Topology-based data reduction for Green Deep Learning

Javier Perera-Lago

University of Seville

10th November, 2023

REXASI-PRO project

As a partner of the European Project **REXASI-PRO** (REliable & eXplAinable Swarm Intelligence for People with Reduced mObility), the University of Seville is involved in the Task T6.2, called *Topology-based energy consumption optimization of Pedestrian Detection algorithm*.

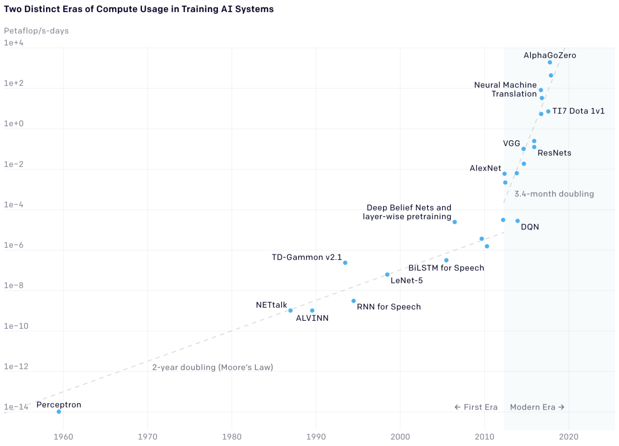
Artificial Intelligence: the training problem

Artificial Intelligence usually relies on Machine Learning models.

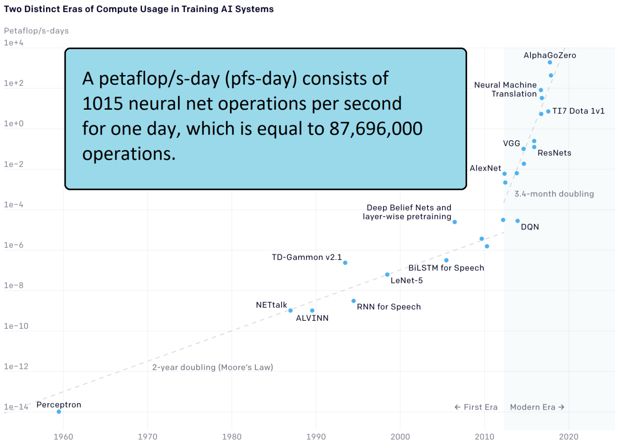
These models depend on a set of parameters that need to be adjusted. The setting or *learning* of the parameters requires a lot of real-world data.

Nowadays, we have more and more sophisticated models and more massive data sets. Because of this, the costs derived from developing new AI are growing continually.

Increasing computations in AI



Increasing computations in AI



A petaflop/s-day (pfs-day) consists of 1015 neural net operations per second for one day, which is equal to 87,696,000 operations.

*Chart taken from the OpenAI blog: AI and compute

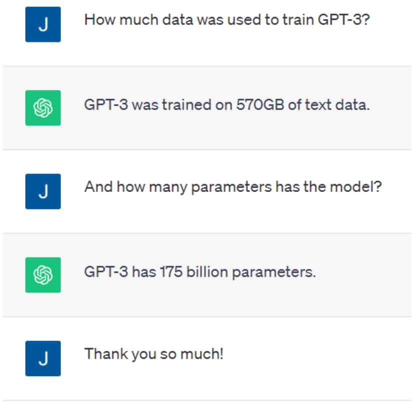
Increasing computations in Language Processing

Increasement of datasets and models for Natural Language Processing models.:

Model	Year	Dataset Size	Number of parameters
BERT-Large	2018	13 GB	350 M
GPT-2-XL	2019	40 GB	1.5 B
ROBERTA	2019	160 GB	125 M
XLNet-Large	2020	158 GB	340 M
T5-11B	2020	750 GB	11 B

Increasing computations in Language Processing

What about OpenAI's GPT-3?
Let's ask the model itself.



J How much data was used to train GPT-3?

GPT-3 was trained on 570GB of text data.

J And how many parameters has the model?

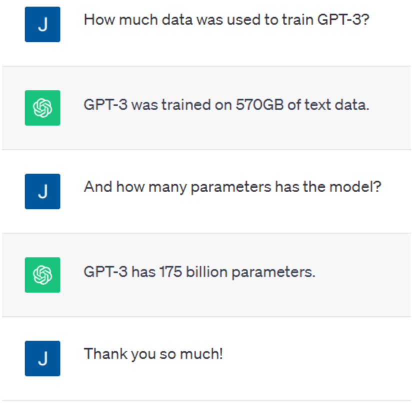
GPT-3 has 175 billion parameters.

J Thank you so much!

Increasing computations in Language Processing

What about OpenAI's GPT-3?
Let's ask the model itself.

Be polite to the assistant!



J How much data was used to train GPT-3?

GPT-3 was trained on 570GB of text data.

J And how many parameters has the model?

GPT-3 has 175 billion parameters.

J Thank you so much!

Red AI vs Green AI

- Red AI: AI research that seeks to improve the performance of models through the use of massive computational power without taking costs into account.

- Green AI: AI research that, in addition to seeking good results, seeks to reduce the consumption of resources.

Green AI: How to measure the efficiency

Proposed efficiency measures:

- Carbon emission
- Electricity usage
- Elapsed running time
- FPO: Floating-Point Operations

Green DL

$$\left.\begin{array}{c}\text{Green AI}\\ +\\ \text{Deep Learning}\end{array}\right\} \Longrightarrow \text{Green Deep Learning}$$

Green DL: 4 approaches

According to the literature, there are four main ways to reduce the costs in Deep Learning:

- Compact Architecture Design
- Energy-efficient Training Strategies
- Energy-efficient Inference
- Efficient Data Usage

Green DL: 4 approaches

According to the literature, there are four main ways to reduce the costs in Deep Learning:

- Compact Architecture Design
- Energy-efficient Training Strategies
- Energy-efficient Inference
- Efficient Data Usage $\longleftarrow$ We will focus on this approach

Efficient data usage: Data Reduction

We want to increase the efficiency in data usage by reducing the size of the dataset, trying that the resulting reduced dataset, even containing less information, gives us a good representation of the full dataset.

$$\begin{tikzcd}\boxed{X_F\text{: Full Dataset}} \arrow[r, "\text{Reduction}"] & \boxed{X_R\text{: Reduced Dataset}}\end{tikzcd}$$

$$\text{Properties}(X_F)\approx\text{Properties}(X_R)$$

Efficient data usage: Data Reduction

The idea is, assuming that the reduced dataset has similar properties to the full dataset, to use the reduced dataset for model training instead of the full dataset, making the process less expensive and giving similar results.

$$\begin{tikzcd}
X_F \arrow[r] \arrow[d] & \text{Training Process} \arrow[r] & \text{Model } M_F \\
X_R \arrow[r] & \text{Training Process} \arrow[r] & \text{Model } M_R
\end{tikzcd}$$

Properties(X_F)≈Properties(X_R) $\Rightarrow$ Model M_F≈ModelM_R

Ways to reduce a dataset

Some authors divide the task of reducing the size of a dataset into two main subtasks:

- **Reducing feature size**: eliminating irrelevant or redundant features diminishes the dataset size and mitigates the risk of overfitting.

$$\mathbb{X}_{N\times D} \longrightarrow \mathbb{Y}_{N\times d}\ (d << D)$$

- **Reducing sample size**: discarding redundant or noisy examples and alleviating imbalances between classes can improve the training process.

$$\mathbb{X}_{N\times D} \longrightarrow \mathbb{Z}_{n\times D}\ (n << N)$$

Ways to reduce a dataset

The task of reducing feature size is usually achieved by two main strategies:

- **Feature Selection**: choosing a subset of features.
- **Dimensionality Reduction**: projecting the feature space into a lower-dimensional space with a function (linear or non-linear)

Ways to reduce a dataset

The task of reducing sample size is usually achieved by two main strategies:

- **Instance Selection**: choosing a proper subset of examples.
- **Prototype Generation**: creating a set of synthetic examples that represent the real ones.

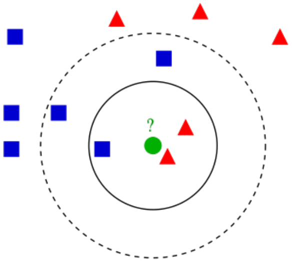
Ways to reduce a dataset

For the remainder of this talk, we will focus only on the task of **reducing the sample size**, which from now on we will simply call **Data Reduction**.

Data Reduction: k-NN algorithm

The interest in Data Reduction began in the 1960s, when the use of the k-NN (k Nearest Neighbours) algorithm became popular.

Given a new unlabeled example, the algorithm looks for its k nearest elements in the dataset and uses them to assign a label to it.



*Picture from the blog https://www.analyticsvidhya.com/

Data Reduction: k-NN algorithm

When the dataset has a large size, the search for the k nearest neighbours can be computationally expensive. In that context, the following question arose:

Data Reduction: k-NN algorithm

When the dataset has a large size, the search for the k nearest neighbours can be computationally expensive. In that context, the following question arose:

Do we really need all that data?

Data Reduction: k-NN algorithm

When the dataset has a large size, the search for the k nearest neighbours can be computationally expensive. In that context, the following question arose:

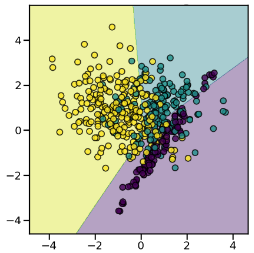
Do we really need all that data?

That is, would we be able to achieve the same classification with the k-NN algorithm but doing the search on a smaller dataset?

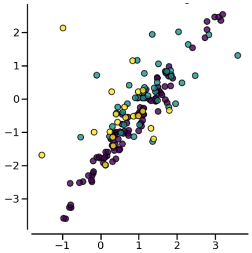
Data Reduction: k-NN algorithm

Basic approaches for Data Reduction for the k-NN algorithm

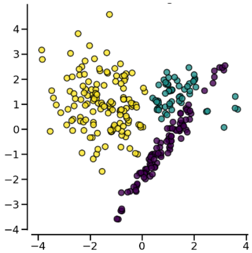
Full dataset



Condensation



Edition



*Pictures from the imbalanced-learn Pyhton library documentation site.

Data Reduction: k-NN algorithm

Starting from these two simple ideas, many new methods were developed to reduce the dataset for a k-NN classifier, such as:

- **CNN**: Condensed Nearest Neighbours (1968)
- **ENN**: Edited Nearest Neighbours (1972)
- **SNN**: Selective Nearest Neighbours (1975)
- **RNN**: Reduced Nearest Neighbours (1972)
- **RENN**: Repeated Edited Nearest Neighbours (1976)
- **All k-NN**: All k Nearest Neighbours (1976)
- **Tomek's Links** (1997)
- ...

Data Reduction: instance-based algorithms

Other Data Reduction methods appeared for other instance-based algorithms, such as:

- **IB2** (1987), **IB3** (1991), **IB4** and **IB5** (1992) (IB stands for *Instance Based*)
- Shrink Algorithm (1987)
- **MCS**: Model Class Selection (1993)
- **TIBL**: Typical Instance Based Learning (1992)
- **RMHC**: Random Mutation Hill Climbing (1994)
- **ELGrow**: Encoding Length Grow (1995)
- **DROP1**, **DROP2**, **DROP3**, **DROP4** and **DROP5** (2000) (DROP stands for *Decremental Reduction Optimization Procedure*)
- . . .

Data Reduction: other classifiers

The list is really long. If we want to reduce the dataset for other classifiers instead of k-NN, we can create new reduction methods:

- **For SVM**: SVM-KM (2000), SV-kNNC (2006), RSVM (2007), PSVM(2010), FCNN-SVM (2010), CCHSVM (2013), MASVM (2014), IPSVM(2018), CBCH (2019), BF-RB (2019), PH scheme (2019) ...
- **For Neural Networks**: Forgetting events score (2019), GraNd Score (2021), EL2N Score (2021), Data Pruning via Influence Functions (2023), Data selection via Proxy (2020) ...

Data Reduction: filter methods

Are all the reduction techniques designed for specific models?

No, there are mainly two kinds of models:

- **Wrapper methods**: the reduction is wrapped in the classifier model.
- **Filter methods**: the reduction is independent of the classifier model.

All the methods listed before are **wrapper methods**.

Data Reduction: filter methods

The list of filter methods is also long. Let's see some of them:

- Random sampling, Grid Selection (2022), MaxmMin Selection (2021), Distance-Entropy Selection (2022), Weighted Prototypes (2000), Pattern by Ordered Projections (2003), Pair Opposite Class-Nearest Neighbor (2005), kd-tress (2006), Chang Algorithm (1974), Generalized-Modified Chang Algorithm (2002), Nearest Sub-class Classifier (2005), Clustering centroids selection (2006), Object selection by clustering (2007), Prototype Selection by Relevance (2008), PH Landmarks (2023), Reduction via Matrix Decomposition (2019), Principal Sample Analysis (2018), ProtoGreedy (2019), ProtoDash(2019)...

Topology-preserving Data Reduction

Among all these Data Reduction methods, we want to focus on one that is:

- Easy to understand.
- Easy to implement.
- Good to preserve topological features of the dataset.

Topology-preserving Data Reduction

Among all these Data Reduction methods, we want to focus on one that is:

- Easy to understand.
- Easy to implement.
- Good to preserve topological features of the dataset.

We are talking about the **Dominating Dataset** method, defined in [1].

[1] Gonzalez-Diaz, R., Gutiérrez-Naranjo, M. A., & Paluzo-Hidalgo, E. (2022). Topology-based representative datasets to reduce neural network training resources. Neural Computing and Applications, 34(17), 14397-14413.

Dominating Datasets: the task

Let's assume we are trying to solve a classification task, and our dataset $\mathcal{D}$ is defined:

$$\mathcal{D} = \{(x, c_x) | x \in X \subset \mathbb{R}^n, c_x \in [[0, k]]\}$$

where $[[0, k]] = \{0, 1, 2, \cdots, k\}$. For each point $x \in X$, there is a label c_x that tells us its class. Each point belongs to one and only one class.

The classification task consists of, given a new example $y \notin X$, to predict the correct class c_y.

Dominating Datasets: representative points

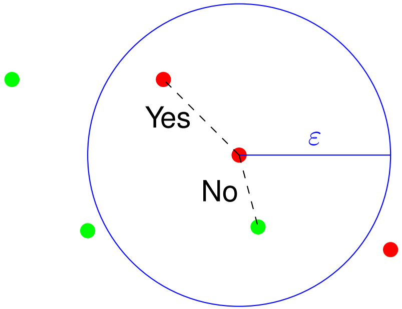
The aim of the Dominating Dataset method is to find a dataset $\tilde{\mathcal{D}}$, with $|\tilde{\mathcal{D}}| << |\mathcal{D}|$ such that $\tilde{\mathcal{D}}$ is an ε-representative of $\mathcal{D}$.

Definition

Given a real number $\varepsilon > 0$ which we call the representation error, a labelled point (x, c_x) is ε-representative of $(\tilde{x}, c_{\tilde{x}})$ if $c_x = c_{\tilde{x}}$ and $||x - \tilde{x}|| \leq \varepsilon$. We denote $x \approx_\varepsilon \tilde{x}$.

Dominating Datasets: representative points

Example of ε-representative points.



Dominating Datasets: representative datasets

We extend ε-representativeness between pair of points to define the ε-representativeness between datasets:

Definition

A dataset $\tilde{\mathcal{D}} = \{(\tilde{x}, c_{\tilde{x}}) | \tilde{x} \in \tilde{X} \subset \mathbb{R}^n, c_{\tilde{x}} \in [[0,k]]\}$ is ε-representative of $\mathcal{D} = \{(x, c_x) | x \in X \subset \mathbb{R}^n, c_x \in [[0,k]]\}$ if there exists an isometric transformation $f : \tilde{X} \to \mathbb{R}^n$, such that for any $(x, c_x) \in \mathcal{D}$ there exists $(\tilde{x}, c_{\tilde{x}}) \in \tilde{\mathcal{D}}$ satisfying that $f(\tilde{x}) \approx_\varepsilon x$.

Dominating Datasets: controlling persistent homology

ε-representative datasets preserve persistent homology:

Dominating Datasets: controlling persistent homology

ε-representative datasets preserve persistent homology:

Theorem 1 [1]

If the dataset $\tilde{\mathcal{D}}$ is ε-representative of $\mathcal{D}$, then

$$d_B(\mathrm{Dgm}_q(X), \mathrm{Dgm}_q(\tilde{X})) \leq 2\varepsilon$$

where $q \leq n$, $\mathrm{Dgm}_q(X)$ and $\mathrm{Dgm}_q(\tilde{X})$ are the persistence diagrams of the Vietoris-Rips filtrations computed from X and $\tilde{X}$, and d_B denotes the bottleneck distance between their persistence diagrams.

Dominating Datasets: proximity graphs

How do we get a subset $\tilde{\mathcal{D}} \subset \mathcal{D}$ that is ε-representative of $\mathcal{D}$?

Dominating Datasets: proximity graphs

How do we get a subset $\tilde{\mathcal{D}} \subset \mathcal{D}$ that is ε-representative of $\mathcal{D}$?

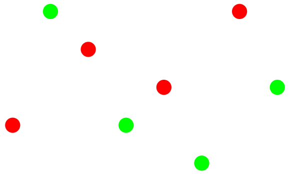
Modelling the ε-representativeness with graphs:

Definition

Given a dataset $\mathcal{D} = \{(x, c_x) | x \in X \subset \mathbb{R}^n, c_x \in [[0, k]]\}$ and given $\varepsilon > 0$, the ε-proximity graph of $\mathcal{D}$ is the graph $G_\varepsilon(\mathcal{D}) = (X, E)$ where $(x, y) \in E$ if and only if $x \approx_\varepsilon y$.

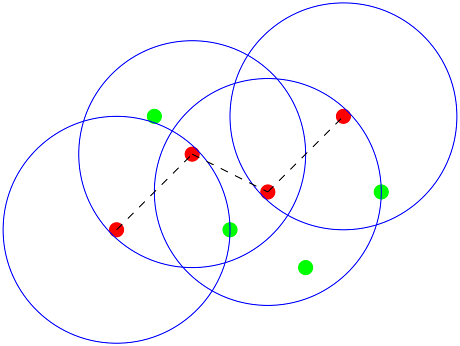
Dominating Datasets: proximity graphs

Let's calculate the ε-proximity graph over this dataset with classes, red and green.



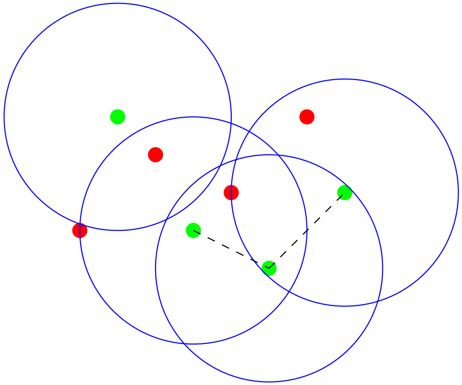
Dominating Datasets: proximity graphs

We start computing the distances between the red points.



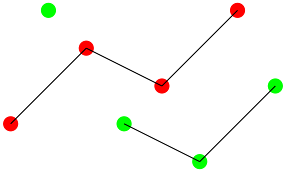
Dominating Datasets: proximity graphs

We then compute the distances between the green points.



Dominating Datasets: proximity graphs

The total ε-proximity graph is the union of the graphs obtained before.



Dominating Datasets: dominating set of a graph

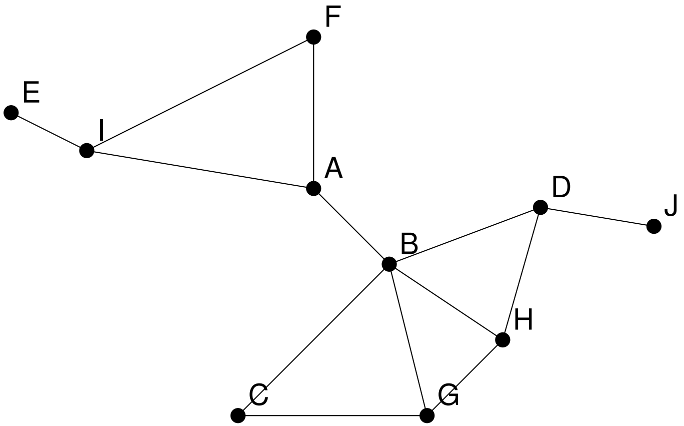
Given this ε-proximity graph, we can find a ε-representative subset $\tilde{\mathcal{D}}$ via dominating sets (also known as vertex covers):

Definition

Given a graph $G = (X, E)$, a dominating set or vertex cover is a set $\tilde{X} \subseteq X$ such that for any $x \in X$, either $x \in \tilde{X}$ or there exists $y \in \tilde{X}$ adjacent to x in G.

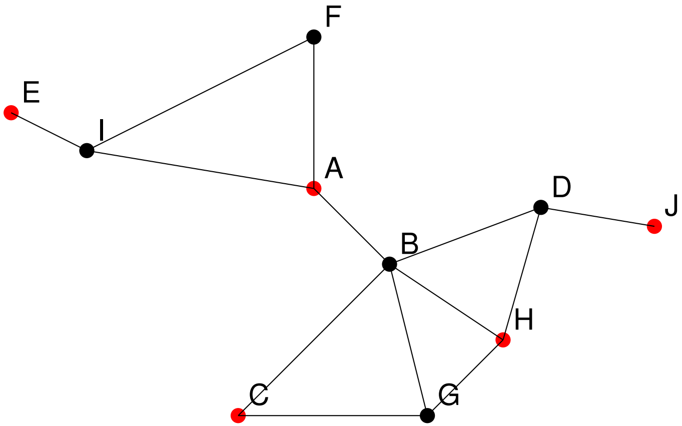
Dominating Datasets: dominating set of a graph

Let’s consider the following graph.



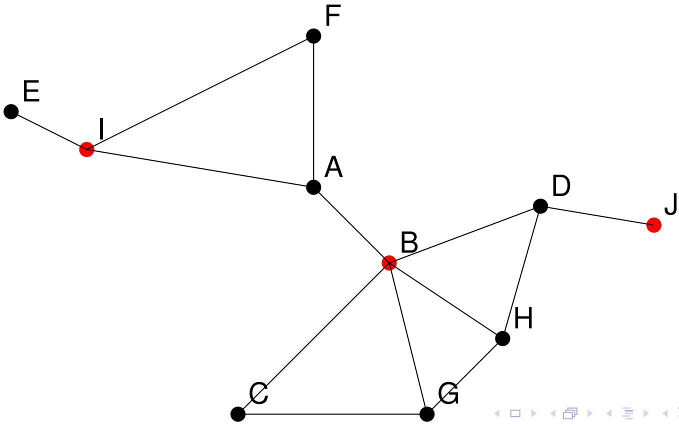
Dominating Datasets: dominating set of a graph

This set of red vertices is a dominant set for the graph.



Dominating Datasets: dominating set of a graph

This other set of red vertices is also a dominant set for the graph.



Dominating Datasets: Algorithm

Summarizing, given a dataset $\mathcal{D}$, we can find a ε-representative sub-dataset $\tilde{\mathcal{D}}$ by applying the following algorithm:

Dominating Dataset Algorithm

Input: A dataset $\mathcal{D} = \{(x, c_x) | x \in X \subset \mathbb{R}^n, c_x \in [[0, k]]\}$ and a parameter $\varepsilon > 0$

Steps:

1. Build the ε-proximity graph $G_\varepsilon(\mathcal{D})$.
2. Find a dominating set $\tilde{X}$ for $G_\varepsilon(\mathcal{D})$.

Output: A dataset $\tilde{\mathcal{D}} = \{(x, c_x) | (x, c_x) \in \mathcal{D}, x \in \tilde{X}\}$

$\tilde{\mathcal{D}}$ is said to be a Dominating Dataset of $\mathcal{D}$.

Dominating Datasets: λ-balanced datasets

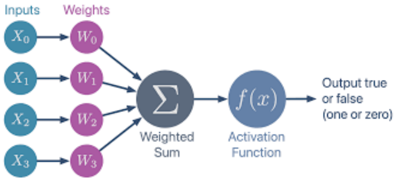
In addition, we want the Dominating Dataset to be balanced.

Definition

Let be $\tilde{\mathcal{D}} = \{(\tilde{x}, c_{\tilde{x}})\}$ a ε-representative dataset of $\mathcal{D} = \{(x, c_x)\}$. $\tilde{\mathcal{D}}$ is said to be λ-balanced if for each $(\tilde{x}, c_{\tilde{x}}) \in \tilde{\mathcal{D}}$ the set $\{(x, c_x) | f(\tilde{x}) \approx_\varepsilon x\}$ has exactly λ points and for each $(x, c_x) \in \mathcal{D}$ there exists only one $(\tilde{x}, c_{\tilde{x}})$ such that $f(\tilde{x}) \approx_\varepsilon x$.

Dominating Dataset: the perceptron case

Let's focus now on the particular case of a binary perceptron $\mathcal{N}_w$.



The perceptron $\mathcal{N}_w$ classifies the point $x = (x_1, \cdots, x_n)$ with the rule:

$$\mathcal{N}_w(x) = \begin{cases} 1 & \text{if} \quad \sigma(w^T x) \geq 1/2 \\ 0 & \text{if} \quad \sigma(w^T x) < 1/2 \end{cases}$$

*Picture from the blog https://medium.com/

Dominating Dataset: the perceptron case

Given the dataset $\mathcal{D}$, the weights vector w is trained to minimize the error function:

$$\mathbb{E}(w, \mathcal{D}) = \frac{1}{|X|} \sum_{x \in X} (c_x - \sigma(w^T x))^2$$

Dominating Dataset: the perceptron case

After the training, we want to measure the goodness of the fit on our dataset $\mathcal{D}$. First, we test if it fits correctly each individual point:

$$\mathcal{I}_w(x) = \begin{cases} 1 & \text{if} \quad c_x = \mathcal{N}_w(x) \\ 0 & \text{if} \quad c_x \neq \mathcal{N}_w(x) \end{cases}$$

Dominating Dataset: the perceptron case

After the training, we want to measure the goodness of the fit on our dataset $\mathcal{D}$. First, we test if it fits correctly each individual point:

$$\mathcal{I}_w(x) = \begin{cases} 1 & \text{if} \quad c_x = \mathcal{N}_w(x) \\ 0 & \text{if} \quad c_x \neq \mathcal{N}_w(x) \end{cases}$$

And define the global accuracy as:

$$\mathbb{A}(\mathcal{D}, \mathcal{N}_w) = \frac{1}{|X|} \sum_{x \in X} \mathcal{I}_w(x)$$

Dominating Datasets: the perceptron case

Theorem 2 [1]

Let be $\mathcal{D}$ a binary dataset (with only two classes), and let be $\tilde{\mathcal{D}}$ an ε-representative dataset of $\mathcal{D}$. Let be $\mathcal{N}_w$ a perceptron with weights $w \in \mathbb{R}^{n+1}$. Then,

$$\varepsilon \leq \min \left\{ \frac{||wx||}{||w||} : (x, c_x) \in \mathcal{D} \right\} \Rightarrow \mathbb{A}(\mathcal{D}, \mathcal{N}_w) = \mathbb{A}(\tilde{\mathcal{D}}, \mathcal{N}_w)$$

where $\mathbb{A}$ denotes the accuracy of the classifier.

Dominating Datasets: the perceptron case

Theorem 3 [1]

Let $\delta > 0$. Let be $\mathcal{D}$ a binary dataset, and let be $\tilde{\mathcal{D}}$ an ε-representative dataset of $\mathcal{D}$. Let be $\mathcal{N}_w$ a perceptron with weights $w \in \mathbb{R}^{n+1}$. Then,

$$\varepsilon \leq \frac{54}{43||w||}\delta \Rightarrow ||\mathbb{E}(w, \mathcal{D}) - \mathbb{E}(w, \tilde{\mathcal{D}})|| \leq \delta$$

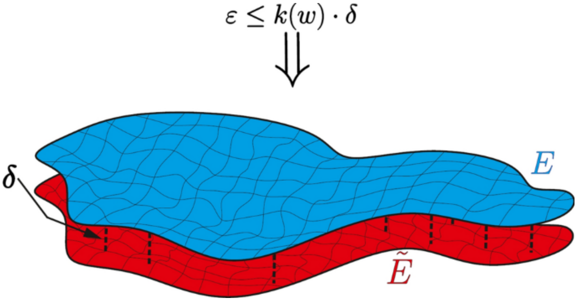
where $\mathbb{E}$ denotes the error function of the perceptron $\mathcal{N}_w$.

Dominating Datasets: the perceptron case

Let us give some intuition to Theorem 3.

Dominating Datasets: the perceptron case

Let us give some intuition to Theorem 3.



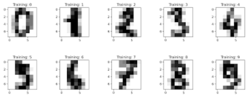
The error function can be understood as an error surface. For a fixed set of weights w, $\mathcal{D}$ and $\tilde{\mathcal{D}}$ define two error functions $\mathbb{E}(w, \mathcal{D})$ and $\mathbb{E}(w, \tilde{\mathcal{D}})$, whose difference is bounded by δ id the condition is satisfied. *Figure from [1]

Dominating Datasets: general neural networks

Although similar theoretical results are not yet available for more complex networks, it can be observed that in practice reducing a dataset using this algorithm works well. The experiments were developed for two classical datasets in classification:

Iris Dataset

Digits 8×8 Dataset



Dominating Datasets: general neural networks

After repeating the experiments 5 times on each dataset, and evaluating all the metrics using the original dataset, the results were:

	Dataset	Acc.	Rec.	Prec.	AUC	MSE
Iris	Original	1	1	1	0.99	0.003
	Dominating	0.9	1	0.9	0.9	0.11
	Random	0.6	0.2	0.2	0.36	0.39
Digits	Original	0.95	0.94	0.96	0.99	0.01
	Dominating	0.77	0.76	0.78	0.95	0.04
	Random	0.63	0.62	0.64	0.89	0.06

*Table from [1]

Dominating Datasets: the minimum dataset problem

Since we want to reduce the dataset as much as possible, we ask ourselves:

Dominating Datasets: the minimum dataset problem

Since we want to reduce the dataset as much as possible, we ask ourselves:

Is it possible to find a Dominating Dataset with minimal size?

Dominating Datasets: the minimum dataset problem

The Dominating Dataset Algorithm explained before computes the dominating dataset as the vertex cover of a graph.

Then, finding a minimal Dominating Dataset is equivalent to finding a minimal vertex cover for a graph.

Dominating Datasets: the minimum dataset problem

The Dominating Dataset Algorithm explained before computes the dominating dataset as the vertex cover of a graph.

Then, finding a minimal Dominating Dataset is equivalent to finding a minimal vertex cover for a graph.

But...

Dominating Datasets: the minimum dataset problem

The Dominating Dataset Algorithm explained before computes the dominating dataset as the vertex cover of a graph.

Then, finding a minimal Dominating Dataset is equivalent to finding a minimal vertex cover for a graph.

But...that problem is **NP-complete!!!**

Dominating Datasets: the minimum dataset problem

That means that we have to use approximate algorithms that run in non-exponential time, such as:

- **Greedy edge algorithm**: iteratively picking and removing both ends of a random edges (it is a 2-approximation).
- **Greedy vertex algorithm**: iteratively picking and removing the vertex with highest degree (it is a 2-approximation).
- **Integer Linear Programming**: Solving the following optimization problem (its relaxation is a 2-approximation):

$$\begin{array}{lll} \min & \sum\limits_{v \in V} x_v & \\ s.t. & x_u + x_v \geq 1 & \forall (u,v) \in E \\ & x_v \in \{0,1\} & \forall v \in V \end{array}$$

Dominating Datasets: our vertex cover algorithm

But in our experiments, the algorithm we apply in our implementations is:

Vertex Cover Algorithm

Input: A graph $G = (X, E)$

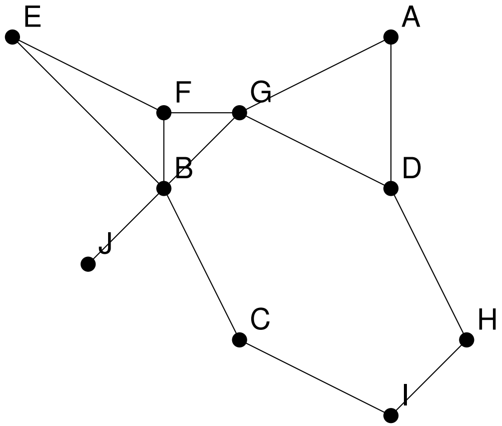
Steps:

```
1. Set $\tilde{X} = \emptyset$.
2. Pick $x \in X$ randomly and add it to $\tilde{X}$.
3. Remove $x$ and all its adjacent vertices from $X$.
4. If $X$ is now empty, stop. Else, go back to 2.
```

Output: A vertex cover $\tilde{X}$.

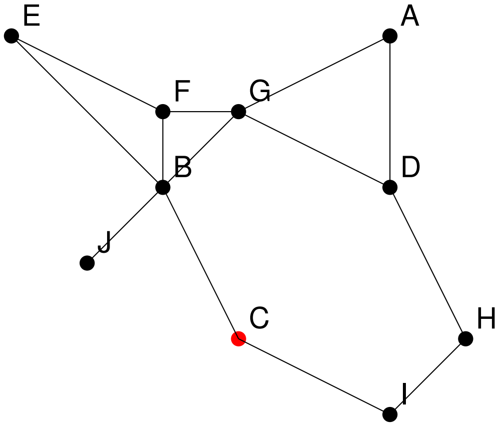
Dominating Datasets: our vertex cover algorithm

Let's consider this graph.



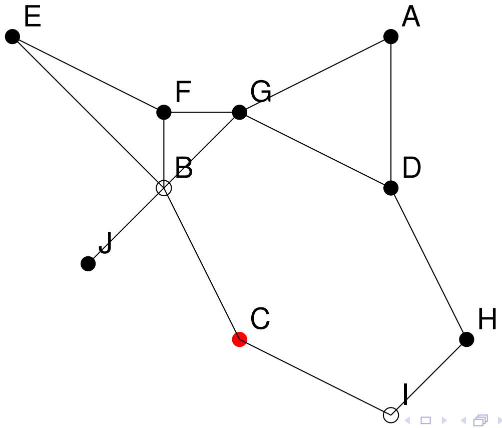
Dominating Datasets: our vertex cover algorithm

We choose a vertex at random, in this case, C.



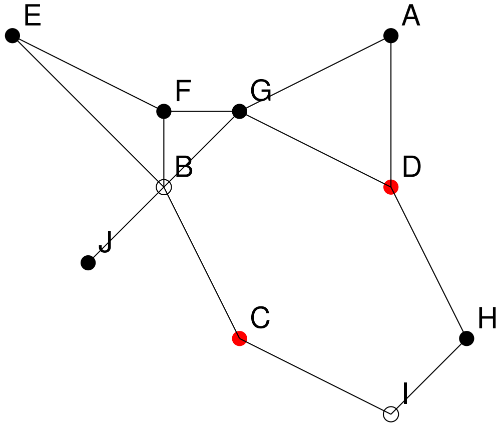
Dominating Datasets: our vertex cover algorithm

We include C in the dominating set and we discard the adjacent nodes.



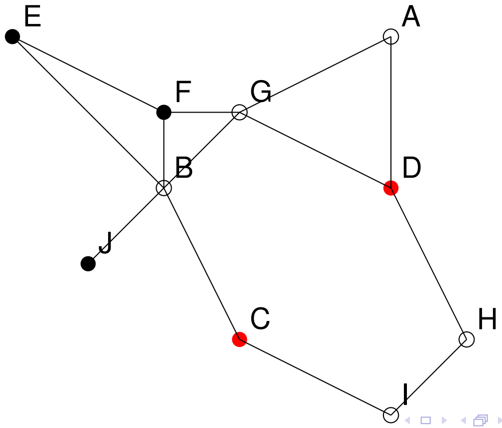
Dominating Datasets: our vertex cover algorithm

Now we choose another vertex at random, this case, D.



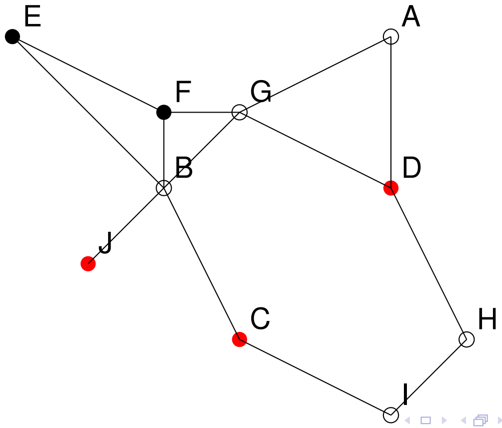
Dominating Datasets: our vertex cover algorithm

We include D in the dominating set and we discard its adjacent nodes.



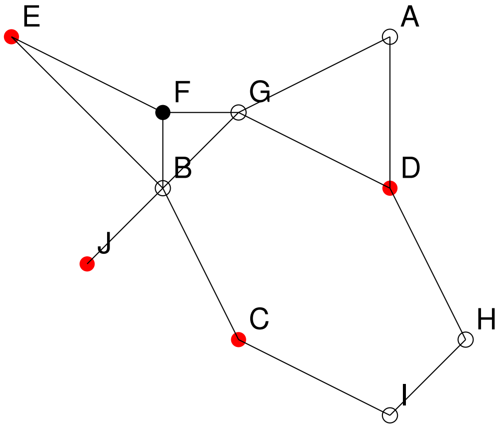
Dominating Datasets: our vertex cover algorithm

We choose again at random, and J is added. There is no vertex to discard this time



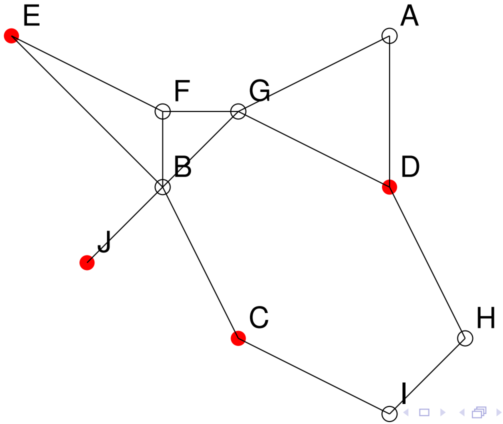
Dominating Datasets: our vertex cover algorithm

There are only two remaining vertices. We choose E.



Dominating Datasets: our vertex cover algorithm

After discarding F, there are no more vertices left, and the algorithm ends. The dominating vertex set is $\{C, D, J, E\}$



Dominating Datasets: our vertex cover algorithm

Advantages:

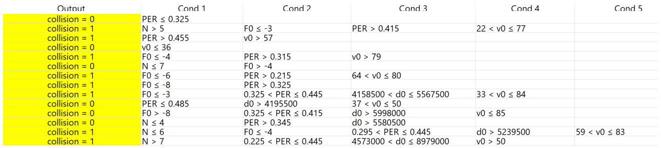
- It is fast. For a dataset $X_{N \times D}$ it only requires $\mathcal{O}(ND)$ computations.
- It allows you to find the Dominating Dataset without even computing the proximity graph.
- In case you already have the graph, the algorithm is already implemented in the Python library *networkx*.

Disadvantages:

- It is not an n-approximation for any $n \geq 2$ (star graph).

Next steps: Collaboration with CNR

Thanks to the REXASI-PRO project, we have started a collaboration with the team of Maurizio Mongelli in the CNR. They are using a **Logic Learning Machine** to solve a classification task in the Platooning dataset. The Machine assigns a class to an item following a list of explainable rules like these:



Output	Cond 1	Cond 2	Cond 3	Cond 4	Cond 5
collision = 0	PER ≤ 0.325				
collision = 1	N > 5	F0 ≤ -3	PER > 0.415	22 < v0 ≤ 77	
collision = 1	PER > 0.455	v0 > 57			
collision = 0	v0 ≤ 36				
collision = 1	F0 ≤ -4	PER > 0.315	v0 > 79		
collision = 0	N ≤ 7	F0 > -4			
collision = 1	F0 ≤ -6	PER > 0.215	64 < v0 ≤ 80		
collision = 1	F0 ≤ -8	PER > 0.325			
collision = 1	F0 ≤ -3	0.325 < PER ≤ 0.445	4158500 < d0 ≤ 5567500	33 < v0 ≤ 84	
collision = 0	PER ≤ 0.485	d0 > 4195500	37 < v0 ≤ 50		
collision = 0	F0 > -8	0.325 < PER ≤ 0.415	d0 > 5998000	v0 ≤ 85	
collision = 0	N ≤ 4	PER > 0.345	d0 > 5580500		
collision = 1	N ≤ 6	F0 ≤ -4	0.295 < PER ≤ 0.445	d0 > 5239500	59 < v0 ≤ 83
collision = 1	N > 7	0.225 < PER ≤ 0.445	4573000 < d0 ≤ 8979000	v0 > 50	

Next steps: Collaboration with CNR

We have been doing some experiments together, testing how does the Dominating Dataset reduction affect the training of this Logic Learning Machine. The experimental results so far are:

Next steps: Collaboration with CNR

We have been doing some experiments together, testing how does the Dominating Dataset reduction affect the training of this Logic Learning Machine. The experimental results so far are:

- For low-dimensional datasets, the reduction does not affect the performance of the final classifier.
- For high-dimensional datasets, the classifier has a decrease in the performance but it preserves the properties of the model trained on the full dataset.

Next steps: Collaboration with CNR

Next steps in this line:

- Study the effect of Dominating Datasets on more complex models.
- Study the relationship between data reduction and explainability.

Thanks for your attention.