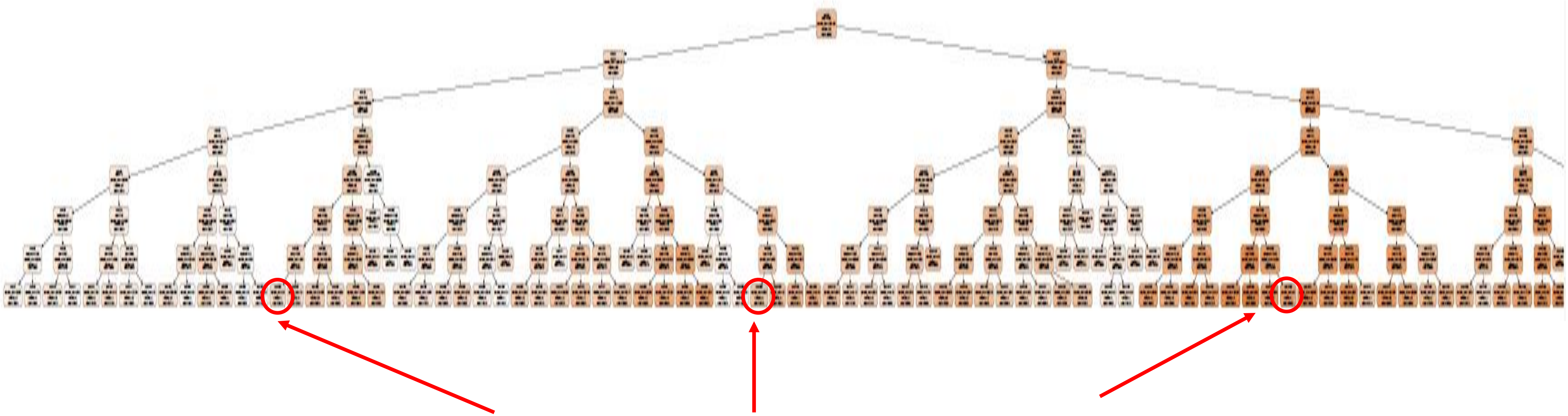
OK, but if we exclude black-box, what options do we have?

- Logistic Regression
  - Good level of interpretability and explainability
  - No built-in mechanism to deal with **noisy features, curse of dimensionality**, and **multicollinearity**

    → Poor performance with high-dimensional datasets

- Decision Trees
  - One of the most favored options when it comes to interpretability and explainability
  - Robust against the curse of dimensionality, irrelevant features, and noisy samples
  - Decision Trees are transparent, but are they explainable and interpretable?

Limited Explainability of Decision Trees

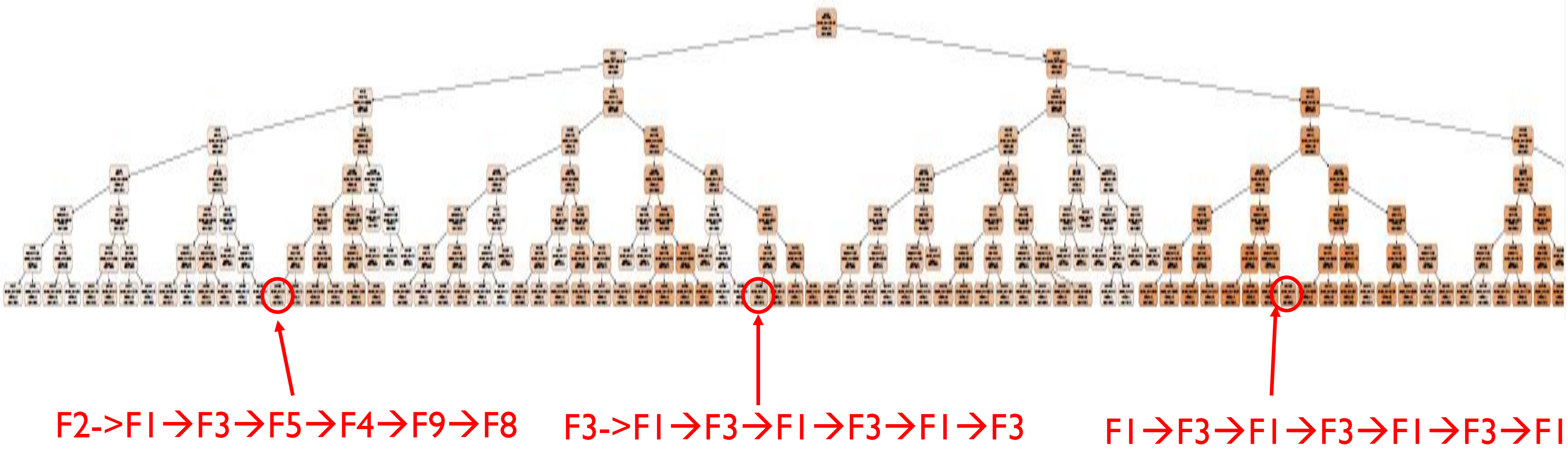


Can a **universal rule** explain the decisions for these people?

Opposed to Logistic regression, decision trees cannot provide a **global explanation** for the decisions.

Local explanations have a **limited value** if they cannot be **generalized**!

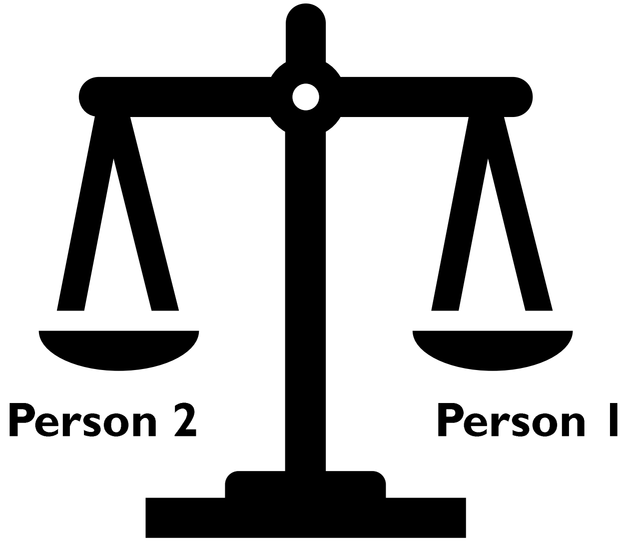
Limited Interpretability of Decision Trees



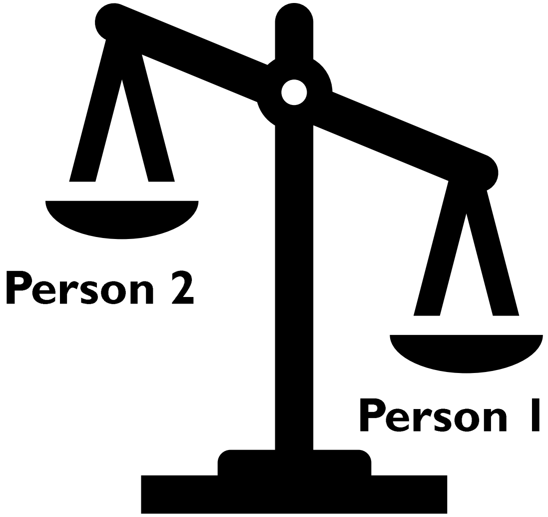
For each decision, different combinations of features are used.
It is **impossible** to infer the **actual contribution of features** at the **global** level.

Analogy in Law

We don't have **numerous versions of laws tailored to different individuals**; rather, there exists a **single universal law** that applies to **everyone**.

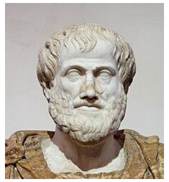


Logistic Regression is able to provide a **fair explanation**



Decision Tree is unable to provide **a fair explanation**

Underlying Philosophy of Decision Trees



Aristotle
(384-322 B.C.)

- The challenges associated with decision trees stem from their underlying philosophy, which is rooted in **Aristotle's categorization theory**
  - Humans use rule-based explanations to categorize concepts.

- Extensive Research in cognitive psychology in 1970s indicated shortcomings in this model, suggesting that **people** likely **do not rely on rule-based definitions** when categorizing objects**.**

Natural Categories (Prototype Theory)

Eleanor Rosch
Professor of Psychology, University of California, Berkeley

- People **categorize** objects and concepts based on their **similarity to a prototype**

Furniture Prototype

More Similar

Object

Image Source: https://slideplayer.com/slide/9817067/

Less similar

Electronics Prototype

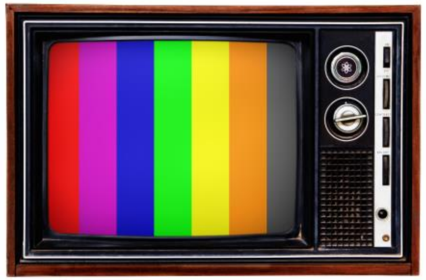


Image Source: https://facts.net/who-invented-color-tv/

natural-learning.cc

Rosch, Eleanor H. "Natural categories." Cognitive psychology 4.3 (1973): 328-350.

Characteristics of Prototype (1): **Typicality**

- Prototype is **the most typical** or central example of a category

Chair is a **prototype**
for Furniture Category

Furniture Category

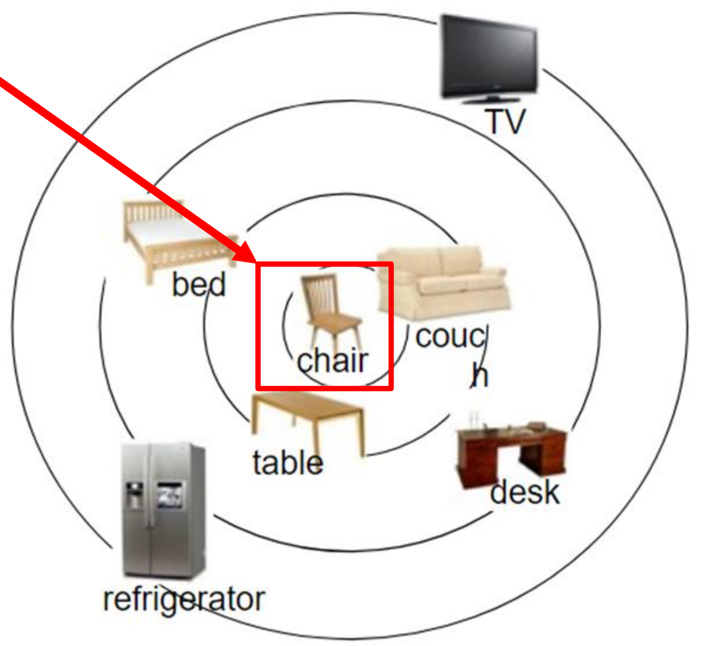


Image Source: https://www.slideserve.com/louise/psy-369-psycholinguistics

Characteristics of Prototype (2): **Core Features**

- **Core features**: **central features of the prototype** that are typically shared by most, if not all, instances within the category and are necessary for **distinguishing** the category from other categories.

- **Saliency Features**: prominent **within a category** but **not necessary** for distinguishing the category from other categories.

- **Peripheral features**: not essential or central to identity of a category

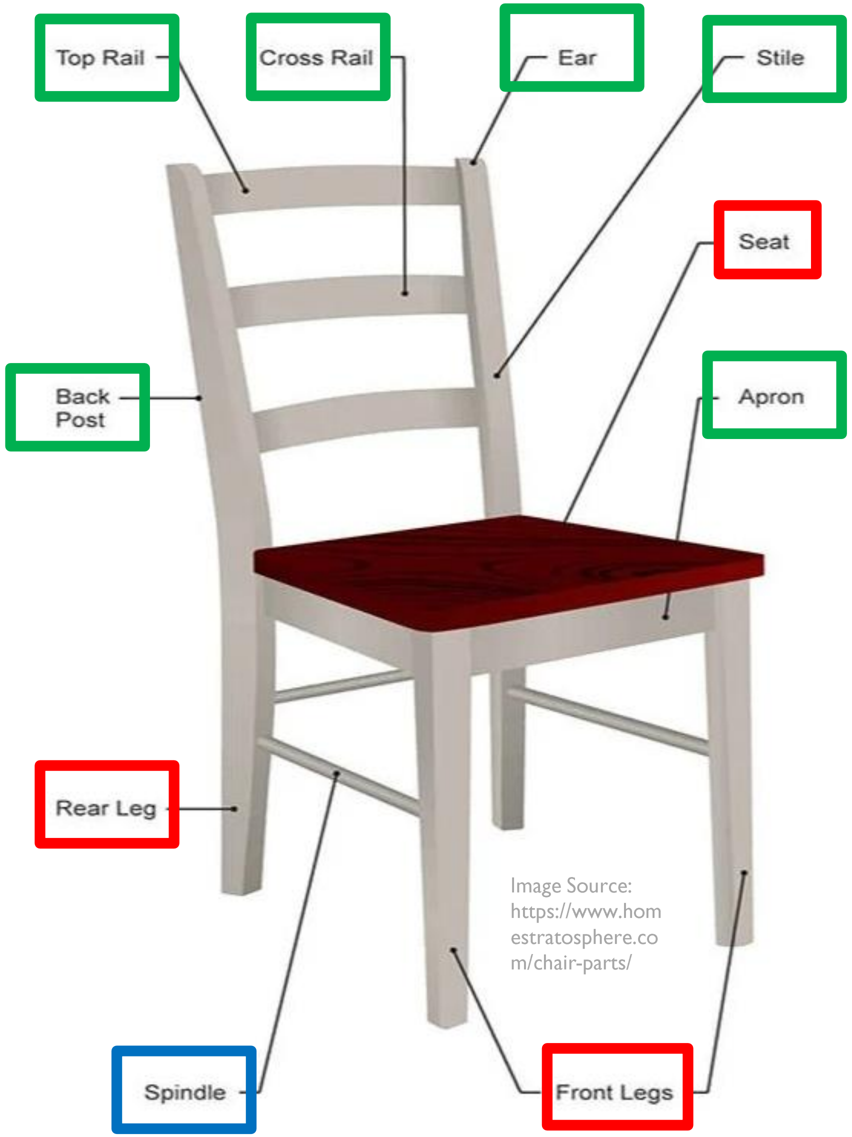


Image Source: https://www.homestratosphere.com/chair-parts/

Characteristics of Prototype (3): **Generalizability**

- Features of a prototype should be **generalizable to other members of the category**, even if those members differ in some respects from the prototype itself

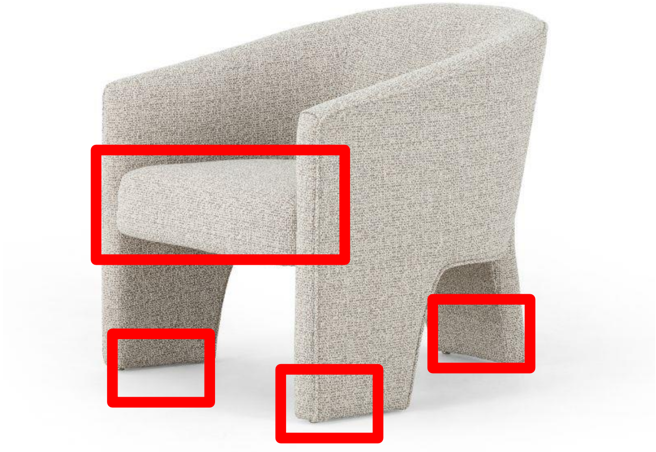
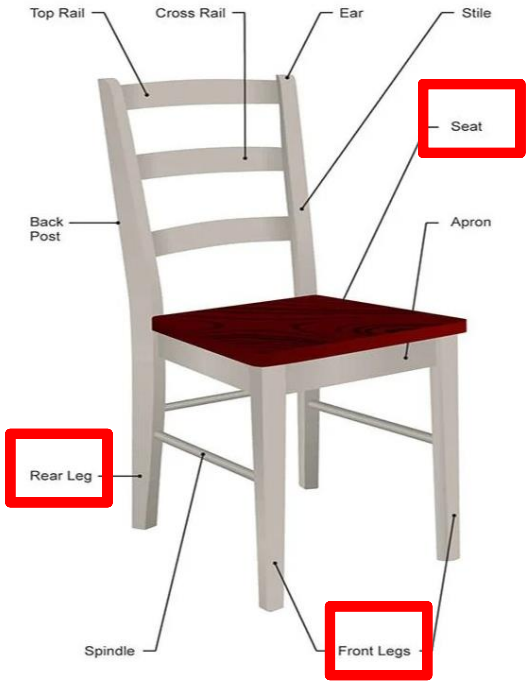


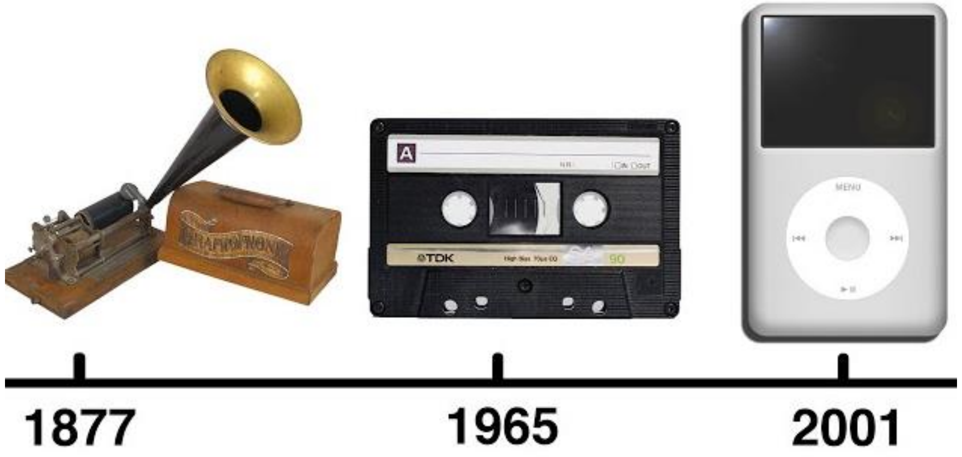
Image Source: mecox.com



Characteristics of Prototype (4): **Flexibility**

- Prototypes are not fixed entities; **they can change** based on new experiences.

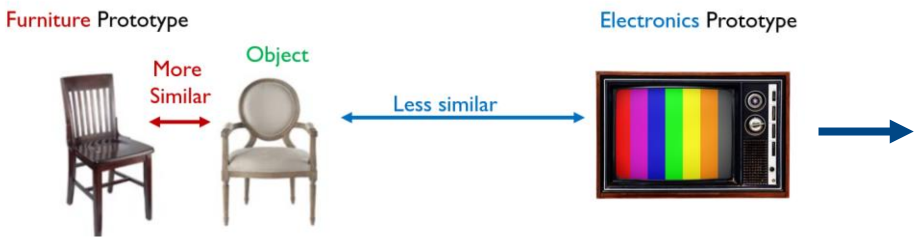
Prototypes for Audio Recording



Source: https://www.youtube.com/watch?app=desktop&v=5Pl2rsLhhwQ

Translation to Machine Learning Language

Prototype Theory	Translation to Machine Learning Language
Typicality	Each class is represented by only **one single prototype**
Core Features	Prototypes have **sparse features**
Generalizability	Prototype features are **generalizable** to samples of class
Flexibility	Learning prototypes is an **incremental** process



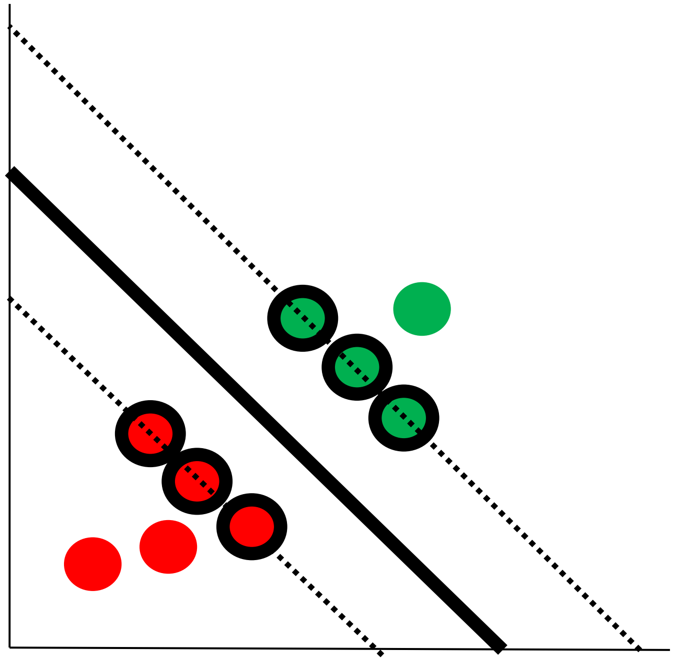
Classification Rule

If the **test sample** is closer to class 0's prototype than class 1's prototype, it is classified as 0; otherwise, it is classified as 1.

natural-learning.cc

Models that are called prototype-based

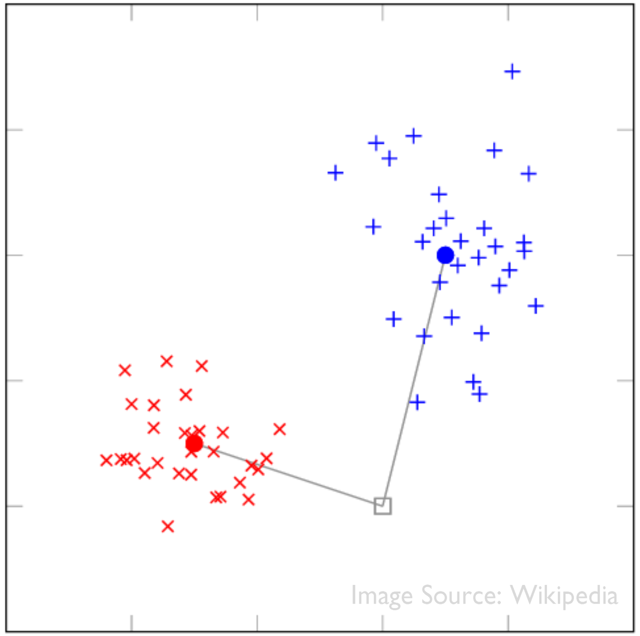
Support Vector Machines (SVM)



Boser, Bernhard E., Isabelle M. Guyon, and Vladimir N. Vapnik. "A training algorithm for optimal margin classifiers." Proceedings of the fifth annual workshop on Computational learning theory. 1992.

Popular in Machine Learning

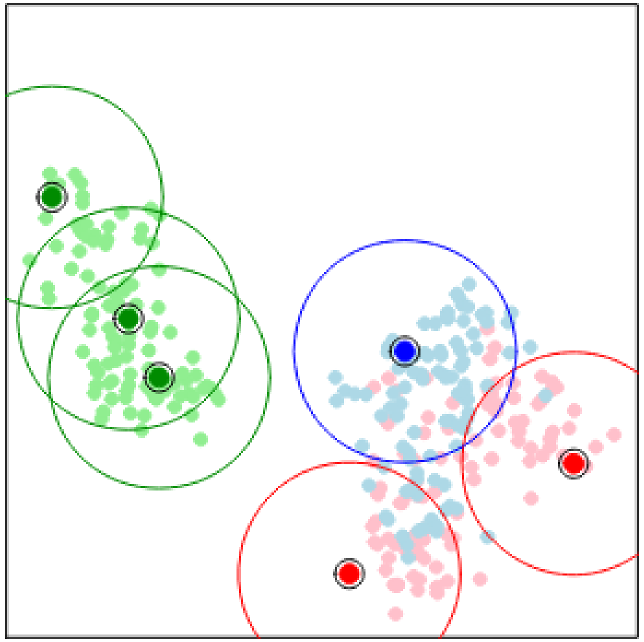
Nearest Centroid Classifier (NCC)



Manning, Christopher; Raghavan, Prabhakar; Schütze, Hinrich (2008). "Vector space classification". Introduction to Information Retrieval. Cambridge University Press.

Popular in information retrieval

Prototype Selection (PS)



Jacob Bien and Robert Tibshirani. Prototype selection for interpretable classification. arXiv preprint arXiv:1202.5933, 2012.

Unsupervised discovery of multiple prototypes from classes (not a direct classifier, a pre-processing method)

Do they match the properties mentioned in prototype theory?

Prototype Theory	Translation to Machine Learning Language	SVM	NCC	PS
Typicality	Each class is represented by only **one single prototype**	✗	✓	✗
Core Features	Prototypes have **sparse features**	✗	✗	✗
Generalizability	Prototype features are **generalizable** to samples of class	✗	✗	✗
Flexibility	Learning prototypes is an **incremental** process	✗	✗	✗

- Unfortunately, the term "**prototype**" is **misused** in machine learning to refer to "**exemplar**" methods.
- This can justify why there is **no authentic implementation of prototype theory** in machine learning.

Can we build a Classifier based on Prototype Theory?

- According to prototype theory, our prototypes of interest should be:
  - **A pair of samples** (One sample from class 0 and one sample from class 1)
  - We should be able to **correctly classify all (or the majority of) samples** based on the following rule:
    - If the test sample is closer to class 0's prototype than class 1's prototype, it is classified as 0; otherwise, it is classified as 1.
  - We don't know what are the core features of the prototypes, but we know that they are the **most generalizable and the sparsest** among all samples.

Brute-Force Approach: Naïve Prototype Classifier

- We can define this classification task as a pure **cross-validation** problem.
- We test all possible pairs of samples from 0 and 1 classes, and with all subsets of features (from length 1 to p) to see which one generalizes best to all samples: X is closer to 0's prototype than 1's prototype, labeled 0 otherwise 1
- We pick the pair with the **lowest error** and **number of features.**

|F|=1

(X1,X3,F1)
(X1,X4,F1)
(X2,X3,F1)
(X2,X4,F1)

(X1,X3,F2)
(X1,X4,F2)
(X2,X3,F2)
(X2,X4,F2)

(X1,X3,F3)
(X1,X4,F3)
(X2,X3,F3)
(X2,X4,F3)

(X1,X3,F4)
(X1,X4,F4)
(X2,X3,F4)
(X2,X4,F4)

|F|=2

(X1,X3,F1,F2)
(X1,X4,F1,F2)
(X2,X3,F1,F2)
(X2,X4,F1,F2)

(X1,X3,F1,F3)
(X1,X4,F1,F3)
(X2,X3,F1,F3)
(X2,X4,F1,F3)

(X1,X3,F1,F4)
(X1,X4,F1,F4)
(X2,X3,F1,F4)
(X2,X4,F1,F4)

(X1,X3,F2,F3)
(X1,X4,F2,F3)
(X2,X3,F2,F3)
(X2,X4,F2,F3)

(X1,X3,F2,F4)
(X1,X4,F2,F4)
(X2,X3,F2,F4)
(X2,X4,F2,F4)

(X1,X3,F3,F4)
(X1,X4,F3,F4)
(X2,X3,F3,F4)
(X2,X4,F3,F4)

|F|=3

(X1,X3, F1,F2,F3}
(X1,X4, F1,F2,F3)
(X2,X3, F1,F2,F3)
(X2,X4, F1,F2,F3)

(X1,X3, F1,F2,F4}
(X1,X4, F1,F2,F4)
(X2,X3, F1,F2,F4)
(X2,X4, F1,F2,F4)

(X1,X3, F1,F3,F4}
(X1,X4, F1,F3,F4)
(X2,X3, F1,F3,F4)
(X2,X4, F1,F3,F4)

(X1,X3, F2,F3,F4}
(X1,X4, F2,F3,F4)
(X2,X3, F2,F3,F4)
(X2,X4, F2,F3,F4)

|F|=4

(X1,X3, F1,F2,F3,F4}
(X1,X4, F1,F2,F3,F4)
(X2,X3, F1,F2,F3,F4)
(X2,X4, F1,F2,F3,F4)

Train Dataset

	F1	F2	F3	F4	y
X1	0	0.25	0.5	0.75	0
X2	0.75	0.5	0.25	1	0
X3	1	1	0.75	0.25	1
X4	0.25	0.25	1	0	1

MATLAB Code: Naïve Prototype Classifier on iris dataset

```
function Mdl=NaivePrototype(X_train,y_train)

    set = 1:size(X_train,2);
    subsets = cell(1, length(set));
    for i = 1:length(set)
        subsets{i} = nchoosek(set, i);
    end
    idn=find(y_train==0);
    idp=find(y_train==1);
    k=0;
    for i=1:numel(subsets)
        for j=1:size(subsets{i},1)
            sfids=subsets{i}(j,:);
            for s=1:size(X_train,1)
                curr_y=y_train(s);
                if curr_y==0
                    nn_neg=knnsearch(X_train(idn,sfids),X_train(s,sfids),'K',2);
                    nn_neg=nn_neg(end);
                    nn_neg=idn(nn_neg);
                    nn_pos=knnsearch(X_train(idp,sfids),X_train(s,sfids),'K',1);
                    nn_pos=idp(nn_pos);
                else
                    nn_neg=knnsearch(X_train(idn,sfids),X_train(s,sfids),'K',1);
                    nn_neg=idn(nn_neg);
                    nn_pos=knnsearch(X_train(idp,sfids),X_train(s,sfids),'K',2);
                    nn_pos=nn_pos(end);
                    nn_pos=idp(nn_pos);
                end
                yt=y_train([nn_neg,nn_pos]);
                yhat=yt(knnsearch(X_train([nn_neg,nn_pos],sfids),X_train(:,sfids),'K',1));
                k=k+1;
                err(k) = sum(yhat~=y_train)/numel(y_train);
                svs(k,1)=nn_neg;
                svs(k,2)=nn_pos;
                nn_features{k}=sfids;
            end
        end
    end
    [minerr,bestk]=min(err);
    Mdl.PrototypeSampleIDs=svs(bestk,1:2);
    Mdl.PrototypeFeatureIDs=nn_features{bestk};
    Mdl.Error=minerr;
    Mdl.Subsets=subsets;
    Mdl.L=0;
    Mdl.MX=X_train(Mdl.PrototypeSampleIDs,Mdl.PrototypeFeatureIDs);
    Mdl.My=y_train(svs(bestk,1:2));
```

```
clear
load fisheriris
y=grp2idx(species)-1;
X=meas;
ids=find(y==0|y==1);
y=y(ids);
X=X(ids,:);
[N,M]=size(X);
rng(42);
indices = randperm(N);
numTestSamples = round(0.2 * N);
trainIdx = indices(numTestSamples+1:end);
testIdx = indices(1:numTestSamples);
X_train = X(trainIdx, :);
X_test = X(testIdx, :);
y_train = y(trainIdx, :);
y_test = y(testIdx, :);

Mdl=NaivePrototype(X_train,y_train);
y_test_NP=Mdl.My(knnsearch(Mdl.MX,X_test(:,Mdl.Protot
ypeFeatureIDs),'K',1));
acc_test=sum(y_test_NP==y_test)/numel(y_test);
```

100% Accuracy with Extreme Sparsity (iris data)

- Test Accuracy = 100%

	PrototypeSampleIDs	[5,6]
	PrototypeFeatureIDs	3
	Error	0
	Subsets	*1x4 cell*
	L	0
	MX	[1.4000;3.3000]
	My	[0;1]

- Prototype for Setosa (Sample 5):
  - Petal length=1.40
- Prototype for Versicolour (Sample 6) :
  - Petal length= 3.30

- **Simple Rule for Classification**: If Petal Length of test sample is closer to Sample 5's petal length (1.4) than Sample 6's petal length (3.3), the prediction is Setosa (0), otherwise it is Versicolour(1).

- Test this rule yourself. It works for all samples! (both train & test)

Sample ID	sepal length	sepal width	petal length	petal width	
1	5	3.6	1.4	0.2	0
2	5.4	3.4	1.5	0.4	0
3	5.8	4	1.2	0.2	0
4	5.7	4.4	1.5	0.4	0
5	5	3.3	1.4	0.2	0
6	4.9	2.4	3.3	1	1
7	6.1	2.8	4	1.3	1
8	5	3.4	1.6	0.4	0
9	4.3	3	1.1	0.1	0
10	5.1	3.8	1.9	0.4	0
11	5.9	3	4.2	1.5	1
12	5.6	2.9	3.6	1.3	1
13	5.1	3.8	1.5	0.3	0
14	4.6	3.6	1	0.2	0
15	5.4	3.9	1.3	0.4	0
16	5.5	3.5	1.3	0.2	0
17	5.4	3	4.5	1.5	1
18	5.1	3.8	1.6	0.2	0
19	6	3.4	4.5	1.6	1
20	5.2	2.7	3.9	1.4	1
21	5.8	2.7	3.9	1.2	1
22	6.1	2.9	4.7	1.4	1
23	6	2.9	4.5	1.5	1
24	5.1	3.3	1.7	0.5	0
25	5.1	3.5	1.4	0.2	0
26	5	2	3.5	1	1
27	6.2	2.9	4.3	1.3	1
28	5.7	3.8	1.7	0.3	0
29	5.1	3.4	1.5	0.2	0
30	4.8	3.4	1.9	0.2	0
31	5.5	2.5	4	1.3	1
32	5.7	3	4.2	1.2	1
33	4.5	2.3	1.3	0.3	0
34	5.2	3.5	1.5	0.2	0
35	4.6	3.2	1.4	0.2	0
36	5.7	2.9	4.2	1.3	1
37	5.1	3.5	1.4	0.3	0
38	6.7	3.1	4.4	1.4	1
39	5.3	3.7	1.5	0.2	0
40	5	2.3	3.3	1	1
41	5.2	3.4	1.4	0.2	0
42	6.5	2.8	4.6	1.5	1
43	4.6	3.1	1.5	0.2	0
44	4.4	2.9	1.4	0.2	0
45	4.8	3.1	1.6	0.2	0
46	5.4	3.4	1.7	0.2	0
47	5.5	2.4	3.7	1	1
48	6.3	2.3	4.4	1.3	1
49	4.8	3	1.4	0.3	0
50	4.4	3	1.3	0.2	0
51	6.4	2.9	4.3	1.3	1
52	4.9	3.1	1.5	0.1	0
53	6.1	3	4.6	1.4	1
54	6.6	3	4.4	1.4	1
55	6.7	3.1	4.7	1.5	1
56	4.7	3.2	1.3	0.2	0
57	5.8	2.6	4	1.2	1
58	5.6	2.7	4.2	1.3	1
59	6.8	2.8	4.8	1.4	1
60	5.9	3.2	4.8	1.8	1
61	6.4	3.2	4.5	1.5	1
62	5	3	1.6	0.2	0
63	5.8	2.7	4.1	1	1
64	5	3.2	1.2	0.2	0
65	6.1	2.8	4.7	1.2	1
66	6	2.2	4	1	1
67	4.8	3	1.4	0.1	0
68	5.5	2.4	3.8	1.1	1
69	5	3.4	1.5	0.2	0
70	5.6	3	4.1	1.3	1
71	5.5	2.3	4	1.3	1
72	5	3.5	1.6	0.6	0
73	5.7	2.8	4.5	1.3	1
74	6.9	3.1	4.9	1.5	1
75	5.5	4.2	1.4	0.2	0
76	4.9	3	1.4	0.2	0
77	4.9	3.1	1.5	0.2	0
78	7	3.2	4.7	1.4	1
79	4.8	3.4	1.6	0.2	0
80	5.6	2.5	3.9	1.1	1

But why nobody has ever tried this classifier before?

Pat Langley
Stanford University

- *"Most courses in Machine Learning* ***ignore older methods with links to cognitive psychology****. Few graduate students read papers more than ten years old, so they are not exposed to the classic literature",* Pat Langley

Langley, Pat. "The computational gauntlet of human-like learning." AAAI 2022.

Weak Connection between ML and Cognitive Psychology

K-Nearest Neighbors resembles exemplar theory, but they are developed independently without any connection between them

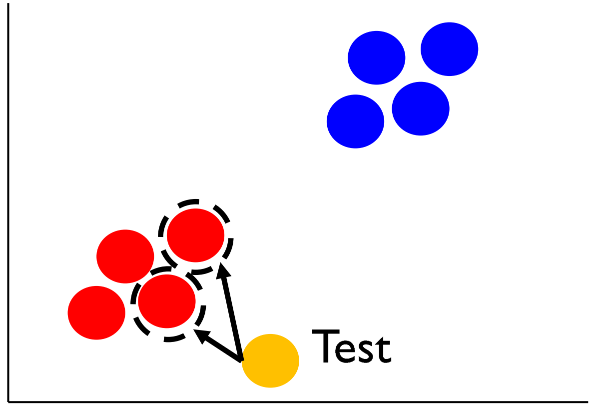
Cognitive Psychology 1975

- **Exemplar Theory of Categorization**
- Individuals categorize objects or events based on their previous experiences with **specific examples**, or exemplars, of those categories.

Rosch, E. (1975). "Cognitive Representations of Semantic Categories." Journal of Experimental Psychology: General, 104(3), 192–233.

1951 Statistics

K-Nearest Neighbors



Fix, E., & Hodges Jr, J. L. (1951). "Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties." Project 21-49-004, U.S. Air Force School of Aviation Medicine

Naive Prototype is not a practical classifier!

- Inherently **Interpretable and Explainable**
- Inherently **robust to label noise (noisy samples)**
- **It is not scalable**: $O(n^3 2^p)$
- It is **vulnerable to the curse of dimensionality**
  - Nearest neighbors become meaningless in higher dimensions
- It is **not robust to noisy features**

- Prototype Theory does not offer any solution for the above problems.
- These problem only arises in computers → **needs a computing solution**

Scalability

- The number of ways to choose **k** features from a set of **p** features (p=total number of features) without regard to the order of selection:

$$\binom{p}{k} = \frac{p!}{(p-k)!k!}$$

- Our sparse features can be in length of k=[1,2,..,p]:

$$\frac{p!}{(p-1)!1!} + \frac{p!}{(p-2)!2!} + \ldots + \frac{p!}{1!(p-1)!} = 2^p - 2 \rightarrow +1 \text{ for } k=p \rightarrow 2^p - 1$$

- Every pair of + samples and – samples are potential candidates: Cost of Pair of samples: $n^+ n^- \approx O(n^2)$

- We also need to cross-validate all combinations of sample pairs and feature subsets: O(2n)

- **Total time Complexity = $O(n^3 2^p)$**
  - Example (only in terms of p): p=784 (MNIST data) → $2^{784}-1 \approx 10^{236}$ > number of atoms in universe (10^{80})

Required Properties

Prototype Theory	Machine Learning	SVM	NCC	PS	NP
Typicality	Each class is represented by only **one single prototype**	✗	✓	✗	✓
Core Features	Prototypes have **sparse features**	✗	✗	✗	✓
Generalizability	Prototype features are **generalizable** to samples of class	✗	✗	✗	✓
Flexibility	Learning prototypes is an **incremental** process	✗	✗	✗	✓
	Robustness to **noisy labels**	✓	✓	✓	✓
	Interpretability (what features are used in the decision?)	✗	✗	✗	✓
	Explainability (clear trajectory from input to output)	○	✓	○	✓
	Robustness to **curse of dimensionality**	✓	✗	✗	✗
	Robustness to **noisy features**	✗	✗	✗	✗
	Computationally **scalable**	○	✓	○	✗

We need to solve these issues to build an **authentic and practical replica of prototype theory** for machine learning

natural-learning.cc

Additionally, we want a natural solution

- **Hyperparameter-free**: Nature does not use hyperparameters
- **Optimization-free:** Optimization does not exist in nature
  - Engineering tricks made by humans
  - Nature instead uses evolution, natural selection, and self-organization
- **Purely based on Nearest neighbor**
  - Brain runs a Nearest neighbor algorithm in an efficient way
    - See the evidence in the next slide

Bypassing curse of dimensionality with nature's algorithm


Fruit Fly

- Naïve prototype classifier is dependent to nearest neighbor search. In higher dimensions , nearest neighbors become irrelevant, because relevant samples become dissimilar, and irrelevant samples become similar.

- Prototype theory does not explain **how humans find the nearest neighbor**, but recent evidence has been found in **fruit fly's brain** (Dasgupta , et. al, 2017)
  - Fruit fly's brain uses a version of **locality-sensitive-Hashing (LSH) algorithm** for nearest neighbor search.

Dasgupta, Sanjoy, Charles F. Stevens, and Saket Navlakha. "A neural algorithm for a fundamental computing problem." **Science** 358.6364 (2017): 793-796.

Bypassing curse of dimensionality via LSH

Piotr Indyk

Rajeev Motwani

- LSH is an efficient solution to nearest-neighbor search by **mapping high-dimensional data points into a lower-dimensional space** in such a way that **similar points are more likely to be hashed into the same bucket** with high probability.

- LSH, focuses on a subset of potential candidates, thereby providing both **computational efficiency** and robustness to the **curse of dimensionality.**

- So, if we use LSH for our nearest neighbor search, we have already solved the first problem!

[1] Indyk, Piotr, and Rajeev Motwani. "Approximate nearest neighbors: towards removing the curse of dimensionality." Proceedings of the thirtieth annual ACM symposium on Theory of computing. 1998.
[2] Charikar, Moses S. "Similarity estimation techniques from rounding algorithms." Proceedings of the thiry-fourth annual ACM symposium on Theory of computing. 2002.

Properties of Interest

Prototype Theory	Machine Learning	SVM	NCC	PS	NP
Typicality	Each class is represented by only **one single prototype**	✗	✓	✗	✓
Core Features	Prototypes have **sparse features**	✗	✗	✗	✓
Generalizability	Prototype features are **generalizable** to samples of class	✗	✗	✗	✓
Flexibility	Learning prototypes is an **incremental** process	✗	✗	✗	✓
	Robustness to **noisy labels**	✓	✓	✓	✓
	Interpretability (what features are used in the decision?)	✗	✗	✗	✓
	Explainability (reasoning the decision)	○	✓	○	✓
	Robustness to **curse of dimensionality**	✓	✗	✗	✓
	Robustness to **noisy features**	✗	✗	✗	✗
	Computationally **scalable**	✗	✓	○	✗

We have less problems to solve now

natural-learning.cc

Attacking computational scalability with respect to *n*

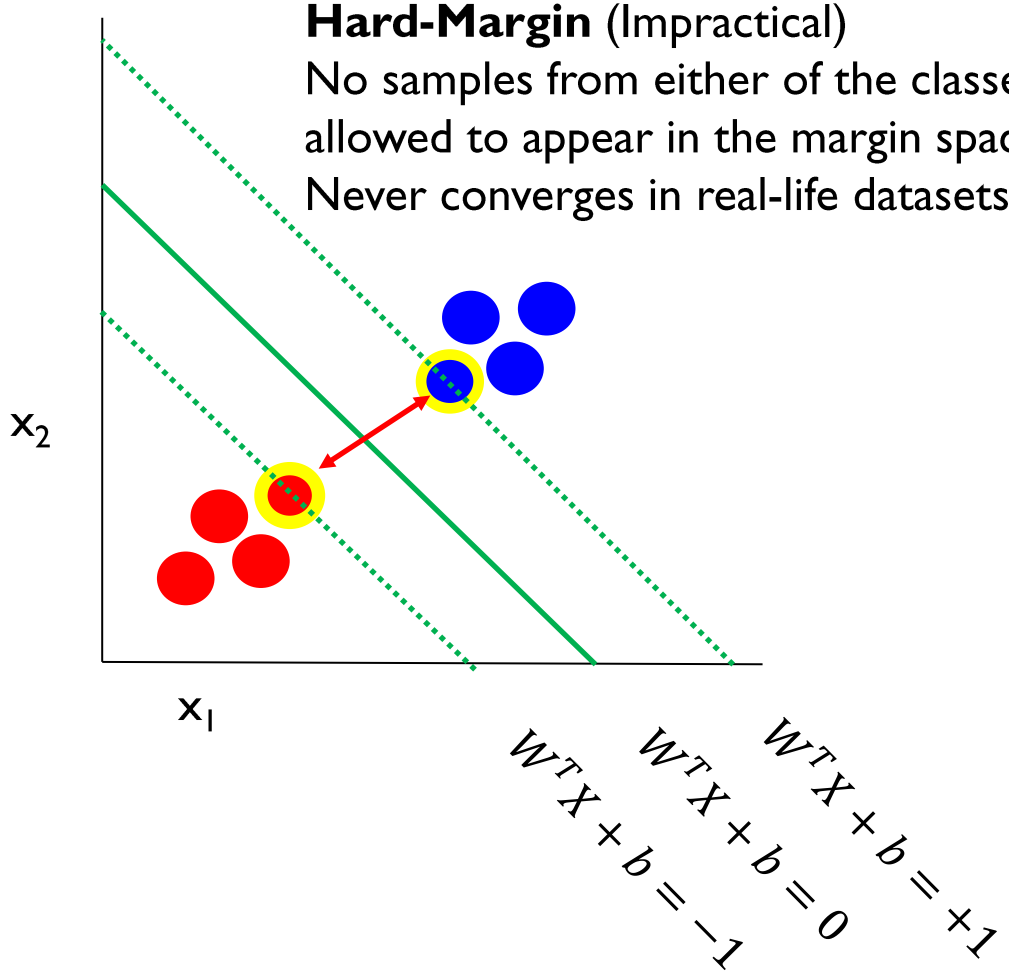
- To solve this, we need to time travel to 1995, take some lessons from Soft-Margin SVM and return back.

Image source: stock.adobe.com

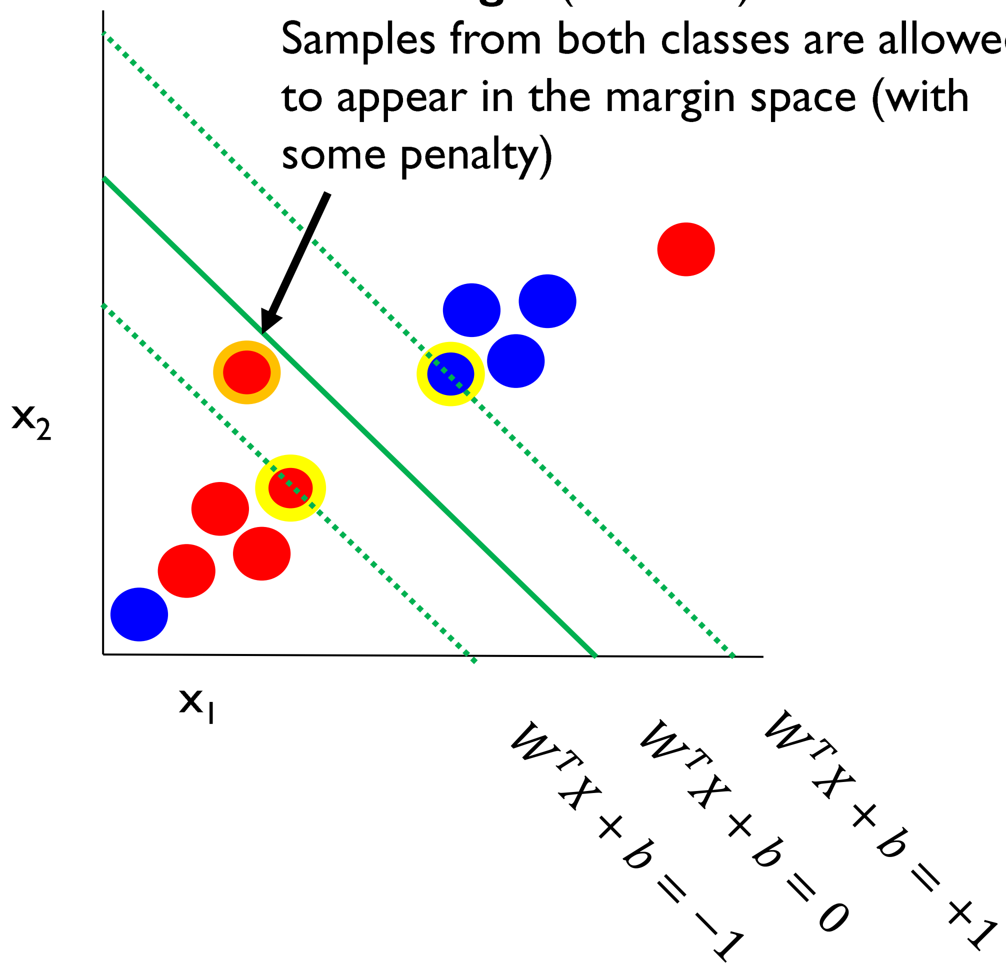
Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273–297.

Hard Margin SVM vs. Soft Margin SVM

Hard-Margin (Impractical)
No samples from either of the classes are allowed to appear in the margin space
Never converges in real-life datasets



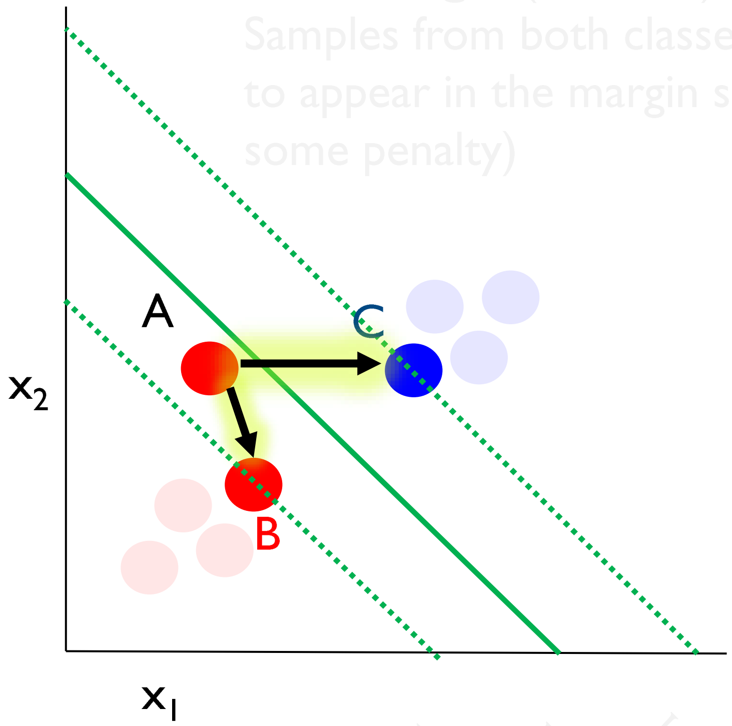
Soft-Margin (Practical)
Samples from both classes are allowed to appear in the margin space (with some penalty)



Re-designing SVM with typicality principle of prototype theory

- Prototype theory suggests that **only one support vector exists for each class.** This assumption can simplify the problem of finding support vectors to a regular cross-validation.
- **A** can serve as an ideal **pivot** to find support vectors **B** and **C**
- The Nearest Neighbor of **A** from its own class is **B**, and its nearest neighbor from the other class is **C**.
- If we put **A** as a **pivot**, it shows us the path to reach support vectors **B** and **C**.
- Since **B** and **C** are **actual support vectors**, if we cross-validate their **generalizability**, they **pass** the test!
- So, **margin violation** samples are a very informative source for finding support vectors without the need for optimization!

Soft-Margin (Practical)
Samples from both classes are allowed to appear in the margin space (with some penalty)

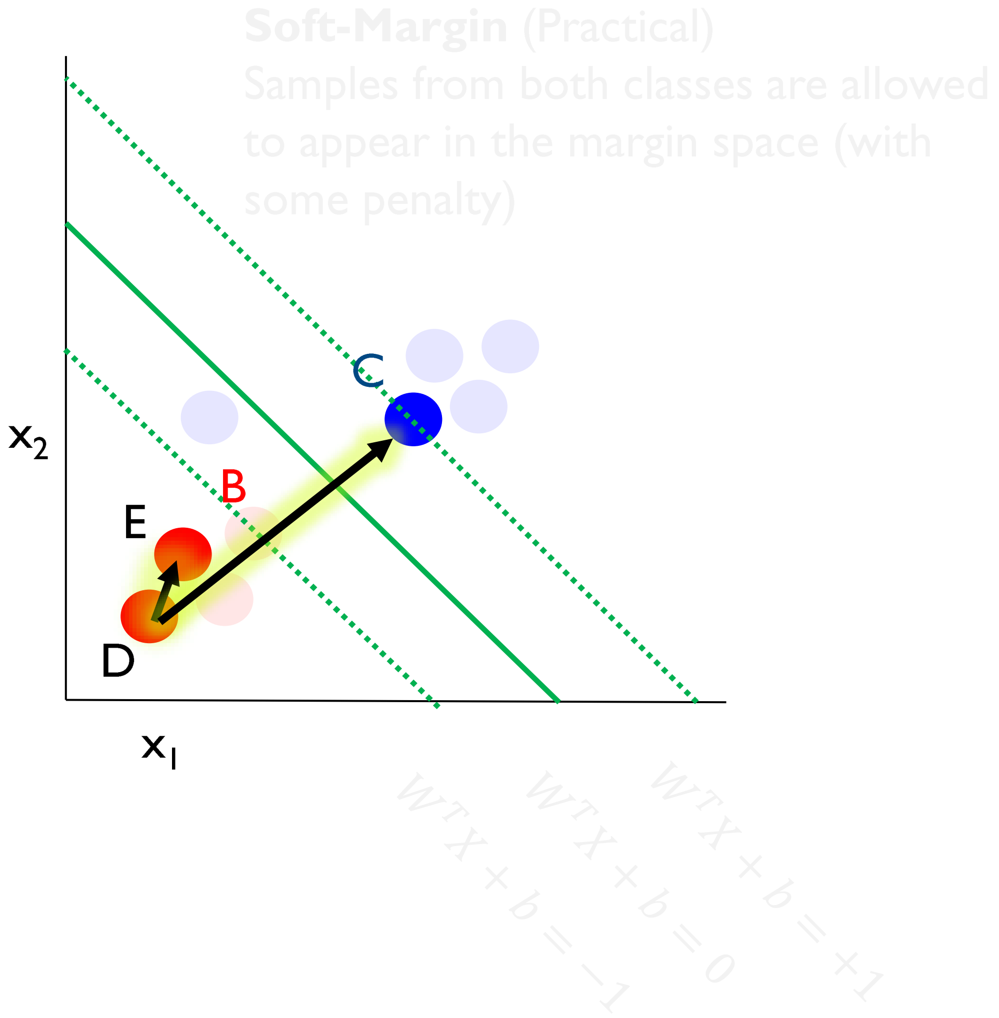


Re-designing SVM with typicality principle of prototype theory

- What about other regular samples?

- The nearest neighbor of **D** is **E** from its class and **C** from the other class. Again, **C** can be found as a support vector, but **E** is a little far from the actual support vector **B**.

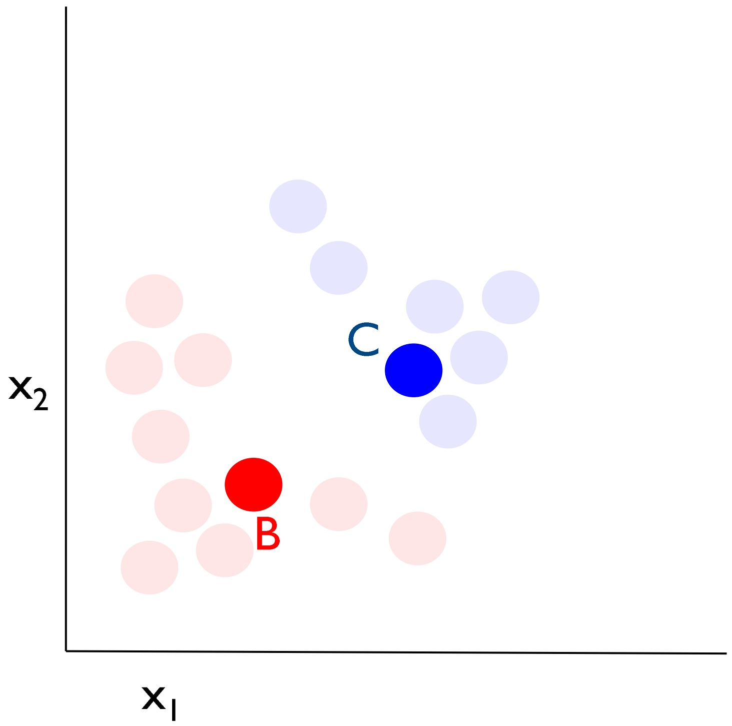
- In cross-validation, it is likely that the decision boundary will **not generalize as well as the decision boundary between B and C**, so it automatically will be beaten by actual support vectors (**B** and **C**) suggested by **A** as a pivot.



Fuzzy decision boundary vs. SVM's linear boundary

- We can now enjoy a **fuzzy decision boundary,** which gives us more flexibility

- Removes margin width as a hyperparameter
  - We keep going **hyperparameter-free**



Computational benefit

- Instead of testing all pairs of samples, we can **limit** our **search** to **triplets**
  1. Sample
  2. The nearest neighbor from its class (support vector candidate 1)
  3. The nearest neighbor from the other class (support vector candidate 2)

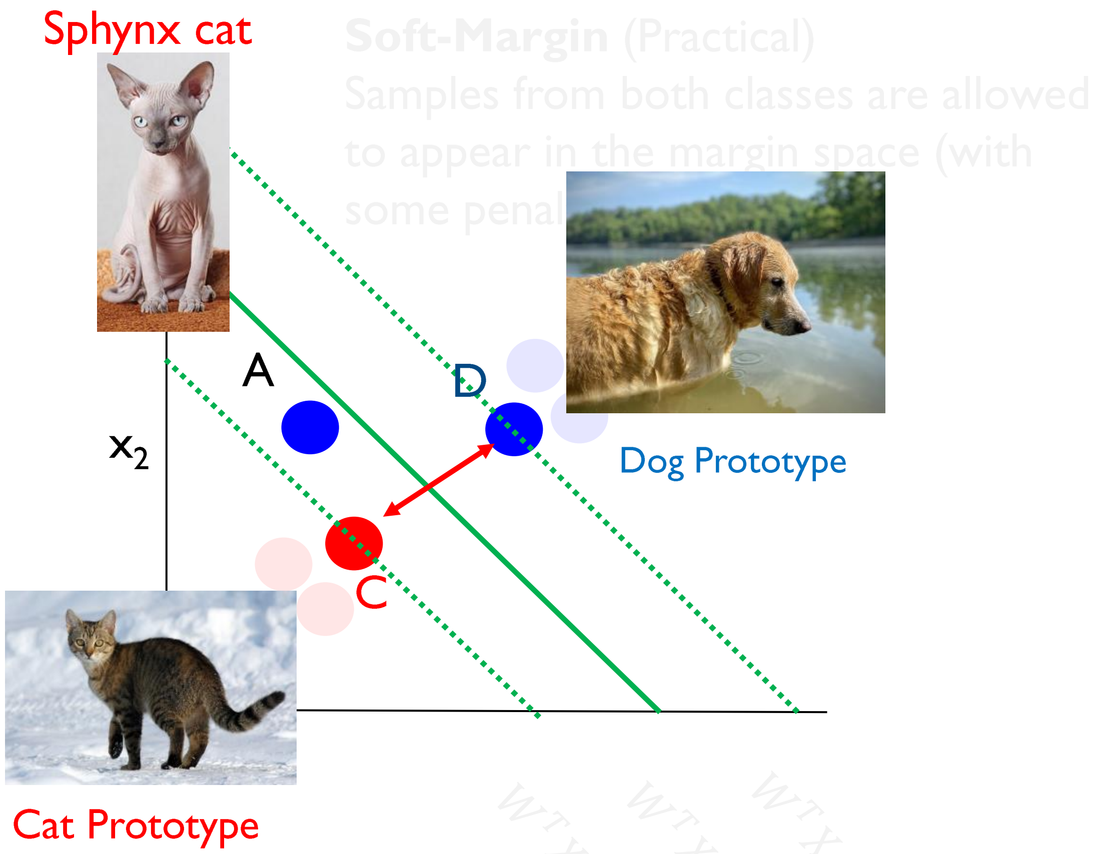
- This reduces the complexity of search in terms of $\boldsymbol{n}$ from $\boldsymbol{O(n^2)}$ **to** $\mathbf{O(n)}$

- Cross-validation costs only $\mathbf{O(2n)}$ due to relaxed assumption of **prototype theory**
  - Distance of all samples to only **2 support vectors**

Intuitive Example of Triplets

- Sphynx cat is a distinctive cat breed often confused for a dog because of its unique physical characteristics.
  - Sphynx cat is a margin violation sample

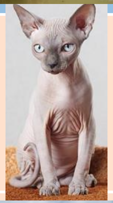
- Triplets helps us gain efficiency not only in terms of "n" but also in terms of "p". How?



Example of a triplet

Applying "Generalizability Principle"

non-core feature candidate

		Size	Weight	Mustache
Nearest Neighbor from the opposite class (dog) Dog prototype candidate		0.80	0.60	0
		↑ 0.20	↑ 0.10	↑ 0
Pivot (Cat) Sphynx		0.05<0.2 0.60	0.01<0.1 0.50	1 ≮ 0 0
		↓ 0.05	↓ 0.01	↓ 1
Nearest Neighbor from its own class (Cat) Cat prototype Candidate		0.55	0.51	1

Size and **Weight** can still be the core features of prototypes because they make a cat seem closer to a cat as expected.

Mustache is a candidate for **a non-core feature:** it makes pivot (cat) seem closer to the prototype of the opposite class (dog), the class it does not belong to.

natural-learning.cc

Cat/dog photos from Wikipedia

Applying "Generalizability Principle"

- Test Generalization of All samples derived without feature "Mustache"

Feature Matrix with censored non-core features

	Size	Weight	Mustache
Dog 1	0.5	0.5	0
Dog 2	0.4	0.1	0
Dog 3	0.3	0.2	0
Dog 4	0.80	0.60	0
Cat 1	0.60	0.50	1
Cat 2	0.8	0.9	1
Cat 3	0.5	0.8	1
Cat 4	0.55	0.51	0

Prototype Candidates 1 **nominated by Sample 1** after removing non-core candidate

	Size	Weight	Mustache
	0.5	0.5	0
	0.6	0.5	1

Prediction: Dog Because it is closer to dog

Cat/dog photos from Wikipedia

Applying "Generalizability Principle"

- Test Generalization of All samples derived without feature "Mustache"

Feature Matrix with censored non-core features

	Size	Weight	Mustache
Dog 1	0.5	0.5	0
Dog 2	0.4	0.1	0
Dog 3	0.3	0.2	0
Dog 4	0.80	0.60	0
Cat 1	0.60	0.50	1
Cat 2	0.8	0.9	1
Cat 3	0.5	0.8	1
Cat 4	0.55	0.51	0

Prototype Candidates 1 **nominated by Sample 1** after removing non-core candidate

	Size	Weight	Mustache
	0.5	0.5	0
	0.6	0.5	1

natural-learning.cc

Cat/dog photos from Wikipedia

Applying "Generalizability Principle"

- Test Generalization of All samples derived without feature "Mustache"

- Error =0.125

Prototype Candidates **1 nominated by Sample 1** after removing non-core candidate

	Size	Weight	Mustache
	0.5	0.5	0
	0.6	0.5	1

Feature Matrix with censored non-core features

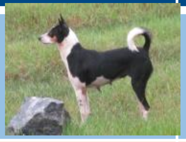
	Size	Weight	Mustache
Dog 1	0.5	0.5	0
Dog 2	0.4	0.1	0
Dog 3	0.3	0.2	0
Dog 4	0.80	0.60	0
Cat 1	0.60	0.50	1
Cat 2	0.8	0.9	1
Cat 3	0.5	0.8	1
Cat 4	0.55	0.51	0

Cat/dog photos from Wikipedia

Applying "Generalizability Principle"

- The next pivot nominates two different samples with different non-core features.
- Error =0.375

Prototype Candidates **2 nominated by Sample 2** after removing non-core candidate

	Size	Weight	Mustache
	0.4	0.1	0
	0.5	0.8	1

Feature Matrix with censored non-core features

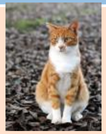
	Size	Weight	Mustache
Dog 1	0.5	0.5	0
Dog 2	0.4	0.1	0
Dog 3	0.3	0.2	0
Dog 4	0.80	0.60	0
Cat 1	0.60	0.50	1
Cat 2	0.8	0.9	1
Cat 3	0.5	0.8	1
Cat 4	0.55	0.51	0

Cat/dog photos from Wikipedia

Applying "Generalizability Principle"

- The next pivot nominates two different samples with different non-core features.
- Error =0.25

Prototype Candidates ***n* nominated by Sample *n*** after removing non-core candidate

	Size	Weight	Mustache
	0.80	0.6	1
	0.55	0.51	1

Feature Matrix with censored non-core features

	Size	Weight	Mustache
Dog 1	0.5	0.5	0
Dog 2	0.4	0.1	0
Dog 3	0.3	0.2	0
Dog 4	0.80	0.60	0
Cat 1	0.60	0.50	1
Cat 2	0.8	0.9	1
Cat 3	0.5	0.8	1
Cat 4	0.55	0.51	0

Cat/dog photos from Wikipedia

Applying "Generalizability Principle"

The minimum Generalization Error is obtained for the Removal of "**Mustache**" (Error = 0.125)

	Size	Weight
	0.80	0.6
	0.55	0.51

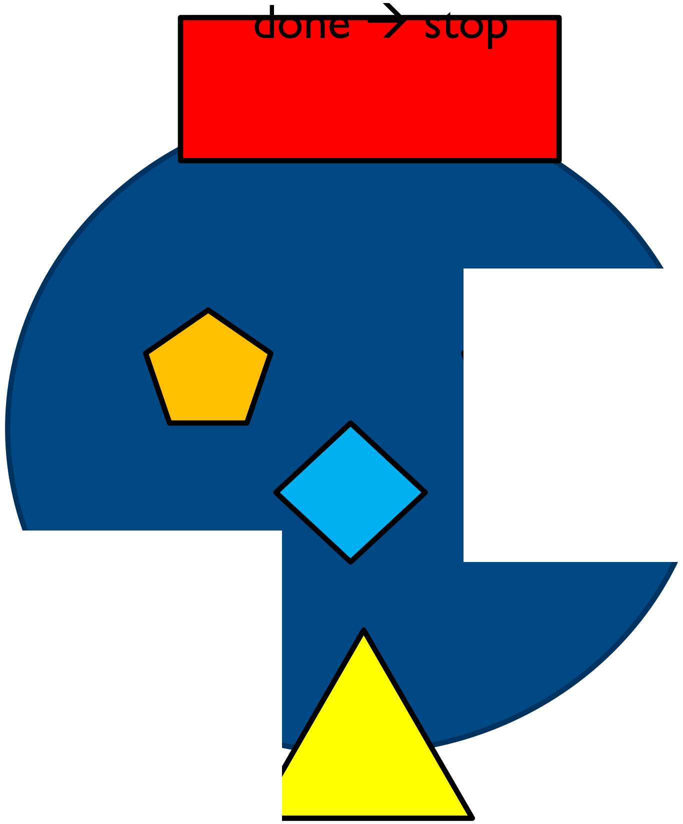
Removing **Mustache** globally from the feature matrix

	Size	Weight
Dog 1	0.5	0.5
Dog 2	0.4	0.1
Dog 3	0.3	0.2
Dog 4	0.80	0.60
Cat 1	0.60	0.50
Cat 2	0.8	0.9
Cat 3	0.5	0.8
Cat 4	0.55	0.51

Cat/dog photos from Wikipedia

Intuitive Example (1)

No more pruning can be done → stop

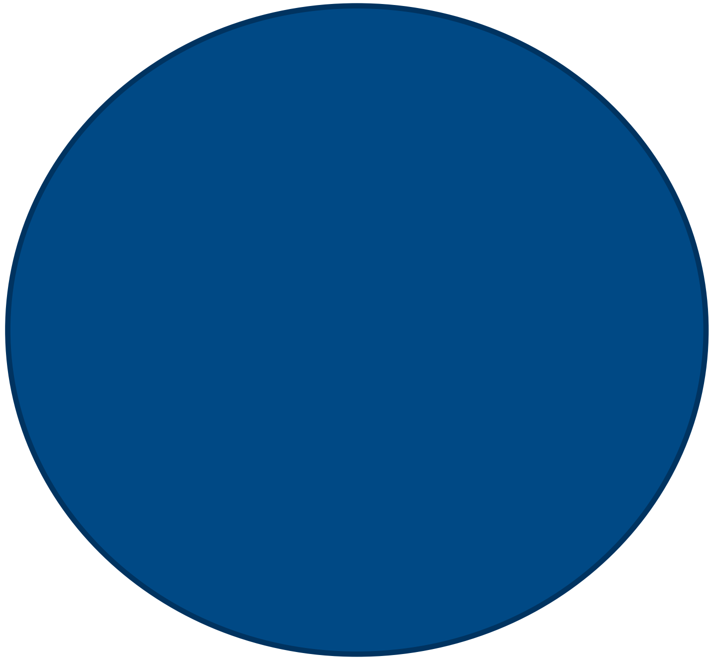


Non-prototype Example

We can't turn this complex object into a simpler prototype in **one step** without having a **feedback** after pruning part of complexity

We remove this part, but it was part of core features, so it does not generalize well, thus we discard the pruning.

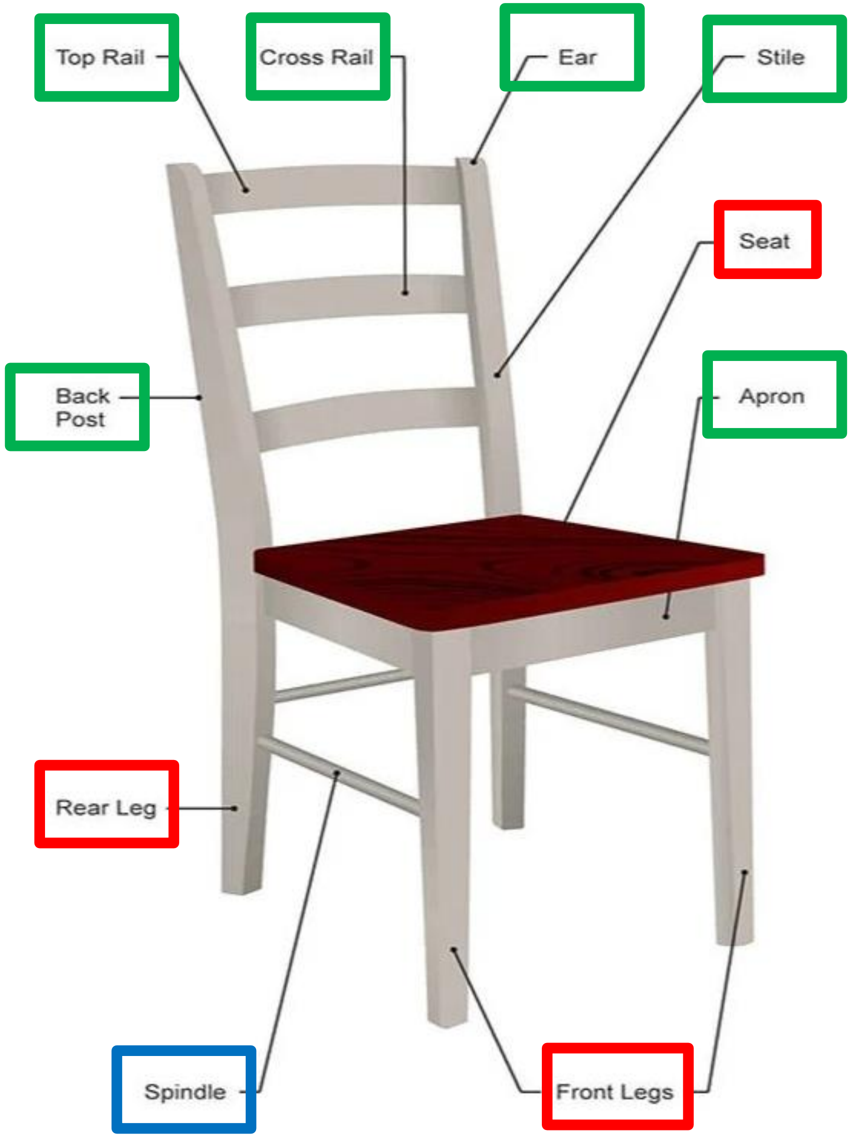
We remove this part, it was not part of core features, so it generalized well, thus we accept the pruning.



Basic Level (Sparse Prototype)
Core Feature : Circle

Intuitive Example (2)

- Iteration #1
- Iteration #2
- Iteration #3
- Iteration #4
- Iteration #5
- Stop



"You know you've achieved ***perfection****, not when you have nothing more to add, but* ***when you have nothing to take away****"* (**Saint-Exupery**, Airman's Odyssey, 1943)

Saint-Exupery, 1943

Analogy in Academic Writing

- 2000 characters limit in conference paper submission

- Iteration #1" You write your first version: 5000 characters
- Iteration #2: Remove some non-core information: 3500 characters
- …
- Iteration #10: Remove non-core information: 2000 characters

Iteration #2

New non-core feature candidate at iteration #2

	Size	Weight	
New Nearest Neighbor from Dog Class at **iteration# 2** 	0.80	0.45	
	↑ 0.2	↑ 0.05	
Pivot (Cat) Sphynx  0.05<0.2	0.60	0.50	0.1 ≮ 0.05
	↓ 0.05	↓ 0.1	
New Nearest Neighbor from Cat Class at **iteration# 2** 	0.55	0.40	

The triplet space of Sphynx cat has **new neighbors** because we are in **a new feature space** and have a **better similarity relevance** due to **removed non-core (noisy) features.** We also have **purer** classes.

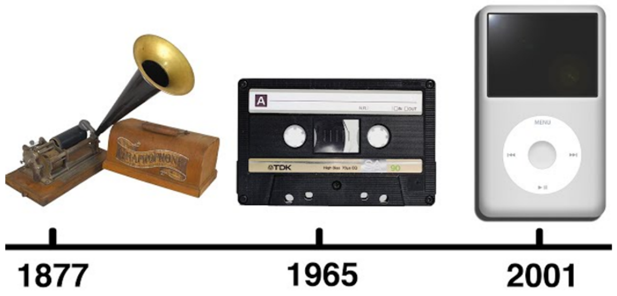
“Flexibility Principle”

– As it can be seen, prototypes can change during each iteration.
  - **Flexibility condition of prototype theory** → Incremental property

Characteristics of Prototype (4): **Flexibility**

- Prototypes are not fixed entities; **they can change** based on new experiences.

Prototypes for Audio Recording



Applying "Typicality Principle"

- If we cannot prune more features, that means that we have reached the **core features of prototypes**
- After removing weight, **Size** becomes the core feature.
- Those prototype candidates that generalize better with the "Size" feature as the core feature become our **final prototype samples**

Final Prototype	Size
	0.80
	0.55

Generalization Error = 0.125

Cat/dog photos from Wikipedia

Massive time complexity reduction for feature search

- By iterative pruning of non-core features, we naturally reach to the core features of prototypes in a very efficient way

- So, with this method, we **no longer need to test all subsets of features**.
  - Reducing complexity for feature search from $\boldsymbol{O(2^p)}$ to $\boldsymbol{O(pL)}$
    - Where L is the number of pruning iterations (e.g., for MNIST 0 vs. 1, L=10)

Problem Solved

- **Scalable**
  - Reducing Time Complexity from $O(n^3 2^p)$ to $O(n^2 pL)$, where L is number pruning Iterations.
  - The algorithm design allows **high parallelization with GPUs**.
    - Each pruning/validation task can be parallelized or distributed for each sample independently.
- Robust to **Curse of Dimensionality**
  - **Thanks to LSH**
- Robustness to **Noisy Features**
  - By Iterative pruning of non-core features, features of sparse prototypes shows up naturally
  - **No hyperparameter** required for number of pruning iterations
    - Stopping criteria: **when no more features can be pruned**

Problem Solved

Prototype Theory	Machine Learning	SVM	NCC	PS	NP
Typicality	Each class is represented by only **one single prototype**	✗	✓	✗	✓
Core Features	Prototypes have **sparse features**	✗	✗	✗	✓
Generalizability	Prototype features are **generalizable** to samples of class	✗	✗	✗	✓
Flexibility	Learning prototypes is an **incremental** process	✗	✗	✗	✓
	Robustness to **noisy labels**	✓	✓	✓	✓
	Interpretability (what features are used in the decision?)	✗	✗	✗	✓
	Explainability (reasoning the decision)	○	✓	○	✓
	Robustness to **curse of dimensionality**	✓	✗	✗	✓
	Robustness to **noisy features**	✗	✗	✗	✓
	Computationally **scalable**	✗	✓	○	✓

This new algorithm is now called "Natural Learning (NL)"

Prototype Theory	Machine Learning	SVM	NCC	PS	NL
Typicality	Each class is represented by only **one single prototype**	✗	✓	✗	✓
Core Features	Prototypes have **sparse features**	✗	✗	✗	✓
Generalizability	Prototype features are **generalizable** to samples of class	✗	✗	✗	✓
Flexibility	Learning prototypes is an **incremental** process	✗	✗	✗	✓
	Robustness to **noisy labels**	✓	✓	✓	✓
	Interpretability (what features are used in the decision?)	✗	✗	✗	✓
	Explainability (reasoning the decision)	○	✓	○	✓
	Robustness to **curse of dimensionality**	✓	✗	✗	✓
	Robustness to **noisy features**	✗	✗	✗	✓
	Computationally **scalable**	✗	✓	○	✓

After Rosch's 1973 paper "Natural Categorizes"

Rosch, Eleanor H. "Natural categories." Cognitive psychology 4.3 (1973): 328-350.

natural-learning.cc

Training Algorithm in 20 lines!

Algorithm 1 NLTrain

Input: training set (x, y) (n samples and p features), $y_i = \{0, 1\}$, and features of best prototype M
Output: prototype samples (s_{best} and o_{best}), and their labels, prototype features M
if M is null **then**
    $M \leftarrow \{1, 2, ..., p\}$ *//initialization of prototype features*
end if
$x = x(:, M)$ *// Copy of x with features in M*
$e_{best} \leftarrow \infty$ *//initialization of best error. Allowing NL to learn better prototypes at each iteration.*
for each sample i in x **do**
    $s \leftarrow$ index of x_i's nearest neighbor from same class using LSH *//prototype sample candidate*
    $o \leftarrow$ index of x_i's nearest neighbor from opposite class using LSH *//prototype sample candidate*
    $C \leftarrow$ indices of features in M that make x_i closer to x_s than x_o *// prototype features candidate*
    $\hat{y} \leftarrow NLPredict(x_s, x_o, y_s, y_o, C, x)$ *// test the generalization of prototype candidate*
    $e \leftarrow \sum(y \neq \hat{y})$
    if $e < e_{best}$ & $|C| > 1$ **then**
        $(s_{best}, o_{best}) \leftarrow (s, o)$ *// Best prototype samples*
        $C_{best} \leftarrow C$ *//Best prototype features*
        $e_{best} \leftarrow e$ *//Best error so far*
    end if
end for
if $|C_{best}| \neq |M|$ **then**
    $M \leftarrow C_{best}$
    $NLTrain(x, y, M)$
end if

Hyperparameter-free

Self-explainable Algorithm

Code available in MATLAB, Python, and R

Algorithm 2 NLPredict

Input: data (x), prototype samples (x_o and x_s) and corresponding labels (y_o and y_s) and features (M)
Output: $\hat{y}$ (Predicted labels)
$x \leftarrow x(:, M)$ // copy of x with prototype features (M or C_i)
for each sample i in x **do**
    $(d_s, d_o) \leftarrow D(x_i, x_s, x_o)$ *//Distance of example to prototype samples s and o*
    $\hat{y}_i = y_s$
    if $d_o < d_s$ **then**
        $\hat{y}_i = y_o$
    end if
end for

Illustrative Example: Iteration # 1

1 – Feature Matrix

	F1	F2	F3	F4	y
X1	0	0.25	0.5	0.75	0
X2	0.75	0.5	0.25	1	0
X3	1	1	0.75	0.25	1
X4	0.25	0.25	1	0	1

2. BestError = inf

NNS	NNO	Pivot
X2	X4	X1
X1	X3	X2
X4	X2	X3
X3	X1	X4

3 – For each sample, find the Nearest Neighbor via **Locality-Sensitive Hashing** from the same class (NNS) and the opposite class (NNO)

4- **Prune** features that make Pivot closer to NNO comparing NNS

		F1	F2	F3	F4
NNS	X2	0.75	0.5	0.25	1
Pivot	X1	0	0.25	0.5	0.75
NNO	X4	0.25	0.25	1	0

		F1	F2	F3	F4
NNS	X1	0	0.25	0.5	0.75
Pivot	X2	0.75	0.5	0.25	1
NNO	X3	1	1	0.75	0.25

		F1	F2	F3	F4
NNS	X4	0.25	0.25	1	0
Pivot	X3	1	1	0.75	0.25
NNO	X2	0.75	0.5	0.25	1

		F1	F2	F3	F4
NNS	X3	1	1	0.75	0.25
Pivot	X4	0.25	0.25	1	0
NNO	X1	0	0.25	0.5	0.75

5- Prototype Candidates

	F3	F4
S0	0.25	1
S1	1	0

	F2	F3	F4
S0	0.25	0.5	0.75
S1	1	0.75	0.25

	F3	F4
S0	0.25	1
S1	1	0

	F3	F4
S0	0.5	0.75
S1	0.75	0.25

6- Cross Validation

	D(S0)	D(S1)	ŷ
X1	**0.35**	0.90	0
X2	**0.00**	1.25	0
X3	0.90	**0.35**	1
X4	1.25	**0.00**	1

✓

	D(S0)	D(S1)	ŷ
X1	**0.00**	0.94	0
X2	**0.43**	1.03	0
X3	0.94	**0.00**	1
X4	0.90	**0.83**	1

	D(S0)	D(S1)	ŷ
X1	**0.35**	0.90	0
X2	**0.00**	1.25	0
X3	0.90	**0.35**	1
X4	1.25	**0.00**	1

	D(S0)	D(S1)	ŷ
X1	**0.00**	0.56	0
X2	**0.35**	0.90	0
X3	0.56	**0.00**	1
X4	0.90	**0.35**	1

7- Selection of **The Best Prototype**

Prototype Features / **Prototype Samples**

	F3	F4
S0	0.25	1
S1	1	0

8- Update feature matrix by keeping features of the best prototypes F3 F4

9- Repeat 1-8. Stop when best features do not change anymore.

Illustrative Example: Iteration # 2

1 – Feature Matrix

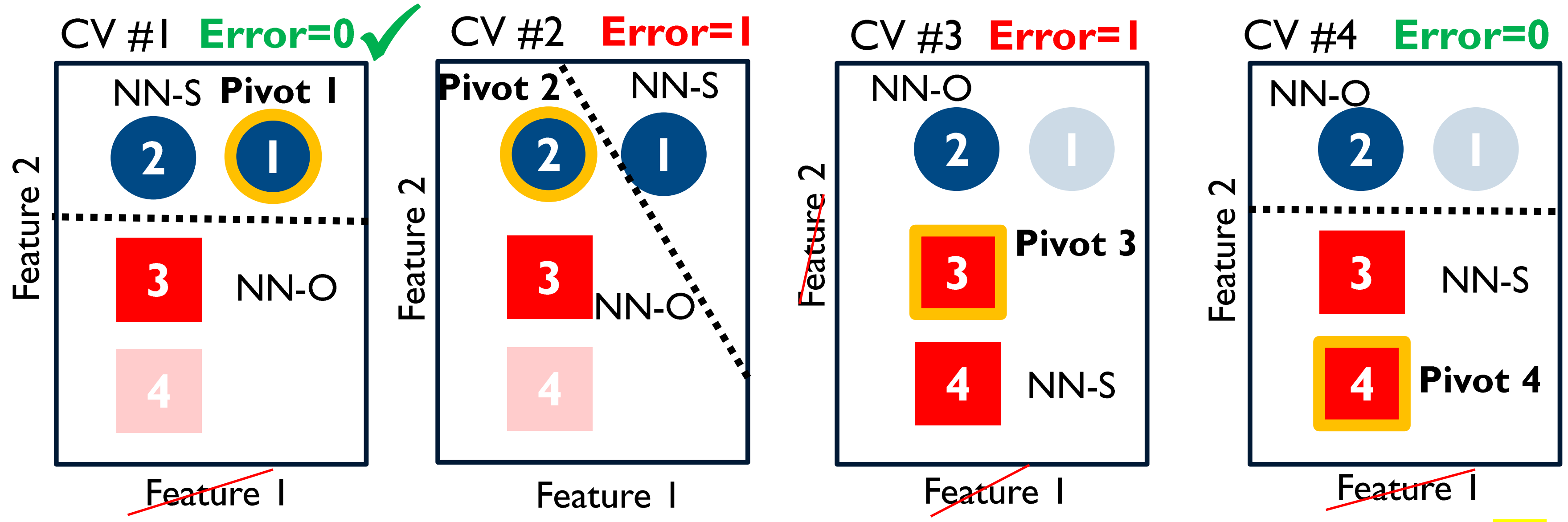
	F1	F2	F3	F4	y
X1	0	0.25	0.5	0.75	0
X2	0.75	0.5	0.25	1	0
X3	1	1	0.75	0.25	1
X4	0.25	0.25	1	0	1

2. BestError = inf

…

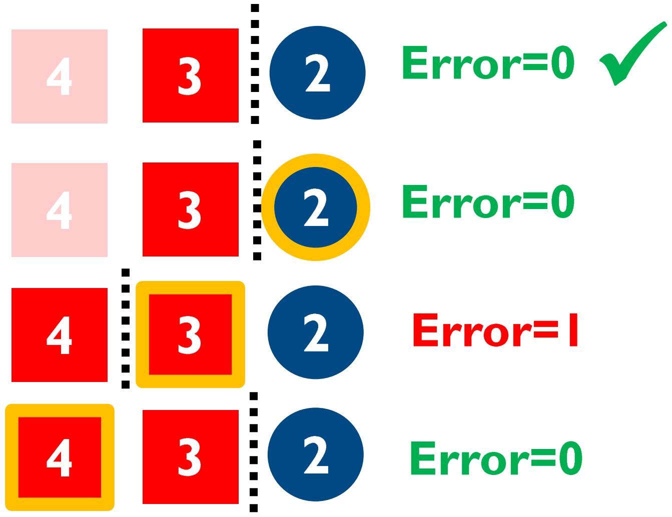
Geometric View of Decision Boundaries

Iteration #1 **Active Features: [1,2]**



Best Prototype's features: [2]
Best Prototype Samples: [2,3]

Iteration #2



Active Features = [2]
Final Prototype samples: [2,3]
Final Prototype features: [2]

Feature 1 has the same distance to pivot's NN-S and NN-O → Non-core pruning Candidate

Both features are relevant, so no pruning candidate

Both features have the same distance to pivot's NN-S and NN-O → cross-validation will not be performed

Feature 1 has the same distance to pivot's NN-S and NN-O → Non-core pruning Candidate

MNIST Dataset (0 vs. 1) – Iteration 1

- Iteration=1 PivotSample=1/12665 PrototypeCandidate=[5845,8205], NumFeatures=166/784, Error=0.0756
- Iteration=1 PivotSample=4/12665 PrototypeCandidate=[844,10217], NumFeatures=153/784, Error=0.0360
- Iteration=1 PivotSample=21/12665 PrototypeCandidate=[3840,11859], NumFeatures=150/784, Error=0.0258
- Iteration=1 PivotSample=76/12665 PrototypeCandidate=[815,11478], NumFeatures=148/784, Error=0.0188
- Iteration=1 PivotSample=208/12665 PrototypeCandidate=[3175,7938], NumFeatures=122/784, Error=0.0121
- Iteration=1 PivotSample=245/12665 PrototypeCandidate=[4403,7780], NumFeatures=110/784, Error=0.0114
- Iteration=1 PivotSample=402/12665 PrototypeCandidate=[513,7078], NumFeatures=125/784, Error=0.0081
- Iteration=1 PivotSample=2062/12665 PrototypeCandidate=[5011,7780], NumFeatures=165/784, **Error=0.0046**

This sample likely is a **margin violation sample** that **guides us towards good support vectors** (e.g. , **A** in our SVM example)

MNIST Dataset (0 vs. 1) – Iteration 2

- Iteration=2 PivotSample=1/12665 PrototypeCandidate=[62,8205], NumFeatures=88/165, Error=0.0882
- Iteration=2 PivotSample=2/12665 PrototypeCandidate=[3652,8205], NumFeatures=86/165, Error=0.0853
- Iteration=2 PivotSample=3/12665 PrototypeCandidate=[4167,10217], NumFeatures=93/165, Error=0.0791
- Iteration=2 PivotSample=5/12665 PrototypeCandidate=[5396,9876], NumFeatures=99/165, Error=0.0711
- Iteration=2 PivotSample=14/12665 PrototypeCandidate=[3567,9987], NumFeatures=64/165, Error=0.0388
- Iteration=2 PivotSample=16/12665 PrototypeCandidate=[4400,9719], NumFeatures=88/165, Error=0.0126
- Iteration=2 PivotSample=124/12665 PrototypeCandidate=[3943,6696], NumFeatures=91/165, Error=0.0066
- Iteration=2 PivotSample=6228/12665 PrototypeCandidate=[6065,1711], NumFeatures=107/165, Error=0.0066
- Iteration=2 PivotSample=7297/12665 PrototypeCandidate=[6447,5814], NumFeatures=88/165, Error=0.0063
- Iteration=2 PivotSample=8095/12665 PrototypeCandidate=[6153,2611], NumFeatures=76/165, **Error=0.0061**

Flexibility principle: support vectors are now changed! They are sparser!

MNIST Dataset (0 vs. 1) – Iteration 3

- Iteration=3 PivotSample=1/12665 PrototypeCandidate=[1257,8183], NumFeatures=54/76, Error=0.0695
- Iteration=3 PivotSample=17/12665 PrototypeCandidate=[4740,9332], NumFeatures=58/76, Error=0.0471
- Iteration=3 PivotSample=27/12665 PrototypeCandidate=[2547,7622], NumFeatures=41/76, Error=0.0385
- Iteration=3 PivotSample=31/12665 PrototypeCandidate=[3388,7931], NumFeatures=28/76, Error=0.0165
- Iteration=3 PivotSample=345/12665 PrototypeCandidate=[5305,7160], NumFeatures=47/76, Error=0.0099
- Iteration=3 PivotSample=779/12665 PrototypeCandidate=[5053,7160], NumFeatures=53/76, Error=0.0099
- Iteration=3 PivotSample=943/12665 PrototypeCandidate=[1627,9332], NumFeatures=41/76, Error=0.0087
- Iteration=3 PivotSample=1203/12665 PrototypeCandidate=[3622,7996], NumFeatures=31/76, **Error=0.0053**

MNIST Dataset (0 vs. 1) – Iteration 4

- Iteration=4 PivotSample=1/12665 PrototypeCandidate=[2068,9477], NumFeatures=17/31, Error=0.1368
- Iteration=4 PivotSample=2/12665 PrototypeCandidate=[5840,9477], NumFeatures=15/31, Error=0.1268
- Iteration=4 PivotSample=3/12665 PrototypeCandidate=[3317,9287], NumFeatures=21/31, Error=0.0582
- Iteration=4 PivotSample=5/12665 PrototypeCandidate=[5698,8702], NumFeatures=26/31, Error=0.0511
- Iteration=4 PivotSample=8/12665 PrototypeCandidate=[2419,10601], NumFeatures=12/31, Error=0.0250
- Iteration=4 PivotSample=12/12665 PrototypeCandidate=[735,9308], NumFeatures=21/31, Error=0.0207
- Iteration=4 PivotSample=26/12665 PrototypeCandidate=[2867,10238], NumFeatures=14/31, Error=0.0115
- Iteration=4 PivotSample=61/12665 PrototypeCandidate=[571,11298], NumFeatures=22/31, Error=0.0073
- Iteration=4 PivotSample=989/12665 PrototypeCandidate=[4792,12441], NumFeatures=20/31, Error=0.0071
- Iteration=4 PivotSample=2860/12665 PrototypeCandidate=[3951,7869], NumFeatures=16/31, Error=0.0060
- Iteration=4 PivotSample=3185/12665 PrototypeCandidate=[5428,8945], NumFeatures=18/31, Error=0.0057
- Iteration=4 PivotSample=6783/12665 PrototypeCandidate=[10423,4083], NumFeatures=24/31, **Error=0.0050**

MNIST Dataset (0 vs. 1) – Iteration 5

- Iteration=5 PivotSample=1/12665 PrototypeCandidate=[5836,11670], NumFeatures=15/24, Error=0.1375
- Iteration=5 PivotSample=2/12665 PrototypeCandidate=[2259,9376], NumFeatures=13/24, Error=0.0747
- Iteration=5 PivotSample=4/12665 PrototypeCandidate=[3753,10240], NumFeatures=20/24, Error=0.0168
- Iteration=5 PivotSample=65/12665 PrototypeCandidate=[119,10240], NumFeatures=15/24, Error=0.0160
- Iteration=5 PivotSample=66/12665 PrototypeCandidate=[4880,10996], NumFeatures=16/24, Error=0.0114
- Iteration=5 PivotSample=305/12665 PrototypeCandidate=[4219,10240], NumFeatures=18/24, Error=0.0114
- Iteration=5 PivotSample=605/12665 PrototypeCandidate=[5486,8936], NumFeatures=16/24, Error=0.0107
- Iteration=5 PivotSample=749/12665 PrototypeCandidate=[2746,6051], NumFeatures=10/24, Error=0.0066
- Iteration=5 PivotSample=2746/12665 PrototypeCandidate=[749,6051], NumFeatures=11/24, Error=0.0060
- Iteration=5 PivotSample=4083/12665 PrototypeCandidate=[3709,12086], NumFeatures=15/24, **Error=0.0051**

MNIST Dataset (0 vs. 1) – Iteration 6

- Iteration=7 PivotSample=2/12665 PrototypeCandidate=[1686,7921], NumFeatures=3/10, Error=0.1171
- Iteration=6 PivotSample=1/12665 PrototypeCandidate=[5836,7622], NumFeatures=12/15, Error=0.1248
- Iteration=6 PivotSample=4/12665 PrototypeCandidate=[5751,10240], NumFeatures=13/15, Error=0.0141
- Iteration=6 PivotSample=13/12665 PrototypeCandidate=[4381,10240], NumFeatures=13/15, Error=0.0140
- Iteration=6 PivotSample=35/12665 PrototypeCandidate=[44,10240], NumFeatures=13/15, Error=0.0139
- Iteration=6 PivotSample=73/12665 PrototypeCandidate=[1694,10240], NumFeatures=13/15, Error=0.0137
- Iteration=6 PivotSample=92/12665 PrototypeCandidate=[3714,6627], NumFeatures=13/15, Error=0.0107
- Iteration=6 PivotSample=209/12665 PrototypeCandidate=[4690,7628], NumFeatures=6/15, Error=0.0098
- Iteration=6 PivotSample=1269/12665 PrototypeCandidate=[702,6121], NumFeatures=7/15, Error=0.0077
- Iteration=6 PivotSample=6209/12665 PrototypeCandidate=[9503,1609], NumFeatures=10/15, **Error=0.0069**

MNIST Dataset (0 vs. 1) – Iteration 7

- Iteration=7 PivotSample=6/12665 PrototypeCandidate=[962,7250], NumFeatures=6/10, Error=0.0325
- Iteration=7 PivotSample=14/12665 PrototypeCandidate=[1219,7555], NumFeatures=5/10, Error=0.0308
- Iteration=7 PivotSample=249/12665 PrototypeCandidate=[2327,6577], NumFeatures=3/10, Error=0.0156
- Iteration=7 PivotSample=337/12665 PrototypeCandidate=[1950,6577], NumFeatures=3/10, Error=0.0090
- Iteration=7 PivotSample=4796/12665 PrototypeCandidate=[4300,9531], NumFeatures=2/10, Error=0.0078
- Iteration=7 PivotSample=5883/12665 PrototypeCandidate=[4482,6577], NumFeatures=4/10, Error=0.0069
- Iteration=7 PivotSample=5995/12665 PrototypeCandidate=[10327,1609], NumFeatures=10/10, Error=0.0063
- Iteration=7 PivotSample=8335/12665 PrototypeCandidate=[9625,1609], NumFeatures=8/10, **Error=0.0061**

MNIST Dataset (0 vs. 1) – Iteration 8

- Iteration=8 PivotSample=1/12665 PrototypeCandidate=[3516,7412], NumFeatures=6/8, Error=0.1189
- Iteration=8 PivotSample=3/12665 PrototypeCandidate=[2655,5985], NumFeatures=3/8, Error=0.1169
- Iteration=8 PivotSample=6/12665 PrototypeCandidate=[962,6506], NumFeatures=7/8, Error=0.0549
- Iteration=8 PivotSample=8/12665 PrototypeCandidate=[4988,7555], NumFeatures=2/8, Error=0.0493
- Iteration=8 PivotSample=11/12665 PrototypeCandidate=[4852,7250], NumFeatures=5/8, Error=0.0264
- Iteration=8 PivotSample=38/12665 PrototypeCandidate=[2700,6506], NumFeatures=7/8, Error=0.0242
- Iteration=8 PivotSample=123/12665 PrototypeCandidate=[1809,6506], NumFeatures=6/8, Error=0.0235
- Iteration=8 PivotSample=275/12665 PrototypeCandidate=[357,7572], NumFeatures=4/8, Error=0.0159
- Iteration=8 PivotSample=1152/12665 PrototypeCandidate=[1899,7818], NumFeatures=2/8, Error=0.0120
- Iteration=8 PivotSample=1910/12665 PrototypeCandidate=[2422,11150], NumFeatures=4/8, Error=0.0107
- Iteration=8 PivotSample=3354/12665 PrototypeCandidate=[3982,8105], NumFeatures=7/8, **Error=0.0069**

MNIST Dataset (0 vs. 1) – Iteration 9

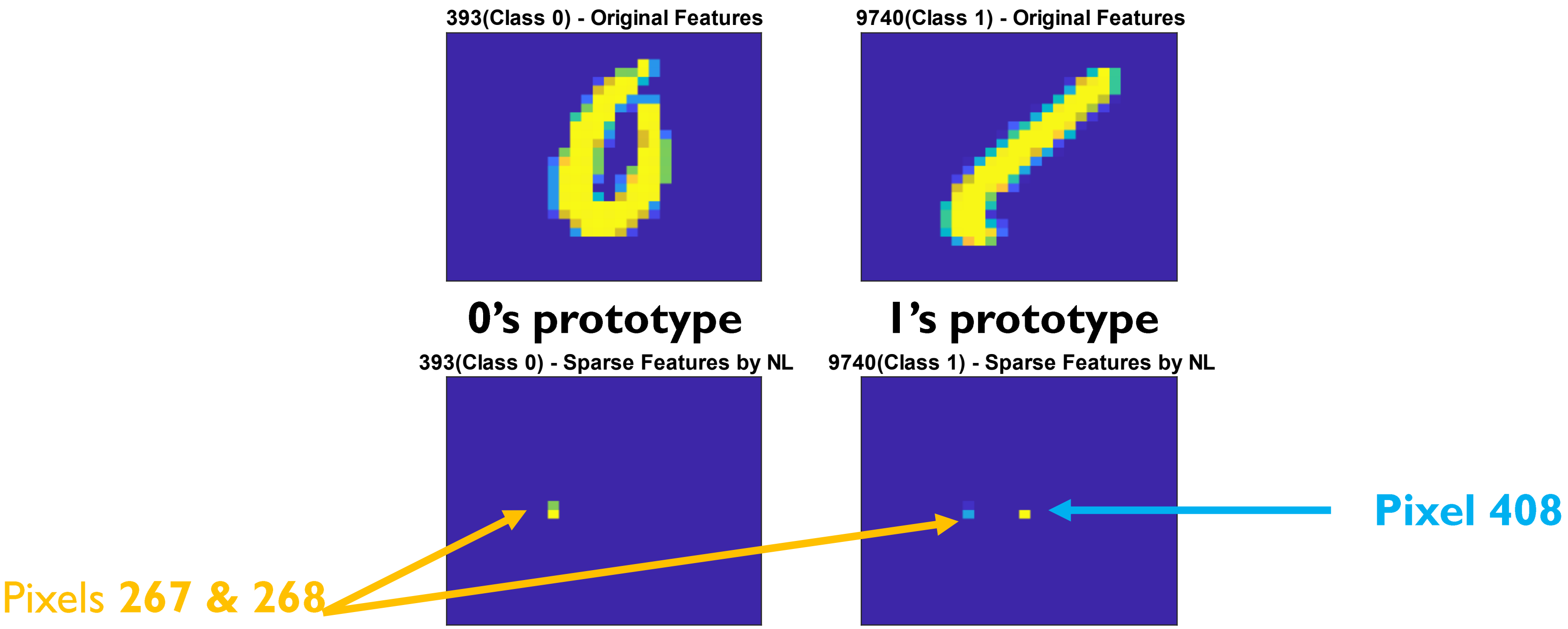
- Iteration=9 PivotSample=1/12665 PrototypeCandidate=[3516,8387], NumFeatures=4/7, Error=0.1526
- Iteration=9 PivotSample=3/12665 PrototypeCandidate=[2655,8387], NumFeatures=3/7, Error=0.1225
- Iteration=9 PivotSample=6/12665 PrototypeCandidate=[962,7412], NumFeatures=3/7, Error=0.1077
- Iteration=9 PivotSample=8/12665 PrototypeCandidate=[3911,12101], NumFeatures=2/7, Error=0.0452
- Iteration=9 PivotSample=11/12665 PrototypeCandidate=[1201,7412], NumFeatures=4/7, Error=0.0449
- Iteration=9 PivotSample=22/12665 PrototypeCandidate=[2105,7555], NumFeatures=3/7, Error=0.0292
- Iteration=9 PivotSample=173/12665 PrototypeCandidate=[3694,9066], NumFeatures=3/7, Error=0.0098
- Iteration=9 PivotSample=473/12665 PrototypeCandidate=[5549,9066], NumFeatures=3/7, Error=0.0094
- Iteration=9 PivotSample=739/12665 PrototypeCandidate=[5656,9066], NumFeatures=3/7, **Error=0.0069**

MNIST Dataset (0 vs. 1) – Iteration 10

- Iteration=10 PivotSample=1/12665 PrototypeCandidate=[224,7622], NumFeatures=2/3, Error=0.7712
- Iteration=10 PivotSample=4/12665 PrototypeCandidate=[38,11310], NumFeatures=2/3, Error=0.5754
- Iteration=10 PivotSample=9/12665 PrototypeCandidate=[10,11310], NumFeatures=2/3, Error=0.5677
- Iteration=10 PivotSample=11/12665 PrototypeCandidate=[2349,7412], NumFeatures=2/3, Error=0.2671
- Iteration=10 PivotSample=12/12665 PrototypeCandidate=[2888,11614], NumFeatures=2/3, Error=0.1240
- Iteration=10 PivotSample=24/12665 PrototypeCandidate=[4712,11859], NumFeatures=3/3, Error=0.0827
- Iteration=10 PivotSample=53/12665 PrototypeCandidate=[5701,9740], NumFeatures=3/3, Error=0.0102
- Iteration=10 PivotSample=103/12665 PrototypeCandidate=[393,9740], NumFeatures=3/3, Error=**0.0067**

- Best Prototype=[Sample **393**(class 0), Sample **9740**(class 1)], Best Error=0.0067, **Core Features=[267 268 408]**

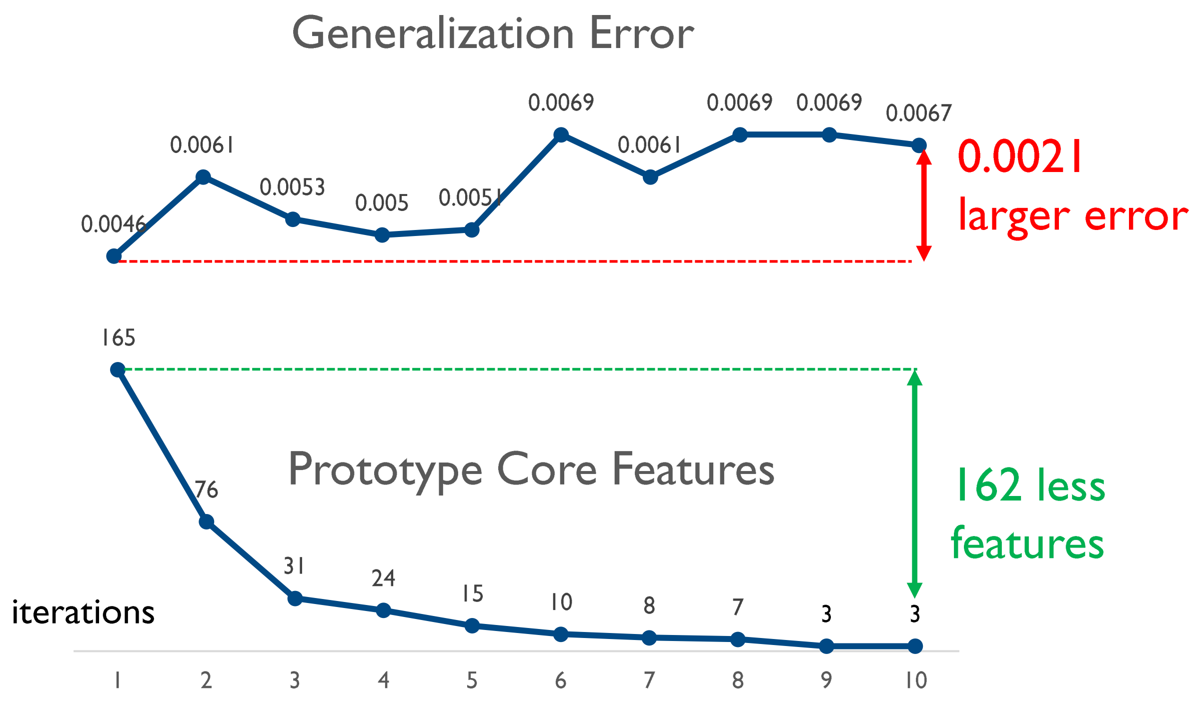
Visualization of Sparse Prototypes found by Natural Learning



If test sample's pixels 267,268, 408, collectively make test example closer to **0's prototype** than **1's prototype**, it is **0,** otherwise it is 1

Accuracy on Train : 99.33%
Accuracy on Test: **99.48%**

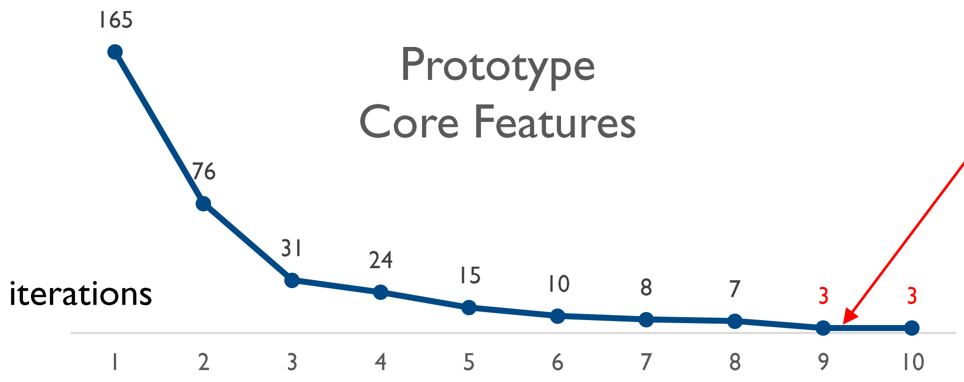
MNIST Dataset (0 vs. 1) – Summary



The key to have **extreme sparsity** is to **sacrifice small accuracy**

natural-learning.cc

MNIST Dataset (0 vs. 1) – Summary



No more features could be pruned → Stop

No hyperparameter is required for the **number of iterations**

Comparison of Properties with other classifiers

Model	Local rules?	Estimate Weight?	Hyperparameters?	Memorize Train set?
Nearest Neighbor (1NN)	No	No	No	Yes
Deep Neural Networks (DNN)	No	Yes	Yes	No
Random Forest (RF)	Yes	No	Yes	No
Decision Trees (DT)	Yes	No	Yes/No	No
Logistic Regression (LR)	No	Yes	No	No
Linear discriminant Analysis (LDA)	No	Yes	No	No
Support Vector Machines (SVM)	No	Yes	Yes	No
Natural Learning (NL)	**No**	**No**	**No**	**Only 2 Samples***

* For binary classification

Connection of NL with other classifiers

- Special case of **Nearest Neighbor Classifier**
  - NL=**Nearest Neighbor Classifier on the compressed training set with size of** s x c
    - s=number of classes (s=2 in binary classification)
    - c=dimension of core features (c<<p)
- Special case of **Support Vector Machines**
  - NL = **Sparse Singular Support Vector Machines** (Hyperparameter-free, with fuzzy boundary)
- Special version of **Decision Trees**
  - Finds a **single multi-attribute rule**
  - e.g., If the test sample's features F1, F25, and F100 are closer to [0.12, 0.26, 0.27] comparing [0.26,0.28, 0.29], it is labeled 1, otherwise 0.
- It shares characteristics with **Linear Discriminant Analysis (LDA)** and **Deep Learning**: simultaneously performs dimension reduction and classification.
  - NL : **Original Space**
  - LDA: Linear Latent Space
  - Deep Learning: Non-linear Latent Space

Superior Compression = Greater Intelligence

Compression Represents Intelligence Linearly

Yuzhen Huang*[1] Jinghan Zhang*[1] Zifei Shan[2] Junxian He[1]
[1]The Hong Kong University of Science and Technology [2]Tencent
{yhuanghj,jzhangjv,junxianh}@cse.ust.hk

Abstract

There is a belief that learning to compress well will lead to intelligence (Hutter, 2006). Recently, language modeling has been shown to be equivalent to compression, which offers a compelling rationale for the success of large language models (LLMs): the development of more advanced language models is essentially enhancing compression which facilitates intelligence. Despite such appealing discussions, little empirical evidence is present for the interplay between compression and intelligence. In this work, we examine their relationship in the context of LLMs, treating LLMs as data compressors. Given the abstract concept of "intelligence", we adopt the average downstream benchmark scores as a surrogate, specifically targeting intelligence related to knowledge and commonsense, coding, and mathematical reasoning. Across 12 benchmarks, our study brings together 30 public LLMs that originate from diverse organizations. Remarkably, we find that LLMs' intelligence – reflected by average benchmark scores – almost *linearly* correlates with their ability to compress external text corpora. These results provide concrete evidence supporting the belief that superior compression indicates greater intelligence. Furthermore, our findings suggest that compression efficiency, as an unsupervised metric derived from raw text corpora, serves as a reliable evaluation measure that is linearly associated with the model capabilities. We open-source our compression datasets as well as our data collection pipelines to facilitate future researchers to assess compression properly.[1]

Huang, Yuzhen, et al. "Compression Represents Intelligence Linearly." arXiv preprint arXiv:2404.09937 (2024).

Experimental Evaluation: Datasets

- 17 benchmark datasets for binary classification from the healthcare domain where NL's strength is supposed to be at the level of black-box models due to noisy labels in this domain (Semenova et al. , 2023)
  - 9 high-dimensional datasets (n<<p)
  - 8 low-dimensional datasets (n>>p)
- 10 Stratified sampling for each dataset (10-fold) to reduce the bias of train/test split
  - 170 train/test set in total

High-Dimensional (Gene Expression) Datasets (N<<P)						Low-Dimensional Datasets (N>>P)					
Dataset	#p	#n	MjClass	ID*	Description	Dataset	#p	#n	MjClass	ID*	Description
AP_Breast_Colon	10935	630	54.60%	1145	Breast vs. Colon Cancer	blood-transfusion	4	748	76.20%	1464	Donor of **Blood Transfusion** (UCI)
AP_Breast_Kidney	10935	604	56.95%	1158	Breast vs. Kidney Cancer	diabetes	8	768	65.10%	42608	**Diabetes** Patient (OpenML)
AP_Breast_Ovary	10935	542	63.47%	1165	breast vs. Ovarian Cancer	Haberman	14	306	73.53%	43	**Breast Cancer** Survival (UCI)
AP_Colon_Kidney	10935	546	52.38%	1137	Colon vs. Kidney Cancer	heart-statlog	13	270	55.56%	53	**Heart Disease** Database (UCI)
OVA_Colon	10935	1545	81.49%	1161	Colon Cancer vs. others	hiva_agnostic	1617	4229	96.48%	1039	**AIDS HIV** infection (ETH Zurich)
OVA_Kidney	10935	1545	83.17%	1134	Kidney Cancer vs. others	ilpd-numeric	10	583	71.36%	41945	Indian **Liver** Patient Dataset (UCI)
OVA_Lung	10935	1545	91.84%	1130	Lung Cancer vs. others	thoracic-surgery	37	470	85.11%	4329	**Lung Cancer** life expectancy (UCI)
OVA_Omentum	10935	1545	95.02%	1139	Omentum Cancer vs. others	wdbc	30	569	62.74%	1510	**Breast Cancer** Wisconsin (UCI)
OVA_Ovary	10935	1545	87.18%	1166	Ovarian Cancer vs. others	* OpenML dataset identifier					

Semenova, et al. "A Path to Simpler Models Starts With Noise.", NeurIPS 2023

Finetuning baseline models

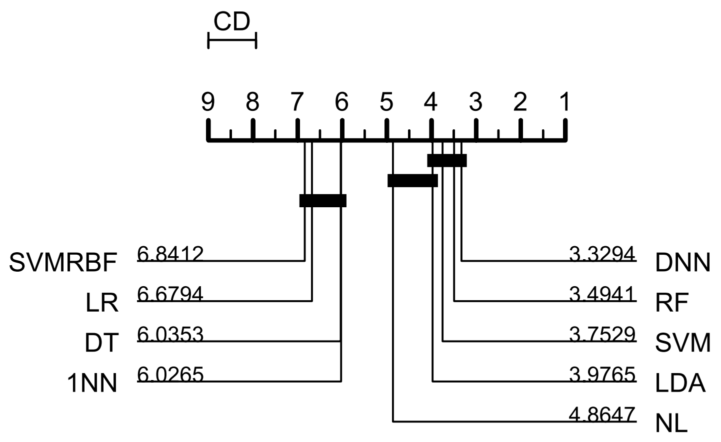
We compare NL versus finetuned baseline models to have a fair comparison. We get the practical configuration settings from applied machine learning sources [1] and [2] for a realistic comparison.

Classifier	Hyperparameter Search	Tested Combinations
Decision Trees	MaxSplits=[1, 5, 10, 20, 50, n], MinLeafSize=[1, 5, 10, 20, 50]	30
Linear SVM	C=[100, 10, 1.0, 0.1, 0.001]	5
SVM-RBF	C=[100, 10, 1.0, 0.1, 0.001], gamma=[2^{-16} … 2^{8}] as suggested by [2] with step of 2^{2}	65
Random Forests (RF)	MaxSplits =[1, 5, 10, 20, 50, n], MinLeafSize=[1, 5, 10, 20, 50], NumTrees= [10,50,100]	90
Deep Neural Networks (DNN)	Batch size=32, Optimizer=Stochastic gradient descent, max epoch of 20, Hidden Layers=[10, 30, 50], Layers=[2, 3, 4], Learning Rate=[0.01, 0.001] and Activation Functions={RelU, Tanh, Sigmoid}	54
Latent Discriminant Analysis (LDA)	Hyperparameter-free	1
Logistic Regression (LR)	Hyperparameter-free	1
Natural Learning (NL)	Hyperparameter-free	1

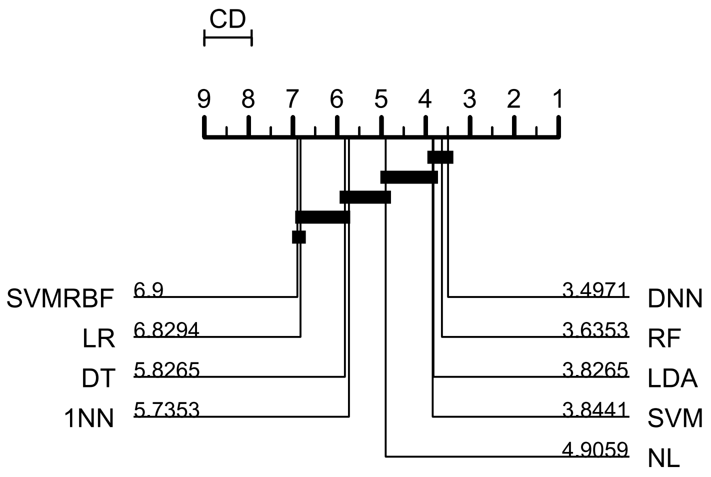
[1] https://machinelearningmastery.com/
[2] Fernández-Delgado, Manuel, et al. "Do we need hundreds of classifiers to solve real world classification problems?." The journal of machine learning research 15.1 (2014): 3133-3181.

Results: Accuracy and F-measure, Winning Ratio

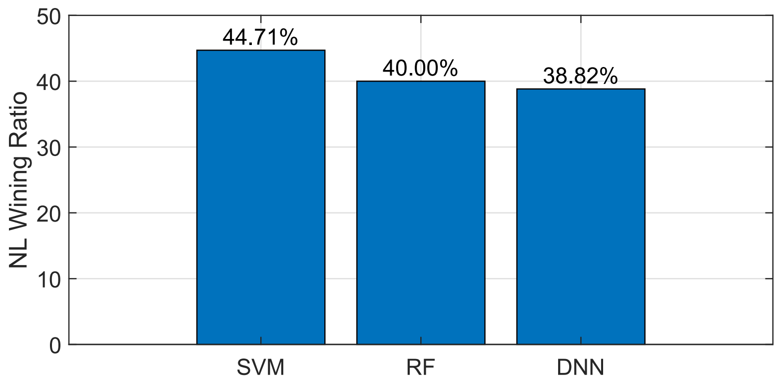
* Critical Difference Diagram, Horizonal line indicates lack of statistical significance at alpha = 0.01 (Nemenyi's test)



Accuracy



F-measure



Winning Ratio(Accuracy)

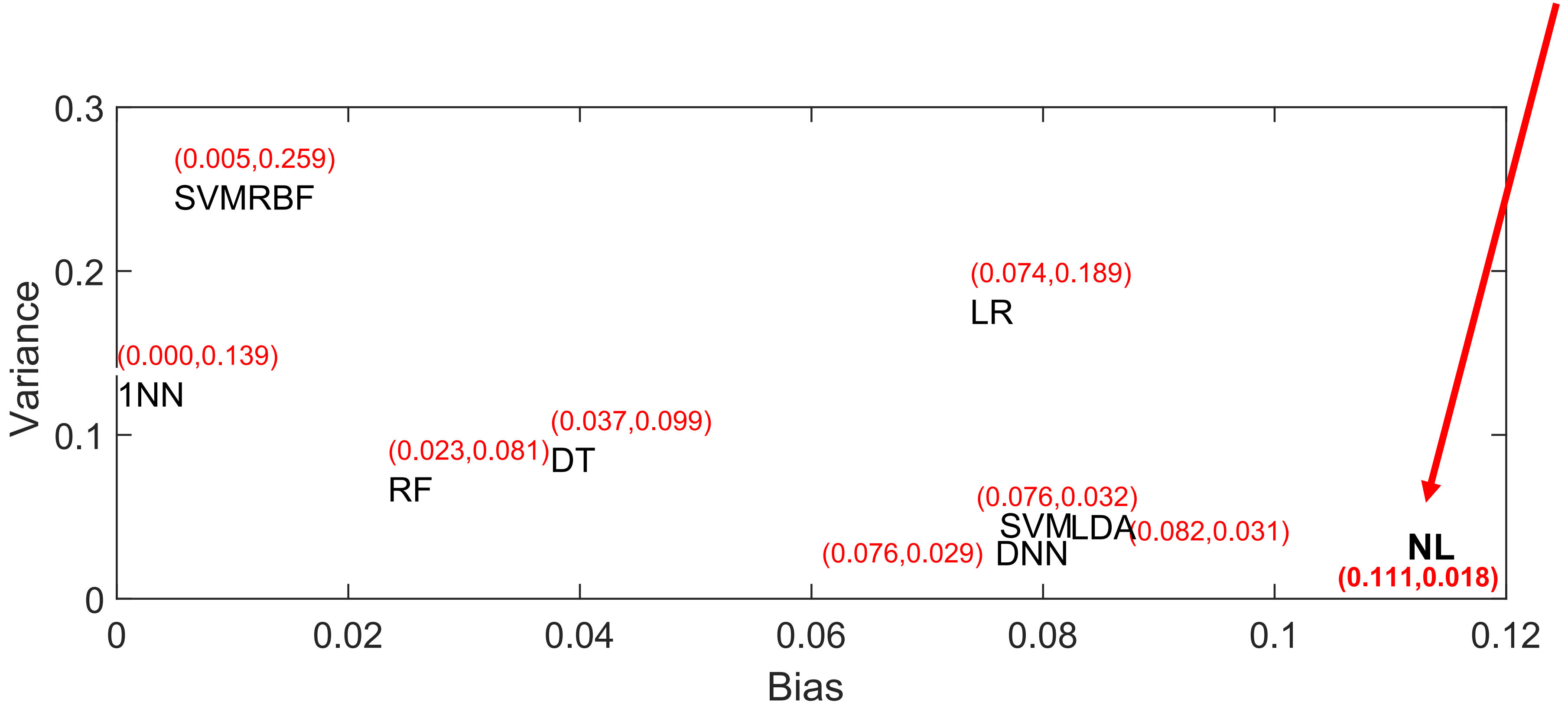
Considering **simplicity** and **extreme sparsity level** of NL comparing black box models, this is an impressive result

natural-learning.cc

* Demšar, Janez. "Statistical comparisons of classifiers over multiple data sets." The Journal of Machine Learning Research 7 (2006): 1-30.

Results: Average Bias-Variance

This **extraordinarily low variance** can be related to the simplicity of the model which results in **larger Rashomon ratio** [1] due to existence of noisy labels [2]



We observed several performance cases where test accuracy was considerably higher than train accuracy.

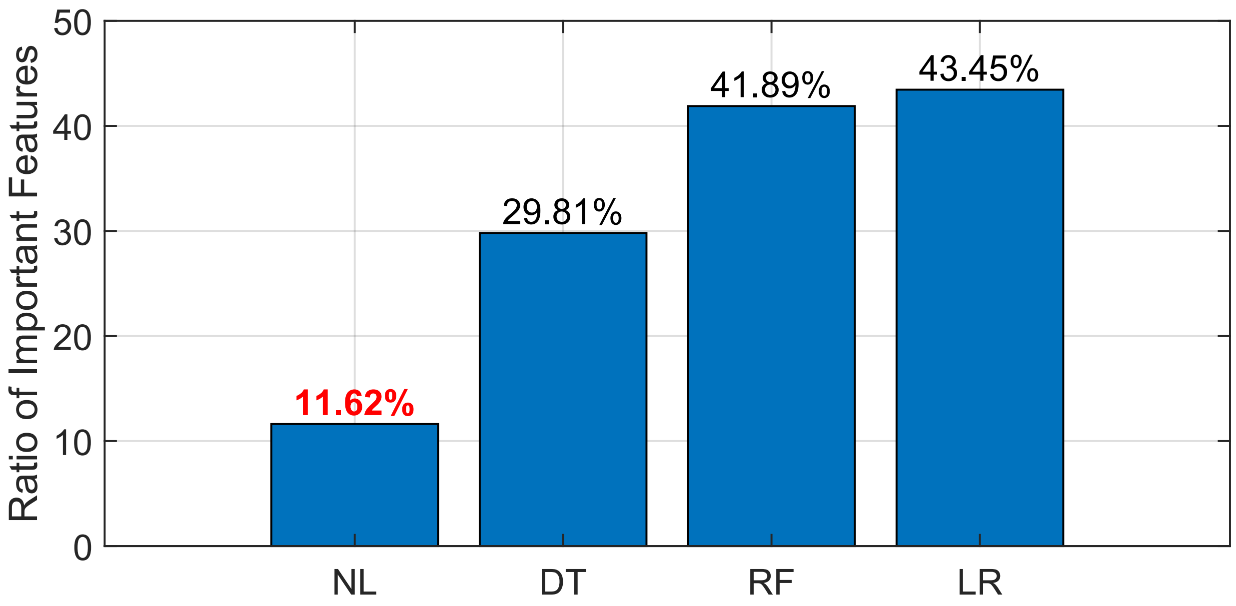
In **humans**, a study revealed that in certain situations, previously unseen prototypes might be classified more accurately during the testing phase than the original training stimuli [3].

[1] Breiman, Leo. "Statistical modeling: The two cultures (with comments and a rejoinder by the author)." Statistical science 16.3 (2001): 199-231.
[2] Semenova, et al. "A Path to Simpler Models Starts With Noise.“, NeurIPS 2023
[3] David R. Shanks, Concept Learning and Representation: Models in Smelser, Neil J., and Paul B. Baltes, eds. International encyclopedia of the social & behavioral sciences. Vol. 11. Amsterdam: Elsevier, 2015.

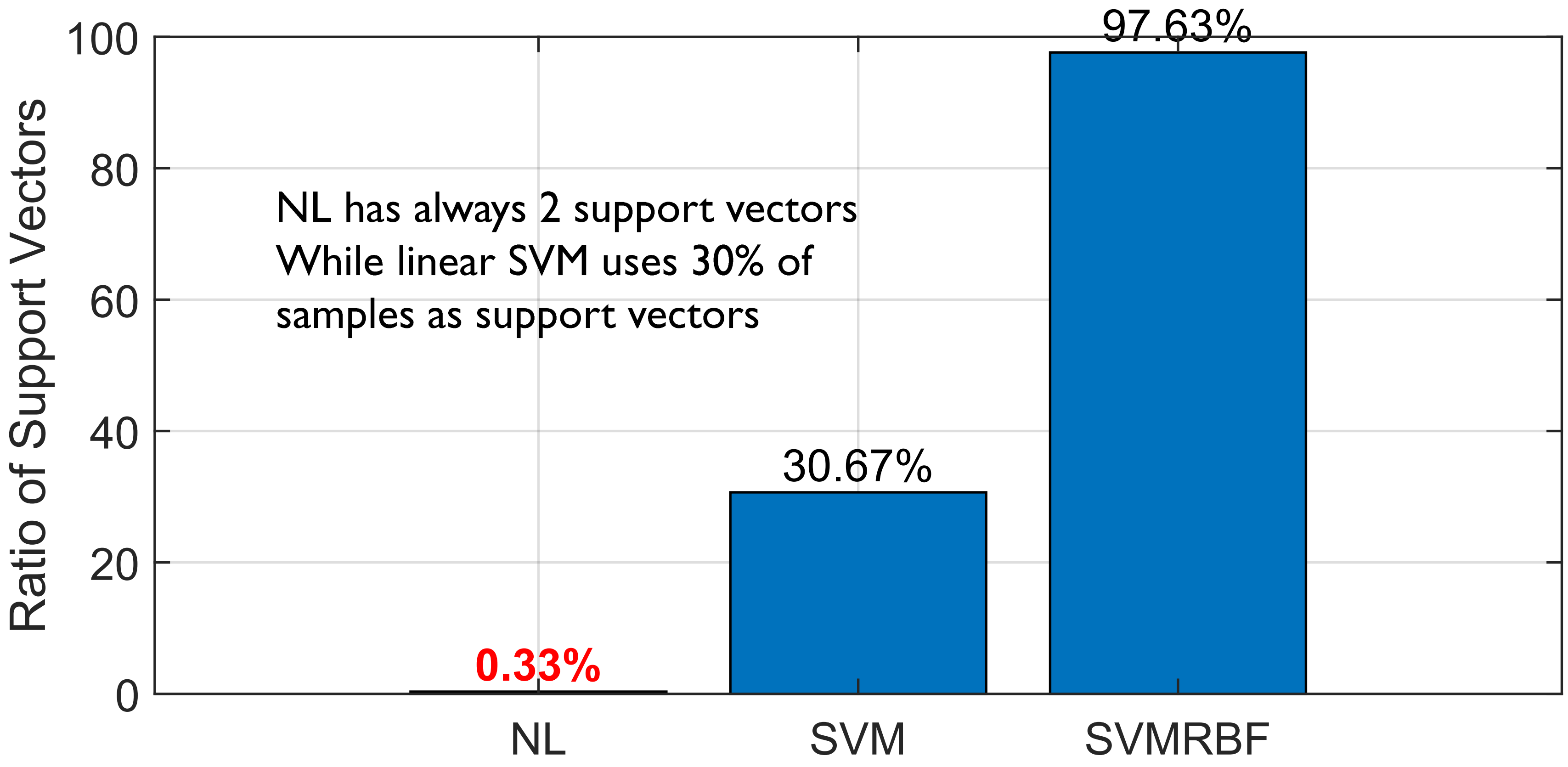
Results: Interpretability

As a quantitative metric, we compare the ratio of important features. But this **does not reflect the real interpretability value of NL**

- NL finds a meaningful subset of features with equal weights for each future
- Makes the interpretability even better than DT and LR

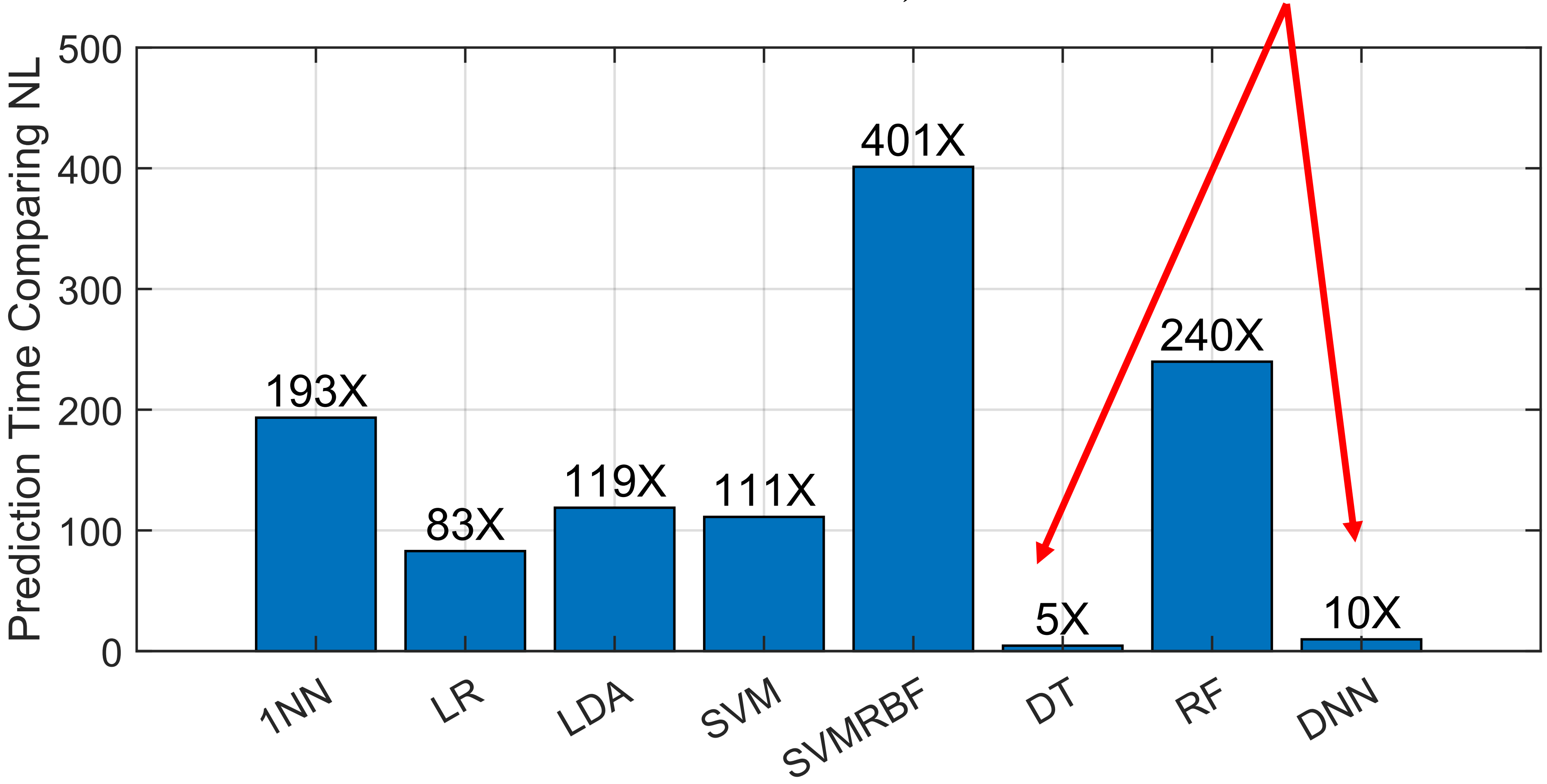


Results: Ratio of Support Vectors



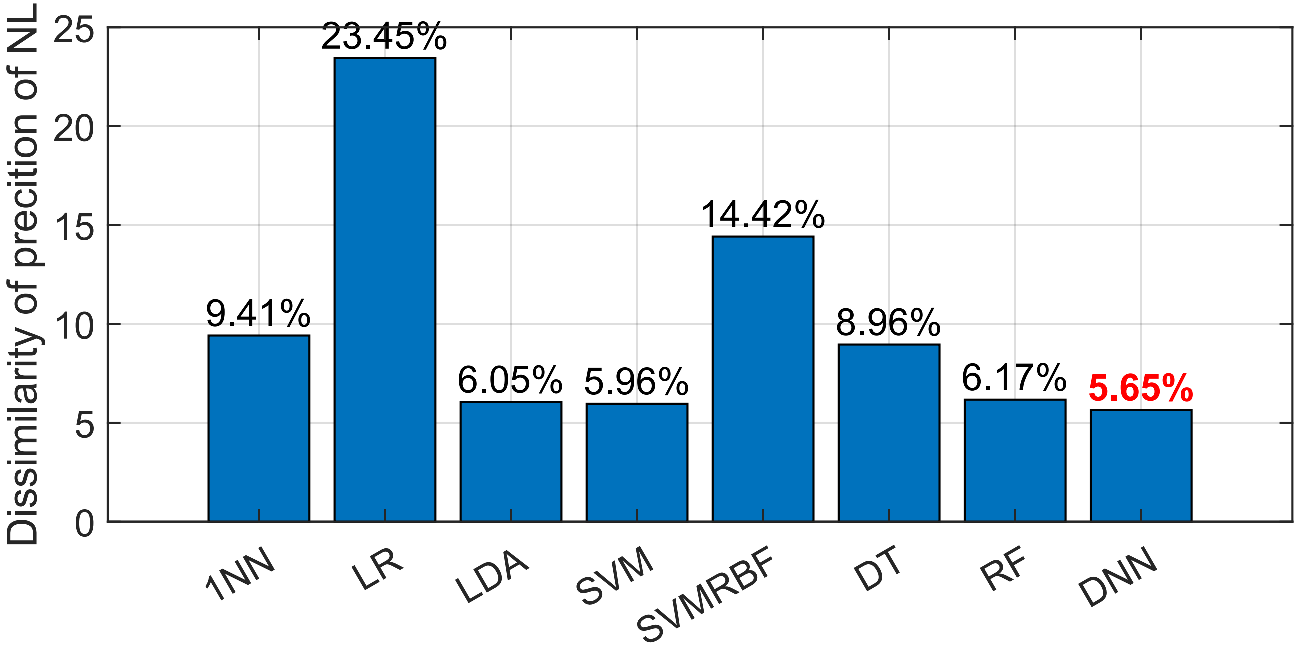
Results: Prediction Runtime

NL advances the **state-of-the-art prediction speed**, 5x of decision trees, and 10x of DNNs.



Prediction Dissimilarity to Other Classifiers

We compared the similarity of NL predictions to other classifiers based on 160k predictions they made on 170 training sets. Deep Neural Networks were found to be the best match with NL in terms of behavior on predictions, with a 5.65% mismatch in their predictions! The less similar classifier was logistic regression.



Example of NL Models: Colon Cancer Gene Expression

Dataset:**OVA_Colon (**1545 patients X 10935 genes)

Natural Learning

6 Minutes

Class 0 Prototype(No Colon Cancer)

Class 1 Prototype (Colon Cancer)

Prototype Patient Index

Prototype's Features (genes)

	x206418_at	**x206430_at**	**x210302_s_at**
SN0(1008)	95.3	498.6	148.2
SN1(155)	639.6	736.9	580.7

Learned Prototype=(2 patients X 3 genes)

Model Sparsity Ratio = **3.55 X 10^{-7}**

Accuracy on Test Set

NL	**98.05**
DNN	97.40
RF	98.05
DT	98.05
SVM	96.75

Examples of Discovered Prototypes by NL and performance

Dataset: OVA_Colon, n-fold=4/10 (seed=42)

Learned Prototype on the Train set (90%)

	x206418_at	x206430_at	x210302_s_at
Class 0 (S#1008)	95.3	498.6	148.2
Class 1 (S#155)	639.6	736.9	580.7

Accuracy on Test Set (10%)

NL	DNN	RF	DT	SVM
98.05	97.40	98.05	98.05	96.75

Dataset: OVA_Ovary, n-fold=9/10 (seed=42)

Learned Prototype on the Train set (90%)

	x1559477_s_at	x219873_at
Class 0 (S# 879)	461.3	1131.1
Class 1 (S# 1378)	606.1	1635.2

Accuracy on Test Set (10%)

NL	DNN	RF	DT	SVM
94.19	91.61	93.55	93.55	91.61

Dataset: OVA_Omentum, n-fold=9/10 (seed=42)

Learned Prototype on the Train set (90%)

	x206067_s_at	x37892_at
Class 0(S# 1419)	8314.6	14771.9
Class 1(S# 304)	11813.7	17220.4

Accuracy on Test Set (10%)

NL	DNN	RF	DT	SVM
94.19	92.26	94.19	94.19	93.55

Dataset: blood-transfusion, n-fold=3/10 (seed=42)

Learned Prototype on the Train set (90%)

	v1	v2	v3
Class 0(S# 678)	23	19	4750
Class 1(S# 45)	4	20	5000

Accuracy on Test Set (10%)

NL	DNN	RF	DT	SVM
80.00	77.33	76.00	76.00	80.00

Dataset: diabetes , n-fold=7/10 (seed=42)

Learned Prototype on the Train set (90%)

	plas	mass
Class 0(S#260)	155	33.3
Class 1(S#670)	154	30.9

Accuracy on Test Set (10%)

NL	DNN	RF	DT	SVM
75.00	72.37	75.00	75.00	71.05

Dataset: HIVa_agnostic , n-fold=1/10 (seed=42)

Learned Prototype on the Train set (90%)

	attr82	attr166	attr1048	attr1324
Class 0 (S# 1542)	0	0	0	0
Class 1 (S# 429)	1	1	1	1

Accuracy on Test Set (10%)

NL	DNN	RF	DT	SVM
97.40	96.45	97.16	96.93	93.14

Examples of Discovered Prototypes by NL and performance

Dataset: **ilpd-numeric,** n-fold=3/10 (seed=42)

Learned Prototype on the Train set (90%)

	v1	v6	v9	v10
Class 0(S#7)	26	16	3.5	1
Class 1(S#531)	22	14	3.8	1.1

Accuracy on Test Set (10%)

NL	DNN	RF	DT	SVM
70.69	67.24	67.24	67.24	67.24

Dataset: **thoracic-surgery**, n-fold=2/10 (seed=42)

Learned Prototype on the Train set (90%)

	v4_2	v4_3	v7_1	v7_2	v4_2
Class 1(S#51)	1	0	1	0	1
Class 0 (S#129)	0	1	0	1	0

Accuracy on Test Set (10%)

NL	DNN	RF	**DT**	SVM
89.36	89.36	87.23	**65.96**	89.36

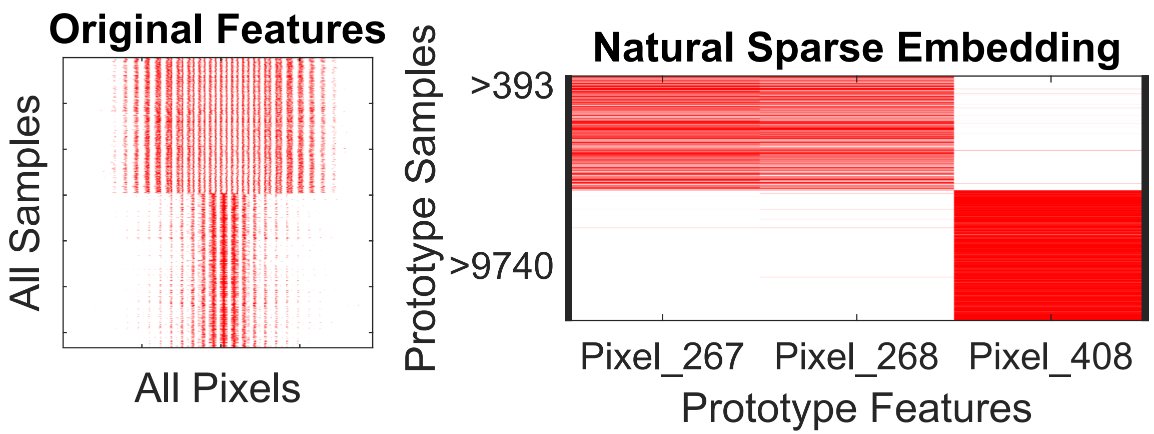
Dataset: **wdbc**, n-fold=4/10 (seed=42)

Learned Prototype on the Train set (90%)

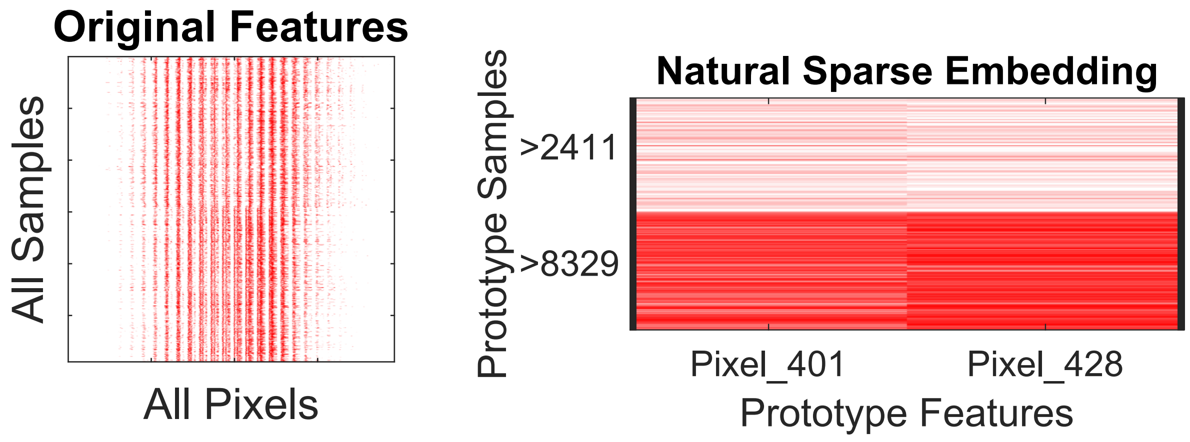
	v2	v11	v13	v18	v22	v23	v25	v28
Class 0 (S#348)	14.74	0.3428	2.537	0.01067	17.93	114.2	0.122	0.1251
Class 1 (S#206)	16.68	0.2711	1.974	0.00826	20.24	117.7	0.149	0.1252

Accuracy on Test Set (10%)

NL	DNN	RF	DT	SVM
98.25	98.25	94.74	94.74	98.25



MNIST **0 vs. 1** (The easiest) – Test Accuracy: 99.48%



MNIST **4 vs. 9** (The most difficult) – Test Accuracy: 85.64%

natural-learning.cc

All datasets

Dataset (Kfold)	TrainAcc	Iterations	Prototype Samples	Prototype Core Features
AP_Breast_Colon(kfold=1)	95.94	19	502 &348	2726, 4765, 6647, 7000,10203
AP_Breast_Colon(kfold=2)	96.83	18	531 &489	2726, 3268, 5519, 5681, 7222, 8866, 9979,10593
AP_Breast_Colon(kfold=3)	96.30	19	279 &454	1807,2958,4465,4576,6111,7034,7251,7829,8281,9265
AP_Breast_Colon(kfold=4)	96.83	17	242 &500	2566,4424,4451,7204,7448
AP_Breast_Colon(kfold=5)	96.47	18	309 & 39	51,2532,2674,4765,5716,6907,7439,8229
AP_Breast_Colon(kfold=6)	95.59	22	193 &500	2532,7845
AP_Breast_Colon(kfold=7)	96.47	20	5 &285	2532,4204,4215
AP_Breast_Colon(kfold=8)	95.94	18	284 &305	459, 2532, 3382, 4215, 6923, 7083, 7845, 9591,10750
AP_Breast_Colon(kfold=9)	93.12	19	190 &373	2726,9961
AP_Breast_Colon(kfold=10)	96.12	19	358 & 95	2097,2958,3268,4501,6111
AP_Breast_Kidney(kfold=1)	96.14	20	349 &143	401, 524,4333,4592,4682,7742
AP_Breast_Kidney(kfold=2)	98.34	23	139 &159	14, 3536, 4504, 6064, 7247, 7548, 7808, 9950,10750
AP_Breast_Kidney(kfold=3)	97.42	20	157 & 6	92, 294, 487, 1023, 1615, 2924, 3083, 3088, 3196, 3259, 3324, 3366, 3536, 3830, 4387, 4913, 6418, 7022, 7096, 7152, 7167, 7247, 7619, 8136, 8723, 9519, 9735, 9760,10024,10152,10459,10565,10581,10603,10623,10657
AP_Breast_Kidney(kfold=4)	96.13	26	47 &393	1049, 1050, 2355, 2393, 2924, 3229, 6418,10753
AP_Breast_Kidney(kfold=5)	97.24	30	221 & 43	2109,2768,3196,3536,7022,9165,9750
AP_Breast_Kidney(kfold=6)	97.98	22	345 &162	3196,3324,7247,7578
AP_Breast_Kidney(kfold=7)	96.32	21	47 &492	3196,9736
AP_Breast_Kidney(kfold=8)	96.14	21	252 &160	3052,7742,8219,8749
AP_Breast_Kidney(kfold=9)	97.79	20	196 & 48	2,3316,4900,6723,7061,8136,8217
AP_Breast_Kidney(kfold=10)	97.24	20	411 &316	2, 1096, 1097, 1842, 2768, 3005, 3025, 3039, 3049, 3083, 3195, 3196, 3229, 3324, 3406, 3449, 3521, 3536, 4713, 5076, 5486, 5834, 7089, 7096, 7220, 7247, 7621, 7795, 7808, 7971, 8843, 9166, 9665, 9795, 9980,10039,10331
AP_Breast_Ovary(kfold=1)	94.88	19	158 &179	3296,4330
AP_Breast_Ovary(kfold=2)	93.84	21	118 &385	1725,2146,2537
AP_Breast_Ovary(kfold=3)	96.11	20	328 & 85	4330,9212
AP_Breast_Ovary(kfold=4)	94.05	19	19 &385	1465, 2537, 9212,10501
AP_Breast_Ovary(kfold=5)	97.13	18	449 &272	2, 2643, 2821, 3043, 4198, 4315, 6983, 7832,10082
AP_Breast_Ovary(kfold=6)	96.52	15	154 &138	285, 2896, 4862, 7300, 7416, 8842, 9247, 9457,10667
AP_Breast_Ovary(kfold=7)	94.67	17	307 &140	2643,4536,7741
AP_Breast_Ovary(kfold=8)	93.03	18	61 &441	4330,7592
AP_Breast_Ovary(kfold=9)	95.49	16	223 &334	50,1713,2643,2966,3179,3194,3565,3976,4308,4494,6287,6342,7286,7592
AP_Breast_Ovary(kfold=10)	96.11	20	175 &422	2608, 4536, 7101, 7592, 9212, 9270, 9929,10082

All datasets

Dataset (Kfold)	TrainAcc	Iterations	Prototype Samples	Prototype Core Features
AP_Colon_Kidney(kfold=1)	98.37	28	299 & 28	1853,2266,4133,4480,4609,6825,7111,7759,8237,8896
AP_Colon_Kidney(kfold=2)	97.56	22	313 &426	3541, 7602,10750
AP_Colon_Kidney(kfold=3)	98.98	23	405 &471	4, 151, 863, 913,1076,1151,1359,1448,1508,1855,2078,2916,3208,3598,4490,4832,5599,6009,7914,7915, 7958,8813
AP_Colon_Kidney(kfold=4)	98.37	20	125 &149	1151, 1448, 1460, 1686, 1806, 2918, 3128, 3378, 3602, 4158, 5459, 5986, 6300, 6899, 6948, 7048, 7069, 7111, 7262, 7560, 7814, 7908, 8179, 8813, 8950, 9049, 9428, 9916, 9931,10138,10142,10305,10766,10767,10841,10866
AP_Colon_Kidney(kfold=5)	97.36	22	280 &158	1448, 1460, 2013, 3208, 8724, 9530,10878
AP_Colon_Kidney(kfold=6)	98.17	23	263 &273	1051,1605,1988,2575,2951,3109,4427,4609,4729,5862,5867,8237,8813,8896,9395
AP_Colon_Kidney(kfold=7)	98.37	18	310 &276	76, 1756, 1848, 2708, 2718, 2812, 2836, 2906, 2962, 4141, 4433, 4543, 5053, 5251, 5256, 6135, 6292, 6910, 7187, 7327, 7531, 7966, 8179, 8526, 9318, 9658, 9677, 9745, 9790,10138
AP_Colon_Kidney(kfold=8)	97.96	19	213 &433	484, 1076, 1151, 1625, 1978, 2024, 2245, 2389, 2603, 3094, 3296, 3598, 4112, 4598, 4609, 5583, 6490, 6824, 6825, 7497, 7778, 8179, 8578, 8657, 9318,10171,10580,10841
AP_Colon_Kidney(kfold=9)	98.78	26	161 &283	1151, 1448, 2519, 2697, 3296, 3528, 3567, 3950,10138,10772,10841
AP_Colon_Kidney(kfold=10)	97.96	29	392 &422	1448,1805,2836,3912,4133,5410,5459,5855,7963,8578
OVA_Colon(kfold=1)	95.32	18	1253 &1241	2453,3357
OVA_Colon(kfold=2)	96.33	19	363 &130	2702, 2983, 4562, 6082, 6857, 7661,10045
OVA_Colon(kfold=3)	96.19	22	827 &1346	4755, 6857,10865,10878
OVA_Colon(kfold=4)	96.76	18	142 &910	3352,3357,4562
OVA_Colon(kfold=5)	96.33	17	326 &133	1916,2702,4432,4562,6856,6945,9818
OVA_Colon(kfold=6)	96.19	18	1231 & 567	5205, 6945, 8215,10045
OVA_Colon(kfold=7)	95.83	19	924 &1097	3567,5894,6045,6856,7787
OVA_Colon(kfold=8)	96.69	18	963 &929	4562,5477,7787
OVA_Colon(kfold=9)	95.47	21	1 &1085	3245,3357
OVA_Colon(kfold=10)	96.48	17	793 &1106	2822,4325,6857,6945,7787,8215
OVA_Kidney(kfold=1)	97.41	25	1231 & 94	2479, 5921, 7179, 7246, 9989,10078
OVA_Kidney(kfold=2)	97.77	22	422 &1242	3147,10018
OVA_Kidney(kfold=3)	98.13	24	369 &1375	3147,3236,3448,3528,4660,5743,7179,9769
OVA_Kidney(kfold=4)	97.48	25	1375 &1156	1412,2055,3181,9989
OVA_Kidney(kfold=5)	97.27	23	125 &1215	3146,5815,7827,8828,9769,9826
OVA_Kidney(kfold=6)	97.99	23	686 &471	991,2867,3147,6834,8828
OVA_Kidney(kfold=7)	98.35	28	1304 & 137	2867,3528,4514,5180,7113,8100,9198,9769,9955
OVA_Kidney(kfold=8)	97.84	21	395 &903	3147, 3528, 6861, 7802, 9769, 9838,10009,10054,10755
OVA_Kidney(kfold=9)	97.91	25	814 &1275	3147,8135,9198,9989
OVA_Kidney(kfold=10)	97.84	27	408 & 75	3528, 9989,10018

All datasets

Dataset (Kfold)	TrainAcc	Iterations	Prototype Samples	Prototype Core Features
OVA_Lung(kfold=1)	97.84	15	1248 & 500	4420, 5138, 5855, 6991,10796,10832
OVA_Lung(kfold=2)	97.55	21	136 &121	4420,10796
OVA_Lung(kfold=3)	98.06	17	971 &380	2903, 4420, 5855, 6991, 7587, 8112, 8134, 8769, 8821, 9770,10796
OVA_Lung(kfold=4)	97.77	16	564 &590	4420,5875,6991,8315
OVA_Lung(kfold=5)	98.13	12	131 &819	2669, 3010, 3370, 4420, 5930, 6991, 8134,10796,10810
OVA_Lung(kfold=6)	97.77	13	1257 & 278	1992, 4420, 9599, 9797,10796
OVA_Lung(kfold=7)	97.92	17	1122 & 828	3068, 4420, 6991, 7167, 8134, 9797,10743,10796
OVA_Lung(kfold=8)	97.99	18	557 &382	5490, 6991, 9991,10796
OVA_Lung(kfold=9)	97.84	18	963 &584	4015, 4420, 7585,10810
OVA_Lung(kfold=10)	97.48	15	841 &833	4420, 6991,10796
OVA_Omentum(kfold=1)	94.17	16	924 &545	943,1304,1922
OVA_Omentum(kfold=2)	95.18	16	584 &697	1950,2528,5200,9033
OVA_Omentum(kfold=3)	95.32	16	139 &1275	3268, 4192, 5016, 5112,10127,10806
OVA_Omentum(kfold=4)	95.33	17	702 &359	3203,5546,9097
OVA_Omentum(kfold=5)	94.97	13	704 &835	2290, 2717, 3264,10443
OVA_Omentum(kfold=6)	94.89	16	358 &1247	3904,4292,5777,9229
OVA_Omentum(kfold=7)	94.68	14	1349 & 832	1799,2246,7965
OVA_Omentum(kfold=8)	95.54	10	277 &717	2, 864, 1004, 1845, 1982, 2030, 2431, 2672, 2674, 2740, 2895, 3360, 4138, 4210, 4249, 4250, 4309, 4337, 4787, 4858, 4889, 5333, 5711, 6643, 6760, 6926, 7635, 7642, 7781, 8304, 8433, 8697, 9142, 9362, 9364,10139,10264,10341,10357,10797,10806
OVA_Omentum(kfold=9)	95.18	15	270 &1277	3268,10806
OVA_Omentum(kfold=10)	95.18	15	536 &842	428, 949,1251,3836,4287,6084,8701,9271
OVA_Ovary(kfold=1)	89.14	15	100 &1384	3616,4249,5042,7131,9664
OVA_Ovary(kfold=2)	91.87	17	920 &1099	7137,7831
OVA_Ovary(kfold=3)	88.71	5	1239 & 715	1537 features
OVA_Ovary(kfold=4)	90.58	16	1388 & 641	246,3007,4152,4369
OVA_Ovary(kfold=5)	93.24	16	1241 &1355	34,7243,7831
OVA_Ovary(kfold=6)	89.64	16	1225 & 406	454,4369
OVA_Ovary(kfold=7)	90.08	18	1035 & 965	6071,8733
OVA_Ovary(kfold=8)	89.36	16	968 &662	3268,8482
OVA_Ovary(kfold=9)	91.44	16	264 &238	246,7243
OVA_Ovary(kfold=10)	89.43	17	557 &1042	3268,4100
blood-transfusion(kfold=1)	76.97	2	4 &84	2,3
blood-transfusion(kfold=2)	76.52	2	448 &312	2,3
blood-transfusion(kfold=3)	76.52	2	43 &610	1,2,3
blood-transfusion(kfold=4)	77.41	2	41 &611	1,2,3
blood-transfusion(kfold=5)	77.12	1	47 &247	1,2,3,4
blood-transfusion(kfold=6)	77.41	2	41 &612	1,2,3
blood-transfusion(kfold=7)	76.71	2	41 &607	1,2,3
blood-transfusion(kfold=8)	76.52	2	42 &610	1,2,3
blood-transfusion(kfold=9)	77.15	2	40 &612	1,2,3
blood-transfusion(kfold=10)	76.23	2	10 &453	2,3

All datasets

Dataset (Kfold)	TrainAcc	Iterations	Prototype Samples	Prototype Core Features
diabetes(kfold=1)	76.70	3	348 &517	2,6,7
diabetes(kfold=2)	76.99	3	353 &509	2,6
diabetes(kfold=3)	76.41	3	311 &265	2,6
diabetes(kfold=4)	73.81	3	553 &443	1,2
diabetes(kfold=5)	76.85	3	348 &525	2,6,7
diabetes(kfold=6)	77.42	3	355 &527	2,6,7
diabetes(kfold=7)	76.73	3	598 &236	2,6
diabetes(kfold=8)	77.86	3	350 &516	2,6
diabetes(kfold=9)	77.60	3	352 &519	2,6
diabetes(kfold=10)	76.99	2	575 &482	1,2,6,7
haberman(kfold=1)	76.00	1	212 &215	2,4
haberman(kfold=2)	76.00	2	85 &72	1, 2,12
haberman(kfold=3)	76.81	1	103 & 98	2,6
haberman(kfold=4)	75.64	1	212 &214	2,4
haberman(kfold=5)	76.45	2	235 &225	1,2
haberman(kfold=6)	76.73	1	96 &99	2,6
haberman(kfold=7)	75.36	1	213 &216	2,4
haberman(kfold=8)	74.55	1	214 &217	2,4
haberman(kfold=9)	76.09	1	154 &158	2,4
haberman(kfold=10)	74.55	1	212 &215	2,4

All datasets

Dataset (Kfold)	TrainAcc	Iterations	Prototype Samples	Prototype Core Features
heart-statlog(kfold=1)	60.49	5	16 &213	3,10
heart-statlog(kfold=2)	78.19	2	32 &53	3,12
heart-statlog(kfold=3)	60.91	3	102 &168	10,12
heart-statlog(kfold=4)	67.90	2	86 & 4	9,12
heart-statlog(kfold=5)	77.37	4	160 &164	3,12
heart-statlog(kfold=6)	78.19	2	7 &76	3,13
heart-statlog(kfold=7)	77.78	4	162 &166	3,12
heart-statlog(kfold=8)	78.19	2	35 &58	3,12
heart-statlog(kfold=9)	69.14	3	163 &222	10,12
heart-statlog(kfold=10)	78.19	2	63 & 3	3,13
hiva_agnostic(kfold=1)	96.53	3	1397 & 381	83, 167,1049,1325
hiva_agnostic(kfold=2)	61.32	3	786 &960	862,1272
hiva_agnostic(kfold=3)	96.61	2	2604 &2648	83, 197, 765,1049,1325
hiva_agnostic(kfold=4)	96.03	4	99 &2006	76,892
hiva_agnostic(kfold=5)	91.20	3	1323 &3583	337,618
hiva_agnostic(kfold=6)	96.61	3	2596 &3276	83, 197, 765,1049,1325
hiva_agnostic(kfold=7)	96.43	3	1428 & 791	200,1234
hiva_agnostic(kfold=8)	91.28	3	1316 &3582	337,618
hiva_agnostic(kfold=9)	96.66	3	193 &2492	83,1049,1325
hiva_agnostic(kfold=10)	96.58	4	200 &797	83,1049,1325
ilpd-numeric(kfold=1)	71.76	3	386 &251	6,7
ilpd-numeric(kfold=2)	72.90	3	470 & 75	1,6
ilpd-numeric(kfold=3)	73.90	3	476 & 7	1, 6, 9,10
ilpd-numeric(kfold=4)	72.57	5	301 &367	1,6
ilpd-numeric(kfold=5)	73.71	2	65 &409	1,7
ilpd-numeric(kfold=6)	71.81	4	475 & 75	1,6
ilpd-numeric(kfold=7)	73.14	3	476 & 6	1, 6, 9,10
ilpd-numeric(kfold=8)	73.52	3	477 & 6	1, 6, 9,10
ilpd-numeric(kfold=9)	74.29	3	480 & 7	1, 6, 9,10
ilpd-numeric(kfold=10)	72.71	3	479 & 7	1, 6, 9,10
thoracic-surgery(kfold=1)	85.11	1	319 & 35	6,8
thoracic-surgery(kfold=2)	85.34	1	44 &118	12,13,18,19
thoracic-surgery(kfold=3)	85.82	1	49 &115	12,13,18,19
thoracic-surgery(kfold=4)	84.87	3	69 &71	3,12
thoracic-surgery(kfold=5)	84.87	1	44 &113	12,13,18,19
thoracic-surgery(kfold=6)	86.29	1	46 &117	12,13,18,19
thoracic-surgery(kfold=7)	85.58	3	267 & 13	1, 3, 5,12
thoracic-surgery(kfold=8)	86.76	1	48 &116	12,13,18,19
thoracic-surgery(kfold=9)	85.34	1	43 &116	12,13,18,19
thoracic-surgery(kfold=10)	85.82	1	49 &122	12,13,18,19
wdbc(kfold=1)	94.73	5	207 &446	22,23,25,26,28,30
wdbc(kfold=2)	94.73	4	214 &450	22,23,25,29
wdbc(kfold=3)	94.92	5	208 &446	22,23,25,26,28,30
wdbc(kfold=4)	94.73	5	313 &187	2,11,13,18,22,23,25,28
wdbc(kfold=5)	93.36	4	167 &203	1, 4, 7,24,25
wdbc(kfold=6)	93.55	6	312 &184	13,22,23,25,28
wdbc(kfold=7)	95.13	5	209 &445	19,22,23,25,26,28,29,30
wdbc(kfold=8)	94.14	4	122 &185	2, 5, 6, 7,12,13,17,22,23,25,27,29
wdbc(kfold=9)	94.92	4	385 &189	2,22,23,27,29
wdbc(kfold=10)	93.75	5	99 &431	3, 5, 6, 7, 8,10,15,23,25,26,27,28,29

Advantages of Prototype Theory in Machine Learning

1. Model's **Transparency**: can be explained to **non-technical people**.
2. **Explainable** Decisions
   - Your loan is rejected because you resemble a rejected reference case compared to an accepted one.
3. **Interpretable** Decisions
   - Your loan is rejected because your **income and credit** are more like the rejected reference case than an accepted one.
4. **Fair** rule: Human-friendly reasoning with a **universal rule** that works for the **majority** (ideally, all).
5. Humans with **limited working memory** better understand the model due to the extreme **sparsity**
   - In Gene Expression data, OVA_Colon: (1545 patientsX10935 genes) → (2 patients x 3 genes) → the sparsity of 3.55X 10^{-7}
6. Low model variance due to sparsity→ better **generalization** to **very different unseen cases**
7. **Ultra-fast prediction speed** due to small model size
8. **Simple to implement** and code: math-free, optimization-free, no package dependency
9. **Inherent robustness to noisy labels** (great applications in **healthcare**, criminal **justice**, **finance**)

Applications: Alternative for Decision Trees/Logistic Regression

- In applications prioritizing interpretability, explainability, and transparency, such as High-Stakes Decisions, where slight differences in accuracy are acceptable compared to black-box models, NL can **replace or complement decision trees and logistic regression** due to its **more accurate**, **simple, human-friendly, and fair explanations**

Applications: Performance near to Black-box with Noisy Labels

- In applications where humans are the sample, such as **healthcare, criminal justice, and finance**, NL can provide a high value.
  - In these domains, typically, **labels are noisy**, and black-box models provide the same performance as simple models
- Another reason: **existence of a prototype is guaranteed**
  - A **clinical case** in healthcare.
  - A **case study** in finance.
  - A **precedent case** in criminal justice.

Applications: key player in discriminant analysis of omics data

- In **discriminant analysis of high-dimensional omics data** (e.g., gene expression) NL can overcome the **curse of dimensionality** and the **challenge of limited samples** and generate **highly sparse and interpretable models** that are essential in these domains.

Applications: State-of-the-art in prediction speed

- **NL models are extremely small**, making them suitable for **real-time applications** where prediction speed is crucial
  - e.g., defense, online trading

Applications: State-of-the-art in embedded machine learning

- For **embedded systems** (e.g., wearable devices) where processing and memory constraints exist, NL's extremely sparse models require much lower computing resources (processing and memory)

Applications: Natural choice for binary input data

- In handling **high-dimensional binary data** where dimensionality reduction or representation learning do not provide added value, NL offers a promising alternative;

Applications: Ultra-fast classification of trivial cases in vision

- In the field of **vision**, **NL does not appear to be competitive** due to **lack of a mechanism for representation learning and violation of one of its assumptions (singularity of prototypes)**

- NL can still be useful for **ultra-fast classifying of trivial cases**
  - (e.g., digit 0 vs. 1 in MINIST: 99.48% accuracy with model of 2x3 matrix)
  - frog vs. airplane in CIFAR-10, 86 % accuracy with model of 2x10 matrix)
  - Main applications
    - **better prediction speed**
    - **interpretability**
    - **explainability**

- It also can be used for detection of **discriminatory noise**

Example of Application of NL in vision

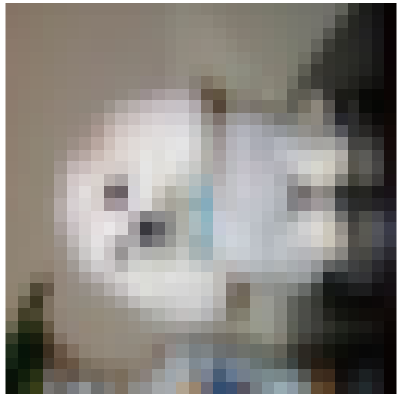
CIFAR-10: Cat vs. Dog

X_train: [10000x 3072] X_test: [2000x 3072]

Model	Test Accuracy
Vanilla DNN Hidden Layers=3 Hidden Nodes=50 ReLU activation MaxEpochs=20 LearnRate=0.01 MiniBatchSize=32	63.20%
Random Forest 100 trees	65.55%
Linear SVM C=1 (default)	58.45%
Natural Learning	**59.05%**

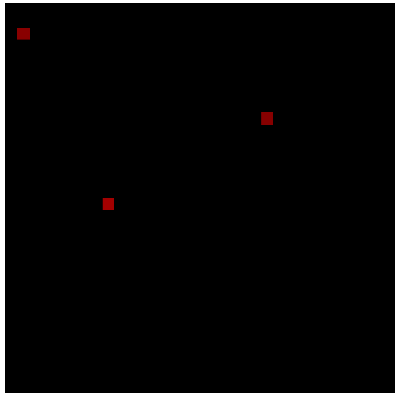
2 samples and 3 features
(only 6 values from the training set)

Dog Prototype (Sample#2778) - Original Features

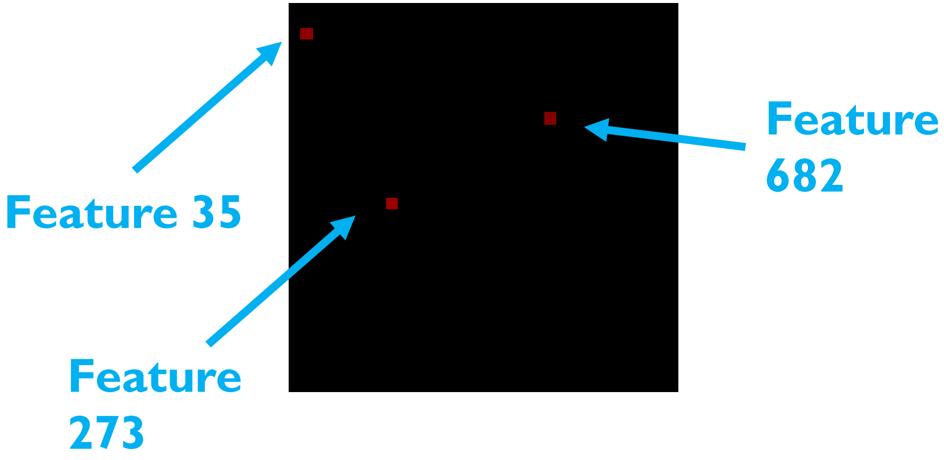


Cat Prototype (sample#3194) - Original Features

Dog Prototype (Sample#2778) - Sparse Features by NL

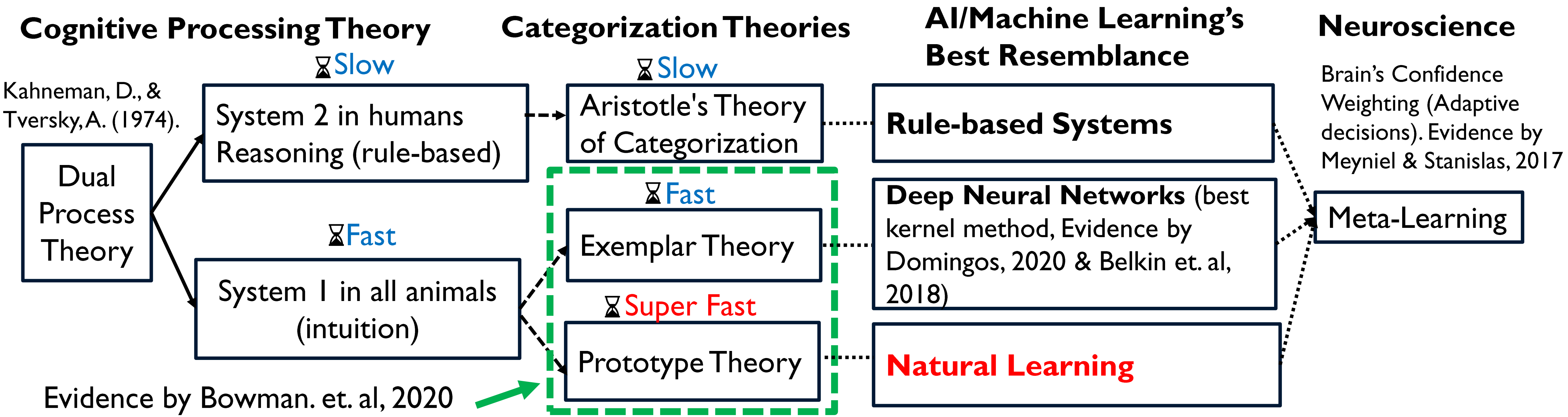


Cat Prototype (sample#3194) - Sparse Features by NL



Dual Process Theory and Natural Learning

Natural Learning Emulates the Brain's System 1 's superfast processing



Kahneman, D., & Tversky, A. (1974). Judgment under Uncertainty: Heuristics and Biases. **Science**, 185(4157), 1124–1131

Brain runs both exemplar and prototype categorization

eLife

RESEARCH ARTICLE

Tracking prototype and exemplar representations in the brain across learning

Caitlin R Bowman[1,2]*, Takako Iwashita[1], Dagmar Zeithamova[1]*

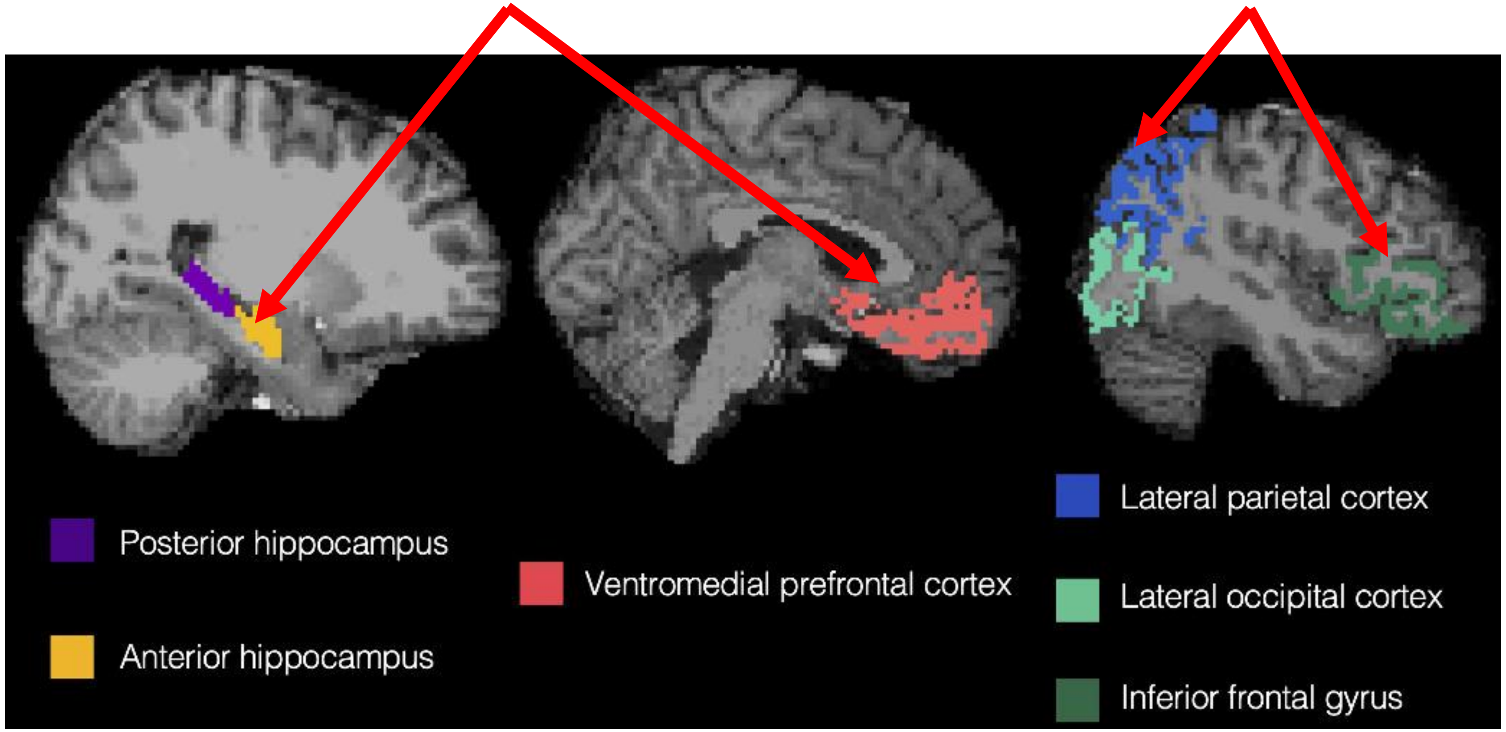
[1]Department of Psychology, University of Oregon, Eugene, United States; [2]Department of Psychology, University of Wisconsin-Milwaukee, Milwaukee, United States

Abstract There is a long-standing debate about whether categories are represented by individual category members (exemplars) or by the central tendency abstracted from individual members (prototypes). Neuroimaging studies have shown neural evidence for either exemplar representations or prototype representations, but not both. Presently, we asked whether it is possible for multiple types of category representations to exist within a single task. We designed a categorization task to promote both exemplar and prototype representations and tracked their formation across learning. We found only prototype correlates during the final test. However, interim tests interspersed throughout learning showed prototype and exemplar representations across distinct brain regions that aligned with previous studies: prototypes in ventromedial prefrontal cortex and anterior hippocampus and exemplars in inferior frontal gyrus and lateral parietal cortex. These findings indicate that, under the right circumstances, individuals may form representations at multiple levels of specificity, potentially facilitating a broad range of future decisions.

Brain runs both exemplar and prototype categorization

Natural Learning
(**Prototype** based Categorization)

Deep Learning, SVM, K-NN
(**Exemplar** based Categorization)



Bowman, Caitlin R., Takako Iwashita, and Dagmar Zeithamova. "Tracking prototype and exemplar representations in the brain across learning." elife 9 (2020): e59360.

natural-learning.cc

References

- **(Bowman. et. al, 2020)** Bowman, Caitlin R., Takako Iwashita, and Dagmar Zeithamova. "Tracking prototype and exemplar representations in the brain across learning." elife 9 (2020): e59360.
- **(Domingos, 2020)** Domingos, Pedro. "Every model learned by gradient descent is approximately a kernel machine." arXiv preprint arXiv:2012.00152 (2020).
- **(Belkin et. al, 2018)** Belkin, Mikhail, Siyuan Ma, and Soumik Mandal. "To understand deep learning we need to understand kernel learning." ICML 2018
- **(Meyniel & Stanislas, 2017)** Meyniel, Florent, and Stanislas Dehaene. "Brain networks for confidence weighting and hierarchical inference during probabilistic learning." Proceedings of the National Academy of Sciences 114.19 (2017): E3859-E3868.

Conclusion

- Prototype theory has been recognized as a **Copernican revolution** in categorization theory because it departed from the Aristotelian rule-based approach.

- Now, **we expect the same effect in machine learning:** a **transition from decision trees (Aristotelian theory of categorization) towards natural learning (prototype theory of categorization)** that provides much better human-like reasoning and, as we showed, can be more accurate than decision trees in noisy environments such as healthcare.

- NL's simple and highly-interpretable models **will provide new insights in many domains**.

Future Work

- Is it possible to implement a local representation learning or feature transformation in NL's local triplet space? We believe meaningful results in this direction can result in a **white-box version of DNNs**.

- How can we boost NL's performance without harming its attractive explainability?

- Is it possible to extend NL for regression?

Natural Learning

Learning Machine,
Inspired by Humans,
for Humans

Email: info@natural-learning.cc
Web: www.natural-learning.cc