(Deep) Machine Learning for Science

Julien Mairal

Univ. Grenoble Alpes, Inria

AlgoSB school, Cargèse, 2024

Machine Learning and Science

Machine learning and the ideal world

1. **observe** the world (gather data);
2. **propose models** of the world (design and learn);
3. **test** on new data (estimate the generalization error).

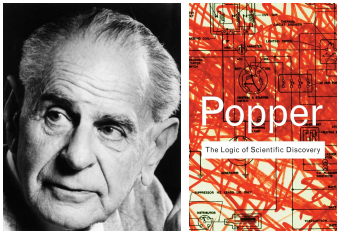
This paradigm may seem compatible with Popper's view of the scientific method.

Machine Learning and Science

Machine learning and the ideal world

1. **observe** the world (gather data);
2. **propose models** of the world (design and learn);
3. **test** on new data (estimate the generalization error).

This paradigm may seem compatible with Popper's view of the scientific method.



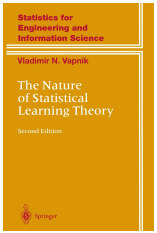
- Karl Popper introduced in the 1930's the **falsifiability** criterion to distinguish scientific statements from other claims—that is, the ability to submit claims to **testing**.

Machine Learning and Science

Machine learning and the ideal world

1. **observe** the world (gather data);
2. **propose models** of the world (design and learn);
3. **test** on new data (estimate the generalization error).

This paradigm may seem compatible with Popper's view of the scientific method.



- Vladimir Vapnik links Popper's view of the scientific method with machine learning theory.
- Hypothesis classes that are "too large" may be able to explain anything and lead to **unfalsifiable** theories.

Machine Learning and Science

Machine learning and the ideal world

1. **observe** the world (gather data);
2. **propose models** of the world (design and learn);
3. **test** on new data (estimate the generalization error).

This paradigm may seem compatible with Popper's view of the scientific method.

J Gen Philos Sci (2009) 40:51–58
DOI 10.1007/s10838-009-9091-3

ARTICLE

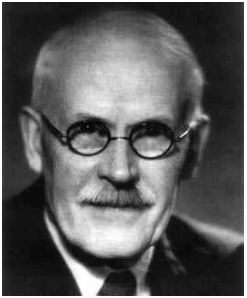
Falsificationism and Statistical Learning Theory: Comparing the Popper and Vapnik-Chervonenkis Dimensions

David Corfield · Bernhard Schölkopf · Vladimir Vapnik

- Is it really compatible?
- Is the concept of **generalization error** a missing/useful piece in Popper's theory?

Machine Learning and Science: Should we Aim for Simplicity?

(a) Dorothy Wrinch
1894–1980



(b) Harold Jeffreys
1891–1989

The existence of simple laws is, then, apparently, to be regarded as a quality of nature; and accordingly we may infer that it is justifiable to prefer a simple law to a more complex one that fits our observations slightly better.

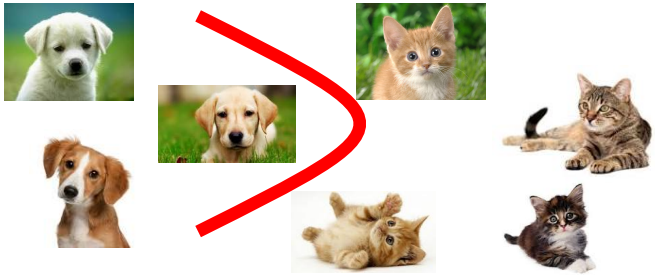
[Wrinch and Jeffreys, 1921]. Philosophical Magazine Series.

Part I: A small tour into the perfect machine learning world

Supervised learning and empirical risk minimization

In supervised learning, we learn a **prediction function** $h : \mathcal{X} \to \mathcal{Y}$ given labeled i.i.d. training data $(x_i, y_i)_{i=1,\dots,n}$ with x_i in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

$$\min_{h \in \mathcal{H}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} L(y_i, h(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(h)}_{\text{regularization}} .$$



Supervised learning and empirical risk minimization

In supervised learning, we learn a **prediction function** $h : \mathcal{X} \to \mathcal{Y}$ given labeled i.i.d. training data $(x_i, y_i)_{i=1,\dots,n}$ with x_i in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

$$\min_{h \in \mathcal{H}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} L(y_i, h(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(h)}_{\text{regularization}} .$$

The labels y_i are in

- $\{-1, +1\}$ for **binary** classification.
- $\{1, \dots, K\}$ for **multi-class** classification.
- $\mathbb{R}$ for **regression**.
- $\mathbb{R}^k$ for **multivariate regression**.
- any general set for **structured prediction**.

Supervised learning and empirical risk minimization

In supervised learning, we learn a **prediction function** $h : \mathcal{X} \to \mathcal{Y}$ given labeled i.i.d. training data $(x_i, y_i)_{i=1,\dots,n}$ with x_i in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

$$\min_{h \in \mathcal{H}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} L(y_i, h(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \Omega(h)}_{\text{regularization}} .$$

Example 1: linear models

- assume there exists a linear relation between y in $\mathbb{R}$ and features x in $\mathbb{R}^p$.
- $h(x) = w^\top x + b$ is parametrized by w, b in $\mathbb{R}^{p+1}$.
- L is often a **convex** loss function.
- $\Omega(h)$ is often the squared ℓ_2-norm $\|w\|^2$.

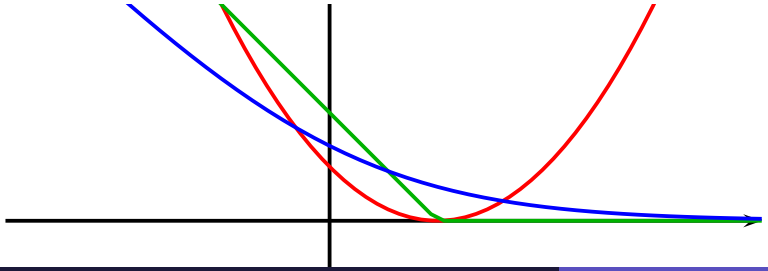
Supervised learning and empirical risk minimization

A few examples of linear models with no bias b:

Ridge regression: $$\min_{w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2}(y_i - w^\top x_i)^2 + \lambda \|w\|_2^2.$$

Linear SVM: $$\min_{w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i w^\top x_i) + \lambda \|w\|_2^2.$$

Logistic regression: $$\min_{w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + e^{-y_i w^\top x_i}\right) + \lambda \|w\|_2^2.$$



Loss as a function of $w^\top x$ with $y = 1$.

Supervised learning and empirical risk minimization

In supervised learning, we learn a **prediction function** $h : \mathcal{X} \to \mathcal{Y}$ given labeled training data $(x_i, y_i)_{i=1,\dots,n}$ with x_i in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

$$\min_{(w,b)\in\mathbb{R}^{p+1}} \underbrace{\frac{1}{n}\sum_{i=1}^{n} L(y_i, w^\top x_i + b)}_{\text{empirical risk, data fit}} + \underbrace{\lambda\|w\|_2^2}_{\text{regularization}} .$$

Example 1: Why the ℓ_2-regularization for linear models $h(x) = w^\top x + b$?

- Intuition: if x and x' are similar, so should $h(x)$ and $h(x')$ be:

$$|h(x) - h(x')| \leq \|w\|_2 \|x - x'\|_2.$$

- Because we have **theory** for it (and it works in practice)!

Supervised learning and empirical risk minimization

In supervised learning, we learn a **prediction function** $h : \mathcal{X} \to \mathcal{Y}$ given labeled training data $(x_i, y_i)_{i=1,\dots,n}$ with x_i in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

$$\min_{(w,b)\in\mathbb{R}^{p+1}} \underbrace{\frac{1}{n}\sum_{i=1}^{n} L(y_i, w^\top x_i + b)}_{\text{empirical risk, data fit}} + \underbrace{\lambda\|w\|_1}_{\text{regularization}} .$$

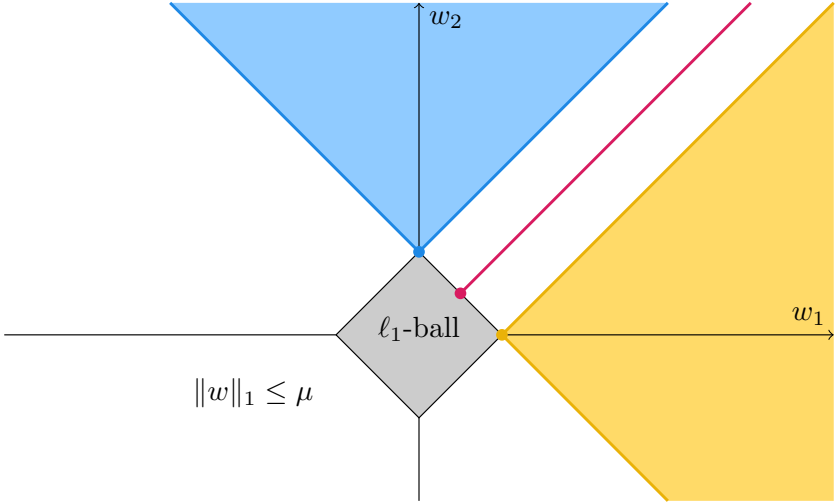
Example 1: Why the ℓ_1-regularization for linear models $h(x) = w^\top x + b$?

- Intuition: induces sparsity, encourages simple models.
- Because we have (too much) **theory** for it!

ℓ_1 and its variants lead to **composite optimization problems**.

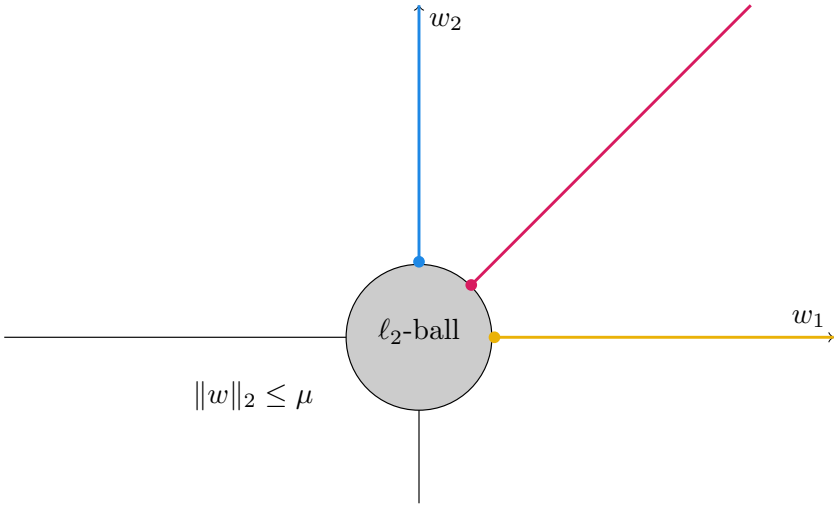
[van de Geer, 2010, Wainwright, 2009, Zhao and Yu, 2006, Candes and Tao, 2005, Chen, Donoho, and Saunders, 1999, Tibshirani, 1996, Olshausen and Field, 1996, Claerbout and Muir, 1973]...

Interlude: Why does the ℓ_1-norm induce sparsity?



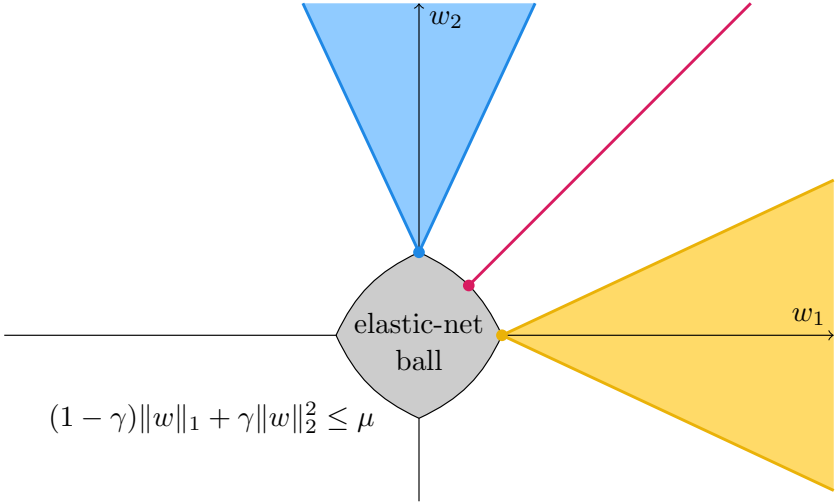
Projection onto convex sets is "biased" towards singularities.

Interlude: Why does the ℓ_1-norm induce sparsity?



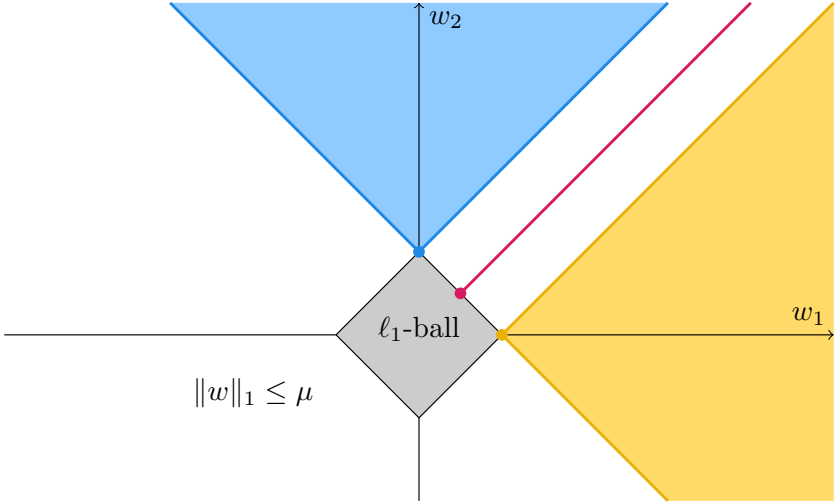
The ℓ_2-ball is isotropic.

Interlude: Why does the ℓ_1-norm induce sparsity?



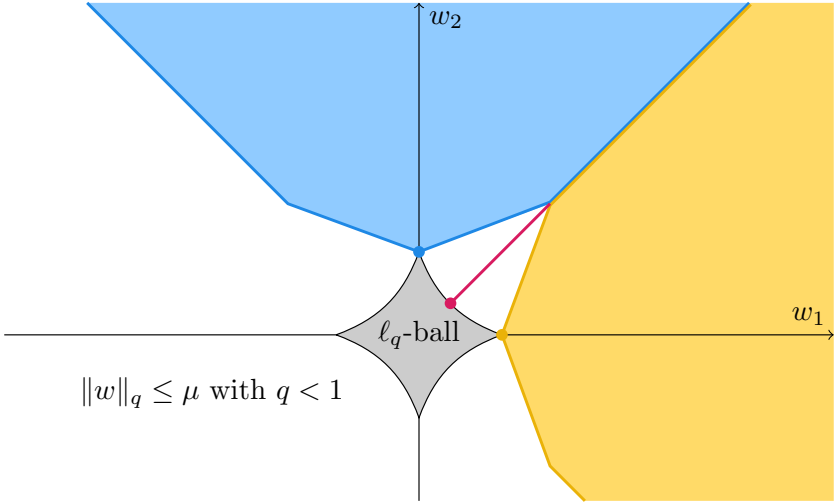
The Elastic-net penalty interpolates between ℓ_2 and ℓ_1.

Interlude: Why does the ℓ_1-norm induce sparsity?



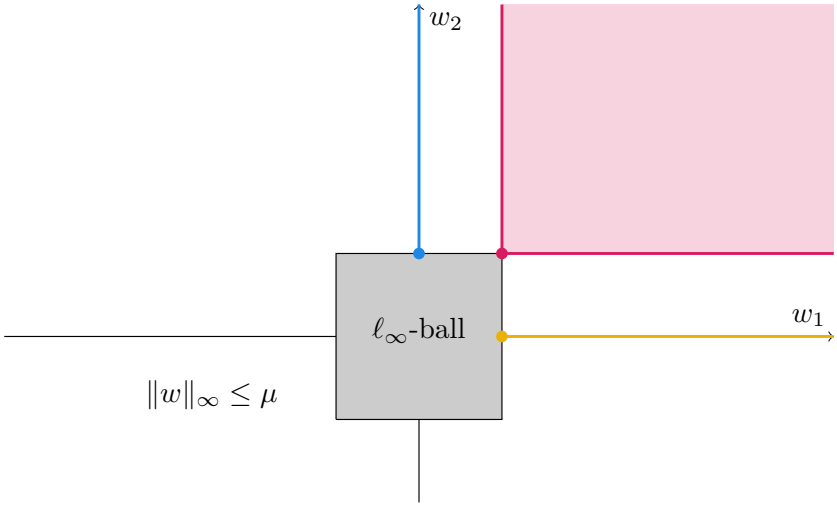
ℓ_1 again: the sparsity-inducing effect is more aggressive.

Interlude: Why does the ℓ_1-norm induce sparsity?



the sparsity-inducing effect is even more aggressive with non-convex penalties.

Interlude: Why does the ℓ_1-norm induce sparsity?

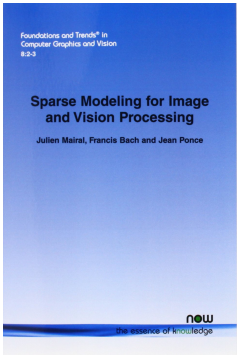


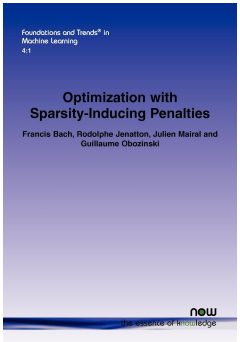
The ℓ_∞-ball encourages solutions such that $|w_1| = |w_2|$.

Material on sparse estimation (free on arXiv)

long tutorial: `http://thoth.inrialpes.fr/people/mairal/resources/pdf/BigOptim.pdf`

J. Mairal, F. Bach and J. Ponce. *Sparse Modeling for Image and Vision Processing*. Foundations and Trends in Computer Graphics and Vision. 2014.





F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. *Optimization with sparsity-inducing penalties*. Foundations and Trends in Machine Learning, 4(1). 2012.

Supervised learning and empirical risk minimization

In supervised learning, we learn a **prediction function** $h : \mathcal{X} \to \mathcal{Y}$ given labeled training data $(x_i, y_i)_{i=1,\dots,n}$ with x_i in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

$$\min_{h \in \mathcal{H}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} L(y_i, h(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \|h\|_{\mathcal{H}}^2}_{\text{regularization}} .$$

Example 2: kernel methods

- $\mathcal{H}$ is a **Hilbert** space (called RKHS) of functions;
- $\mathcal{H}$ and φ are **defined implicitly** through a positive definite kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$:
- Data points are mapped to the same Hilbert space through $\varphi : \mathcal{X} \to \mathcal{H}$;
- $h(x) = \langle h, \varphi(x) \rangle_{\mathcal{H}}$ is linear after mapping data to $\mathcal{H}$;

Supervised learning and empirical risk minimization

In supervised learning, we learn a **prediction function** $h : \mathcal{X} \to \mathcal{Y}$ given labeled training data $(x_i, y_i)_{i=1,\dots,n}$ with x_i in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

$$\min_{h \in \mathcal{H}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} L(y_i, h(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \|h\|_{\mathcal{H}}^2}_{\text{regularization}} .$$

Example 2: kernel methods.

- **Why and how?** This is a 1-slide summary of a 24-hours course on kernel methods: `https://mva-kernel-methods.github.io/course-2023-2024/`

Supervised learning and empirical risk minimization

In supervised learning, we learn a **prediction function** $h : \mathcal{X} \to \mathcal{Y}$ given labeled training data $(x_i, y_i)_{i=1,\dots,n}$ with x_i in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

$$\min_{\theta} \underbrace{\frac{1}{n}\sum_{i=1}^{n} L(y_i, h_\theta(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda\|\theta\|^2}_{\text{regularization}} .$$

Example 3: neural networks

- we parametrize h by $\theta = \{W_1, \dots, W_k\}$ as follows:

$$h_\theta(x) = \sigma_k(W_k\sigma_{k-1}(W_{k-1}\dots\sigma_1(W_1 x)).$$

- This simple model is called **multilayer perceptron**.
- Ω is called the "weight decay" parameter.

Supervised learning and empirical risk minimization

In supervised learning, we learn a **prediction function** $h : \mathcal{X} \to \mathcal{Y}$ given labeled training data $(x_i, y_i)_{i=1,\dots,n}$ with x_i in $\mathcal{X}$, and y_i in $\mathcal{Y}$:

$$\min_{\theta} \underbrace{\frac{1}{n}\sum_{i=1}^{n} L(y_i, h_\theta(x_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda\|\theta\|^2}_{\text{regularization}} .$$

Example 3: neural networks

- optimized with **variants of the stochastic gradient descent** algorithm

$$\theta_{t+1} \leftarrow (1 - 2\lambda\eta_t)\theta_t - \eta_t \nabla L(y_{i_t}, h_{\theta_t}(x_{i_t}))$$

- gradient computed by **automatic differentiation** (clever application of the chain rule).
- in practice, use **mini-batches** of data.

Statistical learning

Setting

- We draw i.i.d. pairs (x_i, y_i) from some unknown distribution P.
- The objective is to minimize over all functions the **expected risk**:

$$\min_h \left\{ R(h) = \mathbb{E}_{(x,y)\sim P}[L(y, h(x))] \right\}.$$

Statistical learning

Setting

- We draw i.i.d. pairs (x_i, y_i) from some unknown distribution P.
- The objective is to minimize over all functions the **expected risk**:

$$\min_h \left\{ R(h) = \mathbb{E}_{(x,y)\sim P}[L(y, h(x))] \right\}.$$

But

Statistical learning

Setting

- We draw i.i.d. pairs (x_i, y_i) from some unknown distribution P.
- The objective is to minimize over all functions the **expected risk**:

$$\min_h \left\{ R(h) = \mathbb{E}_{(x,y)\sim P}[L(y, h(x))] \right\}.$$

But

1. we do minimize over **a class of functions** $\mathcal{H}$ only.

Statistical learning

Setting

- We draw i.i.d. pairs (x_i, y_i) from some unknown distribution P.
- The objective is to minimize over all functions the **expected risk**:

$$\min_h \left\{ R(h) = \mathbb{E}_{(x,y)\sim P}[L(y, h(x))] \right\}.$$

But

1. we do minimize over **a class of functions** $\mathcal{H}$ only.
2. datasets are often finite and we minimize instead the **empirical risk**:

$$\min_{h\in\mathcal{H}} \left\{ R_n(h) = \frac{1}{n}\sum_{i=1}^{n}[L(y_i, h(x_i))] \right\}.$$

Statistical learning

Setting

- We draw i.i.d. pairs (x_i, y_i) from some unknown distribution P.
- The objective is to minimize over all functions the **expected risk**:

$$\min_h \left\{ R(h) = \mathbb{E}_{(x,y)\sim P}[L(y, h(x))] \right\}.$$

But

1. we do minimize over **a class of functions** $\mathcal{H}$ only.
2. datasets are often finite and we minimize instead the **empirical risk**:

$$\min_{h\in\mathcal{H}} \left\{ R_n(h) = \frac{1}{n}\sum_{i=1}^{n}[L(y_i, h(x_i))] \right\}.$$

3. we minimize **approximately**.

Statistical learning

$$\hat{h}_n \in \arg\min_{h \in \mathcal{H}} R_n(h).$$

Approximation/Estimation:

$$R(\hat{h}_n) - \min_h R(h) = \underbrace{R(\hat{h}_n) - \min_{h \in \mathcal{H}} R(h)}_{\text{estimation error}} + \underbrace{\min_{h \in \mathcal{H}} R(h) - \min_h R(h)}_{\text{approximation error}}$$

- Controlled with **regularization** (bias/variance, over/under-fitting)

Statistical learning

$$\hat{h}_n \in \arg\min_{h \in \mathcal{H}} R_n(h).$$

Approximation/Estimation/Optimization:

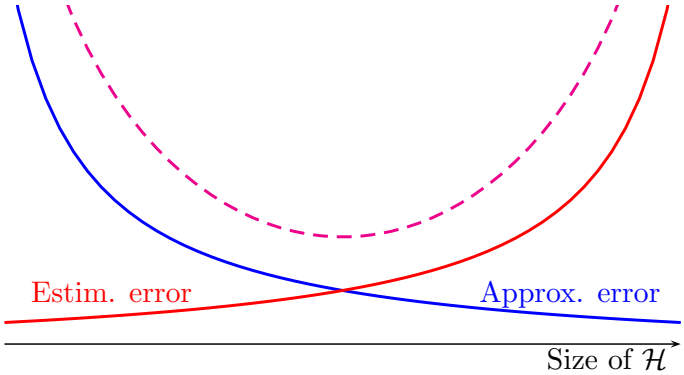
$$R(\hat{h}_n) - \min_h R(h) = \underbrace{R(\hat{h}_n) - \min_{h \in \mathcal{H}} R(h)}_{\text{estimation error}} + \underbrace{\min_{h \in \mathcal{H}} R(h) - \min_h R(h)}_{\text{approximation error}}$$

- Controlled with **regularization** (bias/variance, over/under-fitting)
- $\hat{h}_n$ is obtained **approximately** by optimization:

$$R(\tilde{h}_n) - \min_h R(h) = \underbrace{R(\tilde{h}_n) - R(\hat{h}_n)}_{\text{optimization error}} + R(\hat{h}_n) - \min_h R(h)$$

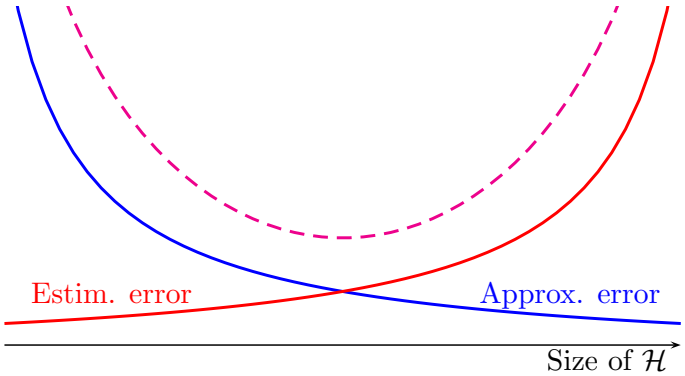
- Insight of Bottou and Bousquet (2008): **no need to optimize below statistical error!**

Statistical learning



- Illustration of the Approximation/Estimation trade-off without considering optimization cost, inspired from L. Bottou's tutorial.

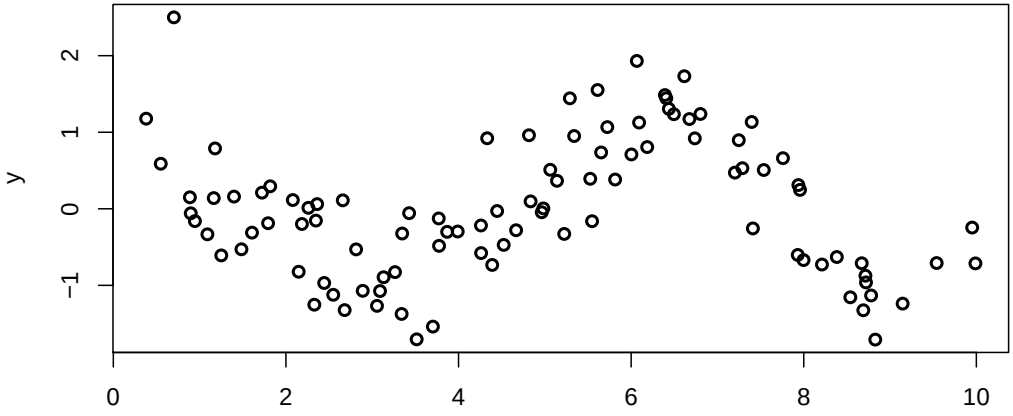
Statistical learning



- Illustration of the Approximation/Estimation trade-off without considering optimization cost, inspired from L. Bottou's tutorial.
- . . . but when optimization comes into play, things become more complicated, especially when the optimization algorithm influences the approximation error!

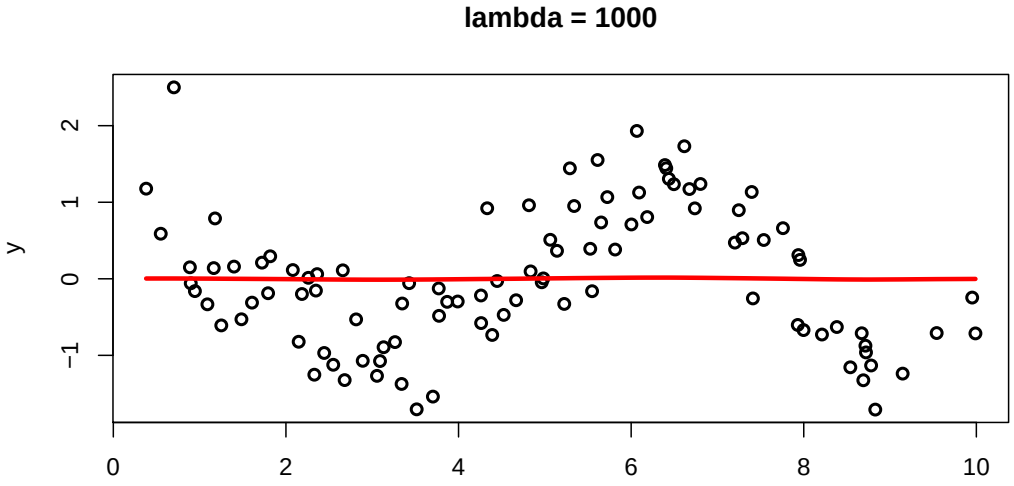
Statistical learning

Gradually increasing the size of the function class in kernel ridge regression:



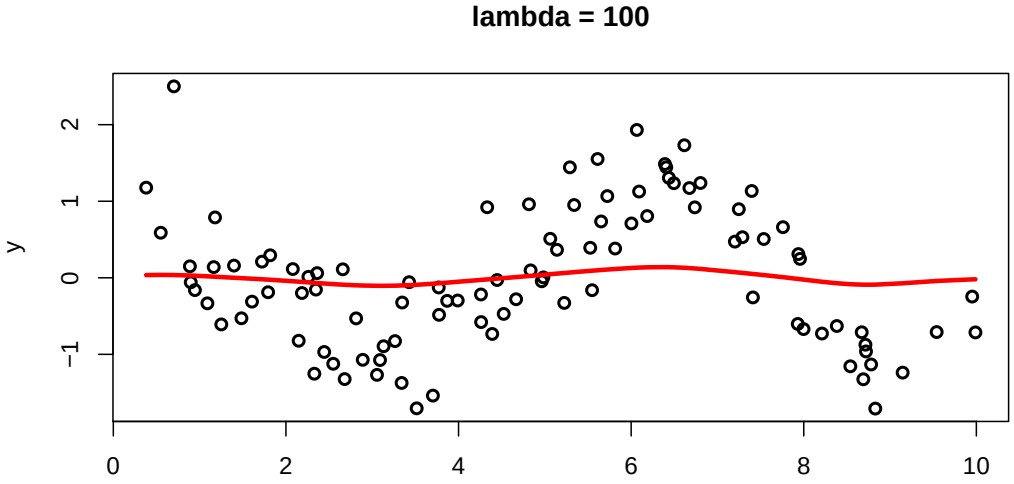
Statistical learning

Gradually increasing the size of the function class in kernel ridge regression:



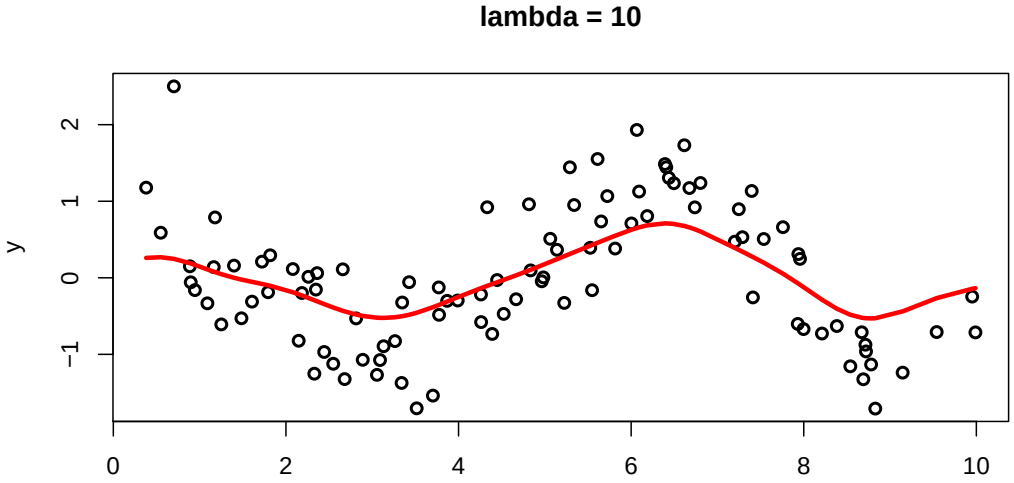
Statistical learning

Gradually increasing the size of the function class in kernel ridge regression:



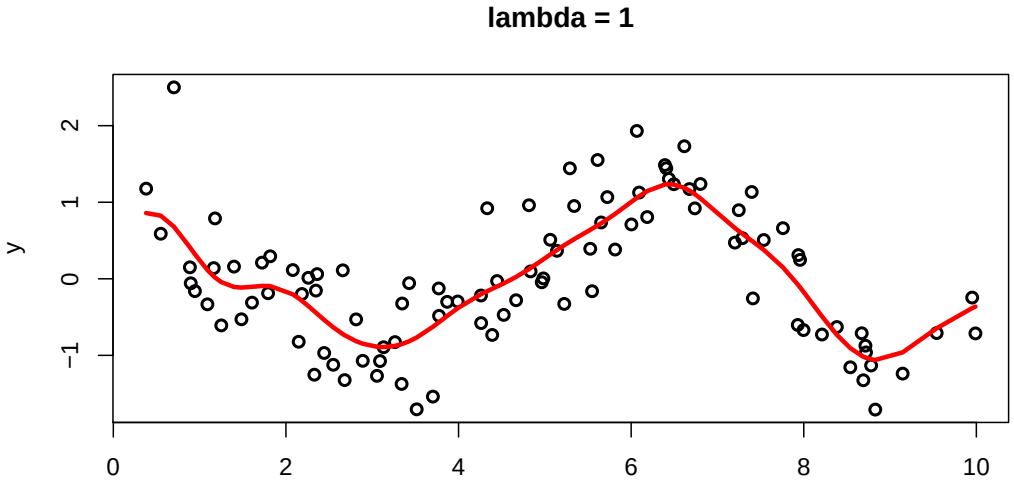
Statistical learning

Gradually increasing the size of the function class in kernel ridge regression:



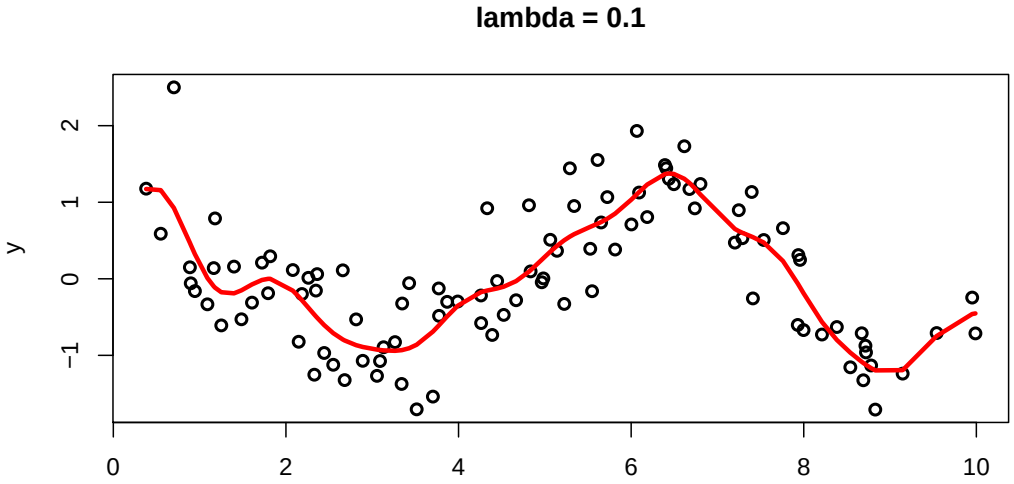
Statistical learning

Gradually increasing the size of the function class in kernel ridge regression:



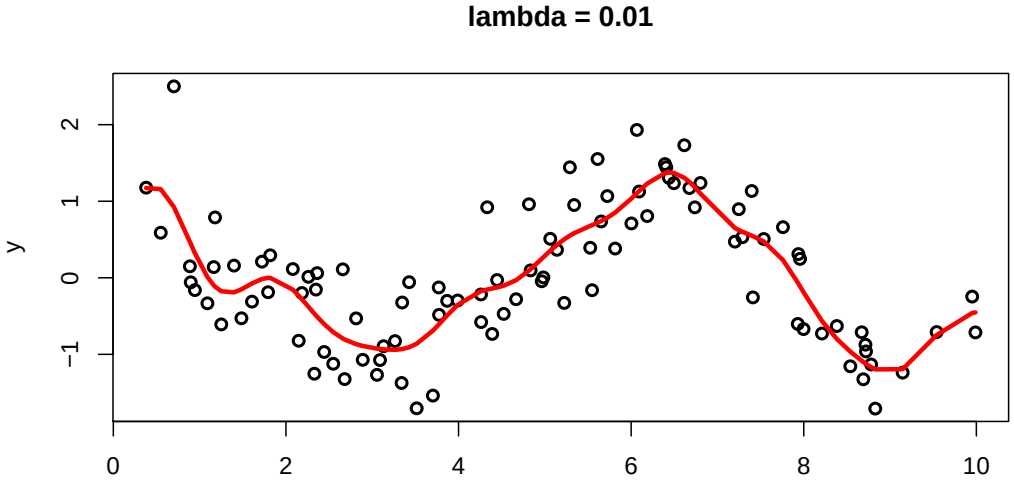
Statistical learning

Gradually increasing the size of the function class in kernel ridge regression:



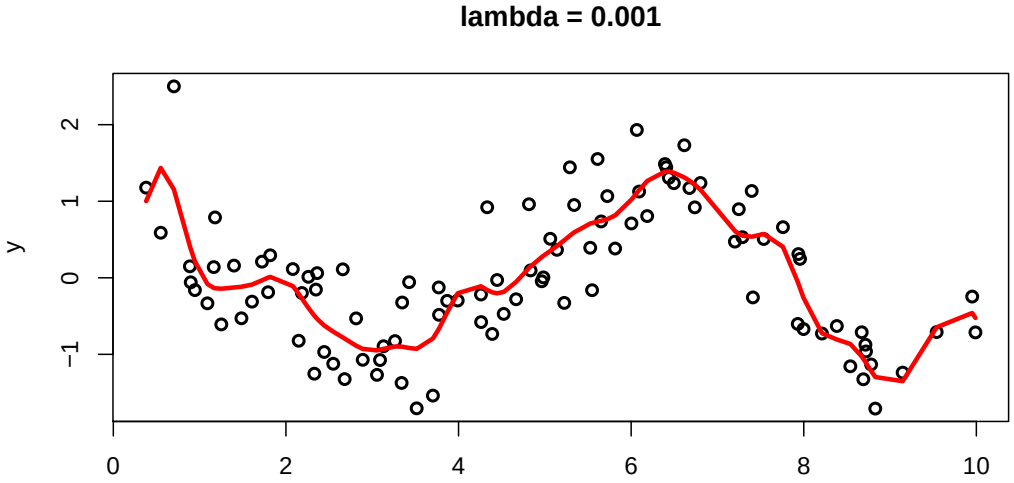
Statistical learning

Gradually increasing the size of the function class in kernel ridge regression:



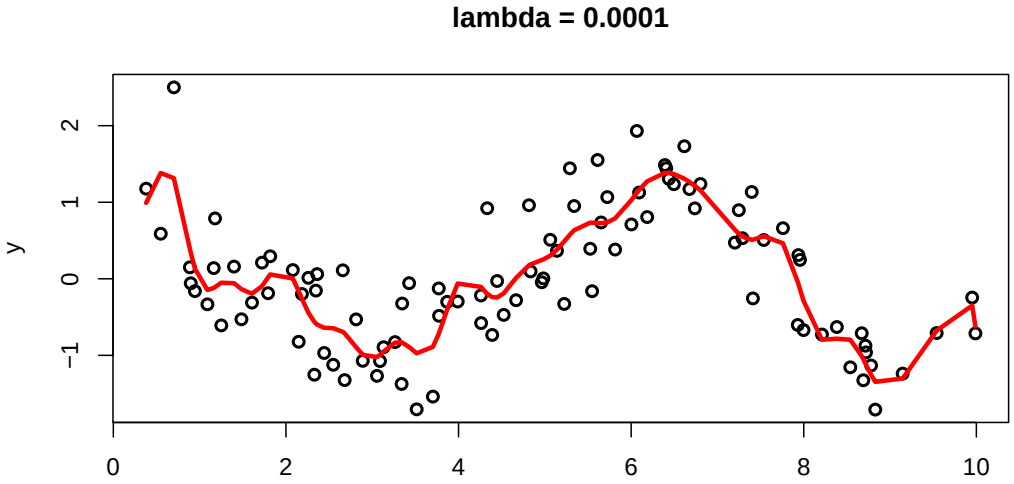
Statistical learning

Gradually increasing the size of the function class in kernel ridge regression:



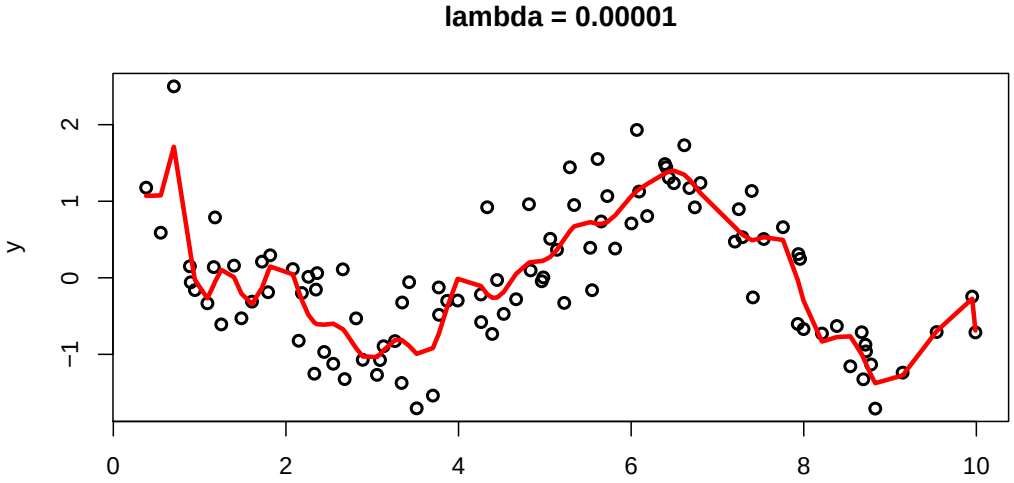
Statistical learning

Gradually increasing the size of the function class in kernel ridge regression:



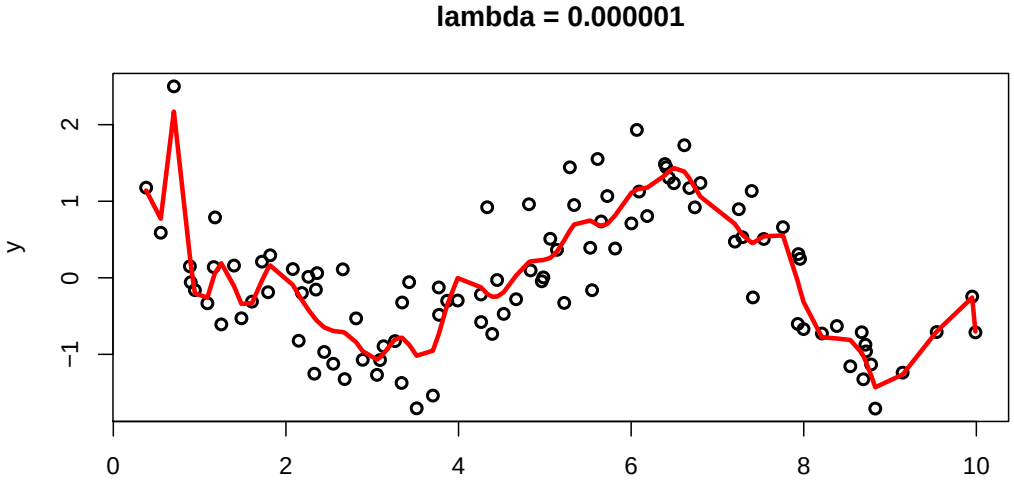
Statistical learning

Gradually increasing the size of the function class in kernel ridge regression:



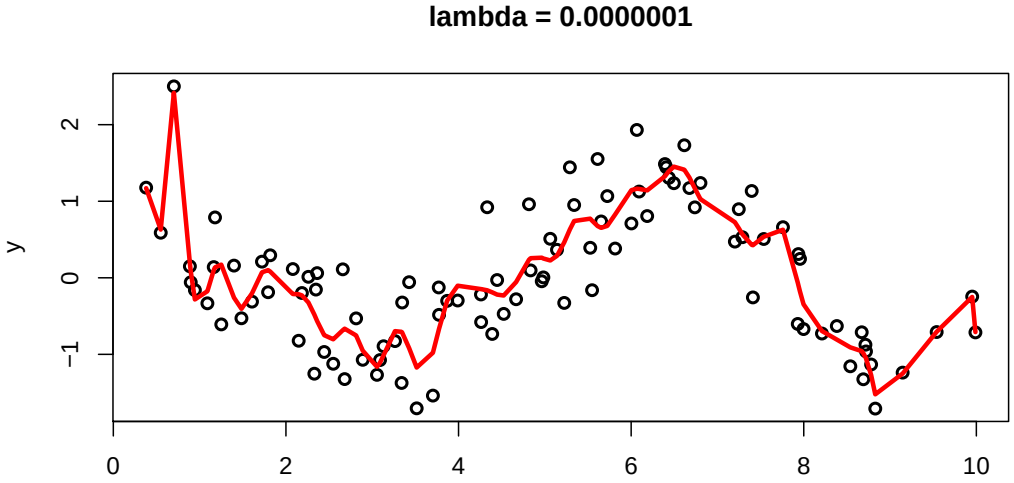
Statistical learning

Gradually increasing the size of the function class in kernel ridge regression:



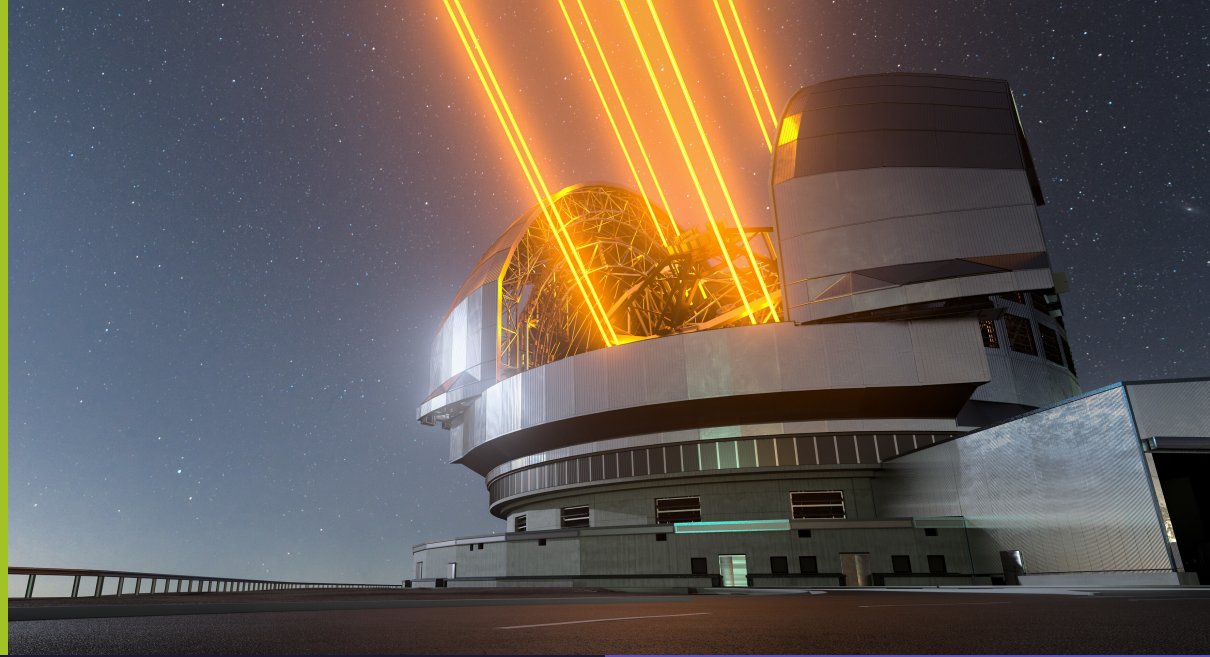
Statistical learning

Gradually increasing the size of the function class in kernel ridge regression:

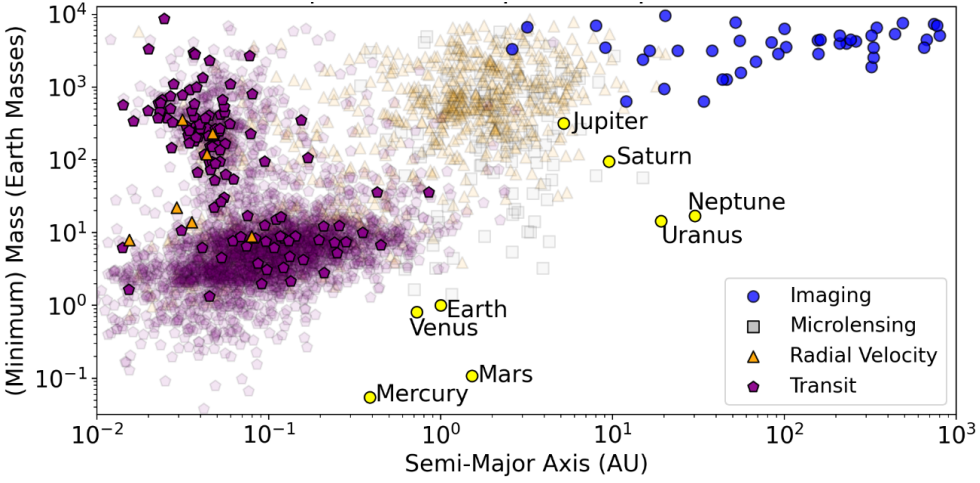


Part II: Scientific machine learning in the real world

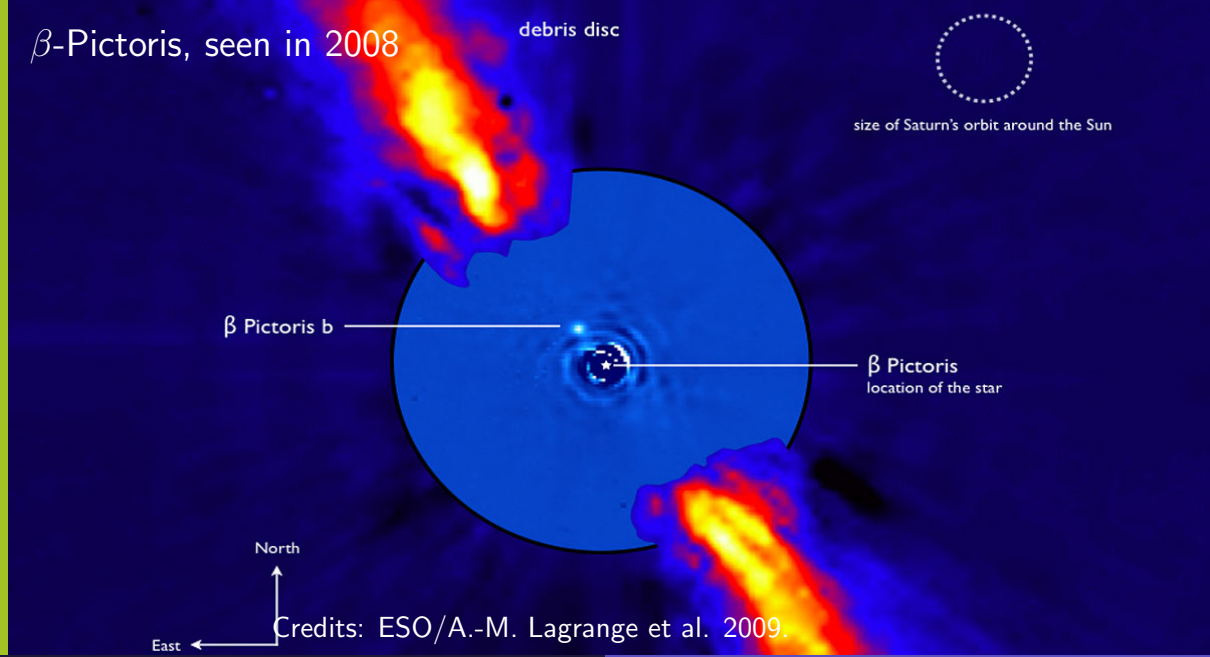
Part II: Scientific machine learning in the real world



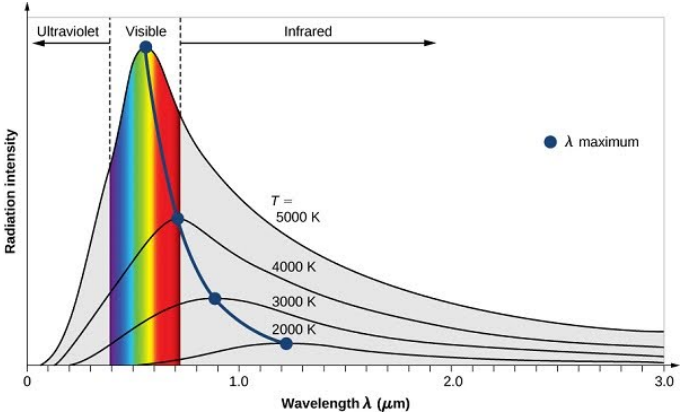
Example 1: Exoplanet detection



Credit: Dmitry Savransky, using data from the NASA Exoplanet Archive.


β-Pictoris, seen in 2008
debris disc
size of Saturn's orbit around the Sun
β Pictoris b
β Pictoris
location of the star
North
East
Credits: ESO/A.-M. Lagrange et al. 2009.

Finding a needle in a haystack . . . (contrast $\approx 10^9$ in visible light)



- Planck's law: increase λ to **increase contrast**.
- Rayleigh criterion $\Theta = 1.22\lambda/D$: decrease λ to **improve resolution**.
- near-infrared improves contrast $10^9 \Longrightarrow 10^6$.

Image Credit: `https://doi.org/10.3390/universe7080276`

A coronograph blocks light emitted by the star.

Contrast improves from 10^6 to 10^4

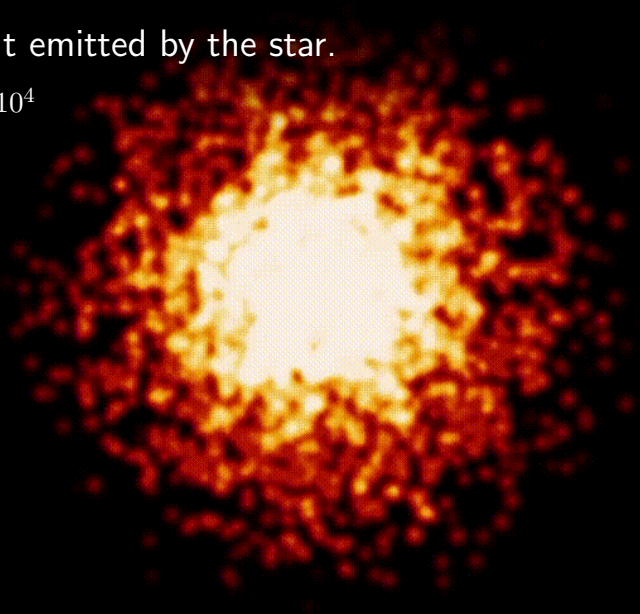
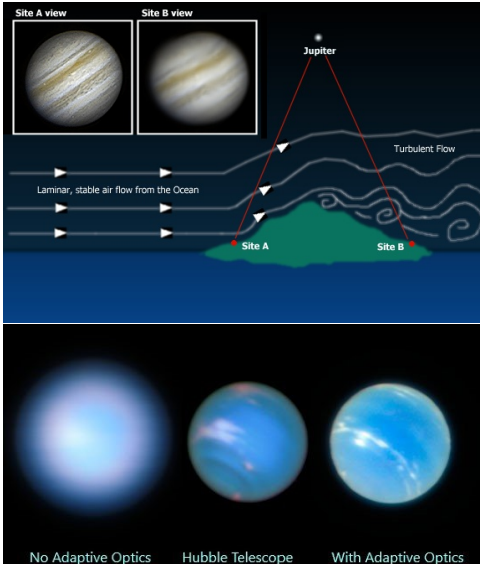
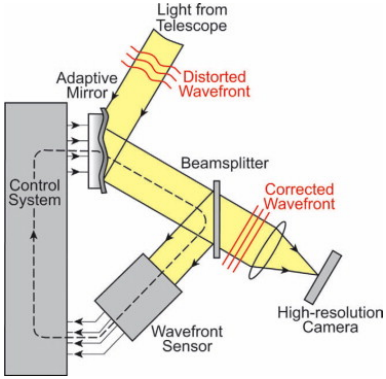


Image Credit: Nasa

Adaptive optics to mitigate atmospheric disturbances



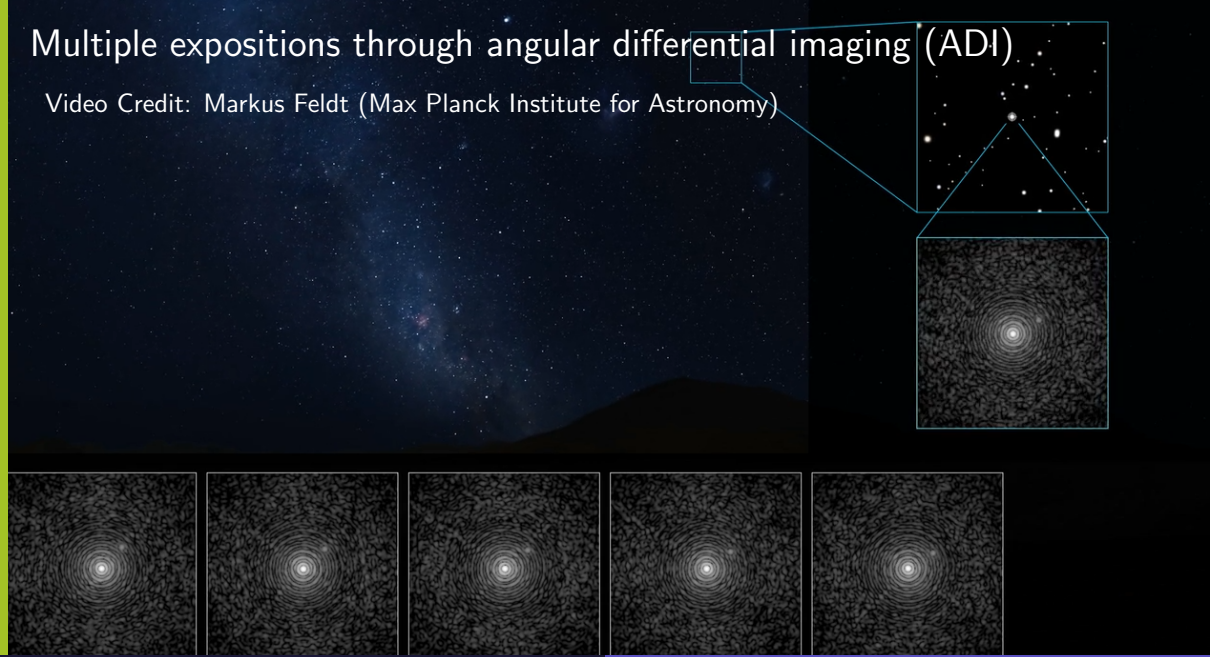


Contrast improves from 10^4 to $\approx 10^3$.

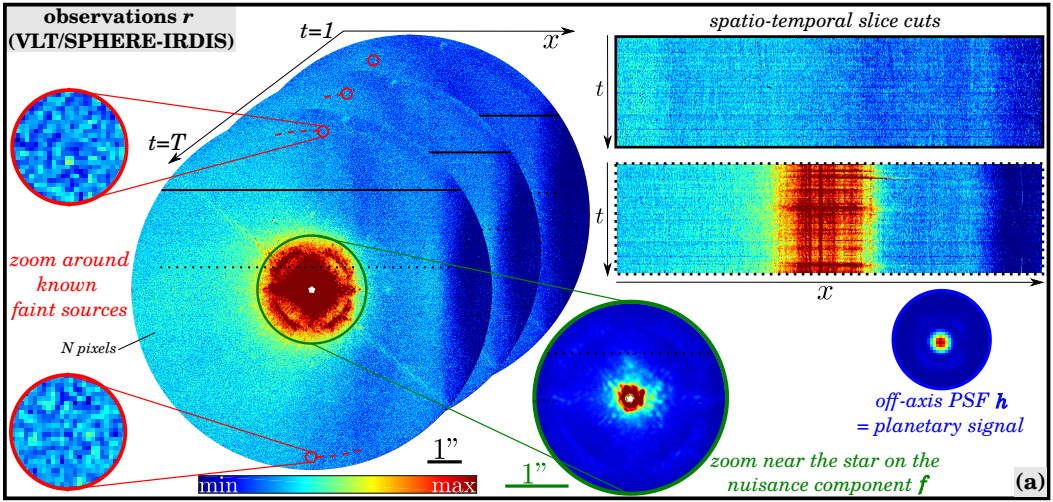
Image credit: Damian Peach and ESO

Multiple expositions through angular differential imaging (ADI)

Video Credit: Markus Feldt (Max Planck Institute for Astronomy)

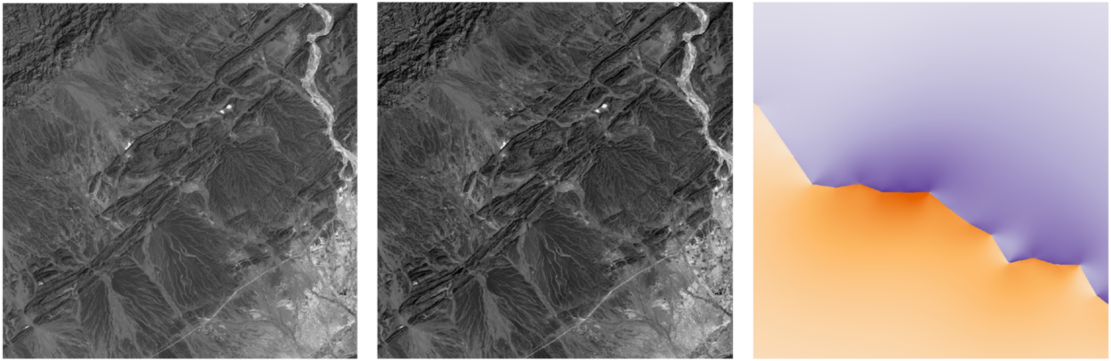


Finally: The Data



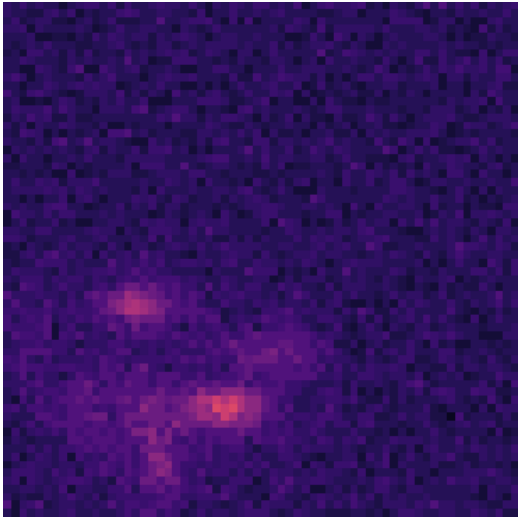
Speckles are temporally quasi-static but spatially non-stationary.

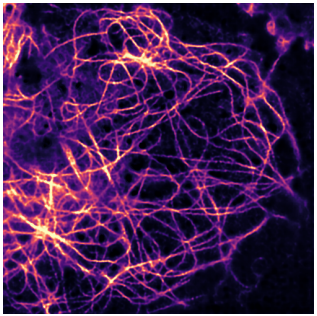
Example 2: Estimating ground deformation in seismic events

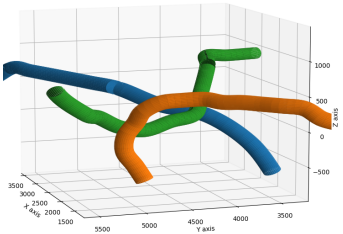


Picture from [Montagnon, Hollingsworth, Pathier, Marchandon, Dalla Mura, Giffard-Roisin, 2022].

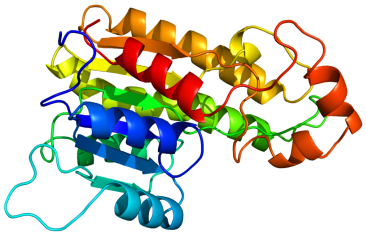
Example 3: Molecular microscopy

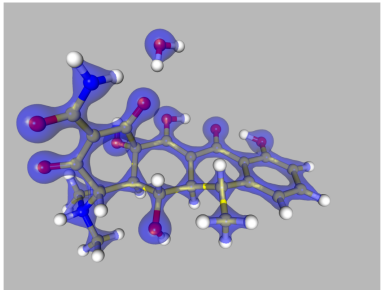


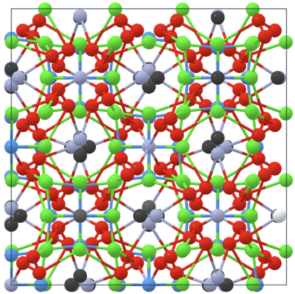




Example 4: Material science and computational biology

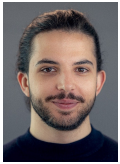






- Evaluating the properties of materials/drugs (solubility/toxicity...)
- Generating materials/drugs with desired properties.

Set of collaborators for the previous scientific applications









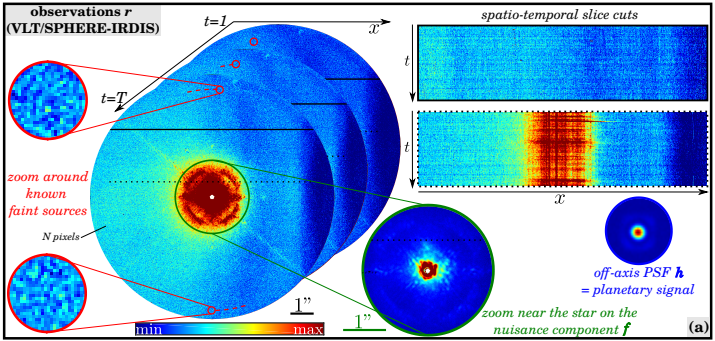

...


What could go wrong ?


What could go wrong ?
(not with the collaborators, with the previous examples)

Issue 1: where is the ground truth data?

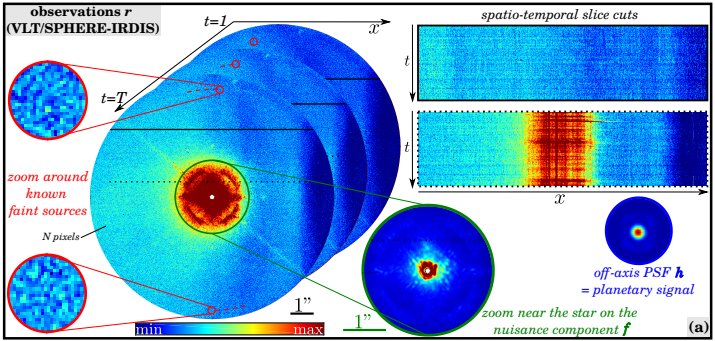
Exoplanet detection



- almost no ground truth data. . . , but we know the psf of potential sources and their apparent trajectory, which allows us to inject synthetic sources in true observations.
- we know how to remove existing sources by **temporal shuffling**. .

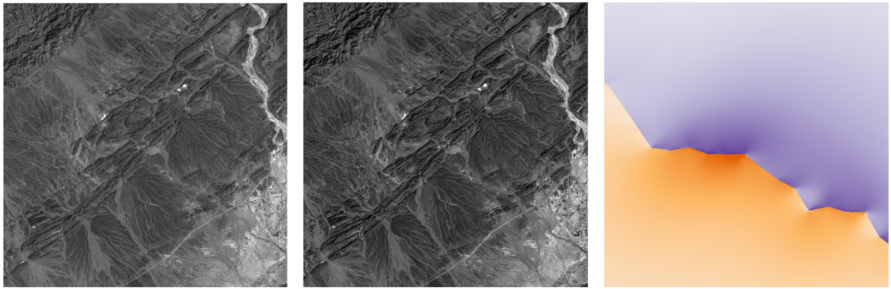
Issue 1: where is the ground truth data?

Exoplanet detection



- almost no ground truth data. . . , but we know the psf of potential sources and their apparent trajectory, which allows us to inject synthetic sources in true observations.
- need to learn with **semi-synthetic data**.

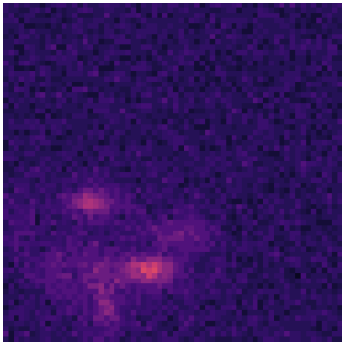
Issue 1: where is the ground truth data?

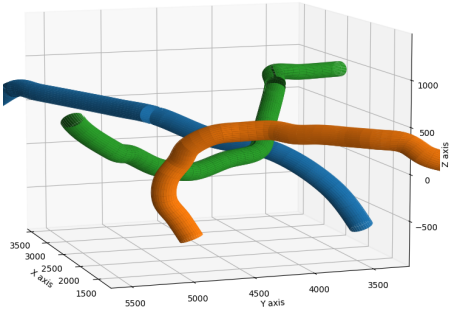


Ground deformation in seismic events

- almost no ground truth data (very sparse), but we know how to simulate seismic events from real satellites imagery. **How good is your simulator?**
- need to learn with **semi-synthetic data**.

Issue 1: where is the ground truth data?

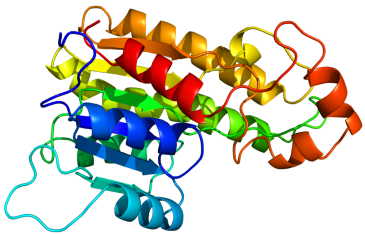


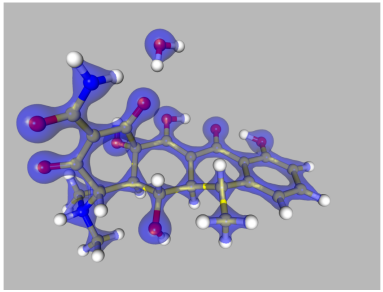


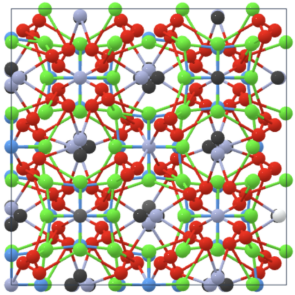
Molecular miscoscopy

- no ground truth data, but we know how to **simulate** observations from pseudo ground-truth data. **How good is your simulator?**
- need to learn with **semi-synthetic/synthetic data**.

Issue 1: where is the ground truth data?





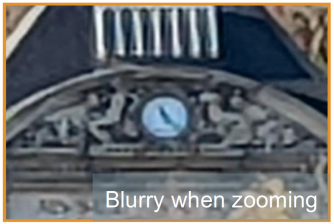


Material science

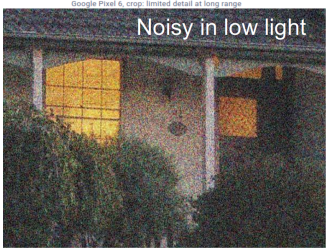
- costly approximations through quantum simulations (DFT) of proxy ground truth variables (electronic band gap).
- . . . but DFT is not a good enough approximation according to many chemists.

Issue 1: where is the ground truth data?

Google Pixel 6, long range tele



Google Pixel 6, crop: limited detail at long range



Small aperture and sensor

- Even for simple image restoration, dealing with **real-world** degradations is hard.


Learning with semi-synthetic data for real scientific applications?

Learning with semi-synthetic data

- Drawing **quantitative conclusions** on a semi-synthetic benchmark is fine. It may help **calibrating the prediction uncertainty**. Extrapolation to real data is questionable, unless the data generation process is considered very realistic.

Learning with semi-synthetic data

- Drawing **quantitative conclusions** on a semi-synthetic benchmark is fine. It may help **calibrating the prediction uncertainty**. Extrapolation to real data is questionable, unless the data generation process is considered very realistic.
- **Qualitative evaluation** on real data is important.

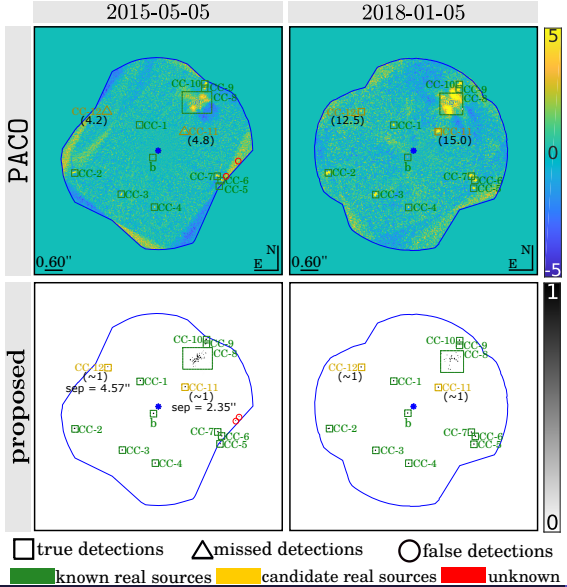
Learning with semi-synthetic data

- Drawing **quantitative conclusions** on a semi-synthetic benchmark is fine. It may help **calibrating the prediction uncertainty**. Extrapolation to real data is questionable, unless the data generation process is considered very realistic.
- **Qualitative evaluation** on real data is important.
- Quantative evaluation on real data is welcome, even on a very sparse dataset.

Learning with semi-synthetic data

- Drawing **quantitative conclusions** on a semi-synthetic benchmark is fine. It may help **calibrating the prediction uncertainty**. Extrapolation to real data is questionable, unless the data generation process is considered very realistic.
- **Qualitative evaluation** on real data is important.
- Quantative evaluation on real data is welcome, even on a very sparse dataset.
- **Making predictions** on real data is fine if the corresponding claims can be tested with further investigations.

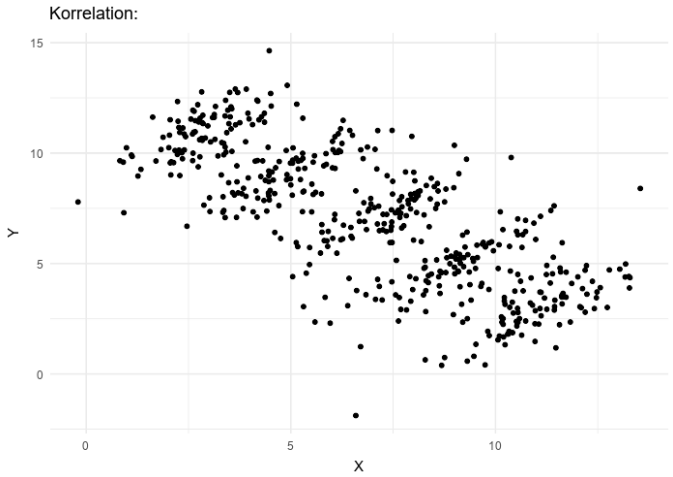
Exoplanet detection: observations of real known sources (HD 95086)



- The star with the largest number of known observed objects.
- Candidate sources are observed on several independent observations.
- Sources may be galaxies, stars, that are very far in the background, or **exoplanets**.

Issue 2: data is imperfect

Do you control the data acquisition process?



Issue 2: data is imperfect

This is an illustration of Simpsons's paradox (not that common, but disturbing).

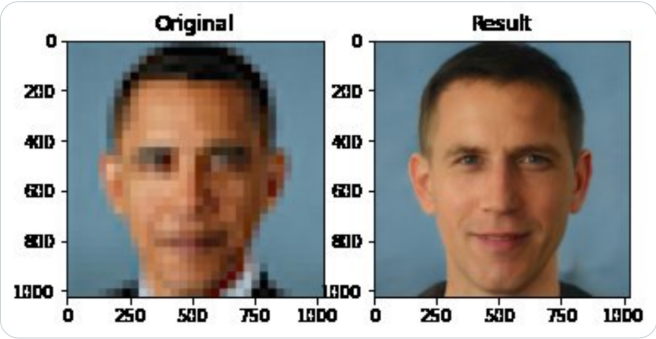
Success Based on Stone Size		
	Small Stones	**Large Stones**
Treatment A	93% (81/87)	73% (192/263)
Treatment B	87% (234/270)	69% (55/80)

Success Rate (success/total)	
Treatment A	**Treatment B**
78% (273/350)	83% (289/350)

Issue 2: data is imperfect

Non-exhaustive checklist

- Do you have **potential biases** in your data?



Issue 2: data is imperfect

Non-exhaustive checklist

- Do you have **potential biases** in your data?
- Is there **missing data**? Is the pattern of missing data random? (important example inspired from a discussion with N. Varoquaux).

Issue 2: data is imperfect

Non-exhaustive checklist

- Do you have **potential biases** in your data?
- Is there **missing data**? Is the pattern of missing data random? (important example inspired from a discussion with N. Varoquaux).
- Will you deploy the method in a **different environment** than the training one? Is this environment changing over time?

Issue 2: data is imperfect

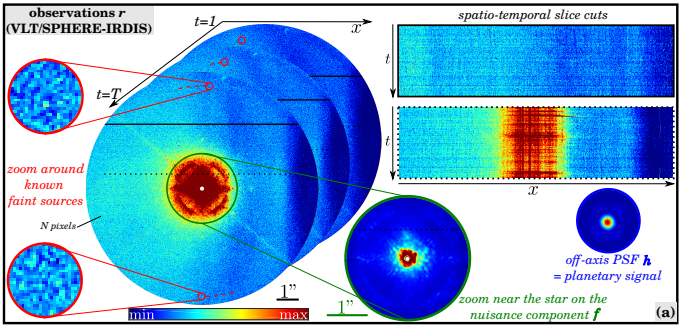
Non-exhaustive checklist

- Do you have **potential biases** in your data?
- Is there **missing data**? Is the pattern of missing data random? (important example inspired from a discussion with N. Varoquaux).
- Will you deploy the method in a **different environment** than the training one? Is this environment changing over time?

Your turn

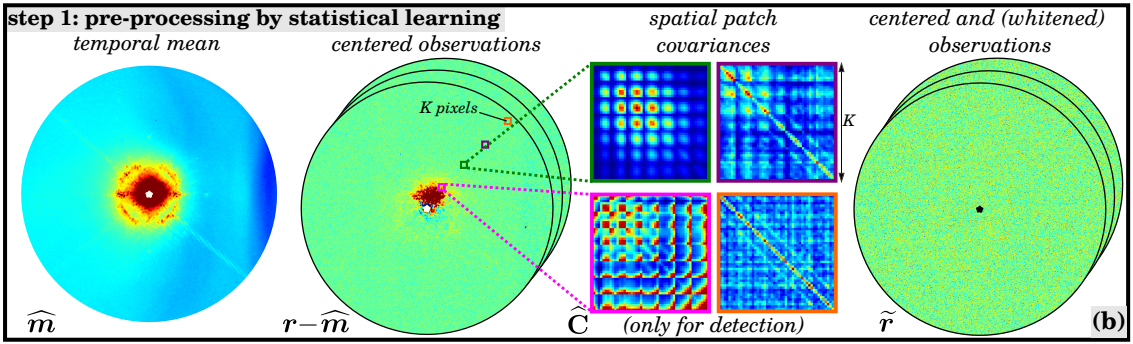
Taking physics into account: data generation

Physics can help us in the data generation process: the case of exoplanet detection



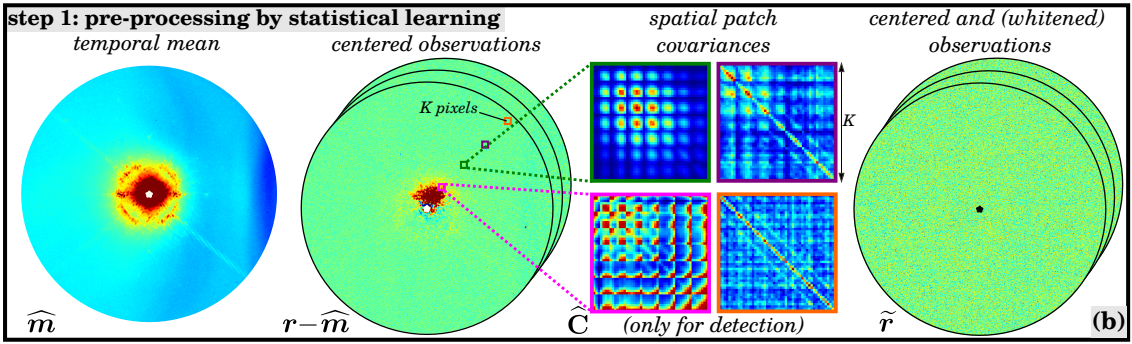
- Speckles are **temporally quasi-static** but **spatially correlated**.
- Sources are punctual with **known PSF** and **known trajectories** (due to Earth rotation).
- We are looking for **few exoplanet/sources**.

Taking physics into account: data generation



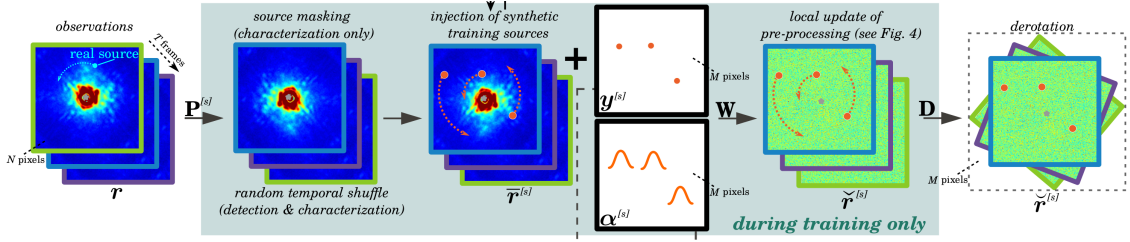
- Speckles are temporally quasi-static and sources rotate around the star (apparent motion thanks to ADI): **centering**
- Speckles are spatially correlated and sources are punctual sources (up to the PSF): **spatial decorrelation by whitening**.

Taking physics into account: data generation



- Procedure proposed in the PACO method [Flasseur et al., 2020] with a **shrinkage estimator** for covariances (good usage of statistics already).
- **Limitation**: close to the star, exoplanets do not move enough and are captured by centering and patch covariances.

Taking physics into account: data generation



- known real sources: **masking**.
- unknown real sources: **temporal shuffling**!
- good physical models of synthetic planetary PSFs: **injection**.
- known planetary apparent motion: **derotation**.

Taking physics into account: invariance

Consider a class of functions $\mathcal{H}$ and a group of transformations G. Say you want to encourage the prediction function to be invariant to G.

Data augmentation

$$\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, h(x_i)) + \lambda \Omega(h).$$

Taking physics into account: invariance

Consider a class of functions $\mathcal{H}$ and a group of transformations G. Say you want to encourage the prediction function to be invariant to G.

Data augmentation

$$\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_g[L(y_i, h(g.x_i))] + \lambda \Omega(h).$$

Taking physics into account: invariance

Consider a class of functions $\mathcal{H}$ and a group of transformations G. Say you want to encourage the prediction function to be invariant to G.

Data augmentation

$$\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_g[L(y_i, h(g.x_i))] + \lambda \Omega(h).$$

How?

- simple to implement by using SGD: at each step, draw a sample i and a random transformation g.
- nothing to do at test time.
- Examples in computer vision: geometric (translations, flips) and color transformations.
- resulting model has no invariance guarantee.

Taking physics into account: invariance

Consider a class of functions $\mathcal{H}$ and a group of transformations G. Say you want to encourage the prediction function to be invariant to G ($h(g.x) = h(x)$ for all g).

Model "layer"

- Pooling layer

$$\tilde{h}(x) = \int_{g \in G} h(g.x)$$

- Fourier/harmonic analysis. For instance, for images and $G =$ translations:

$$\tilde{h}(x) = h(|Fx|) \quad \text{modulus of the Fourier transform.}$$

Taking physics into account: invariance

Consider a class of functions $\mathcal{H}$ and a group of transformations G. Say you want to encourage the prediction function to be **equivariant to** G.

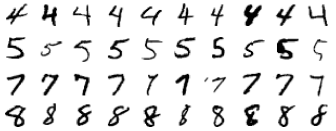
$$h(g.x) = g'.h(x).$$

For instance,

- **convolutions** are equivariant to **translations**.
- **Spherical harmonics** are equivariant to **3D rotations**.

Taking physics into account: stability (near invariance)

For more complex groups of transformations, invariance may be too much. This may be the case of **deformations** in **images** (group of diffeomorphisms).



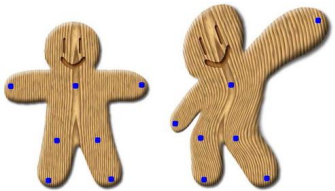
- Representation $h(\cdot)$ is **stable** [Mallat, 2012] if:

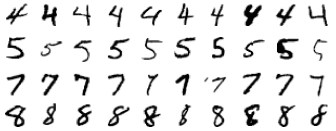
$$\|h(L_\tau x) - h(x)\| \leq (C_1 \|\nabla \tau\|_\infty + C_2 \|\tau\|_\infty)\|x\|$$

- $\|\nabla \tau\|_\infty = \sup_u \|\nabla \tau(u)\|$ controls deformation
- $\|\tau\|_\infty = \sup_u |\tau(u)|$ controls translation
- $C_2 \to 0$: translation invariance

Taking physics into account: stability (near invariance)

For more complex groups of transformations, invariance may be too much. This may be the case of **deformations** in **images** (group of diffeomorphisms).





Typically achieved by

- scattering transform [Mallat, 2012].
- **multilayer** convolutional architectures with **local pooling** [Bietti and Mairal, 2019].

Taking physics into account: physics-informed machine learning

Sometimes, we want to enforce **additional constraints** on the prediction function h.
For instance, we may want $\mathcal{D}(h) \approx 0$, where $\mathcal{D}$ is a differential operator encoding a physical law. A physics-aware objective could be

$$\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, h(x_i)) + \lambda \Omega(h) + \mu \|\mathcal{D}(h)\|.$$

Taking physics into account: physics-informed machine learning

Sometimes, we want to enforce **additional constraints** on the prediction function h.
For instance, we may want $\mathcal{D}(h) \approx 0$, where $\mathcal{D}$ is a differential operator encoding a physical law. A physics-aware objective could be

$$\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, h(x_i)) + \lambda \Omega(h) + \mu \|\mathcal{D}(h)\|.$$

Or, we may want $h(x)$ to be the solution of an auxiliary optimization problem:

$$\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, h(x_i)) + \lambda \Omega(h) \quad \text{s.t.} \quad h(x_i) \in \arg\min_{z} g(x_i, z),$$

which is called a **bilevel** optimization problem.

Taking physics into account: Summary

Take-home message

- improves the data quality (*e.g.*, better SNR, fewer confounding factors)
- better efficiency: **no need to learn what we already know about the problem**.
- better generalization and robustness: **physics law hold across data distributions**

This will of course not solve all of potential issues about imperfect data.

Taking physics into account: Summary

Take-home message

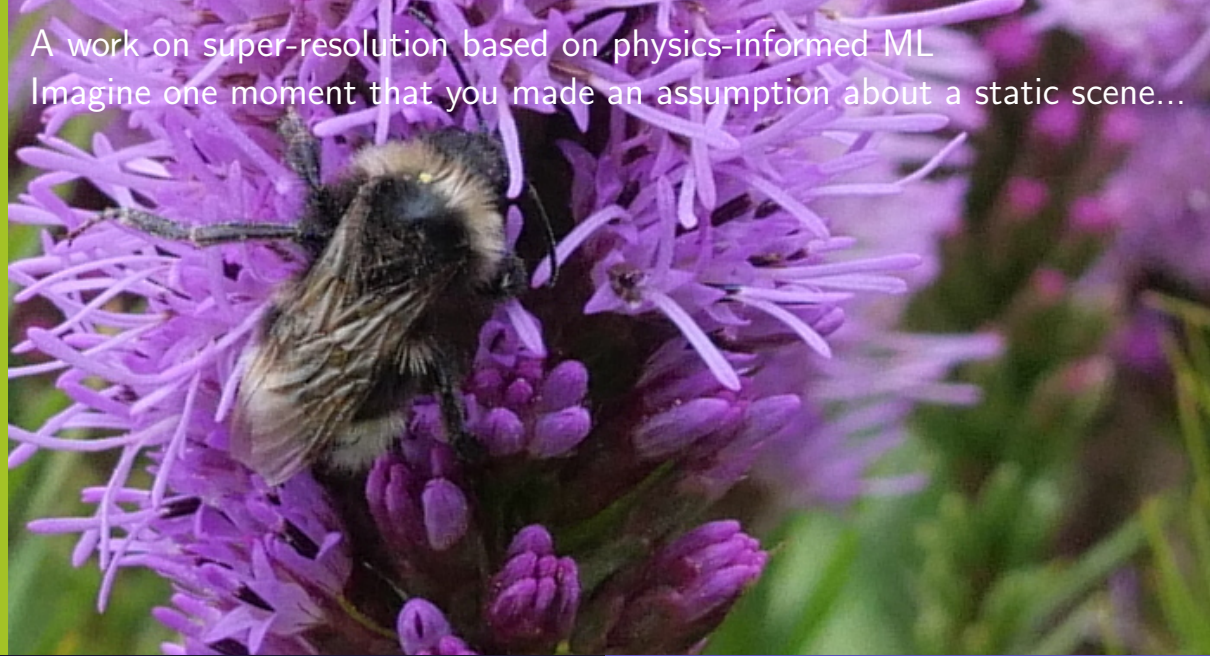
- improves the data quality (*e.g.*, better SNR, fewer confounding factors)
- better efficiency: **no need to learn what we already know about the problem**.
- better generalization and robustness: **physics law hold across data distributions**

This will of course not solve all of potential issues about imperfect data.

but **be careful about wrong/simplistic physical models!**

A work on super-resolution based on physics-informed ML

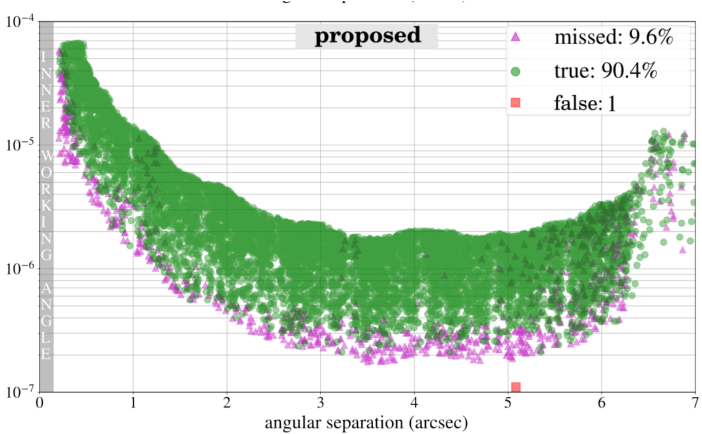
Imagine one moment that you made an assumption about a static scene...



Ooops

Solution: better model

Issue 3: calibration and the need to quantify uncertainty

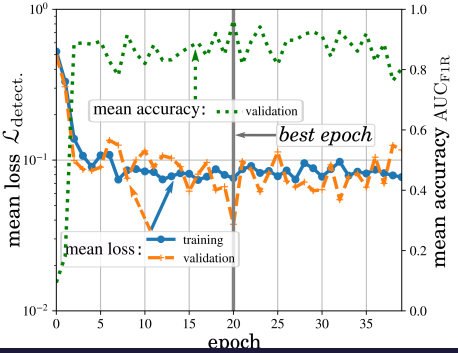


- Identify all **sources of randomness** in your process.
- Learn about **statistical testing, bootstrap, confidence intervals**...

Issue 3: calibration and the need to quantify uncertainty

Unusual features for (deep) machine learners

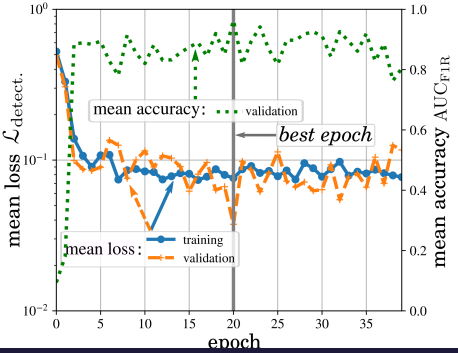
- only possible to inject a small number of sources in a dataset, but we need **thousands** ($\approx 10k$) to obtain meaningful ROC curves and calibrate the model.
- $\Longrightarrow$ repetition of independent evaluations: need to train hundreds/thousands of deep learning models.



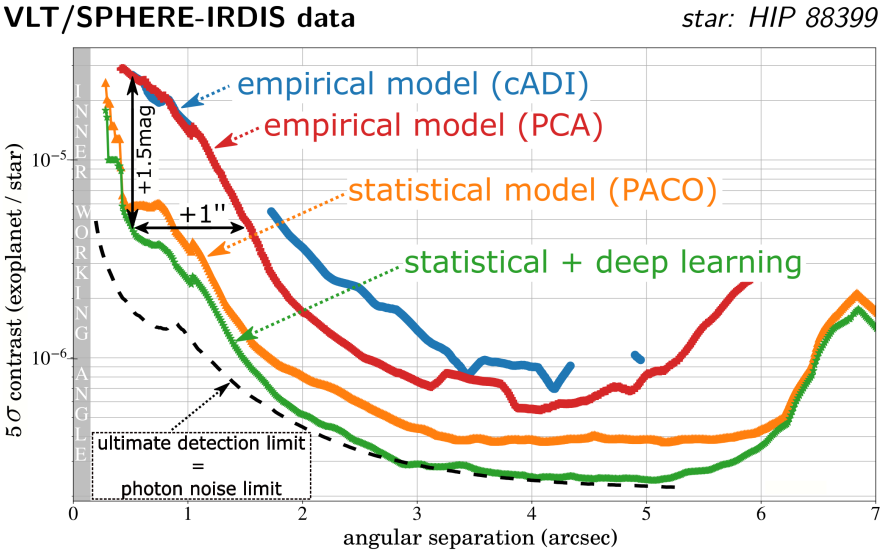
Issue 3: calibration and the need to quantify uncertainty

Unusual features for (deep) machine learners

- only possible to inject a small number of sources in a dataset, but we need **thousands** ($\approx 10k$) to obtain meaningful ROC curves and calibrate the model.
- $\Longrightarrow$ repetition of independent evaluations: need to train hundreds/thousands of deep learning models with **no manual intervention**.



Taking physics into account: knowing physical limits



Issue 4: sometimes, we need to adjust theory

https://x.com/daniela_witten/status/1292293102103748609

Daniela Witten
@daniela_witten

The Bias-Variance Trade-Off & "DOUBLE DESCENT" 🧵

Remember the bias-variance trade-off? It says that models perform well for an "intermediate level of flexibility". You've seen the picture of the U-shape test error curve.

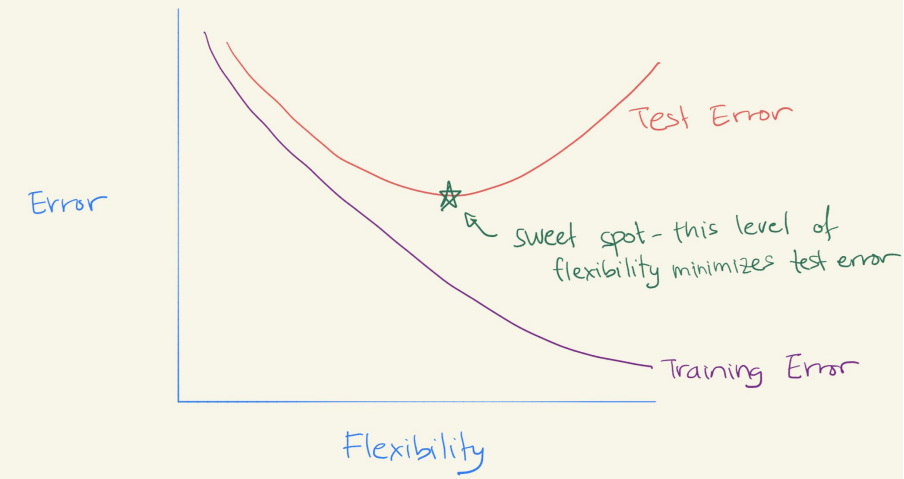
We try to hit the "sweet spot" of flexibility.

1/🧵

Traduire le post

Issue 4: sometimes, we need to adjust theory

https://x.com/daniela_witten/status/1292293102103748609



Issue 4: sometimes, we need to adjust theory

https://x.com/daniela_witten/status/1292293102103748609

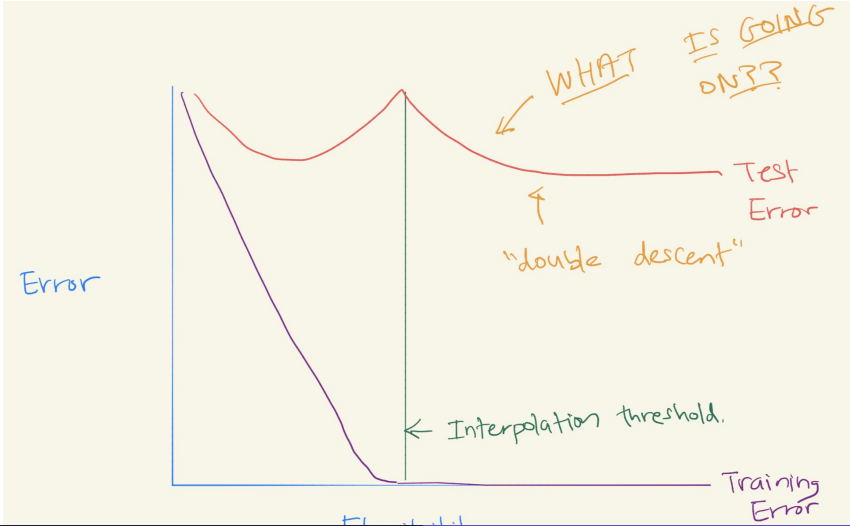
Daniela Witten @daniela_witten · 9 août 2020

In the past few yrs, (and particularly in the context of deep learning) ppl have noticed "double descent" -- when you continue to fit increasingly flexible models that interpolate the training data, then the test error can start to DECREASE again!!

Check it out:
3/

Issue 4: sometimes, we need to adjust theory

https://x.com/daniela_witten/status/1292293102103748609



Issue 4: sometimes, we need to adjust theory

What is going on?

- if **you do not regularize (enough)**, when increasing the model size, you start **interpolating** the dataset—that is, $h(x_i) = y_i$ for all $i = 1, \ldots, n$.

Issue 4: sometimes, we need to adjust theory

What is going on?

- if **you do not regularize (enough)**, when increasing the model size, you start **interpolating** the dataset—that is, $h(x_i) = y_i$ for all $i = 1, \ldots, n$.
- then, the problem admits multiple solutions (**overparametrized** regime). Among these solutions, your learning algorithm may **implicitly** select a good one.

Issue 4: sometimes, we need to adjust theory

What is going on?

- if **you do not regularize (enough)**, when increasing the model size, you start **interpolating** the dataset—that is, $h(x_i) = y_i$ for all $i = 1, \ldots, n$.
- then, the problem admits multiple solutions (**overparametrized** regime). Among these solutions, your learning algorithm may **implicitly** select a good one.
- Then, a natural formulation to **select** a solution would be

$$\min_{h \in \mathcal{H}} \Omega(h) \quad \text{s.t.} \quad \forall i = 1, \ldots, n, \qquad h(x_i) = y_i,$$

where Ω is a "**norm**" characterizing good solutions (smooth, robust...).

Issue 4: sometimes, we need to adjust theory

What is going on?

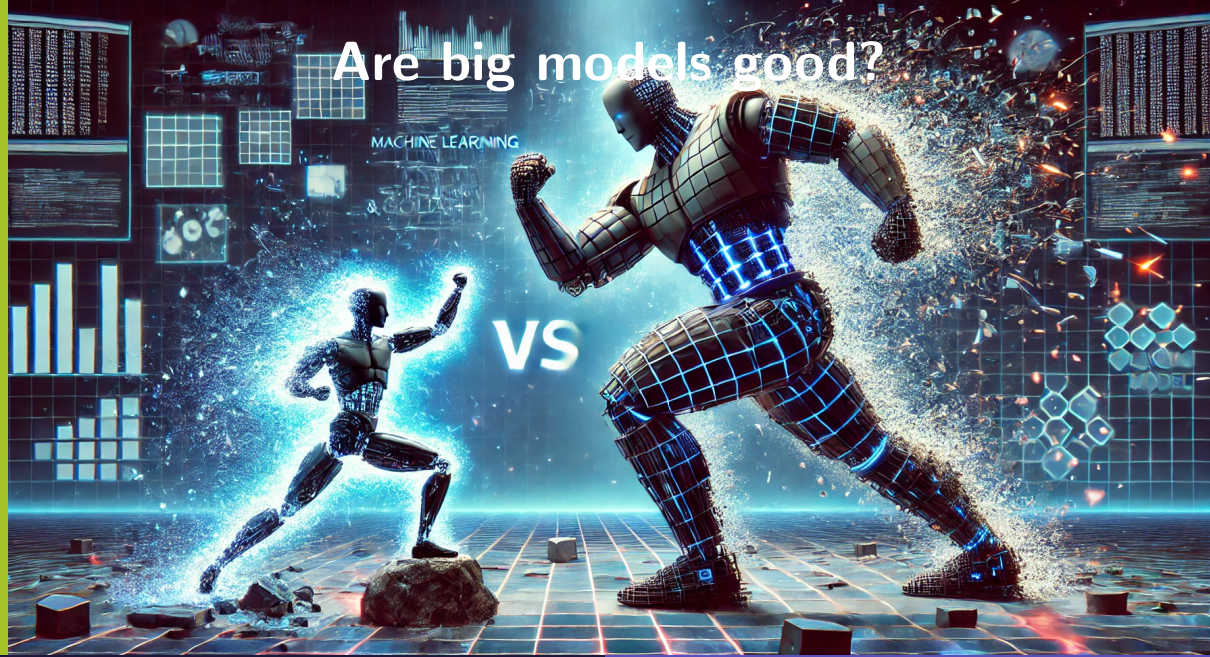
- if **you do not regularize (enough)**, when increasing the model size, you start **interpolating** the dataset—that is, $h(x_i) = y_i$ for all $i = 1, \ldots, n$.
- then, the problem admits multiple solutions (**overparametrized** regime). Among these solutions, your learning algorithm may **implicitly** select a good one.
- Then, a natural formulation to **select** a solution would be

$$\min_{h \in \mathcal{H}} \Omega(h) \quad \text{s.t.} \quad \forall i = 1, \ldots, n, \qquad h(x_i) = y_i,$$

  where Ω is a "**norm**" characterizing good solutions (smooth, robust...).
- Wait... Increasing the model class $\mathcal{H}$ (larger models) potentially yields smaller norm solutions (smoother, more robust)???

Are big models good?

Are big models good?



Are big models good?

Are big models good?

Yes they can be good

- This is a trivial statement: what you can do with p parameters, you can do with $p+1$.
- Successful stories in computer vision and NLP about "big" models.
- Generic robustness requires large models [Bubeck and Selke, 2021].

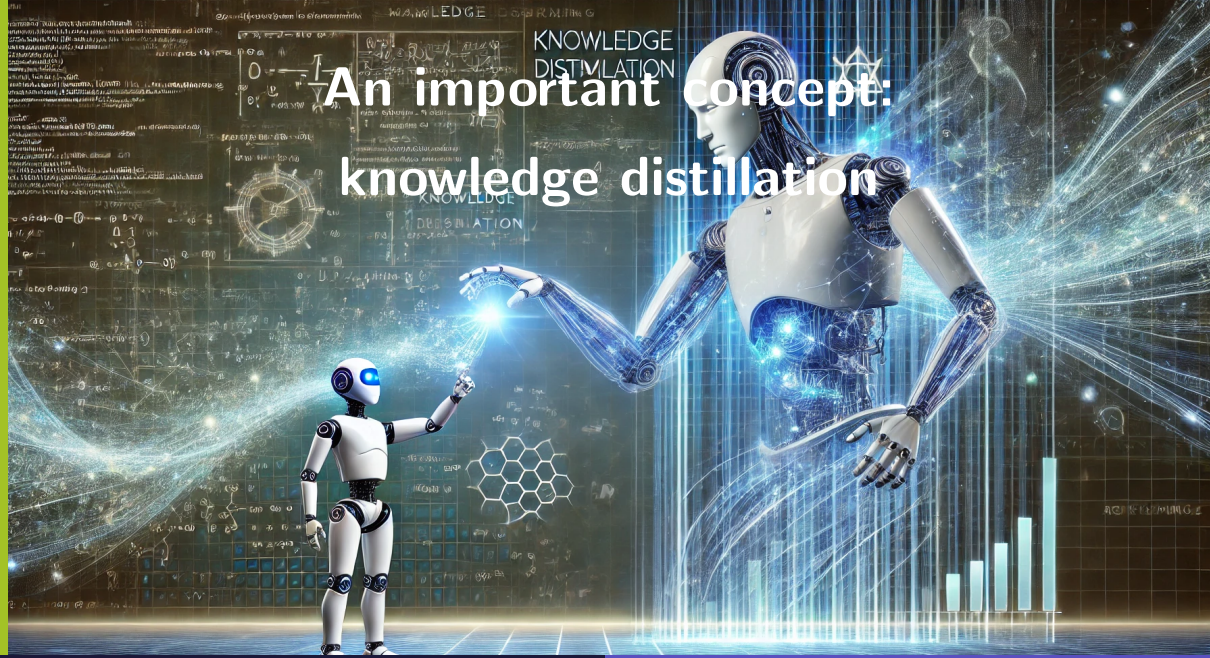
Are big models good?

Yes they can be good

- This is a trivial statement: what you can do with p parameters, you can do with $p+1$.
- Successful stories in computer vision and NLP about "big" models.
- Generic robustness requires large models [Bubeck and Selke, 2021].

But

- We do not control precisely the previous "norm" Ω; we may even not know what it is exactly for current deep learning models.
- This means that for specific problems, big is not necessarily better than small.
- Even when it is better, is it worth it? (diminishing returns).



An important concept: knowledge distillation

An important concept: knowledge distillation

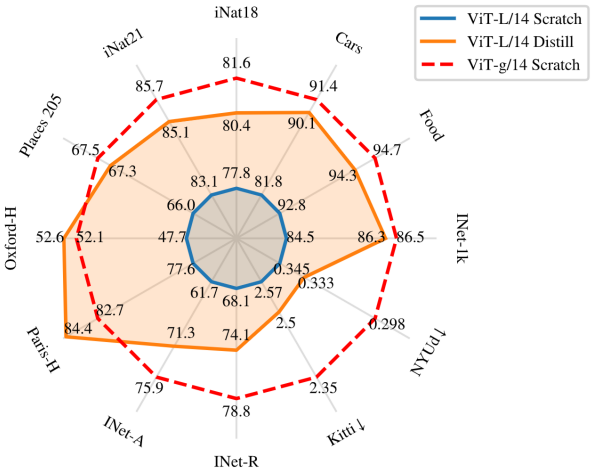
Simple technique introduced by Hinton et al. [2015] to transfer knowledge from a large "**teacher**" model to a smaller one (the **student**).

$$\min_{h_s \in \mathcal{H}_s} \frac{1}{n} \sum_{i=1}^{n} L(h_s(x_i), h_t(x_i))$$

- $h_s(x)$ and $h_t(x)$ often represent **logits** for classification problems.
- can leverage large amounts of unlabeled data.

An important concept: knowledge distillation

From DINOv2 [Oquab et al., 2024]

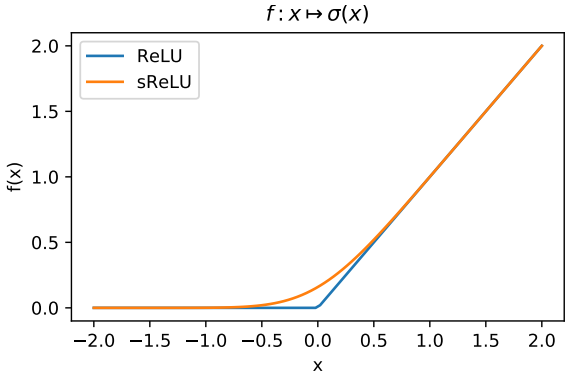


Part III: A few deep learning models

A single neuron

- Input data x lives in $\mathbb{R}^p$.
- A neuron computes $h(x)$ in $\mathbb{R}$ with p parameters:

$$h(x) = \sigma(w^\top x + b).$$

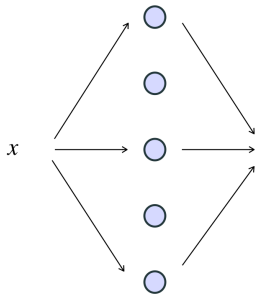


A simple neural network

- Input data x lives in $\mathbb{R}^p$.
- A collection of neurons computes $h(x)$ in $\mathbb{R}$:

$$h(x) = w_2^\top \sigma(W_1 x + b).$$

- With enough neurons, we can already approximate any continuous function! (up to mild technical assumptions).

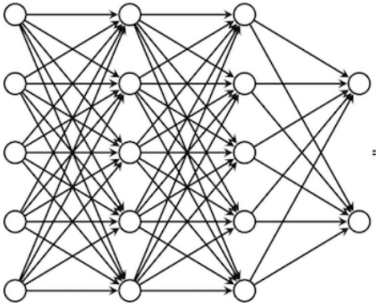


Multilayer perceptron

- Input data x lives in $\mathbb{R}^p$.
- We parametrize h by $\theta = \{W_1, \ldots, W_k\}$ as follows:

$$h_\theta(x) = W_{k+1}\sigma_k(W_k\sigma_{k-1}(W_{k-1}\ldots\sigma_1(W_1x)).$$

We have simplified the notation by removing the optional biases b_is.

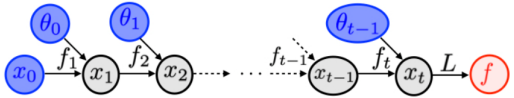


Backpropagation

Gabriel Peyré ✓
@gabrielpeyre

Back-propagation for feed-forward architectures is a special case of reverse-mode automatic differentiation. It computes the gradient of a loss function with a cost comparable to the computation of the loss function itself. en.wikipedia.org/wiki/Backpropa...

Traduire le post



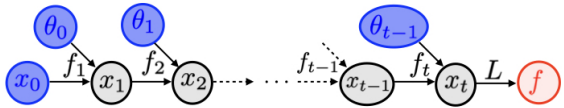
forward

$$\begin{aligned}&\text{for } k = 1, \dots, t-1, t\\ &\quad x_k = f_k(x_{k-1}, \theta_{k-1})\\ &f(\theta) \overset{\text{def.}}{=} L(x_t)\end{aligned}$$

backward

$$\begin{aligned}&\nabla_{x_t} f = \nabla L(x_t)\\ &\text{for } k = t, t-1, \dots, 1\\ &\quad \nabla_{x_{k-1}} f = [\partial_x f_k(x_{k-1}, \theta_{k-1})]^\top \nabla_{x_k} f\\ &\quad \nabla_{\theta_{k-1}} f = [\partial_\theta f_k(x_{k-1}, \theta_{k-1})]^\top (\nabla_{x_k} f)\end{aligned}$$

Backpropagation



forward

$$\texttt{for } k = 1, \dots, t-1, t$$
$$\quad x_k = f_k(x_{k-1}, \theta_{k-1})$$
$$f(\theta) \stackrel{\text{def.}}{=} L(x_t)$$

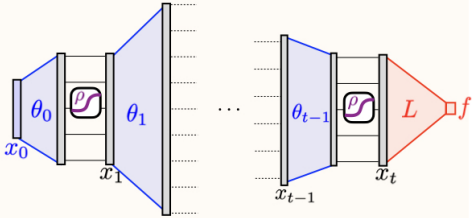
backward

$$\nabla_{x_t} f = \nabla L(x_t)$$
$$\texttt{for } k = t, t-1, \dots, 1$$
$$\quad \nabla_{x_{k-1}} f = [\partial_x f_k(x_{k-1}, \theta_{k-1})]^\top \nabla_{x_k} f$$
$$\quad \nabla_{\theta_{k-1}} f = [\partial_\theta f_k(x_{k-1}, \theta_{k-1})]^\top (\nabla_{x_k} f)$$

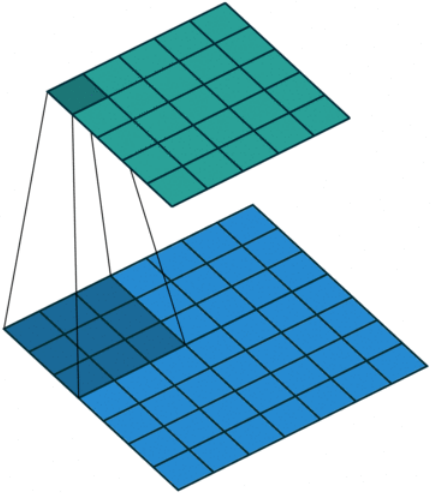
Example: perceptrons

$$f_k(x_{k-1}, \theta_{k-1}) = \rho(\theta_{k-1} x_{k-1})$$

$$[\partial_\theta f_k(x, \theta)]^\top z = w x^\top$$
$$[\partial_x f_k(x, \theta)]^\top z = \theta^\top (w \odot z)$$
$$w \stackrel{\text{def.}}{=} \rho'(Wx)$$

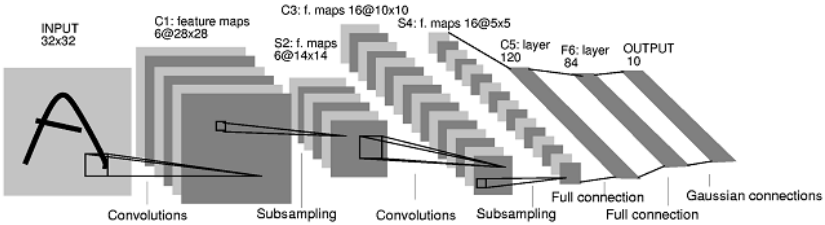


Convolutional Neural Networks



Convolutional Neural Networks

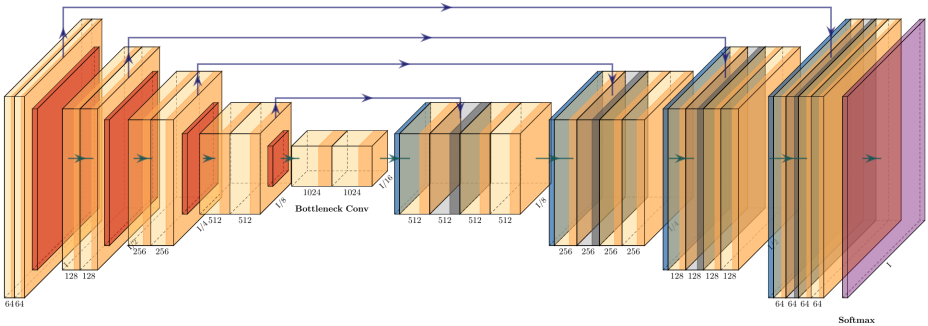
Picture from LeCun et al. [1998]



What are the main features of CNNs?

- they capture **compositional** and **multiscale** structures in images;
- they provide some **invariance**;
- they model **local stationarity** of images at several scales;
- they are **state-of-the-art** in many fields.

U-Net

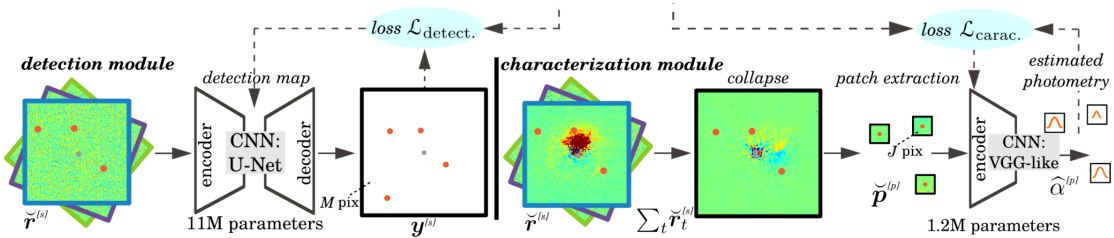


Applications

- image restoration (denoising, super-resolution).
- semantic segmentation.

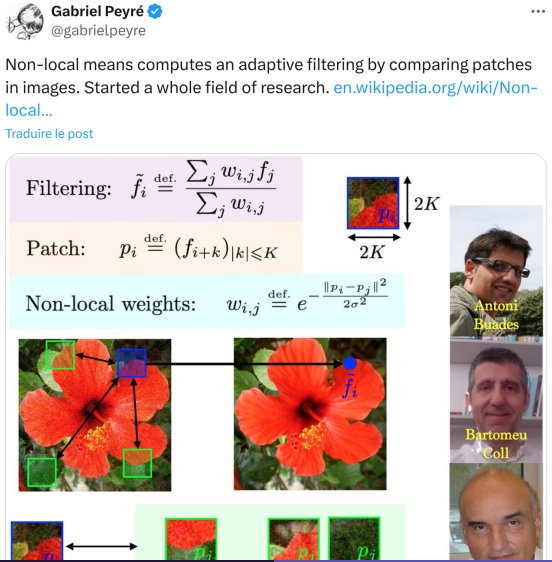
U-Net

Remember the exoplanet detection problem:

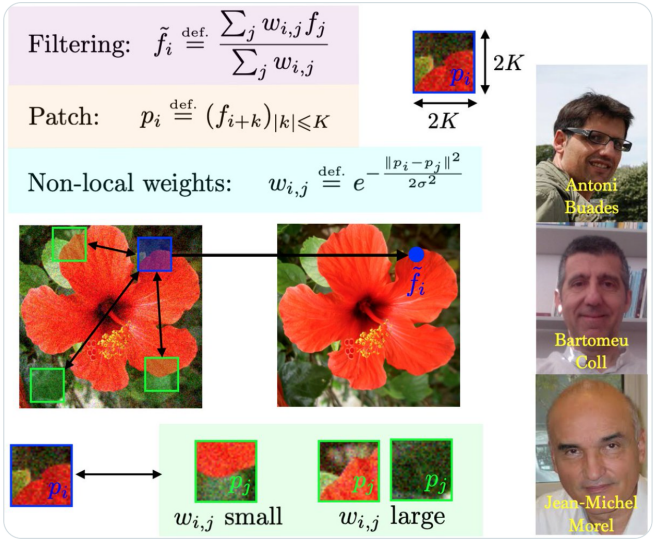


- input data are semi-synthetic **training pairs of observations/detection maps** ($\approx 40K$ injections, max 10 per map).
- architecture and loss function adapted to detection: **U-net with Dice loss**.
- two independent models for **detection** and **characterization** (flux estimation on detected sources, no whitening).

non-local means [Buades et al., 2005].



non-local means [Buades et al., 2005].



From non-local means to attention

Let us decompose an image into patches $p_1, \ldots, p_m$. Assume they have unit norm.

non-local means denoising

$$\tilde{p}_i \leftarrow \frac{\sum_{j=1}^m e^{-\frac{1}{2\sigma^2}\|p_i - p_j\|^2} p_j}{\sum_{j=1}^m e^{-\frac{1}{2\sigma^2}\|p_i - p_j\|^2}}$$

Call $P = [p_1, \ldots, p_m]^\top$ the matrix of patches, then

$$\tilde{P} \leftarrow \text{Softmax}\left(\frac{PP^\top}{\sigma^2}\right) P.$$

Attention layer: building block of the transformer

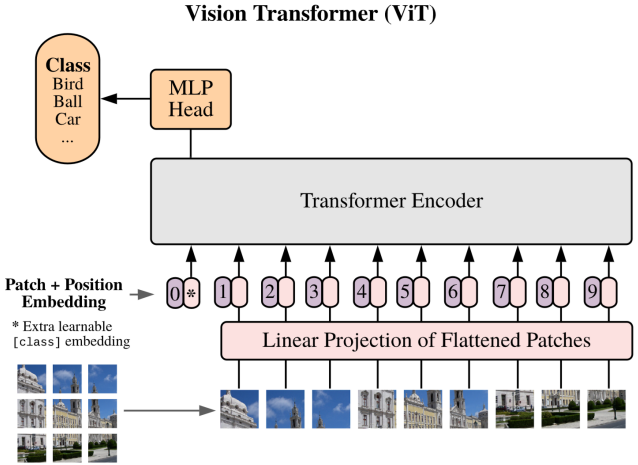
Let us decompose an image into patches $p_1, \ldots, p_m$. Call $P = [p_1, \ldots, p_m]^\top$ the matrix of patches, then

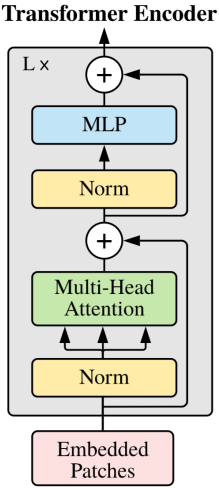
Attention layer

$$\tilde{P}_t \leftarrow \text{Softmax}\left(\frac{1}{\sqrt{d}} \underbrace{PW_Q}_{Q} \underbrace{(PW_K)^\top}_{K}\right) \underbrace{PW_v}_{V}$$

- building block of the transformer (state-of-the-art for LLMs, vision, ...).
- many variants: multiple heads, residual connections, class tokens, registers.
- variants to avoid the naive $O(n^2)$ complexity.
- **positional** encoding.

Vision transformer [Dosovitskiy, 2020]

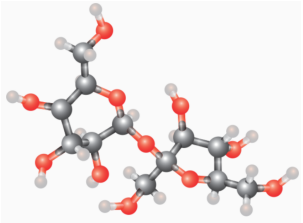




Part IV: The success story of self-supervised learning in computer vision

and some opportunities for molecular representations

Representation learning for molecules



Input x

Encoder

$\varphi(x) \in \mathbb{R}^p$

Representation

Predictor for Task 1

Output y_1

Predictor for Task 2

Output y_2

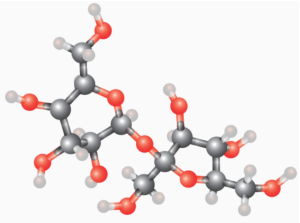
Predictor for Task 3

Output y_3

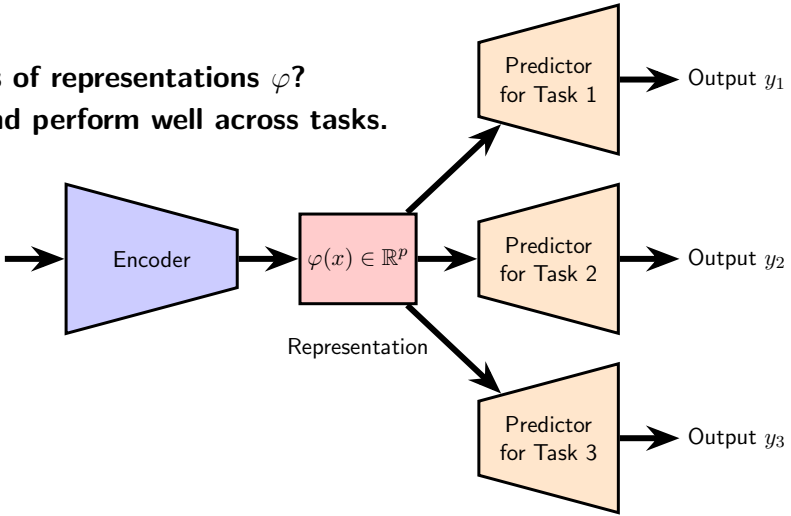
Representation learning for molecules

What is the right class of representations φ?

φ should be generic and perform well across tasks.



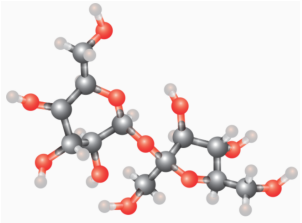
Input x



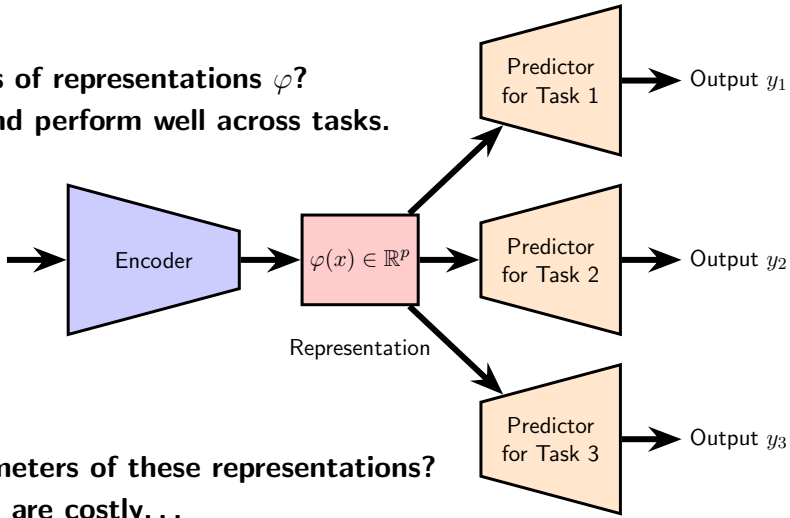
Representation learning for molecules

What is the right class of representations φ?

φ should be generic and perform well across tasks.



Input x



How to learn the parameters of these representations?

Annotations are costly...

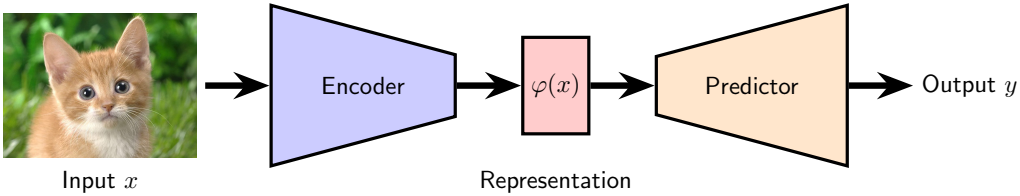
This part:

1. A story that has been **very** successful in computer vision to learn generic image representations $\varphi(x)$, trained on a large corpus of images with no annotations.
   What are the opportunities/difficulties for chemistry?

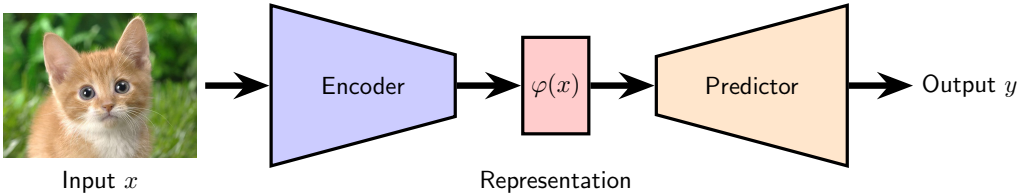
This part:

1. A story that has been **very** successful in computer vision to learn generic image representations $\varphi(x)$, trained on a large corpus of images with no annotations.
   What are the opportunities/difficulties for chemistry?
2. A **very** short survey of classical graph representations in machine learning.
   What are the current challenges?

What is representation learning?



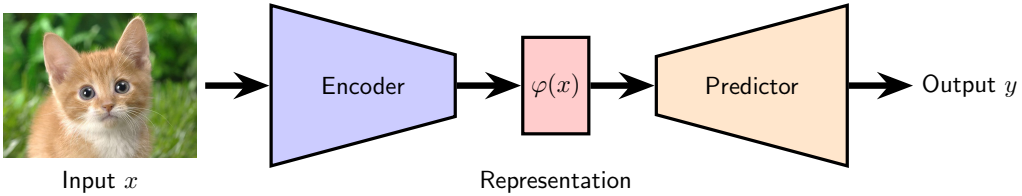
What is representation learning?



Handcrafted representations (encoder is predefined)

- traditional representations based on domain knoweldge (*e.g.*, SIFT [Lowe, 2004]).
- the predictor is **typically linear** $f(x) = W\varphi(x)$.
- $\varphi(x)$ may be very **high-dimensional** (reasonable expressiveness).

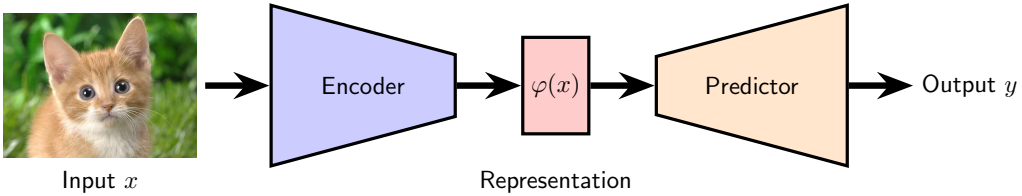
What is representation learning?



Learned representations with neural networks

- the encoder's architecture is **adapted to images** (*e.g.*, convolutional neural networks).
- the predictor is often **simple** (linear model or multilayer perceptron).
- for more complex tasks, the predictor is also **adapted to the output structure** (*e.g.*, U-Net decoder for semantic segmentation in images).

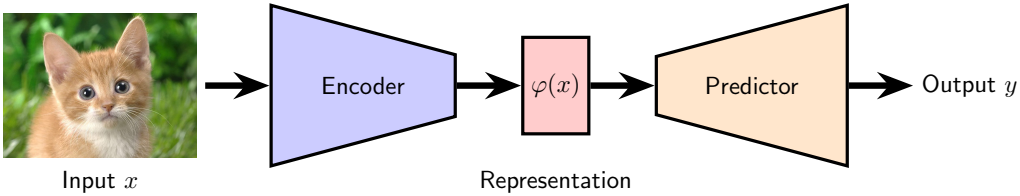
What is self-supervised learning?



Tentative definition and remarks

- learning "good" representations $\varphi(x)$ **with prediction tasks in mind**, but...
- **without having access to any label** y (unsupervised learning).
- achieved by **finding supervisory signals within the data** and/or with **pretext tasks**.

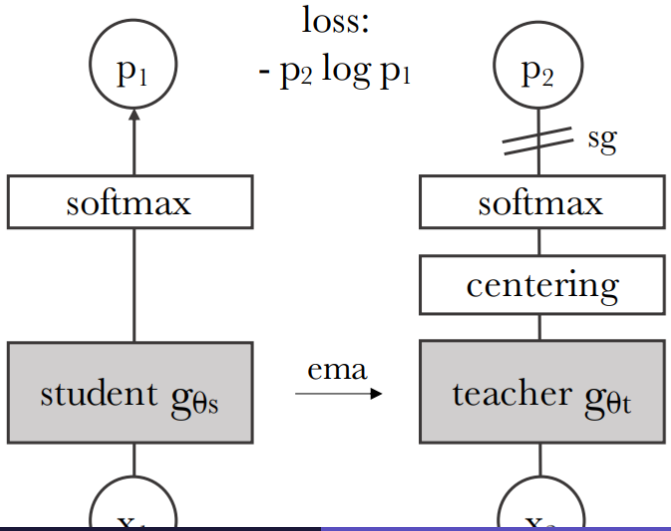
What is self-supervised learning?



Multiple purposes

- finding representations for learning simple predictors when annotations are scarce.
- harnessing information from massive unannotated databases.
- finding **generic** representations that perform well on all visual recognition tasks (foundation models).

from SwAV to DINO with self-distillation



Pretext tasks?

I want to solve task A but I do not have (much) annotated data.

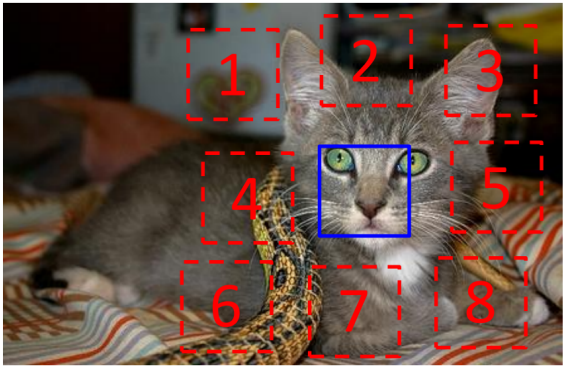
Pretext tasks?

I want to solve task A but I do not have (much) annotated data.

Perhaps a representation $\varphi(x)$ that is good for task B will also be good for task A?

Example: Spatial context prediction

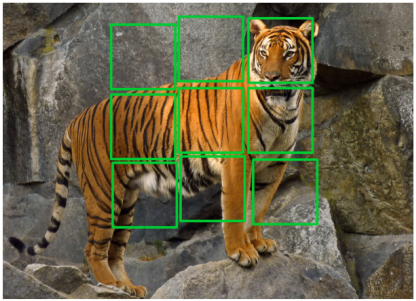
Picture courtesy of Doersch et al. [2015]



X = ( , ); Y = 3

Example: Spatial context prediction

Picture courtesy of Noroozi and Favaro [2016]



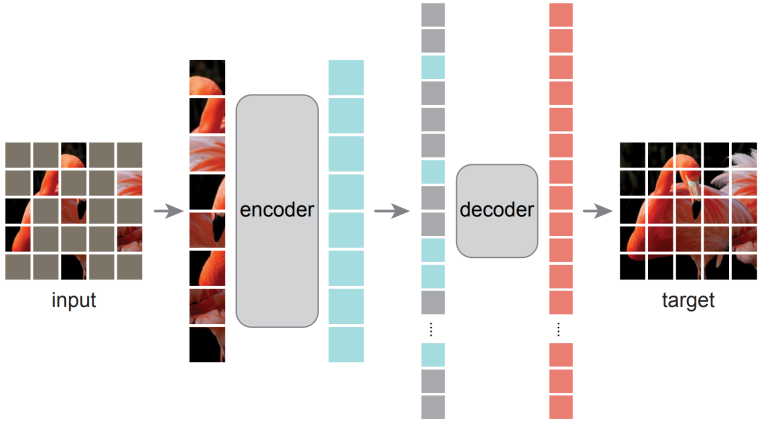
(a)

(b)

(c)

Example: Masked auto-encoders (also context prediction)

Picture courtesy of He et al. [2022]



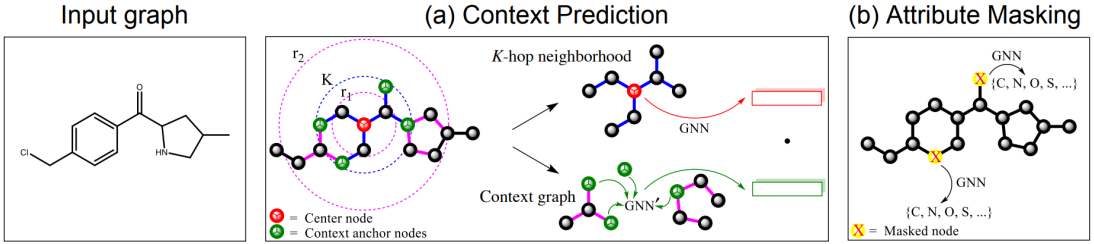
- inspired from masked language modeling [Devlin et al., 2018], revolution in NLP.

Your turn: which pretext tasks for molecular representations?

Your turn: which pretext tasks for molecular representations?

Picture courtesy of Hu et al. [2019].

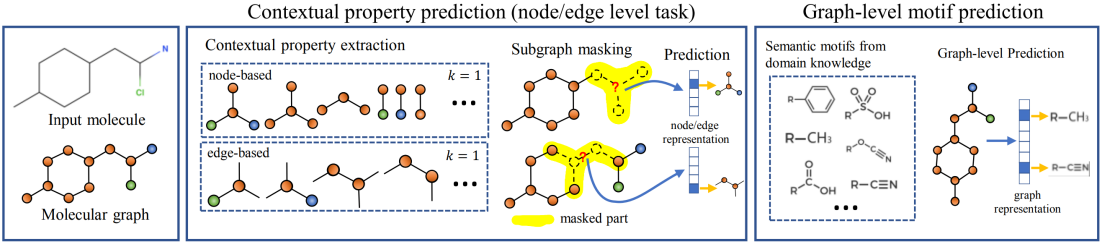
First idea: a good representation φ should be useful for context prediction tasks



Your turn: which pretext tasks for molecular representations?

Picture courtesy of Rong et al. [2020].

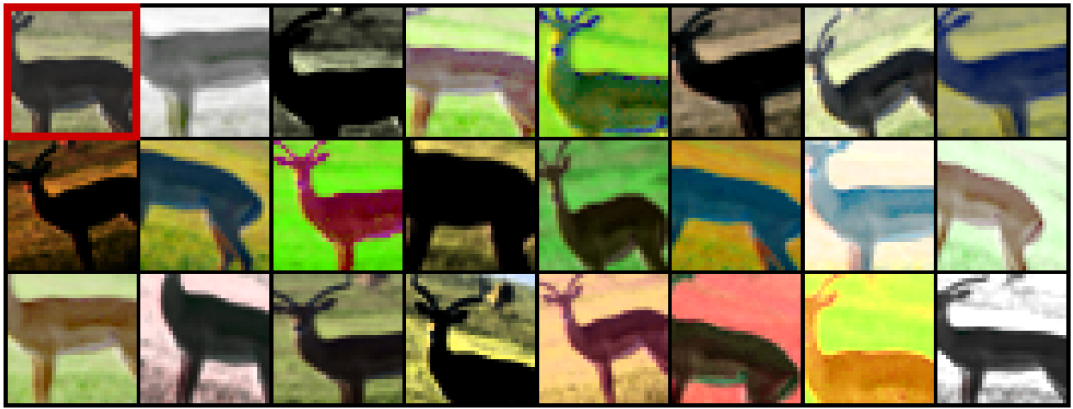
First idea: a good representation φ should be useful for context prediction tasks



Back to computer vision: Harnessing data augmentation

Picture courtesy of Dosovitskiy et al. [2014]

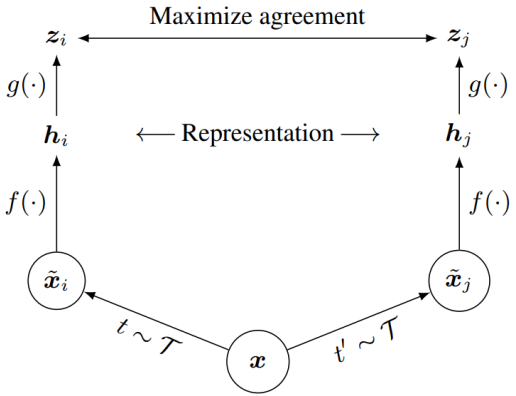
Use data augmentation to create "classes" around each sample.



Harnessing data augmentation and contrastive learning

SimCLR, Picture courtesy of Chen et al. [2020]

Second idea: a good representation φ should make augmented views of the same image closer and push apart different images.

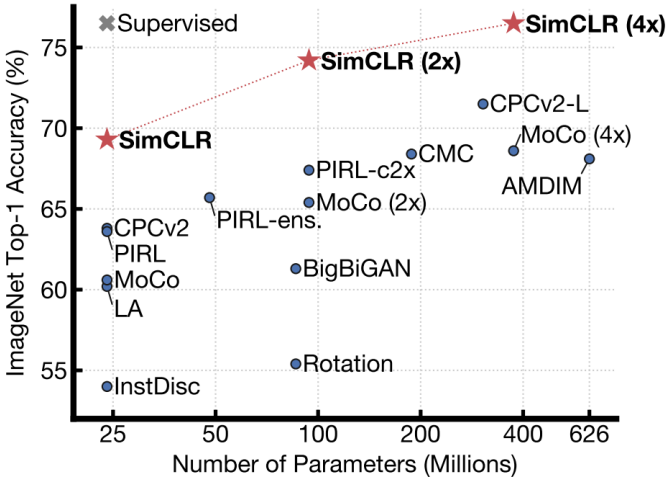


$$\ell_{i,j} = -\log\left(\frac{e^{\mathsf{sim}(\mathbf{z}_i,\mathbf{z}_j)}}{\sum_{i\neq k} e^{\mathsf{sim}(\mathbf{z}_i,\mathbf{z}_k)}}\right).$$

- trained online with large batch sizes.
- strong data augmentation.

Harnessing data augmentation and contrastive learning

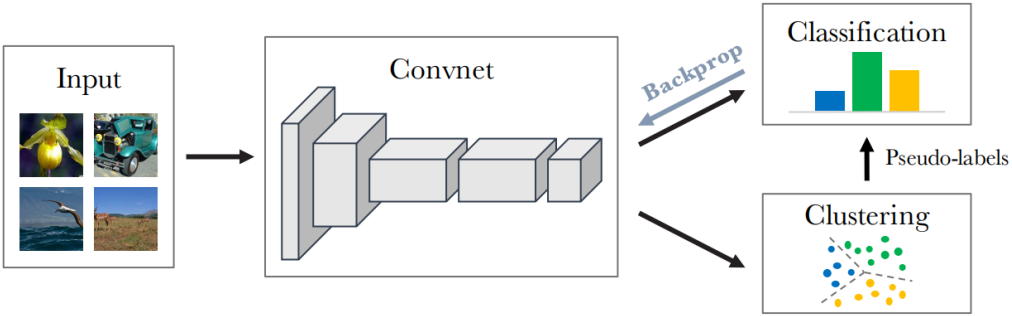
SimCLR, Picture courtesy of Chen et al. [2020]



Uncovering hidden structures in images: DeepCluster

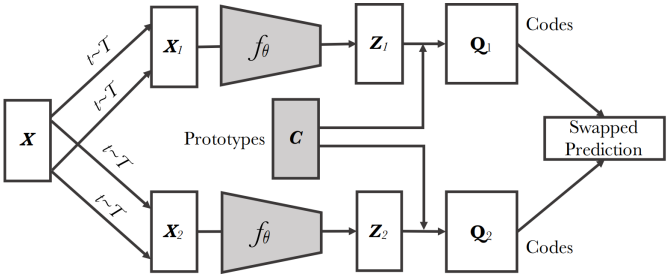
Picture courtesy of Caron et al. [2018]

Third idea: a good representation φ should uncover data clusters.



Clustering, contrastive learning, and context prediction: SwAV

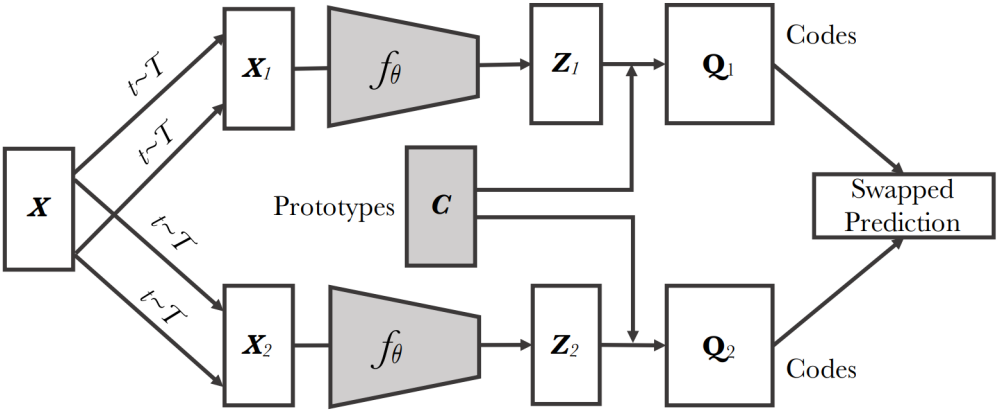
Picture courtesy of Caron, Misra, Mairal, Goyal, Bojanowski, and Joulin [2020]



Recipe

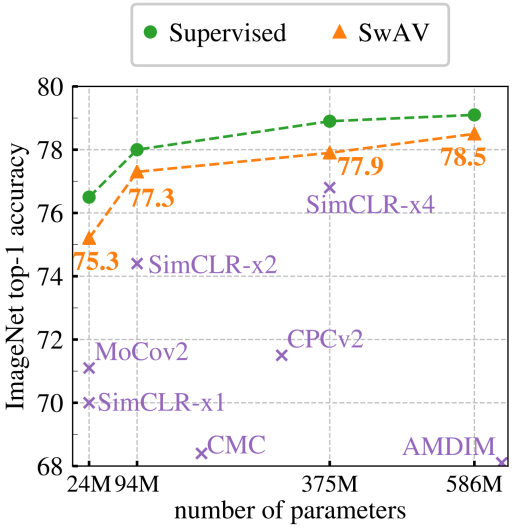
- **clustering**: prototypes $\approx$ centroids. Trivial solutions avoided by optimal transport.
- **contrastive learning with data augmentation** but no explicit negative pairs.
- **context prediction**: predicting global crops from local crops (multicrop).

Clustering, contrastive learning, and context prediction: SwAV



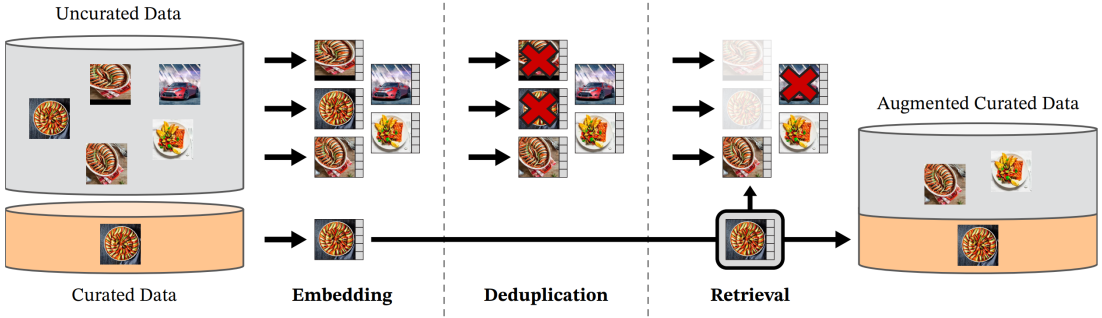
Clustering, contrastive learning, and context prediction: SwAV

Picture courtesy of Caron, Misra, Mairal, Goyal, Bojanowski, and Joulin [2020]



A foundation model for images: DINOv2

- DINO: a more recent model with self-distillation [Caron et al., 2021];
- DINOv2: foundation model trained **well-engineered data** [Oquab et al., 2024].



Opportunities/challenges for molecular representation

How to build a foundation model for molecules/materials?

1. Which **model architecture**? (see second part of this talk).

Opportunities/challenges for molecular representation

How to build a foundation model for molecules/materials?

1. Which **model architecture**? (see second part of this talk).
2. Which **learning algorithm**? Should we follow the self-supervised computer vision recipe? How to design data augmentation strategies?

Opportunities/challenges for molecular representation

How to build a foundation model for molecules/materials?

1. Which **model architecture**? (see second part of this talk).
2. Which **learning algorithm**? Should we follow the self-supervised computer vision recipe? How to design data augmentation strategies?
3. What for? What are the **downstream tasks of interest**?

Opportunities/challenges for molecular representation

How to build a foundation model for molecules/materials?

1. Which **model architecture**? (see second part of this talk).
2. Which **learning algorithm**? Should we follow the self-supervised computer vision recipe? How to design data augmentation strategies?
3. What for? What are the **downstream tasks of interest**?
4. **How to engineer a good dataset?**

Part V: A few machine learning models for molecules

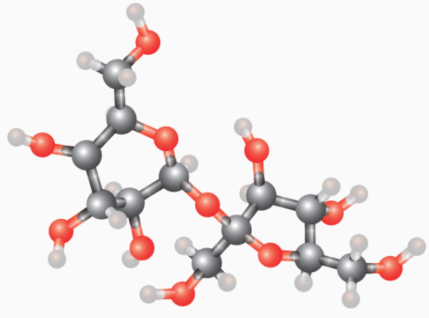
Part V: A few ~~machine~~ learning models for ~~molecules~~

Part V: A few deep learning models for graphs

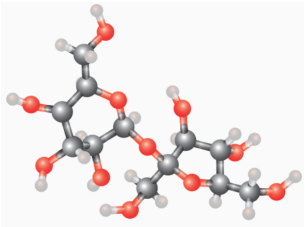
Molecular Graphs for Deep Learning Models

Ex: ZINC or OGB datasets

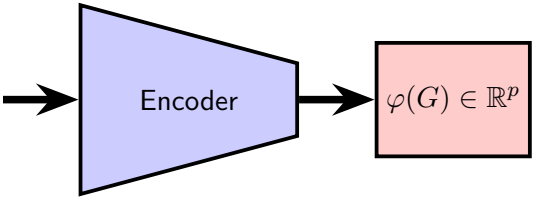
- Nodes are *atoms*, edges are *bonds*.
- Node features can be *atom-type*, *spatial position*, ...
- Edge features are bond types (*single*, *double*, *triple*).



Learning graph representations



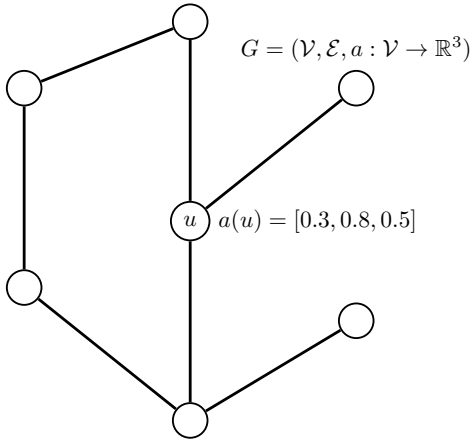
Input G



Representation

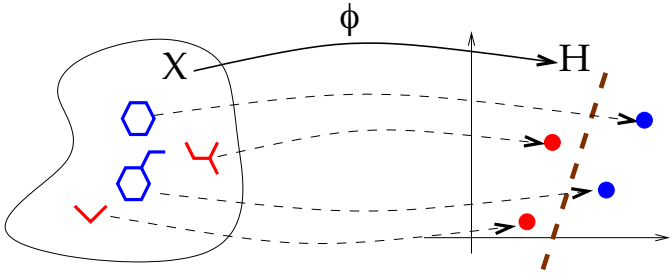
- **Expressiveness**: Find a representation (vector) that is able to discriminate graphs with different structures (distinguish non-isomorphic graphs as best as possible).
- **Tractability**: The representation should be efficiently computable on modern hardware.
- **Learnable**: One should be able to adapt the representation to the task and to the data.
- **Taking into account physics**: long-range potentials, 3D geometry, symmetries...

Graphs with node attributes



- We consider graphs $G = (V, E, a)$ where V and E are the sets of vertices and edges,
- and $a : V \to \mathbb{R}^p$ is a function assigning attributes to each node.

Classical (non-deep) graph representations



- Map each graph G to a vector $\varphi(G)$ in $\mathbb{R}^p$, which lends itself to learning tasks.
- A large class of graph embeddings can be written in the form

$$\varphi(G) := \sum_{u \in \mathcal{V}} \varphi_{\text{base}}(\ell_G(u)) \quad \text{where } \varphi_{\text{base}} \text{ embeds some local patterns } \ell_G(u) \text{ to } \mathbb{R}^p.$$

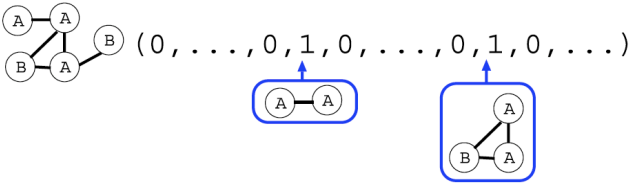
[Shervashidze et al., 2011, Lei et al., 2017, Kriege et al., 2019]

Classical (non-deep) graph representations

Find a high-dimensional representation $\varphi(G)$ for which we can efficiently compute

$$K(G, G') = \langle \varphi(G), \varphi(G') \rangle.$$

There is a very rich literature about graph kernels performing (implicitly or explicitly) **substructure enumeration**.



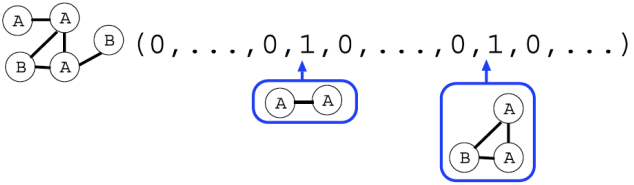
- **subgraphs and path** kernels (NP-hard, [Gärtner et al., 2003]).
- **walk** kernels [Kashima et al., 2003, Mahé et al., 2004].
- **shortest-path** kernels [Borgwardt and Kriegel, 2005].
- **graphlets** kernels [Shervashidze et al., 2009].
- **Weisfeiler-Lehman** kernel [Shervashidze et al., 2011].

Classical (non-deep) graph representations

Find a high-dimensional representation $\varphi(G)$ for which we can efficiently compute

$$K(G, G') = \langle \varphi(G), \varphi(G') \rangle.$$

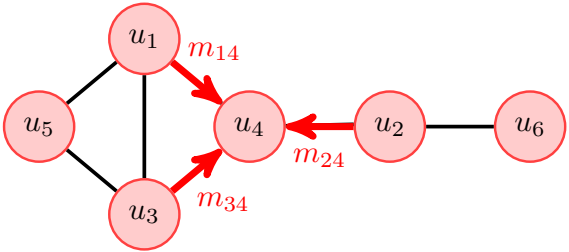
There is a very rich literature about graph kernels performing (implicitly or explicitly) **substructure enumeration**.



For a review, see the course material

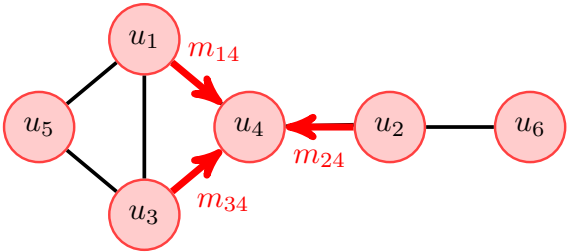
- `https://mva-kernel-methods.github.io/course-2023-2024/`

Graph neural networks with message passing



- A **multi-layer representation**: for each node u and layer k, we store a vector $\varphi_k(u)$.
- By increasing k, $\varphi_k(u)$ contains information about a **larger neighborhood**.
- Final graph representation is obtained by **pooling** $\varphi(G) = \sum_{u \in V} \varphi_K(u) \in \mathbb{R}^p$.

Graph neural networks with message passing

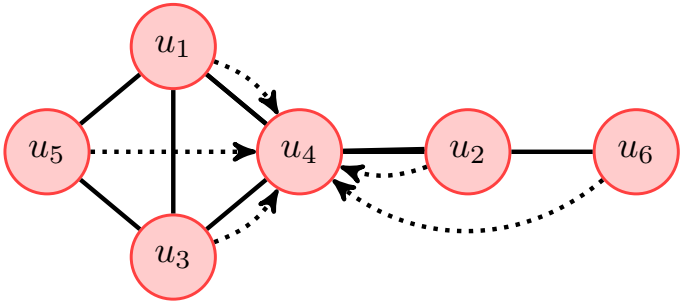


- Layer k is built from layer $k-1$ by **message passing**

$$\begin{aligned}\varphi_k(u) &= \mathsf{Process}(\varphi_{k-1}(u), \{\varphi_{k-1}(v) : v \in \mathcal{N}(u)\}) \\ &= \sum_{v \in \mathcal{N}(u) \cup u} \mathsf{ReLU}(Z_k^\top \varphi_{k-1}(v)) \quad \text{(for example)}.\end{aligned}$$

- There are many, many variants (*e.g.*, GCN [Kipf and Welling, 2017]).

Graph transformers



- G. Mialon, D. Chen, M. Selosse, and J. Mairal. GraphiT: Encoding Graph Structure in Transformers. *arXiv:2106.05667*. 2021.
- R. Menegaux, E. Jehanno, M. Selosse and J. Mairal. Self-Attention in Colors: Another Take on Encoding Graph Structure in Transformers. *TMLR*. 2023.

From GNNs to Graph transformers

An example of GNN layer (GCN, Kipf and Welling, 2017)

$$\varphi_k(u) = \mathsf{ReLU}\left(Z_k^\top \left(\sum_{v \in \mathcal{N}(u) \cup u} w_u \varphi_{k-1}(v)\right)\right) \quad \text{with} \quad w_u = \frac{1}{|\mathcal{N}(u)| + 1}.$$

From GNNs to Graph transformers

An example of GNN layer (GCN, Kipf and Welling, 2017)

$$\varphi_k(u) = \mathsf{ReLU}\left(Z_k^\top \left(\sum_{v \in \mathcal{N}(u) \cup u} w_u \varphi_{k-1}(v)\right)\right) \quad \text{with} \quad w_u = \frac{1}{|\mathcal{N}(u)| + 1}.$$

The basic transformer layer with self attention

$$\varphi_k(u) = \mathsf{ReLU}\left(Z_k^\top \left(\sum_{v \in V} A_k[u,v] \varphi_{k-1}(v)\right)\right)$$
$$\text{with} \quad A_k = \mathsf{Softmax}\left(\varphi_{k-1} Q_k^\top K_k \varphi_{k-1}^\top\right).$$

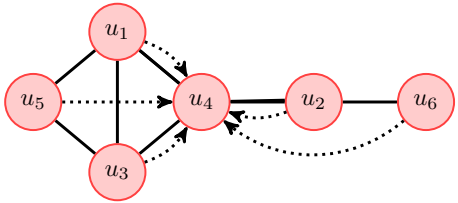
Note that basic details has been removed for simplicity (residual connections).

From GNNs to Graph transformers

The basic transformer layer with self attention

$$\varphi_k(u) = \mathsf{ReLU}\left(Z_k^\top \left(\sum_{v \in V} A_k[u,v]\varphi_{k-1}(v)\right)\right)$$

$$\text{with} \quad A_k = \mathsf{Softmax}\left(\varphi_{k-1} Q_k^\top K_k \varphi_{k-1}^\top\right).$$



From GNNs to Graph transformers

The basic transformer layer with self attention

$$\varphi_k(u) = \text{ReLU}\left(Z_k^\top \left(\sum_{v \in V} A_k[u,v]\varphi_{k-1}(v)\right)\right)$$

$$\text{with} \quad A_k = \text{Softmax}\left(\varphi_{k-1} Q_k^\top K_k \varphi_{k-1}^\top\right).$$

Challenges

- How to encode the **graph structure**? (note that if we multiply elementwise the attention matrix by the adjacency matrix, we are back to message passing)
- How to take into account **edge features**?

Graph transformers: recipes

How to take into account edge features?

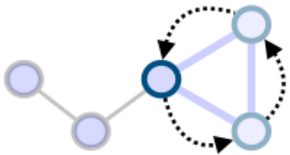
- treat edge features as node features with additional variables $E_k(u, v)$ undergoing "similar" updates.

Local structure encoding

- Enrich input features. A successful feature is based on the diagonals of random walk kernels

$$p(u) = [RW_{uu}, \ldots, RW^p_{uu}]$$

where RW^p_{uu} probability for a p-step random walk to loop back to node u:



[Dwivedi and Bresson, 2020, Rampášek et al., 2022, Lim et al., 2022]

Graph transformers: recipes

Modulate the attention matrix with relative positional encoding

- Graphormer computes an average of the dot-products of edge feature and a learnable embedding along shortest paths

$$A = \text{Softmax}\left(\varphi_{k-1} W_Q^\top W_K \varphi_{k-1}^\top + B_k^{\text{shortest-paths}}\right).$$

- GraphiT weights the attention with a diffusion kernel. This captures both short-range and long-range graph topology

$$A = \text{Normalize}\left(\text{Exp}\left(\varphi_{k-1} W_Q^\top W_K \varphi_{k-1}^\top\right) \circ K_\sigma\right).$$

[Ying et al., 2021, Mialon et al., 2021]

Graph transformers: recipes

Modulate the attention matrix with relative positional encoding

- GraphiT uses a hard-coded kernel and does not include edge features in the attention.
- CSA first enriches original edge features with random walks kernels:

$$E_{uv}^{\mathsf{rw}} = [RW_{uv}, \ldots, RW_{uv}^{p}]$$

and then learns how to exploit these features to modulate the attention matrix

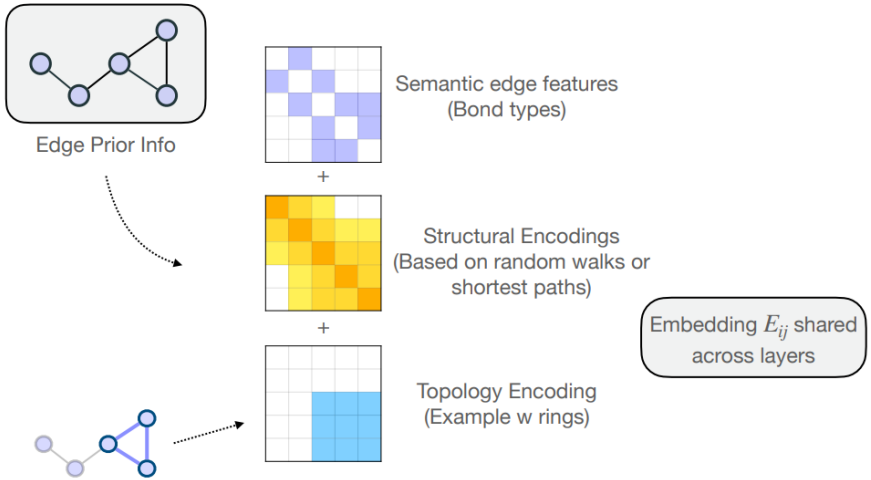
$$A = \mathsf{Softmax}\left(\varphi_{k-1} W_Q^\top W_K \varphi_{k-1}^\top + W_E^\top E_{k-1}\right).$$

Additional tricks

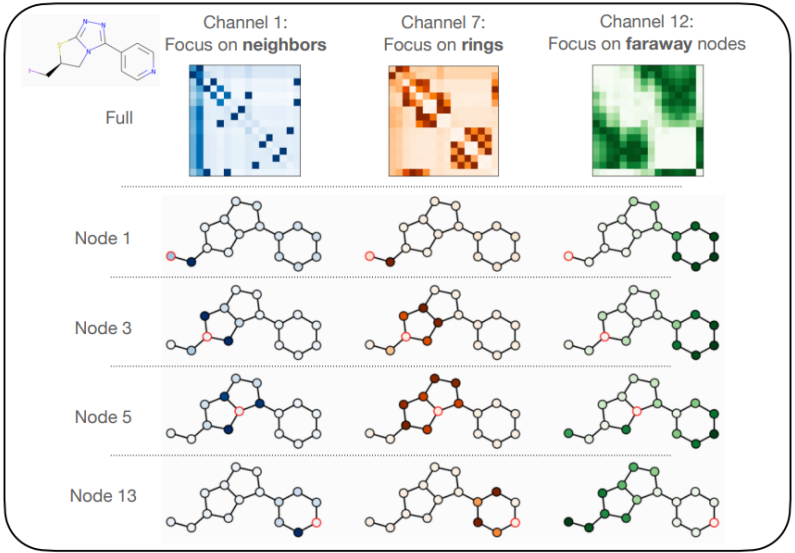
- introduce features for structures that are known to be useful (carbon rings).

[Menegaux et al., 2023]

All of this summarized in a pretty picture



Visualizing self attention



Benchmarks

	Model	ZINC MAE ↓
MPNN	GCN (Kipf & Welling, 2017)	0.367 ± 0.011
	GatedGCN (Dwivedi et al., 2022a)	0.090 ± 0.001
	GPS (Rampášek et al., 2022)	0.070 ± 0.004
h-MPNN	CIN (Bodnar et al., 2021a)	0.079 ± 0.006
	CRaWL (Toenshoff et al., 2021)	0.085 ± 0.004
	GIN-AK+ (Zhao et al., 2022)	0.080 ± 0.001
Transformers	SAN (Kreuzer et al., 2021)	0.139 ± 0.006
	Graphormer (Ying et al., 2021)	0.122 ± 0.006
	SAT (Chen et al., 2022)	0.094 ± 0.008
	EGT (Hussain et al., 2022)	0.108 ± 0.009
	GRPE (Park et al., 2022)	0.094 ± 0.002
	CSA (ours)	$\mathbf{0.070 \pm 0.003}$
	CSA-rings (ours)	$\mathbf{0.056 \pm 0.002}$

(12k graphs)

Benchmarks

	Model	PCQM4Mv2 (4M graphs)	
		Validation MAE ↓	# Param.
MPNN	GCN	0.1379	2.0M
	GCN-virtual	0.1153	4.9M
	GIN	0.1195	3.8M
	GIN-virtual	0.1083	6.7M
Transformers	Graphormer	0.0864	48.3M
	EGT	0.0869	89.3M
	GRPE	0.0890	46.2M
	GPS-small	0.0938	6.2M
	GPS-medium	0.0858	19.4M
	CSA-small (ours)	0.0898	2.8M
	CSA-deep (ours)	**0.0853**	8.3M

Physics and Geometry

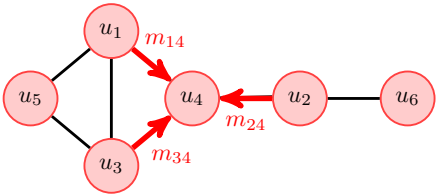
Challenges (not addressed in this presentation)

- Is there **another structure within** the graph? (*e.g.*, chain of amino acids for proteins).
- Is the graph part of a **larger structure** (crystallography)?
- Does the representation model the **right symmetries and inv/equivariances**?
- Is the graph construction satisfactory? What about **long-range interactions**?

Recap: graph representations with deep learning

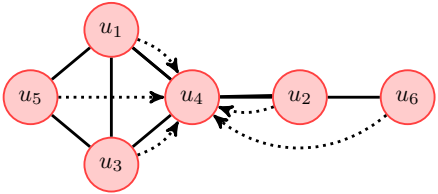
Graph neural networks with message passing

- multi-layer construction.
- sequence of local operations.
- limited expressivity [Xu et al., 2019].



Graph transformers

- non-local operations with attention.
- how to encode the graph structure?



For a detailed review, see

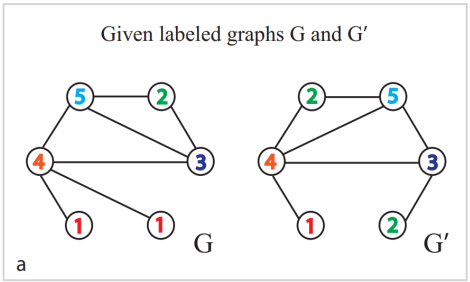
- graph neural networks for 3D atomic systems: [Duval et al., 2023].
- survey on graph transformers: [Müller et al., 2023].
- course material from Xavier Bresson `https://lnkd.in/dZZWay3Z`.

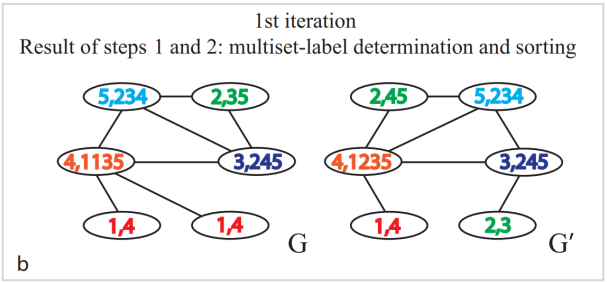
Bonus: Relation between Weisfeler-Lehman and graph neural networks

Principles of the Weisfeiler-Lehman kernel

Consider a graph $G = (V, E, a)$ with discrete labels $l_0(u) = a(u)$ at each vertex u.

- This is a **multi-layer** construction producing new labels $l_k(u)$ for each vertex at layer k.
- A label $l_k(u)$ represents $(l_{k-1}(u), \{l_{k-1}(v) : v \in \mathcal{N}(u)\})$.
- Based on the **graph isomorphism test** of Weisfeiler and Lehman, 1968.



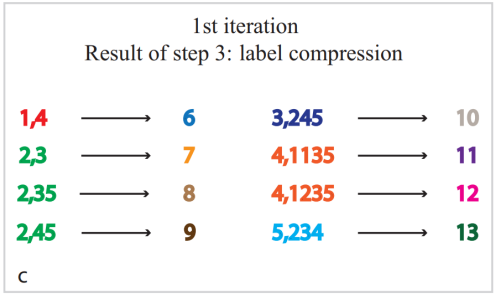


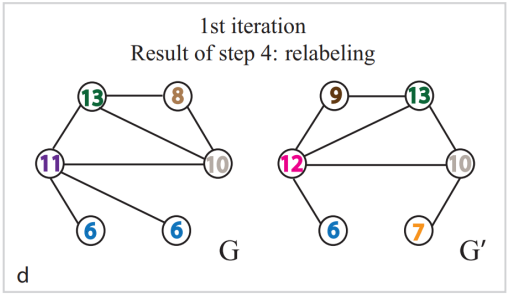
Pictures courtesy of Shervashidze et al. [2011].

Principles of the Weisfeiler-Lehman kernel

Consider a graph $G = (V, E, a)$ with discrete labels $l_0(u) = a(u)$ at each vertex u.

- This is a **multi-layer** construction producing new labels $l_k(u)$ for each vertex at layer k.
- A label $l_k(u)$ represents $(l_{k-1}(u), \{l_{k-1}(v) : v \in \mathcal{N}(u)\})$.
- Based on the **graph isomorphism test** of Weisfeiler and Lehman, 1968.

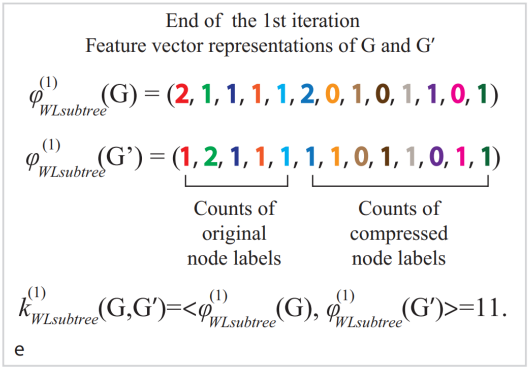




Pictures courtesy of Shervashidze et al. [2011].

Principles of the Weisfeiler-Lehman kernel

- The final representation is a histogram of **label occurences.**
- Extensions with **substructure enumeration**.



Pictures courtesy of Shervashidze et al. [2011].

Principles of the Weisfeiler-Lehman kernel

Given a graph $G = (V, E, a)$ with discrete labels $l_0(u) = a(u)$ in $\mathcal{A}_0$ for all u in V.

The Weisfeiler-Lehmann kernel representation

- **Representation at layer** k: Label $l_k(u) \in \mathcal{A}_k$ for all u in V.
- **Construction of layer** k (**message passing**):

$$l_k(u) = \text{Relabel}(l_{k-1}(u), \{l_{k-1}(v) : v \in \mathcal{N}(u)\}).$$

- **Last layer representation** with **global aggregation**:

$$\varphi_{\text{WL}}(G) = \sum_{v \in V} \text{one-hot-encoding}(l_K(u)) \in \mathbb{R}^{|\mathcal{A}|}.$$

Principles of graph neural networks with message passing

Given a graph $G = (V, E, a)$ with continous attributes $\varphi_0(u) = a(u)$ in $\mathbb{R}^{p_0}$ for all u in V.

Canonical form of message passing architecture

- **Representation at layer** k: $\varphi_k(u) \in \mathbb{R}^{p_k}$ for all u in V.
- **Construction of layer** k (**message passing**):

$$\begin{aligned}\varphi_k(u) &= \mathsf{Process}(\varphi_{k-1}(u), \{\varphi_{k-1}(v) : v \in \mathcal{N}(u)\}) \\ &= \sum_{v \in \mathcal{N}(u) \cup u} \mathsf{ReLU}(Z_k^\top \varphi_{k-1}(v)) \quad \text{(for example)}\end{aligned}$$

- **Last layer representation** with **global pooling**:

$$\varphi_{\mathsf{GNN}}(G) = \sum_{v \in V} \varphi_L(u) \in \mathbb{R}^{p_K}.$$

References I

Alberto Bietti and Julien Mairal. Group invariance, stability to deformations, and complexity of deep convolutional representations. *Journal of Machine Learning Research (JMLR)*, 2019.

Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Fifth IEEE international conference on data mining (ICDM'05)*, pages 8–pp. IEEE, 2005.

A. Buades, B. Coll, and J.-M. Morel. A review of image denoising algorithms, with a new one. *SIAM Journal on Multiscale Modeling and Simulation*, 4(2):490–530, 2005.

E. J. Candes and T. Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, 2018.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 2020.

References II

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2021.

S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20:33–61, 1999.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, 2020.

J. F. Claerbout and F. Muir. Robust modeling with erratic data. *Geophysics*, 38(5):826–844, 1973.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *CVPR*, 2015.

Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

References III

Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. *Advances in neural information processing systems*, 2014.

Alexandre Duval, Simon V Mathis, Chaitanya K Joshi, Victor Schmidt, Santiago Miret, Fragkiskos D Malliaros, Taco Cohen, Pietro Lio, Yoshua Bengio, and Michael Bronstein. A hitchhiker's guide to geometric gnns for 3d atomic systems. *arXiv preprint arXiv:2312.07511*, 2023.

VP Dwivedi and X Bresson. A generalization of transformer networks to graphs. arxiv. *arXiv preprint arXiv:2012.09699*, 2020.

Olivier Flasseur, Loic Denis, Éric Thiébaut, and Maud Langlois. Paco asdi: an algorithm for exoplanet detection and characterization in direct imaging with integral field spectrographs. *Astronomy & Astrophysics*, 637:A9, 2020.

Thomas Gärtner, Peter Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *COLT*, pages 129–143. Springer, 2003.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

References IV

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 321–328, 2003.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Nils M Kriege, Marion Neumann, Christopher Morris, Kristian Kersting, and Petra Mutzel. A unifying view of explicit and implicit feature maps of graph kernels. *Data Mining and Knowledge Discovery*, 33(6):1505–1547, 2019.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *P. IEEE*, 86(11):2278–2324, 1998.

References V

Tao Lei, Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Deriving neural architectures from sequence and graph kernels. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

Derek Lim, Joshua Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and basis invariant networks for spectral graph representation learning. *arXiv preprint arXiv:2202.13013*, 2022.

David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004.

Pierre Mahé, Nobuhisa Ueda, Tatsuya Akutsu, Jean-Luc Perret, and Jean-Philippe Vert. Extensions of marginalized graph kernels. In *Proceedings of the twenty-first international conference on Machine learning*, page 70, 2004.

Stéphane Mallat. Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65 (10):1331–1398, 2012.

References VI

Romain Menegaux, Emmanuel Jehanno, Margot Selosse, and Julien Mairal. Self-attention in colors: Another take on encoding graph structure in transformers. *Transactions on Machine Learning Research (TMLR)*, 2023.

Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. Graphit: Encoding graph structure in transformers. *arXiv preprint arXiv:2106.05667*, 2021.

Luis Müller, Mikhail Galkin, Christopher Morris, and Ladislav Rampášek. Attending to graph transformers. *arXiv preprint arXiv:2302.04181*, 2023.

Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *TMLR*, 2024.

References VII

Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. *Advances in Neural Information Processing Systems*, 33:12559–12571, 2020.

Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*, pages 488–495. PMLR, 2009.

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research (JMLR)*, 12: 2539–2561, 2011.

R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society: Series B*, 58(1):267–288, 1996.

References VIII

S.A. van de Geer. ℓ_1-regularization in high-dimensional statistical models. In *Proceedings of the International Congress of Mathematicians*, volume 4, pages 2351–2369, 2010.

M.J. Wainwright. Sharp thresholds for noisy and high-dimensional recovery of sparsity using ℓ_1-constrained quadratic programming. *IEEE Transactions on Information Theory*, 55(5): 2183–2202, 2009.

D. Wrinch and H. Jeffreys. XLII. On certain fundamental principles of scientific inquiry. *Philosophical Magazine Series 6*, 42(249):369–390, 1921.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888, 2021.

P. Zhao and B. Yu. On model selection consistency of Lasso. *Journal of Machine Learning Research*, 7:2541–2563, 2006.