Representation learning in limited-data settings

Gaël Varoquaux

Inria

Limited-data settings

n to be compared to:

- A measure of the signal-to-noise ratio
- The dimensional of the data *p*

Deep learning does not work well in small-sample regimes

But we can borrow ideas

Google Brain chief: Deep learning takes at least 100,000 examples

BLAIR HANLEY FRANK, ISG @BELRIL OCTOBER 23, 2017 4:06 PM

This talk: No silver bullet, many simple (shallow) tricks

Small-*n* problems are important

- 83% of data scientists[1] never have $n > 1M$
- n is often small for applications such as medicine

Bigger is better (how to not use this talk)

- Get more data (pool related datasets)
- Find a related problem and try transfer

This talk: data that differs from common sources

[1] www.kaggle.com/laurae2/data-scientists-vs-size-of-datasets

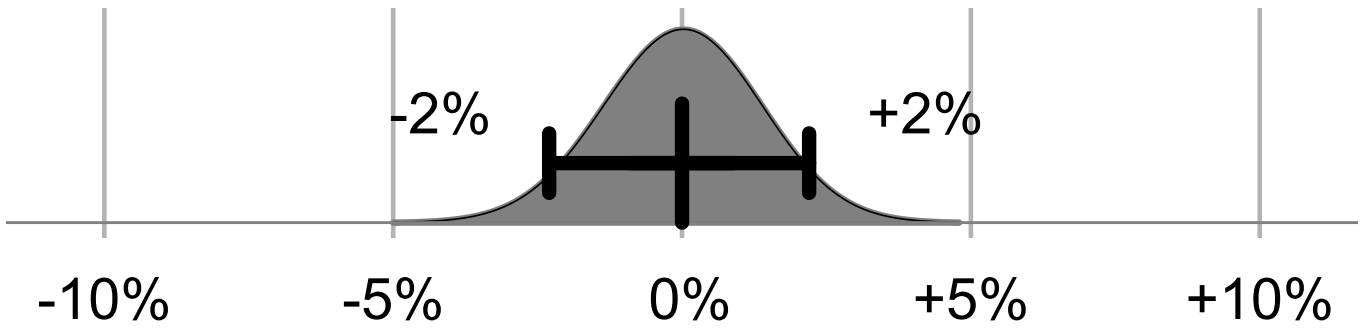
Perils of deep learning with small *n*

Selecting architecture, learning rate…

A deep architecture is validated by its measured accuracy

⚠ overfitting the validation & test set

Sampling noise for $n_{\text{test}} = 1000$:



Binomial distribution of error on test accuracy

Optimizing test accuracy will explore the tails

cf online challenges

Need for guiding principles

[Varoquaux 2018]

Outline

1 Representations for machine learning

- Non-asymptotic supervised learning
- Learning with representations
- Supervised learning of representations

2 Matrix factorization and its variants

- For signals
- For discrete objects

3 Fisher kernels

- Kernels feature maps
- From likelihoods to Kernels

1 Representations for machine learning

- Defining the notion of representations
- Their use for supervised learning

1 Representations for machine learning

Non-asymptotic supervised learning

Learning with representations

Supervised learning of representations

Settings: supervised learning

- Given n pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ drawn *i.i.d.*
  find a function $f : \mathcal{X} \to \mathcal{Y}$ such that $f(x) \approx y$

  Notation: $\hat{y} \overset{\text{def}}{=} f(x)$

Empirical risk minimization

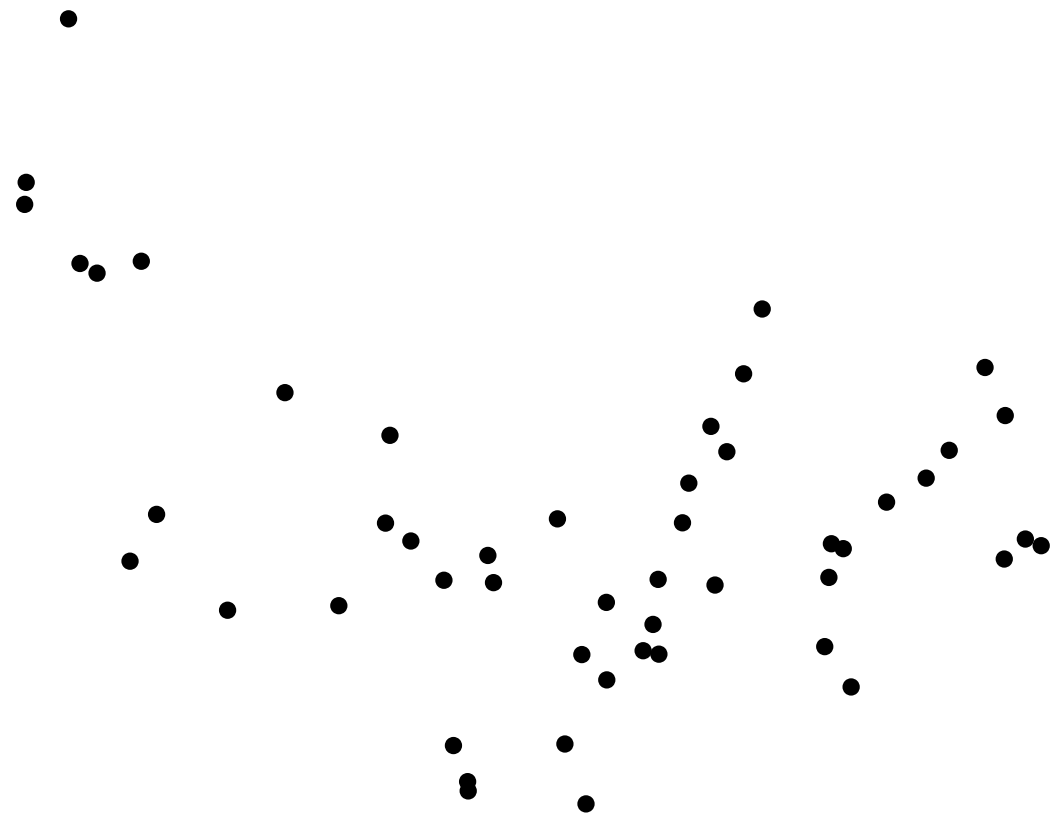
- Loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$

- Estimation of f: $$f^\star = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \, \mathbb{E}\left[l(\hat{y}, y)\right]$$

This course: how to choose good function classes $\mathcal{F}$

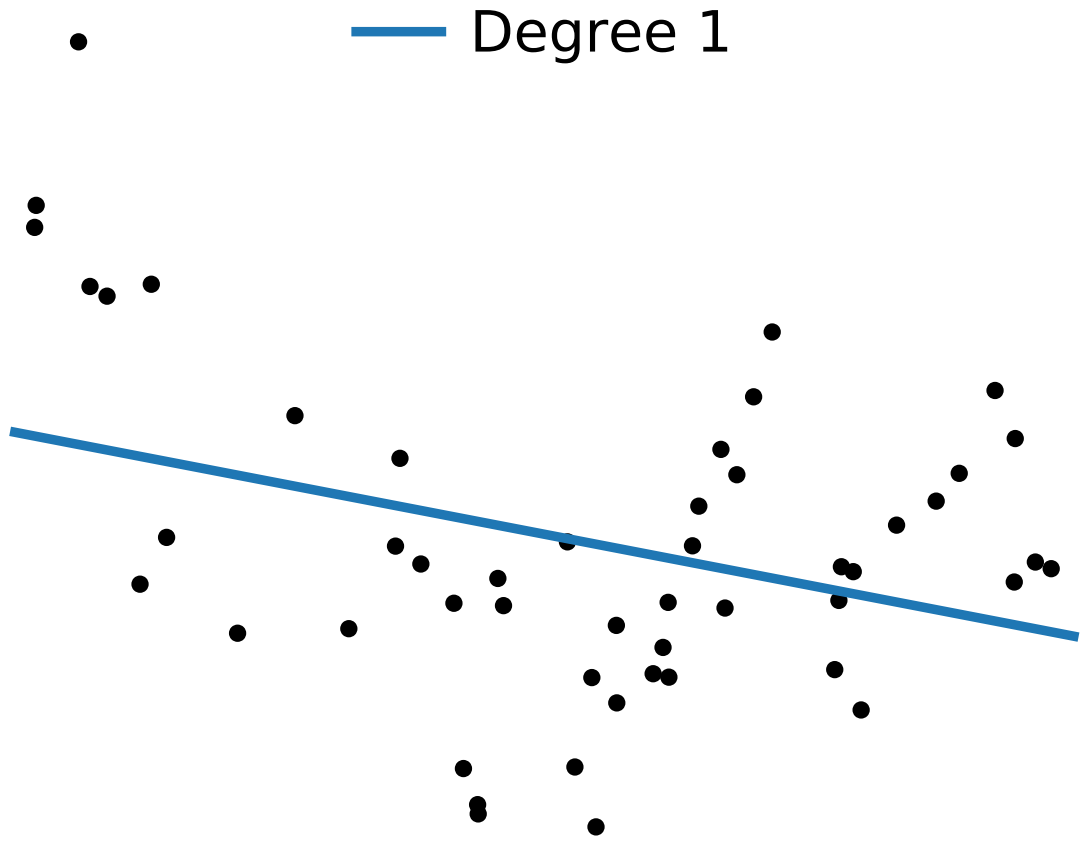
Example: finite-sample estimation of f

- Data generated with 9^{th} order polynomial + noise



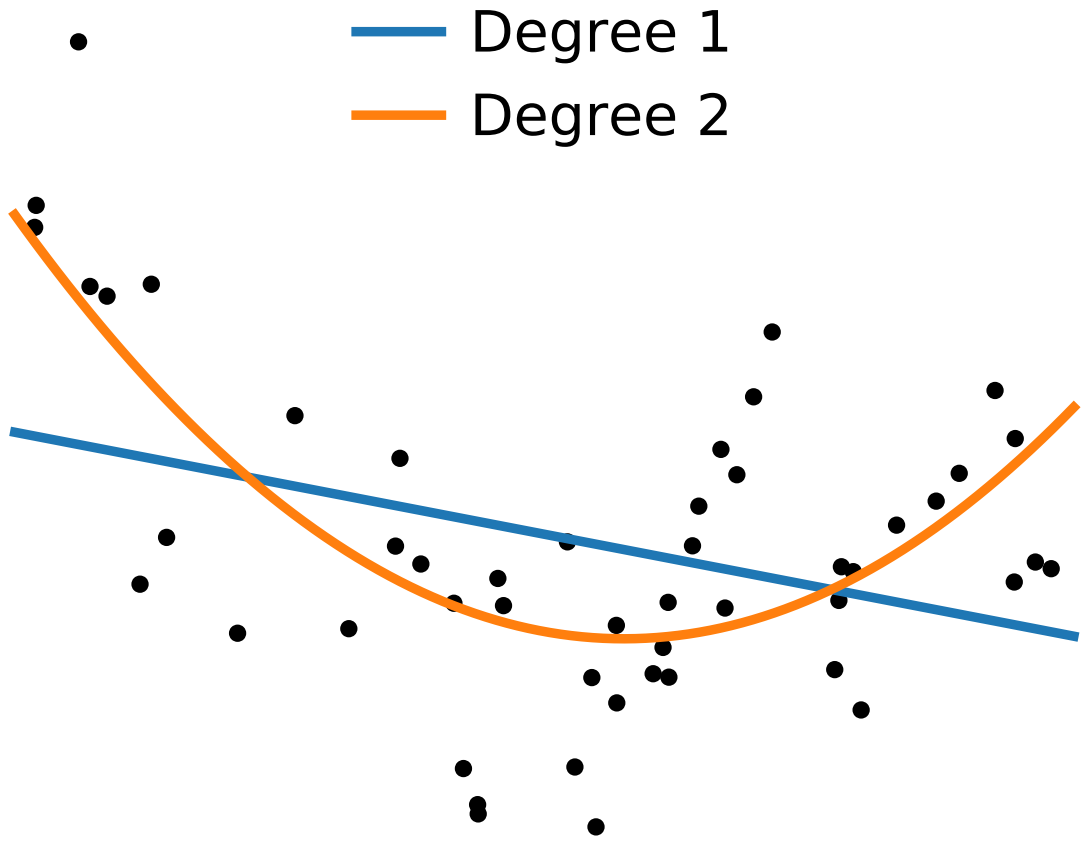
Example: finite-sample estimation of f

- Data generated with 9th order polynomial + noise
- Fit polynomials of various degrees



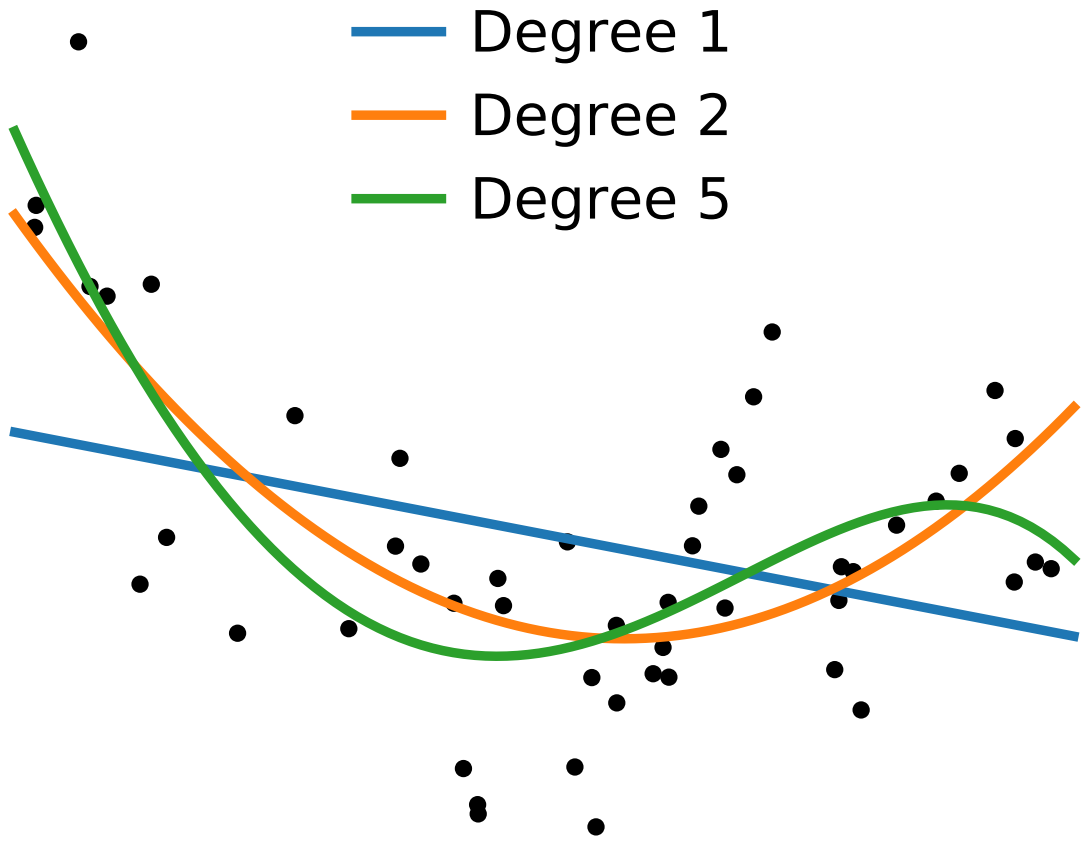
Example: finite-sample estimation of f

- Data generated with 9th order polynomial + noise
- Fit polynomials of various degrees



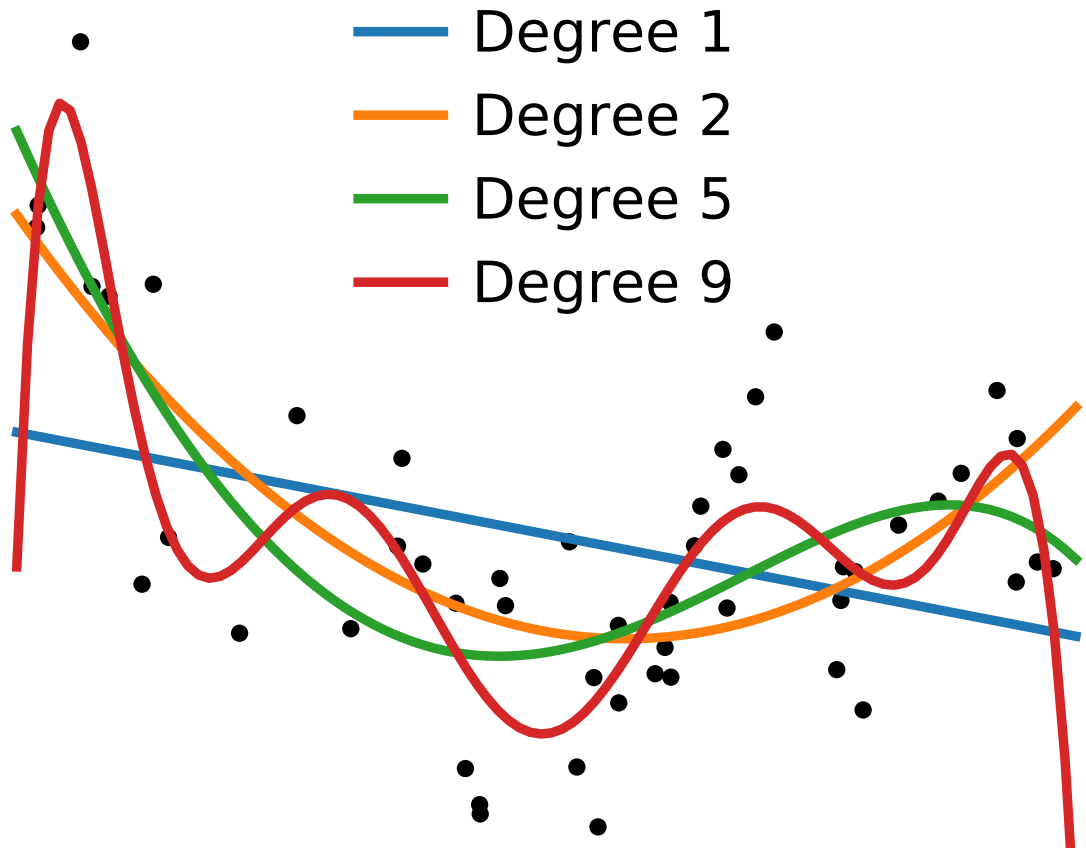
Example: finite-sample estimation of f

- Data generated with 9th order polynomial + noise
- Fit polynomials of various degrees



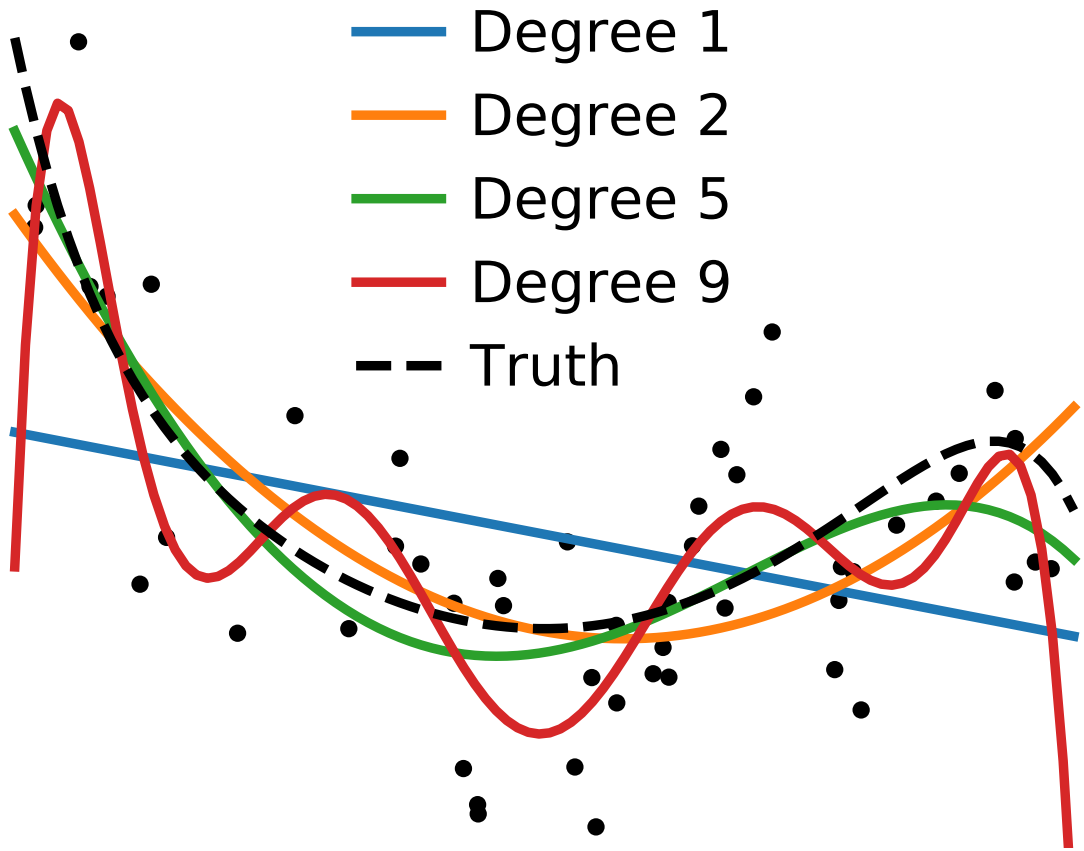
Example: finite-sample estimation of f

- Data generated with 9th order polynomial + noise
- Fit polynomials of various degrees



Example: finite-sample estimation of f

- Data generated with 9th order polynomial + noise
- Fit polynomials of various degrees



- Model too simple: underfit
- Model too complex: overfit

Theory: the generalization error

Generalization error of a prediction function f:

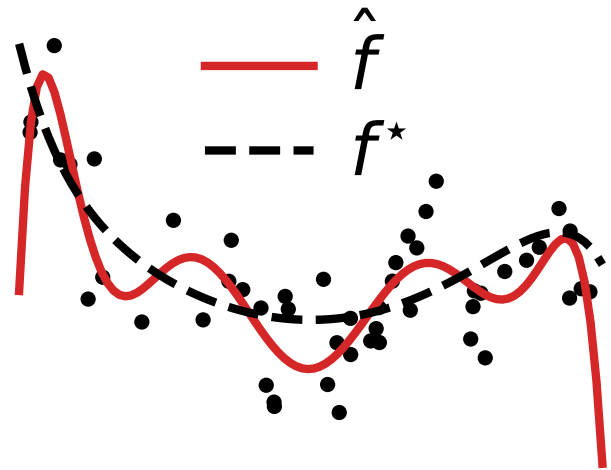
Notation : $\mathcal{E}(f) \overset{\text{def}}{=} \mathbb{E}\left[l(y, f(x))\right]$

Finite-sample regime

Ideally: $f^\star = \underset{f \in \mathcal{F}}{\operatorname{argmin}}\, \mathbb{E}\left[l\big(f(x), y\big)\right]$

In practice: $\hat{f} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \sum_{i=1}^{n} l\big(f(x_i), y_i\big)$

$\mathcal{E}(\hat{f}) \geq \mathcal{E}(f^\star)$



Theory: decomposing the generalization error

Assuming $y = g(x) + e$, e random with $\mathbb{E}[e] = 0$, the generalization error of $\hat{f}$ is:

$$\mathcal{E}(\hat{f}) = \mathbb{E}\left[l(g(x) + e,\ \hat{f}(x))\right]$$

$$= \mathcal{E}(g) \ + \ \left(\mathcal{E}(f^\star) - \mathcal{E}(g)\right) \ + \ \left(\mathcal{E}(\hat{f}) - \mathcal{E}(f^\star)\right)$$

Bayes rate

Best possible prediction

$\mathbb{E}\left[l(g(x) + e,\ g(x))\right]$

Approximation error: $g \notin \mathcal{F}$

Our model is wrong

Estimation

Sampling noise on train data

$\hat{f} \neq f^\star$

Theory: decomposing the generalization error

Assuming $y = g(x) + e$, e random with $\mathbb{E}[e] = 0$, the generalization error of $\hat{f}$ is:

$$\mathcal{E}(\hat{f}) = \mathbb{E}\left[l(g(x) + e,\ \hat{f}(x))\right]$$

$$= \mathcal{E}(g) \ + \ \left(\mathcal{E}(f^\star) - \mathcal{E}(g)\right) \ + \ \left(\mathcal{E}(\hat{f}) - \mathcal{E}(f^\star)\right)$$

Bayes rate

Best possible prediction

$\mathbb{E}\left[l(g(x) + e,\ g(x))\right]$

Due to the noise e

Cannot be avoided

Theory: decomposing the generalization error

Assuming $y = g(x) + e$, e random with $\mathbb{E}[e] = 0$, the generalization error of $\hat{f}$ is:

$$\mathcal{E}(\hat{f}) = \mathbb{E}\left[l(g(x) + e,\ \hat{f}(x))\right]$$

$$= \mathcal{E}(g) \ + \ \left(\mathcal{E}(f^\star) - \mathcal{E}(g)\right) \ + \ \left(\mathcal{E}(\hat{f}) - \mathcal{E}(f^\star)\right)$$

Approximation error: $g \notin \mathcal{F}$

Our model is wrong

Decreases for larger $\mathcal{F}$

Empirical upper bound: train error

Theory: decomposing the generalization error

Assuming $y = g(x) + e$, e random with $\mathbb{E}[e] = 0$, the generalization error of $\hat{f}$ is:

$$\mathcal{E}(\hat{f}) = \mathbb{E}\left[l(g(x) + e,\ \hat{f}(x))\right]$$

$$= \mathcal{E}(g) \ + \ \left(\mathcal{E}(f^\star) - \mathcal{E}(g)\right) \ + \ \left(\mathcal{E}(\hat{f}) - \mathcal{E}(f^\star)\right)$$

Estimation
Sampling noise on train data

$$\hat{f} \neq f^\star$$

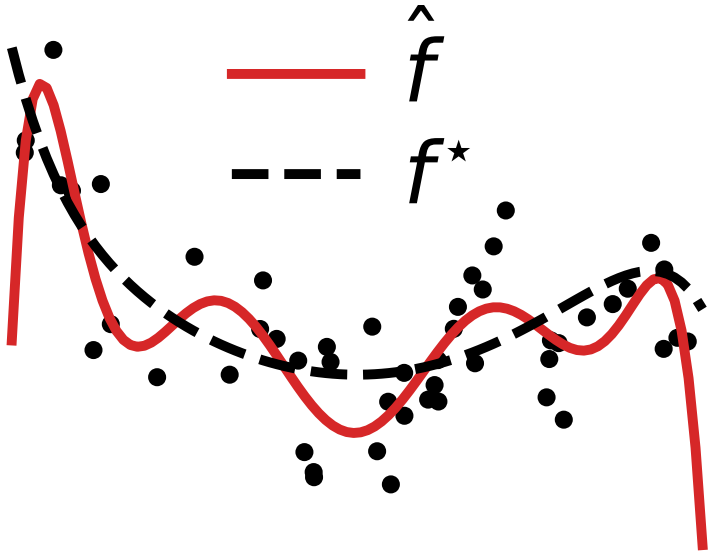
Finite-sample problem

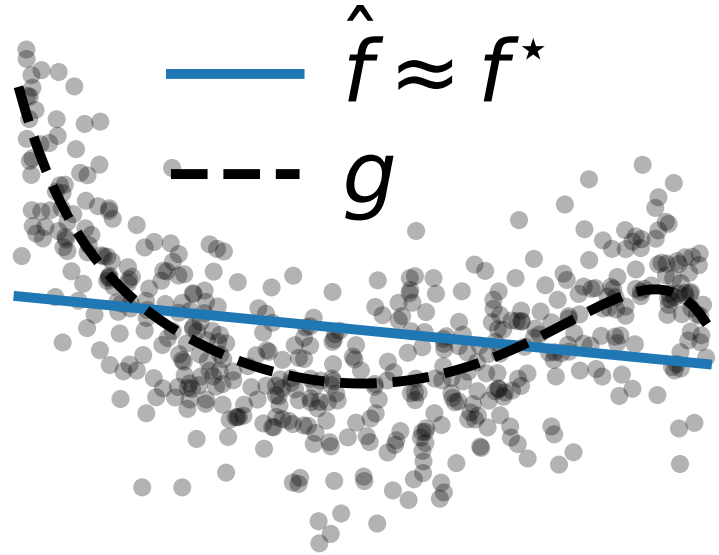
Decreases as n grows

Increases for larger $\mathcal{F}$

Guesstimate: difference between train and test error

Example: polynomial regression degree





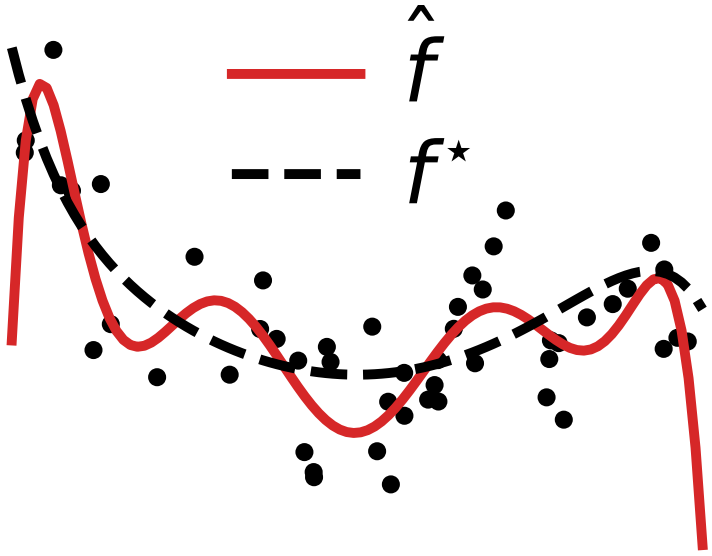
Degree 9, small n

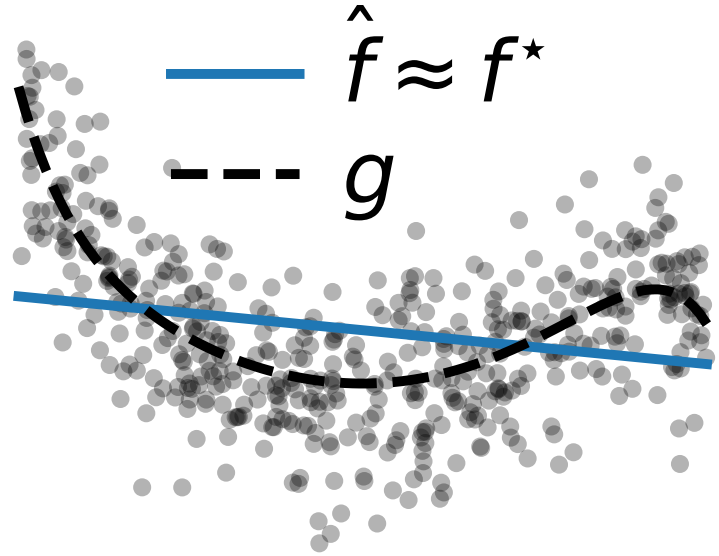
- no approximation error
- large estimation error

Degree 1, large n

- small estimation error
- large approximation error

Example: polynomial regression degree





Degree 9, small n

- no approximation error
- large estimation error

Degree 1, large n

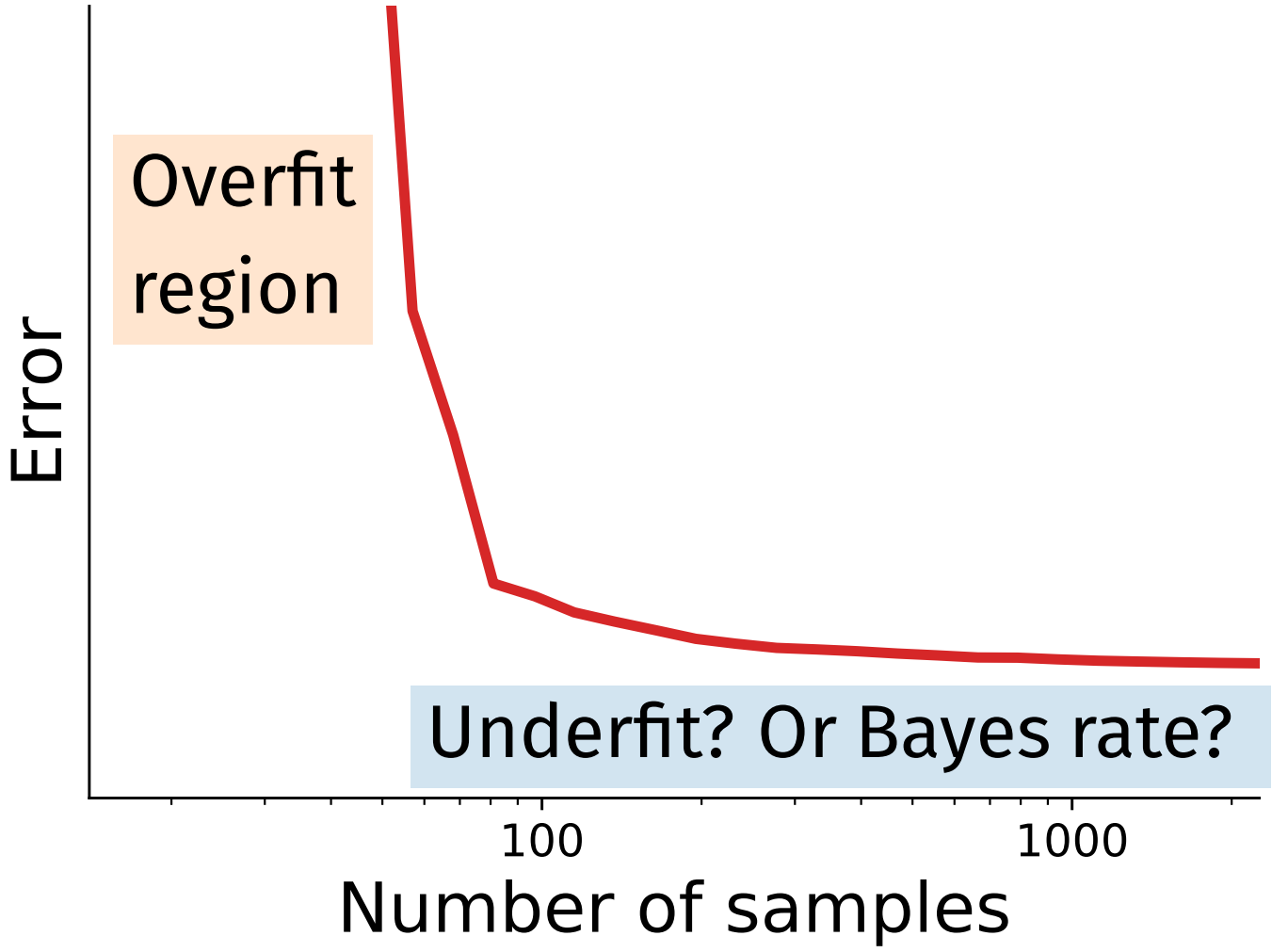
- small estimation error
- large approximation error

$$\hat{f} = \text{argmin}_{f \in \mathcal{F}} \sum_i l(f(x_i), y_i)$$

Function class $\mathcal{F}$ not restrictive enough

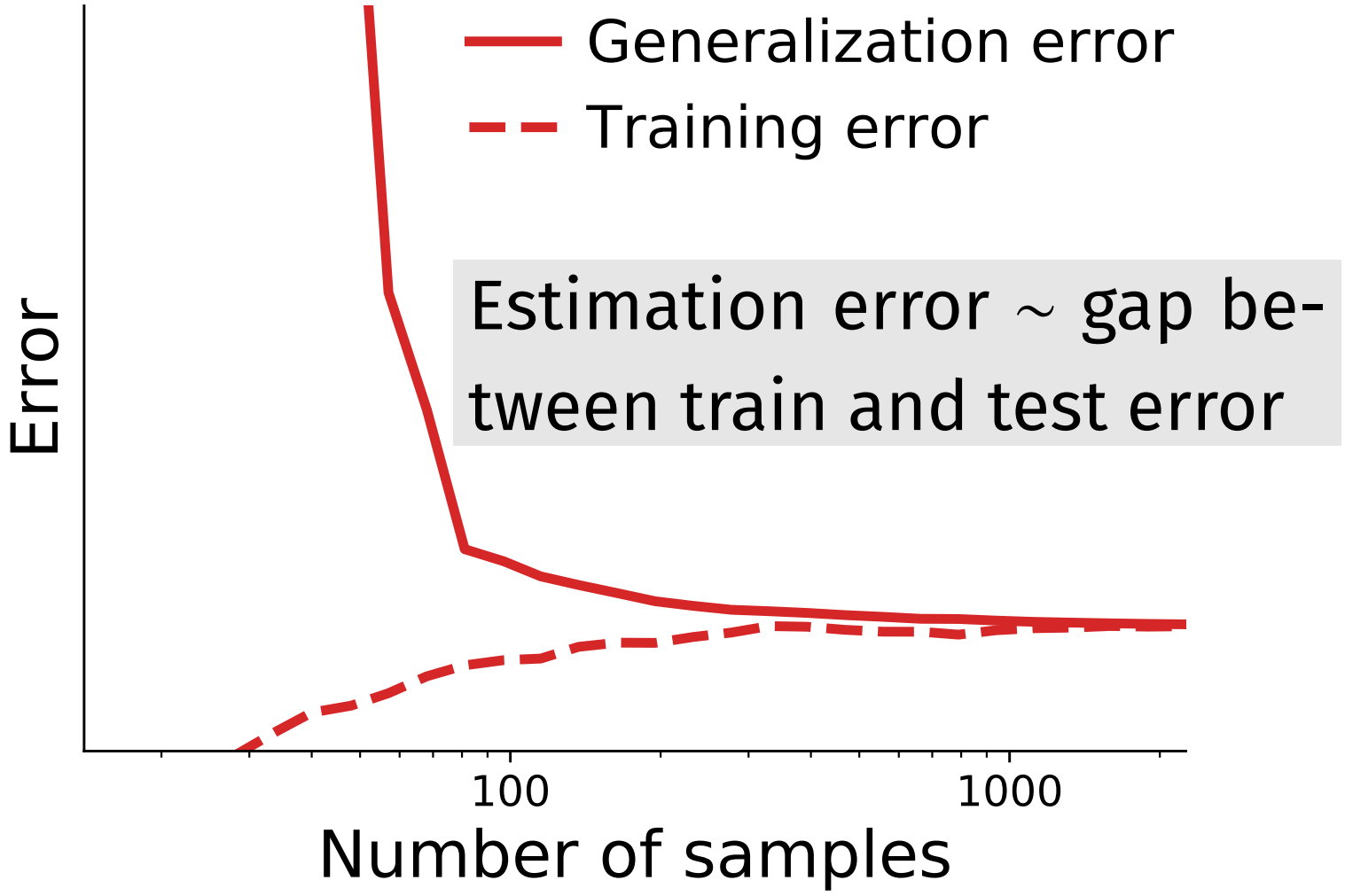
Function class $\mathcal{F}$ too restrictive

Gauging overfit vs underfit: learning curves



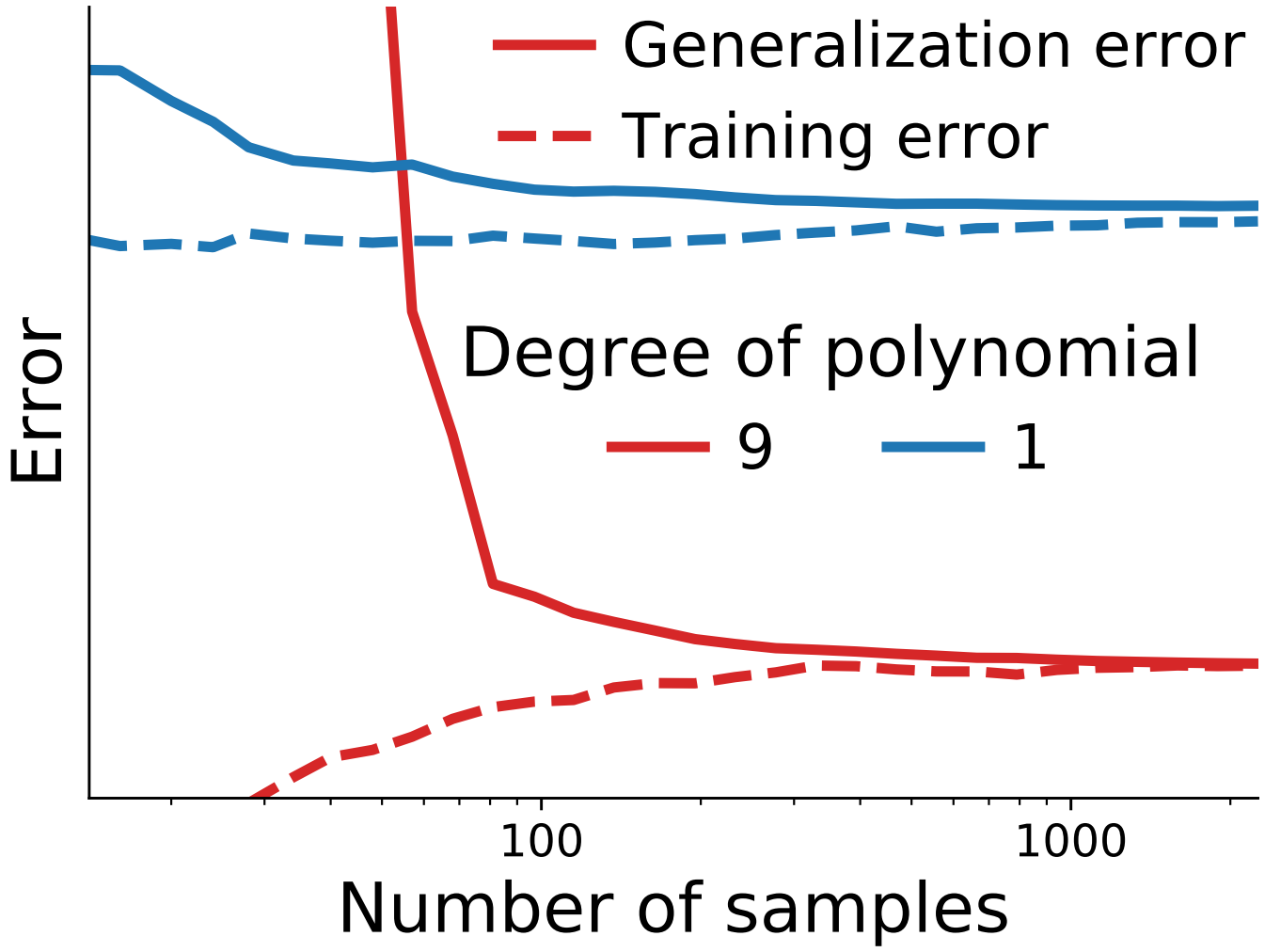
`sklearn.model_selection.learning_curve`

Gauging overfit vs underfit: learning curves



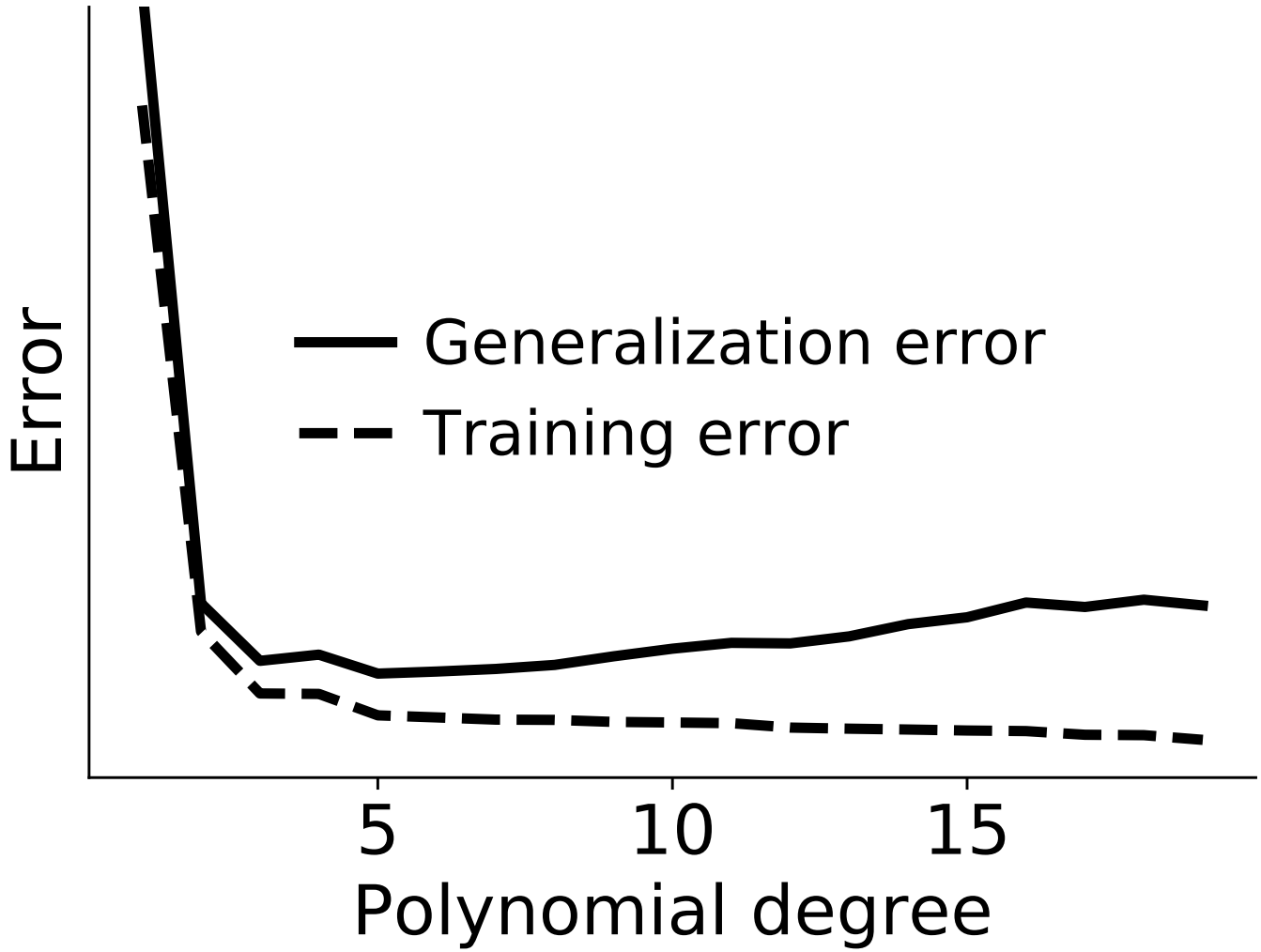
`sklearn.model_selection.learning_curve`

Gauging overfit vs underfit: learning curves



- Simpler models reach the assymptotic regime faster
  (smaller “sample complexity”)
- But can underfit

Gauging overfit vs underfit: validation curves



`sklearn.model_selection.validation_curve`

- Reveals underfits

Linear models for limited-data settings

In high-dimensional limited-data settings,
linear models are often the best choice

For p-dimensional data, $\mathbf{x} \in \mathbb{R}^p$,
they have p parameters

ARTICLE OPEN

Scalable and accurate deep learning with electronic health records

$n \sim 200\,000$

Alvin Rajkomar[1,2], Eyal Oren[1], Kai Chen[1], Andrew M. Dai[1], Nissan Hajaj[1], Michaela Hardt[1], Peter J. Liu[1], Xiaobing Liu[1], Jake Marcus[1], Mimi Sun[1], Patrik Sundberg[1], Hector Yee[1], Kun Zhang[1], Yi Zhang[1], Gerardo Flores[1], Gavin E. Duggan[1], Jamie Irvine[1], Quoc Le[1], Kurt Litsch[1], Alexander Mossin[1], Justin Tansuwan[1], De Wang[1], James Wexler[1], Jimbo Wilson[1], Dana Ludwig[2], Samuel L. Volchenboum[3], Katherine Chou[1], Michael Pearson[1], Srinivasan Madabushi[1], Nigam H. Shah[4], Atul J. Butte[2], Michael D. Howell[1], Claire Cui[1], Greg S. Corrado[1] and Jeffrey Dean[1]

Inpatient Mortality, AUROC (95% CI)	Hospital A	Hospital B
Deep learning	0.95(0.94-0.96)	0.93(0.92-0.94)
Baseline (logistic regression)	0.93(0.92-0.95)	0.91(0.89-0.92)

Theory: Approximating with linear predictors

- Linear predictor[1]: $\hat{y} = \mathbf{x}^T\mathbf{w}, \quad \mathbf{w} \in \mathbb{R}^p$
- Data model: $y = \mathbf{x}^T\mathbf{w}^\star + \delta(\mathbf{x}) + e \quad \mathbb{E}[e] = 0$

  $\mathbf{x}^T\mathbf{w}^\star$: best linear predictor
- Ridge estimator:

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{y}_{\text{train}} - \mathbf{X}_{\text{train}}\mathbf{w}\|^2_{\text{Fro}} + \lambda\|\mathbf{w}\|_2^2$$

Error compared to best linear predictor:

$$\mathbb{E}\left[\|y - \mathbf{x}^T\hat{\mathbf{w}}\|_2^2\right] = \mathbb{E}\left[\|y - \mathbf{x}^T\mathbf{w}^\star\|_2^2\right] + o\left(\sigma^2 p / n_{\text{train}}\right)$$

[Hsu... 2014, sec 2.5]

Random design analysis can characterize the generalization error without assuming a correct data-generating model (miss-specified model) [Hsu... 2014, Rosset and Tibshirani 2018]

[1] Predictor, not model: we do not assume it is a data-generating process.

Theory: Approximating with linear predictors

- Linear predictor[1]: $\hat{y} = \mathbf{x}^\mathsf{T}\mathbf{w}, \qquad \mathbf{w} \in \mathbb{R}^p$
- Data model: $y = \mathbf{x}^\mathsf{T}\mathbf{w}^\star + \delta(\mathbf{x}) + e \qquad \mathbb{E}[e] = 0$

  $\mathbf{x}^\mathsf{T}\mathbf{w}^\star$: best linear predictor
- Ridge estimator:

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{y}_{\text{train}} - \mathbf{X}_{\text{train}}\mathbf{w}\|^2_{\text{Fro}} + \lambda\|\mathbf{w}\|_2^2$$

Error compared to best linear predictor:

$$\mathbb{E}\left[\|y - \mathbf{x}^\mathsf{T}\hat{\mathbf{w}}\|_2^2\right] = \mathbb{E}\left[\|y - \mathbf{x}^\mathsf{T}\mathbf{w}^\star\|_2^2\right] + o\left(\sigma^2 p / n_{\text{train}}\right)$$

Approximation error
Data not linearly generated
$\Rightarrow$ craft more features

Estimation error
Curse of dimensionality
$\Rightarrow$ limit number of features

[1] Predictor, not model: we do not assume it is a data-generating process.

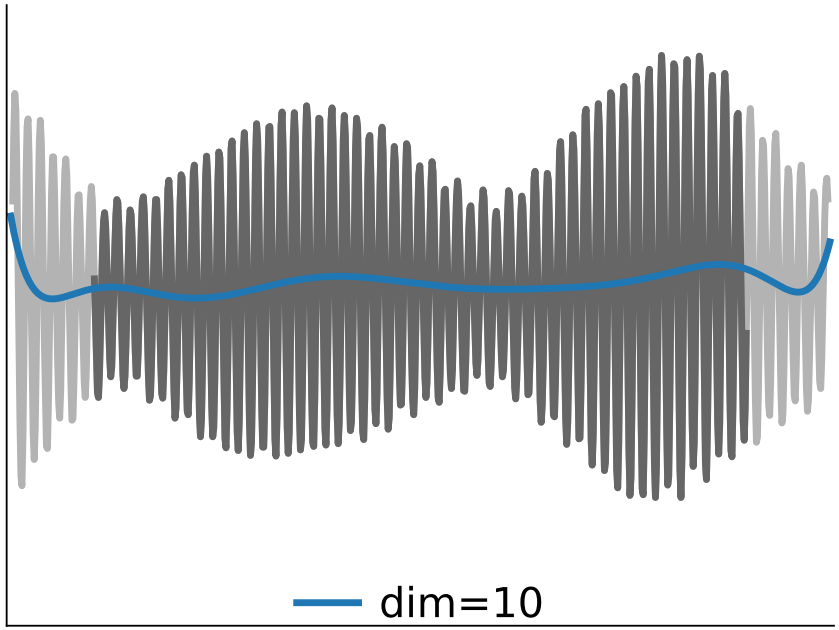
Example: extrapolating sea level (tides)

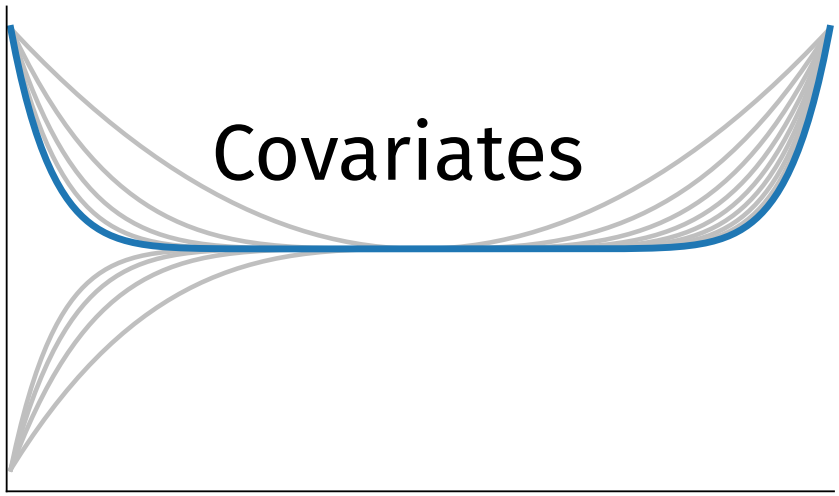
- Predict sea level as a function of time
- Test outside of observed range[1]

[1] Technically, this is not in our theory: test set ≠ train set.

Example: extrapolating sea level (tides)

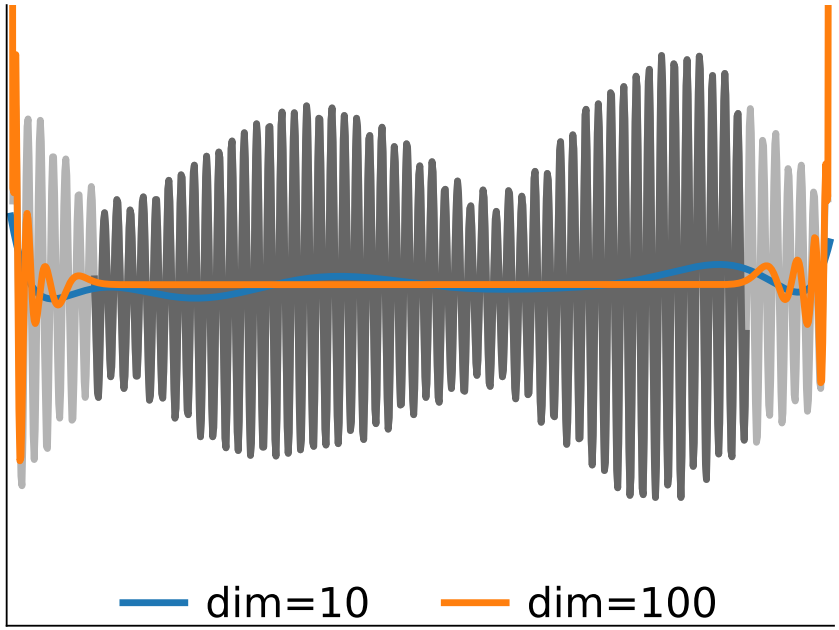
Polynomial regression

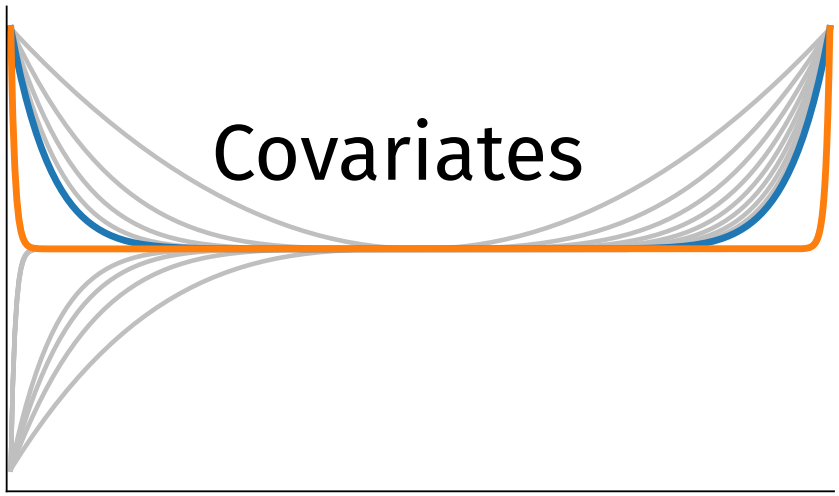




Example: extrapolating sea level (tides)

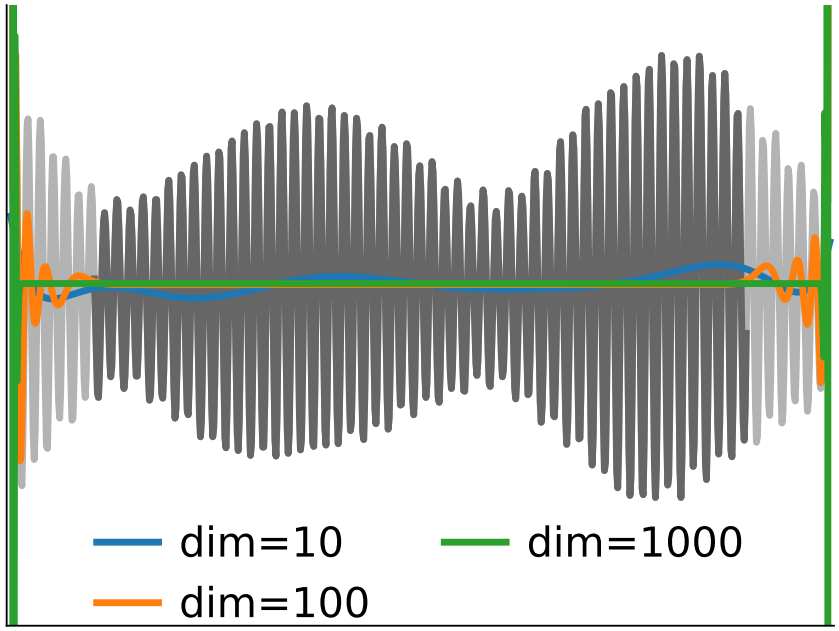
Polynomial regression

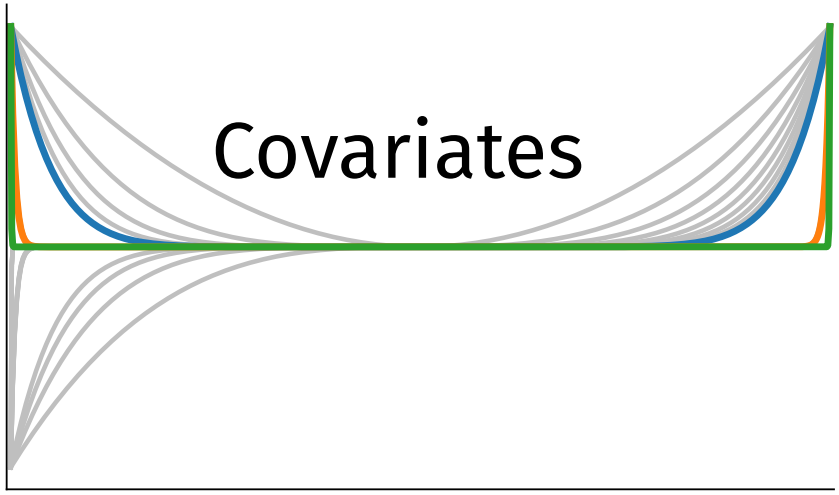




Example: extrapolating sea level (tides)

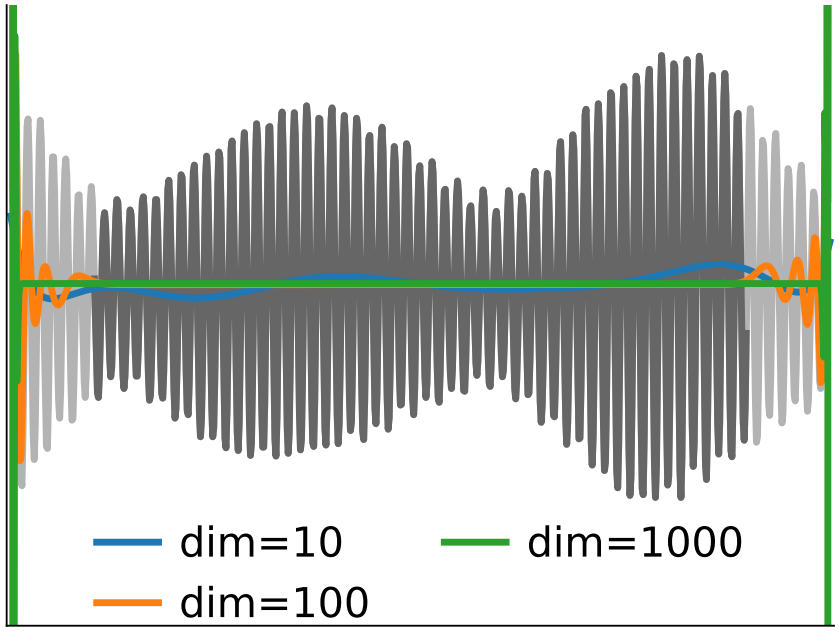
Polynomial regression

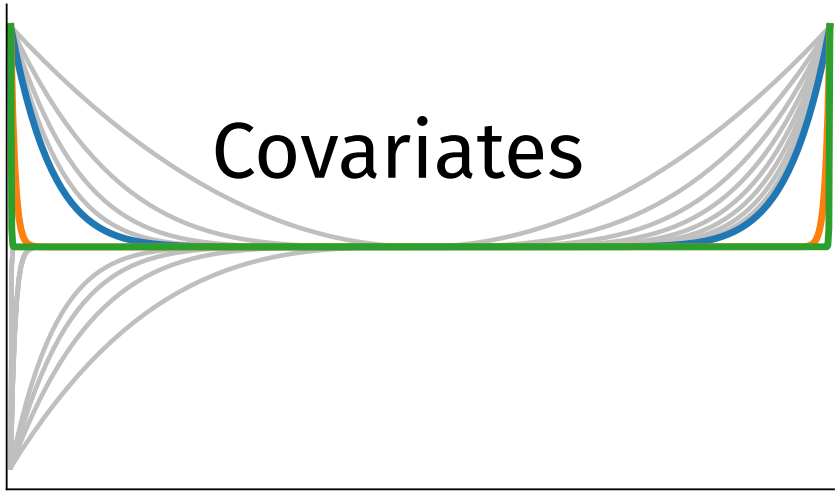




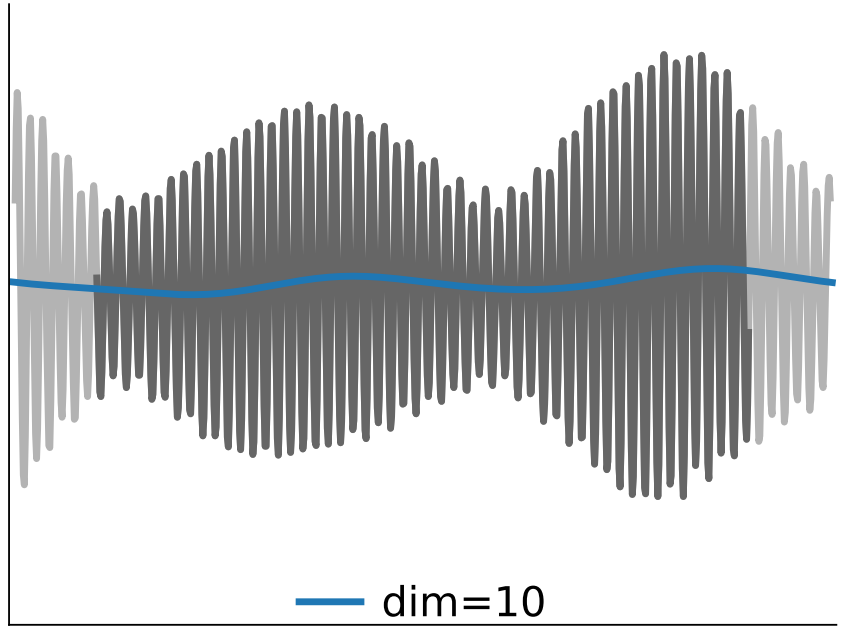
Example: extrapolating sea level (tides)

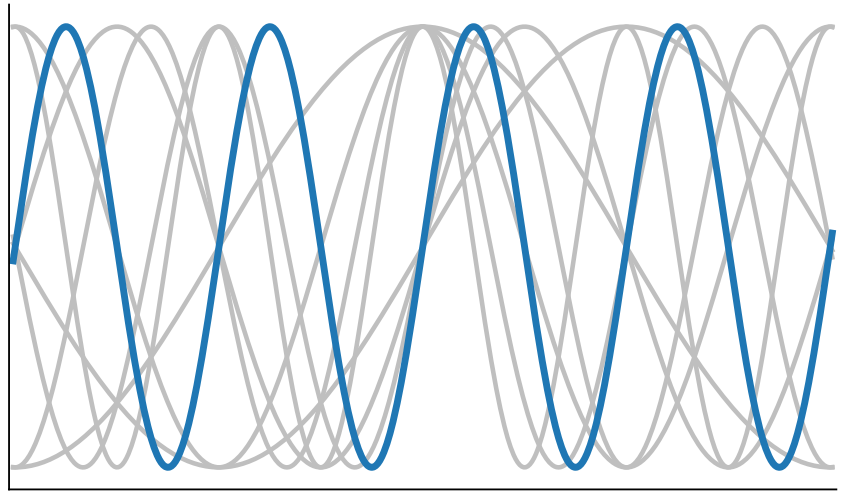
Polynomial regression





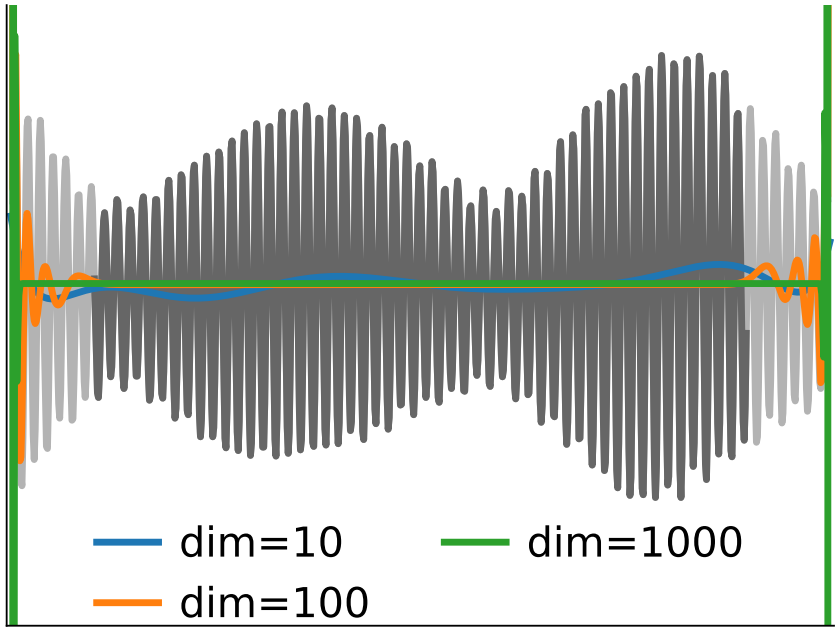
Sines and cosines basis

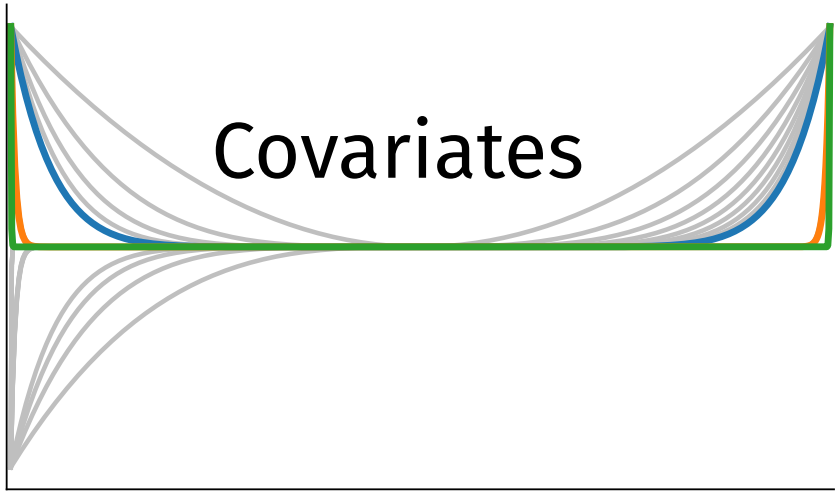




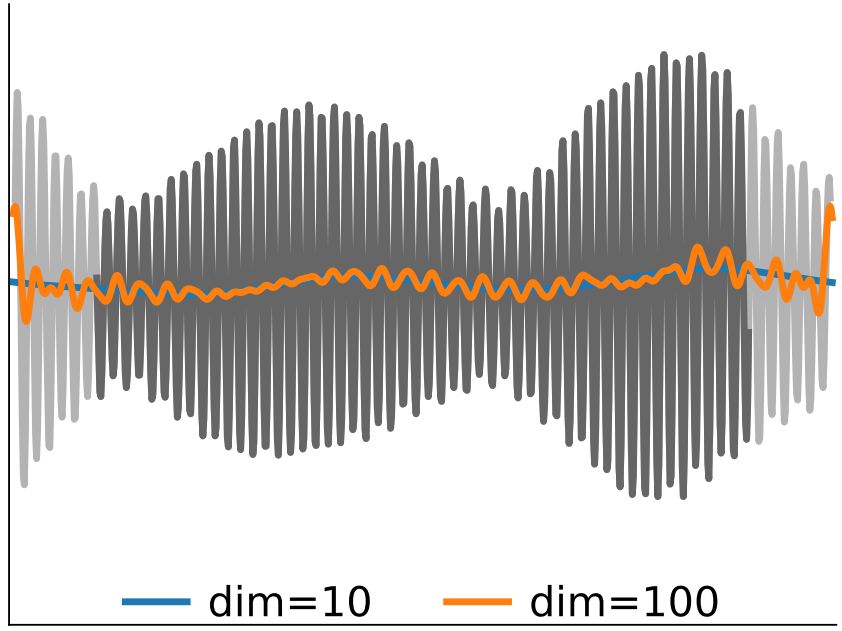
Example: extrapolating sea level (tides)

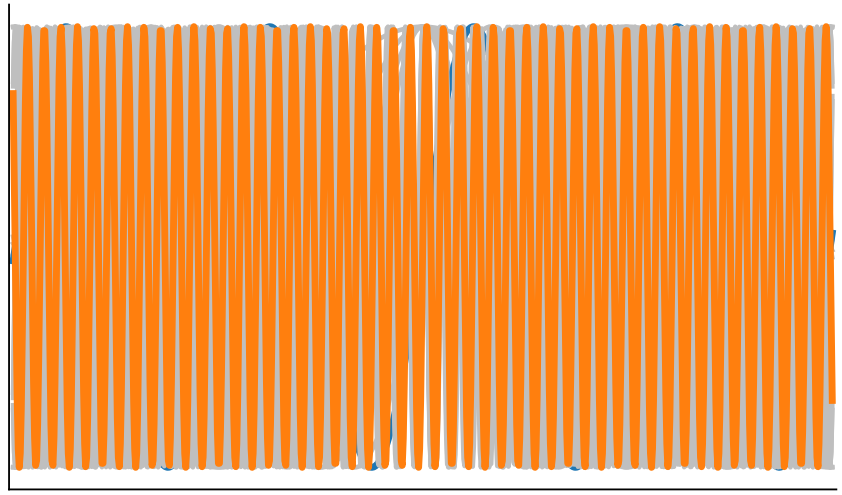
Polynomial regression





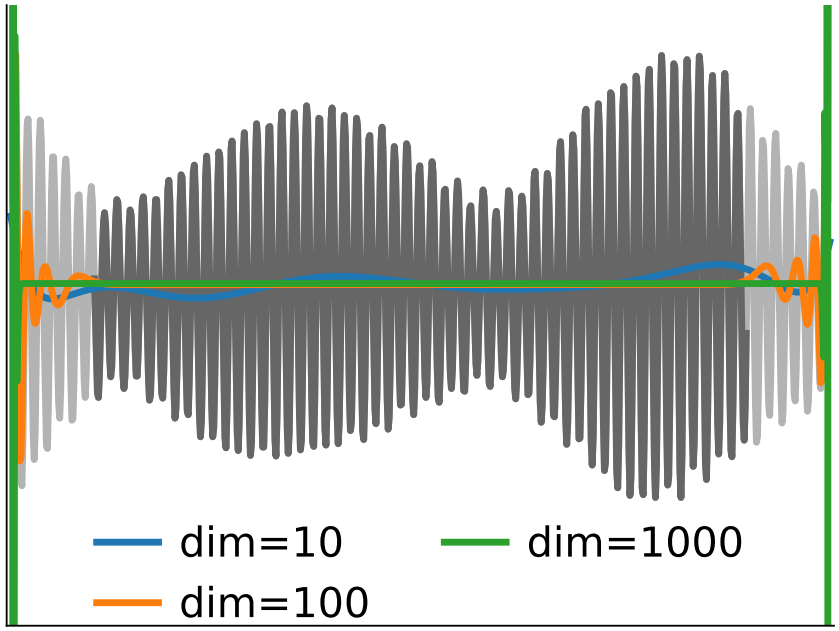
Sines and cosines basis

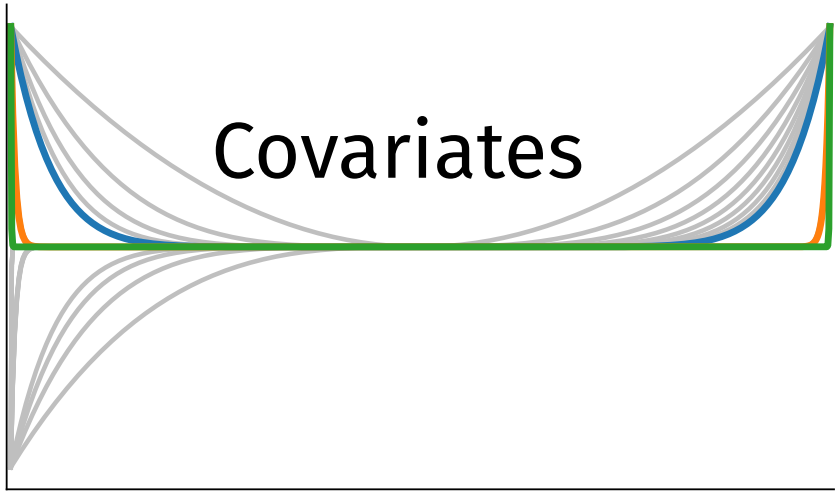




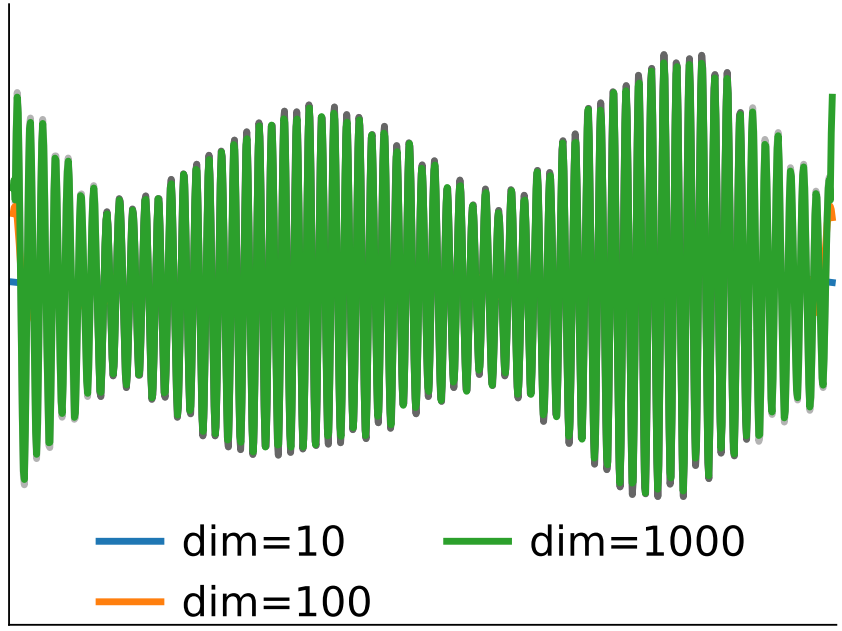
Example: extrapolating sea level (tides)

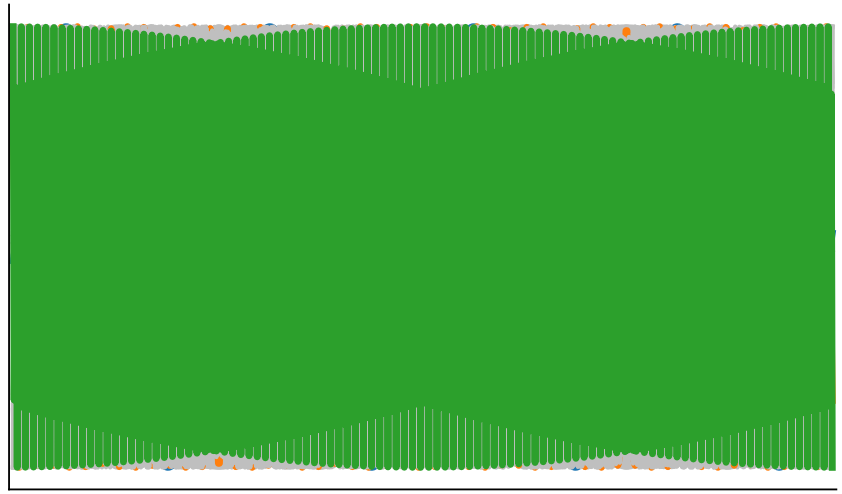
Polynomial regression





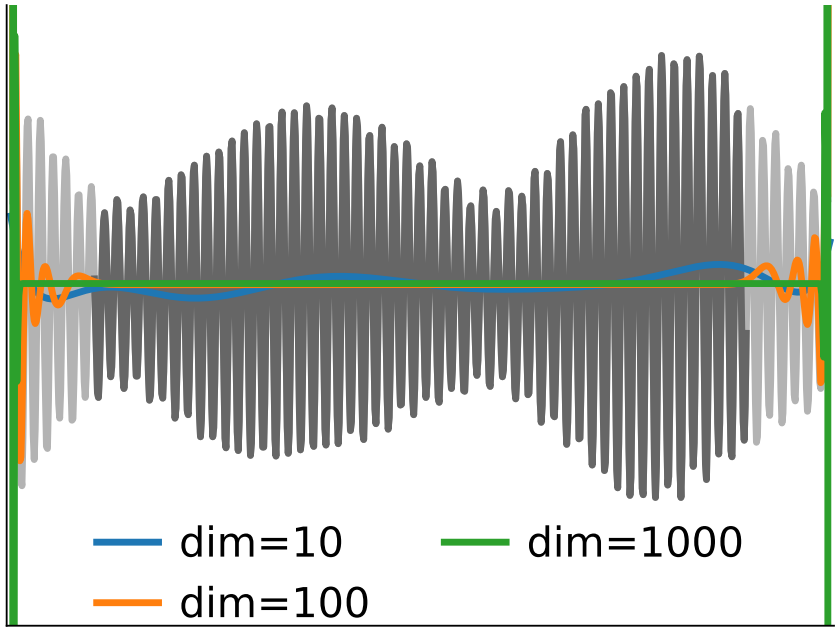
Sines and cosines basis



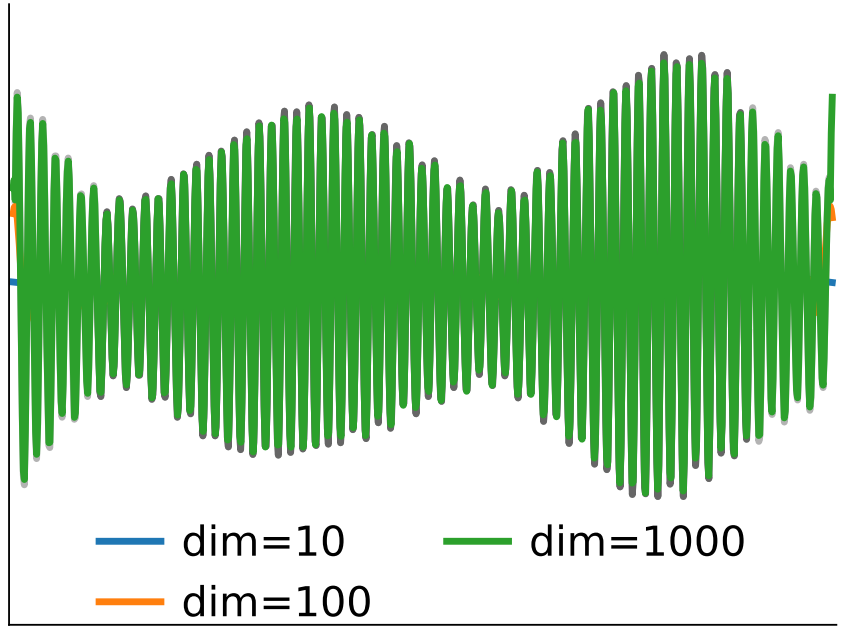


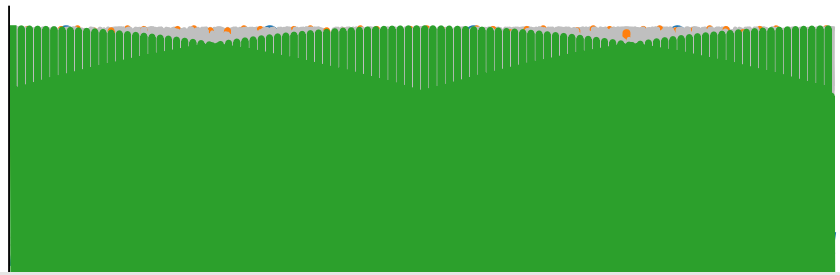
Example: extrapolating sea level (tides)

Polynomial regression



Sines and cosines basis





Choice of covariates / basis / signal representation

$\Rightarrow$ huge difference on approximation error

$\Rightarrow$ huge difference on generalization error

Summary

- $\hat{y} = f(x)$, f chosen in $\mathcal{F}$

  to minimize the observed error $\sum_{i \in \text{train}} l(f(x_i), y)$

- generalization error:
  - approximation error $\Rightarrow \mathcal{F}$ adapted to the data
  - estimation error $\Rightarrow \mathcal{F}$ small

Limited-data settings

- Linear models best option when $p \gg n$
- A good choice of covariates is crucial

1 Representations for machine learning

Non-asymptotic supervised learning

Learning with representations

Supervised learning of representations

Representations to build $\mathcal{F}$

Settings

- $\mathbf{z} = r(\mathbf{x})$: *representation* of the data, $\mathbf{z} \in \mathbb{R}^k$
- Predictor $f : \mathbf{x} \rightarrow \hat{y} = h_{\mathbf{w}}\big(r(\mathbf{x})\big)$

Function composition: “depth”

Representations to build $\mathcal{F}$

Settings

- $\mathbf{z} = r(\mathbf{x})$: *representation* of the data, $\mathbf{z} \in \mathbb{R}^k$
- Predictor $f : \mathbf{x} \rightarrow \hat{y} = h_{\mathbf{w}}\big(r(\mathbf{x})\big)$

Function composition: "depth"

Benefits

- For expressiveness composition $\gg$ basis expansion

Composing L rectifying functions on intermediate representations of dimension k gives $O\left(\left(\frac{k}{p}\right)^{p(L-1)} k^p\right)$ linear regions.

Basis expansion + linear predictor gives $O(k)$

Exponential in depth, linear with dimension [Montufar... 2014]

Representations to build $\mathcal{F}$

Settings

- $\mathbf{z} = r(\mathbf{x})$: *representation* of the data, $\mathbf{z} \in \mathbb{R}^k$
- Predictor $f : \mathbf{x} \rightarrow \hat{y} = h_{\mathbf{w}}\big(r(\mathbf{x})\big)$

Function composition: "depth"

Benefits

- For expressiveness composition $\gg$ basis expansion
- For multi-tasks sharing representations across tasks

  $\mathbf{y}$ multidimensional

Representations to build $\mathcal{F}$

Settings

- $\mathbf{z} = r(\mathbf{x})$: *representation* of the data, $\mathbf{z} \in \mathbb{R}^k$
- Predictor $f : \mathbf{x} \rightarrow \hat{y} = h_{\mathbf{w}}\big(r(\mathbf{x})\big)$

Function composition: "depth"

Benefits

- For expressiveness composition $\gg$ basis expansion
- For multi-tasks sharing representations across tasks
- For limited data $h_{\mathbf{w}}(\mathbf{z}) = \mathbf{w}^{\mathsf{T}}\mathbf{z}$, a linear predictor

A good choice of $\mathbf{z}$ can decrease sample complexity

Representations to build $\mathcal{F}$

Settings

- $\mathbf{z} = r(\mathbf{x})$: *representation* of the data, $\mathbf{z} \in \mathbb{R}^k$
- Predictor $f : \mathbf{x} \rightarrow \hat{y} = h_{\mathbf{w}}\big(r(\mathbf{x})\big)$

Function composition: "depth"

Benefits

- For expressiveness composition $\gg$ basis expansion
- For multi-tasks sharing representations across tasks
- For limited data $h_{\mathbf{w}}(\mathbf{z}) = \mathbf{w}^{\mathsf{T}}\mathbf{z}$, a linear predictor

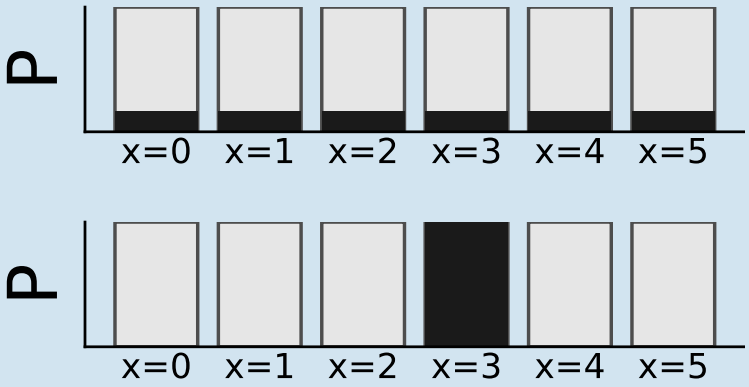
Transfer: r is learned on large data; a simple h used.

Background: Information theory

Entropy = amount of information in x

$$\mathcal{H}(x) = \mathbb{E}_p[-\log p(x)]$$

- Equi-probable distribution
  = high entropy
- Uneven distribution
  = low entropy



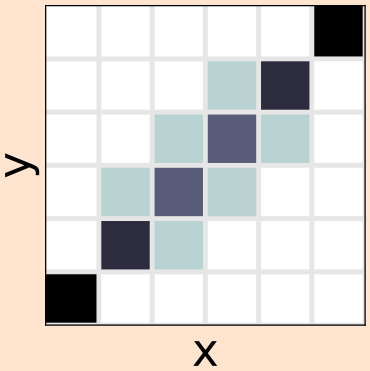
Mutual information between x and y

$$\mathcal{I}(x; y) = \mathcal{H}(x, y) - \mathcal{H}(x) - \mathcal{H}(y)$$

$x \perp\!\!\!\perp y$ (independent) $\Leftrightarrow \mathcal{I}(x; y) = 0$

independence $\Leftrightarrow p(x; y) = p(x)p(y)$

$$\mathcal{H}(x; y) = \mathbb{E}_{(x;y)}\left[\log p(x; y)\right] = \mathbb{E}_{(x;y)}\left[\log p(x) + \log p(y)\right]$$
$$= \mathbb{E}_x\left[\log p(x)\right] + \mathbb{E}_y\left[\log p(y)\right] = \mathcal{H}(x) + \mathcal{H}(y)$$



Theory: information in representations

- A representation z of x is **sufficient for** y if $y \perp\!\!\!\perp x|z$, or equivalently if $\mathcal{I}(z;y) = \mathcal{I}(x;y)$

- x, z, y form a **Markov chain** if $\mathbb{P}(y|x,z) = \mathbb{P}(y|z)$.

$$x \rightarrow z \rightarrow y$$

  Data processing inequality: $\mathcal{I}(x;y) \leq \mathcal{I}(x;z)$

- A sufficient representation z is **minimal** when $\mathcal{I}(x;z)$ is smallest among sufficient representations

[Achille and Soatto 2018]

Nuisances and invariances

- A **nuisance** n: $\mathcal{I}(x, n) \geq 0$, but $\mathcal{I}(y, n) = 0$

- Representation z is **invariant** to the nuisance n if $z \perp\!\!\!\perp n$, or $\mathcal{I}(z; n) = 0$

$\Rightarrow$ We want $\mathcal{I}(z; n)$ low

In a Markov chain $x \rightarrow z_1 \rightarrow z_2 \cdots \rightarrow z_L \rightarrow y$

- If z is a **sufficient representation** for y,

$$\mathcal{I}(z; n) \leq \mathcal{I}(z; x) - \mathcal{I}(x; y)$$

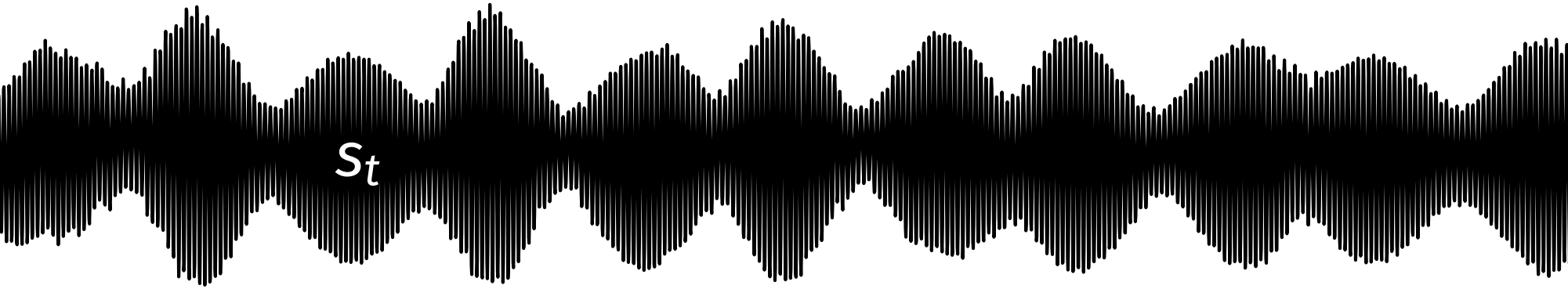
- Communication bottleneck: $\mathcal{I}(z_1; z_2) < \mathcal{I}(z_1; x)$

$$\Rightarrow \mathcal{I}(z_2; n) \leq \mathcal{I}(z_1; z_2) - \mathcal{I}(x; y)$$

Stacking increases invariance

[Achille and Soatto 2018]

Invariant representations on a continous space



Shift invariance representation = Fourier basis

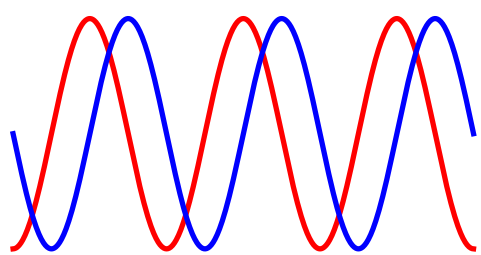
Fourier transform: $F(s)_f = \sum_t e^{-i f t} s_t$

complex i

Shifting the signal: $s_t \rightarrow s'_t = s_{t+k}$

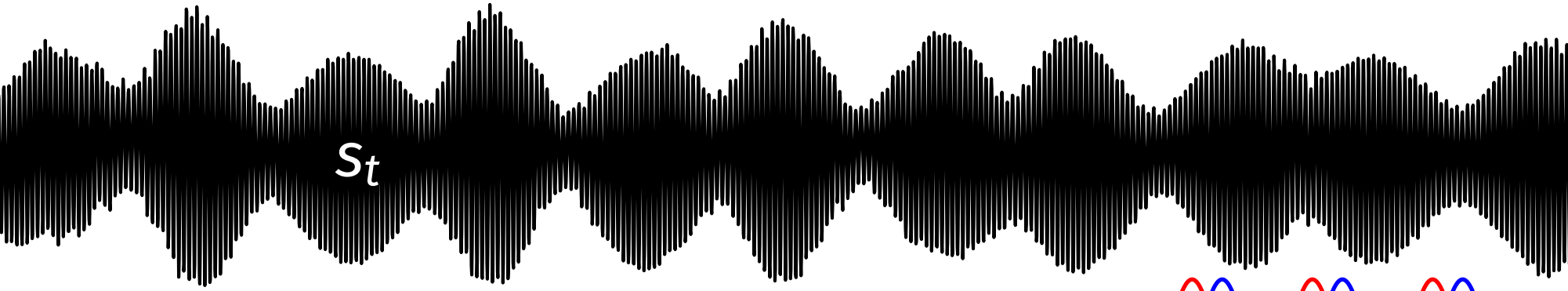
$$F(s')_f = \sum_t e^{-i f t} s_{t+k} = \sum_t e^{-i f (t-k)} s_t = e^{i k f} \sum_t e^{-i f t} s_t$$

$$= e^{i k f} F(s)_f \quad \rightarrow \text{change in phase}$$

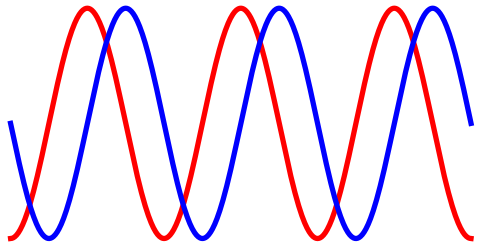


An orthonormal basis
of shift-invariant vectors

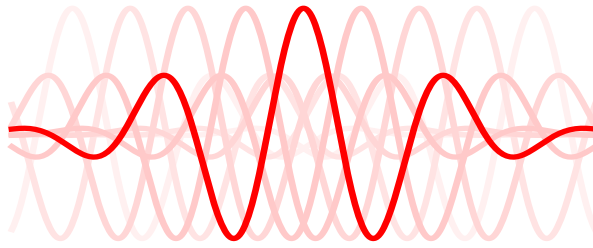
Invariant representations on a continous space



Shift invariance = Fourier basis



Local deformations = Wavelets



- Locally equivalent to Fourier basis
- But without the global extent

Decimated wavelets

Isometric transform of the signal

Higher scales lose shift invariance

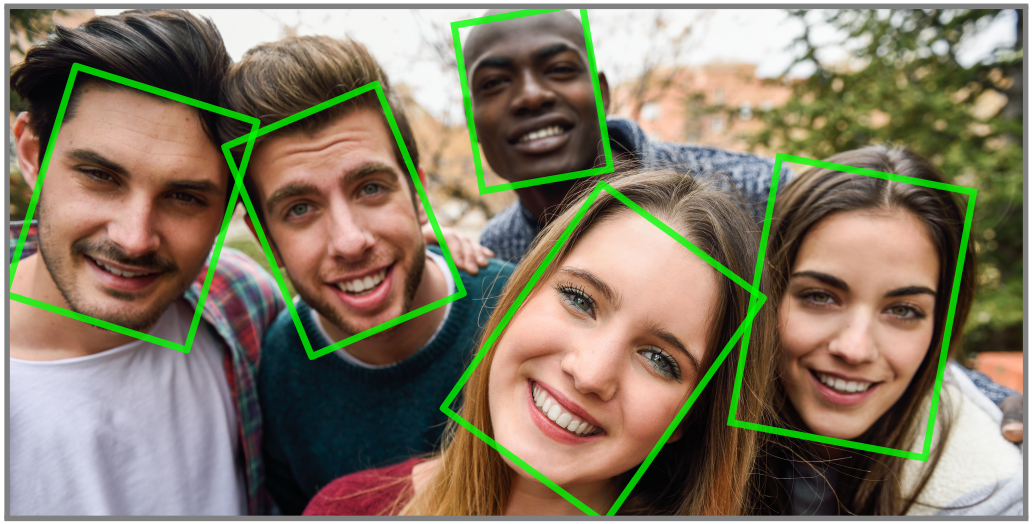
Redundant wavelets

Increase the dimensionality

Good shift invariance

Representations invariant to rich deformations

- Scaling
- Rotations
- Deformations



Ingredients

- Modulus of wavelet / Fourier transform
  $\Rightarrow$ non linearity & filter banks (convolutions)
  + stacking (repeating simple invariants)

Scattering transform
Derived from first principles
Building first-order invariants

Convolutional networks
Learned from data
Pooling across pixels (*eg* max)

[Mallat 2016]

Summary

- Intermediate representations give expressiveness to predictive models
- Good representations keep predictive information and loose nuisance information
- Bottleneck and regularization to loose information

Limited-data settings

Given know invariants of the problem, reusing existing representations helps

eg Headless conv-net, wavelets... [Oyallon... 2017]

1 Representations for machine learning

Non-asymptotic supervised learning

Learning with representations

Supervised learning of representations

The need to supervision

Maximizing $\mathcal{I}(z; y)$ $(\leq \mathcal{I}(x; y))$ sufficient representations

$\Rightarrow$ supervised learning

while minimizing $\mathcal{I}(z; n)$ nuisance

$\Rightarrow$ sampling nuisance / invariants

data augmentation

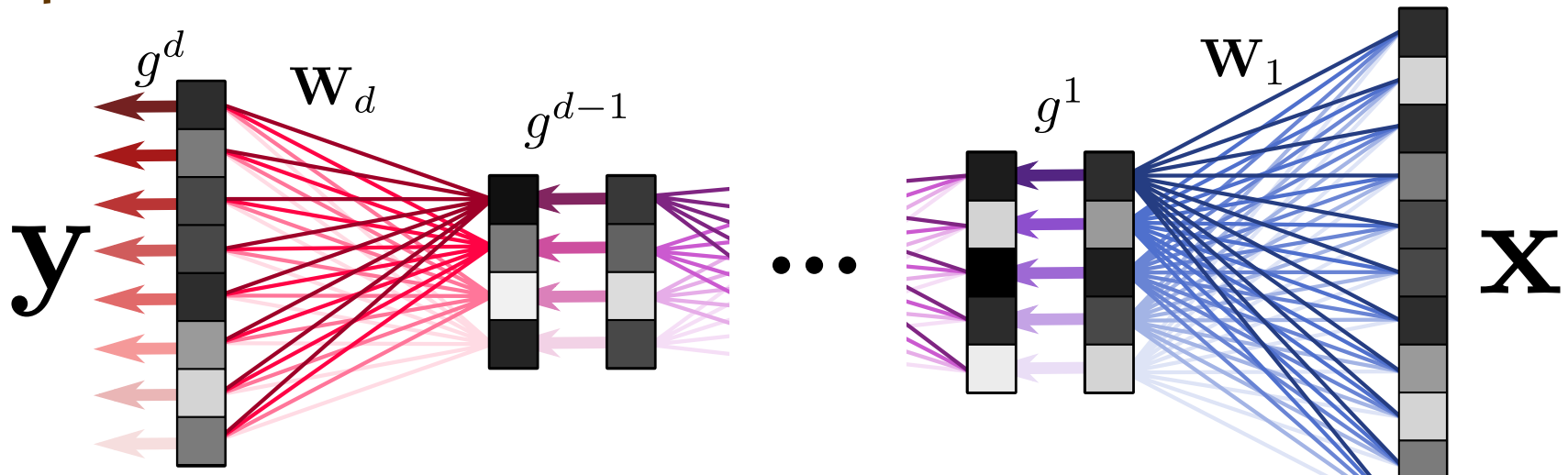
Challenge: amount of labeled data

Pretext tasks

Other targets y' that capture useful information

Finding them needs domain knowledge

Deep architectures



$$\hat{\mathbf{y}} = f^d_{\mathbf{W}_d} \circ \dots \circ f^1_{\mathbf{W}_1}(\mathbf{x})$$

Typically $f^k_{\mathbf{W}_k}(\mathbf{x}) = g^k((\mathbf{W}_k^\mathsf{T}\mathbf{x}))$ and g^k element-wise non-linearity

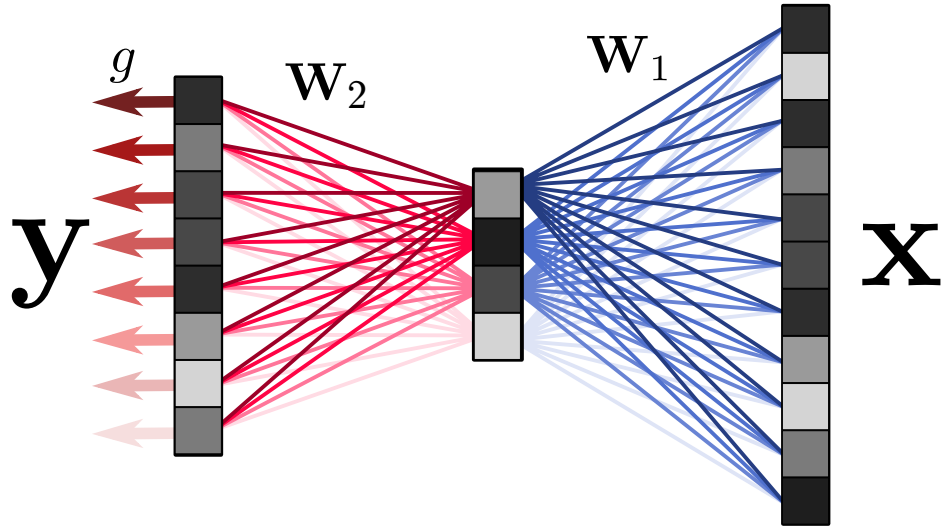
Thus $\hat{\mathbf{y}} = g^d\left(\left(\mathbf{W}_d^\mathsf{T} \dots g^1((\mathbf{W}_1^\mathsf{T}\mathbf{x}))\right)\right)$

Stacked representations: $\mathbf{W}_k$

$\{\mathbf{W}_k\}$ optimized to minimize a prediction error

Shallow architectures for limited data

Keep one latent layer



- Without non-linearity:

$$\hat{\mathbf{y}} = \mathbf{x}^\mathsf{T} \mathbf{W}_1 \mathbf{W}_2, \qquad \mathbf{y} \in \mathbb{R}^k \; \mathbf{W}_1 \in \mathbb{R}^{p \times d} \; \mathbf{W}_2 \in \mathbb{R}^{d \times k},$$

factored / reduced-rank linear model

- Multi-task / multi-output literature

$\Longrightarrow$ structured loss (multiple soft-max's)

- Overparametrization sometimes useful: $d > k$

can be achieved with dropout

[Bzdok... 2015, Mensch... 2018]

Simple case: square loss = reduced rank regression

$$\hat{\mathbf{Y}} = \mathbf{X}\mathbf{W}_1\mathbf{W}_2, \qquad \mathbf{Y} \in \mathbb{R}^{n\times k}\ \mathbf{W}_1 \in \mathbb{R}^{p\times d},\ \mathbf{W}_2 \in \mathbb{R}^{d\times k}$$

$$\hat{\mathbf{W}}_1, \hat{\mathbf{W}}_2 = \underset{\mathbf{W}_1,\mathbf{W}_2}{\operatorname{argmin}} \|\hat{\mathbf{Y}} - \mathbf{Y}_{\text{train}}\|^2_{\text{Fro}}$$

For squared loss the problem is convex

- Full-rank solution[1] ($\mathbf{X}$ and $\mathbf{Y}$ on train set):

$$\hat{\mathbf{W}} = \hat{\mathbf{\Sigma}}_{\mathbf{X}}^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{Y} \qquad \hat{\mathbf{Y}} = \mathbf{X}\hat{\mathbf{W}} = \mathbf{X}\hat{\mathbf{\Sigma}}_{\mathbf{X}}^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{Y}$$

- Rank d solution: [Izenman 1975, Rahim... 2017b]

$$\hat{\mathbf{R}}_d \overset{\text{def}}{=} \mathbf{Y}^{\mathsf{T}}\hat{\mathbf{Y}} \in \mathbb{R}^{k\times k} \overset{\text{SVD}}{\longrightarrow} = \hat{\mathbf{U}}_d\,\hat{\mathbf{s}}_d\hat{\mathbf{V}}_d, \quad \hat{\mathbf{U}}_d \in \mathbb{R}^{k\times d}$$

$$\text{then } \hat{\mathbf{W}}_1 = \mathbf{\Sigma}_{\mathbf{X}}^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{Y}\,\hat{\mathbf{U}}_d \qquad \hat{\mathbf{W}}_2 = \hat{\mathbf{U}}_d^{\mathsf{T}}$$

Full-rank solution — Rank-k projector[2]

[1] No need for pesky SGDs

[2] The projector captures the variance explained on the multiple outputs

Model stacking

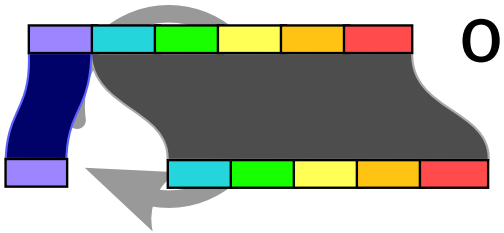
$$x \xrightarrow{f_1} z \xrightarrow{f_2} y$$

Learn f_1 separately

Directly supervising z:

$z = \hat{y}$ for a (simple) predictive model

Trick: "cross-fit" during training



obtain $\hat{y}$ by splitting the training data

(in `sklearn`: `cross_val_predict`)

Model stacking

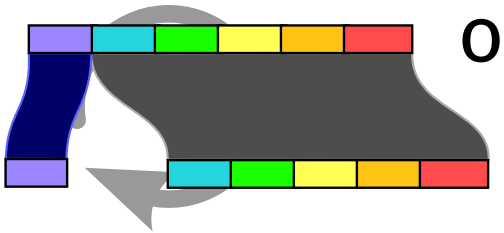
$$x \xrightarrow{f_1} z \xrightarrow{f_2} y$$

Learn f_1 separately

Directly supervising z:

$z = \hat{y}$ for a (simple) predictive model

Trick: "cross-fit" during training



obtain $\hat{y}$ by splitting the training data

(in `sklearn`: `cross_val_predict`)

Application: tackling dimensionality [Rahim... 2017a]

Some features are a high-dimensional signal

eg medical images

- f_1: linear to reduce signal features
- f_2: non-linear (*eg* trees) on all features

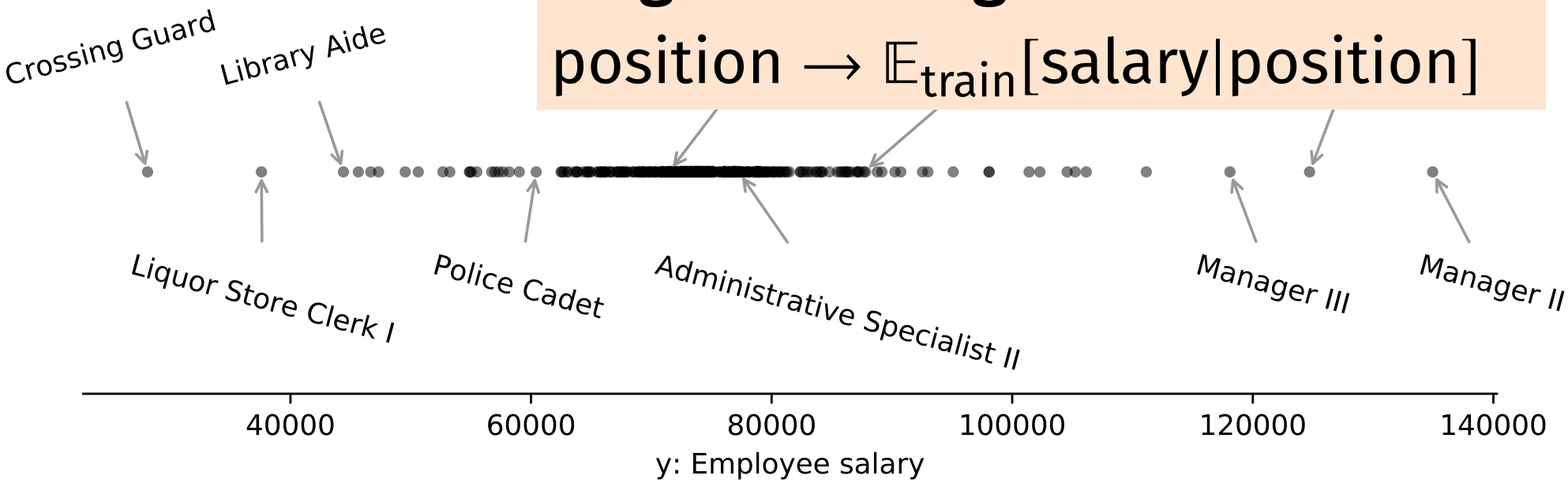
Model stacking to encode discrete items

Sex	Date Hired	Employee Position		Salary
M	09/12/1988	Master Police Officer	predict ⟶	69222.18
F	06/26/2006	Social Worker III		97392.47
M	07/16/2007	Police Officer III		104717.28

Difficulty: number of different positions
what invariants?

Target encoding[1] [Micci-Barreca 2001]

position ⟶ $\mathbb{E}_{\text{train}}[\text{salary}|\text{position}]$



[1] To inject categories in $\mathbb{R}$, before a second level that combines all columns

Summary

Supervision helps selecting the relevant part of the signal

In limited-sample settings, simple models can create representations

- Simple latent-factor models
- Multi-output models
- Stacking: fit a first-level model

Summary of first section

For generalization: small family of functions $f_{\mathbf{w}}$ that **approximate the signal** well

Generalization of a linear predictor:

approximation error + $o(p/n_{\text{train}})$

Predictors by composition: $\hat{y} = f_2(z), \quad z = f_1(x)$

$x \xrightarrow{f_1} z \xrightarrow{f_2} y$ ideally, f_1 makes z invariant to nuisances

Reuse representations with the right invariances:

wavelets, fasttext, pretrained headless neural nets

Simple supervised models

can create representations

- stacking
- multioutput
- pretext tasks

2 Matrix factorization and its variants

Simple unsupervised representation learning

- More unlabeled data than labeled data
- Learn representations and transfer them

Here: Focus on simple models for limited n or low SNR settings

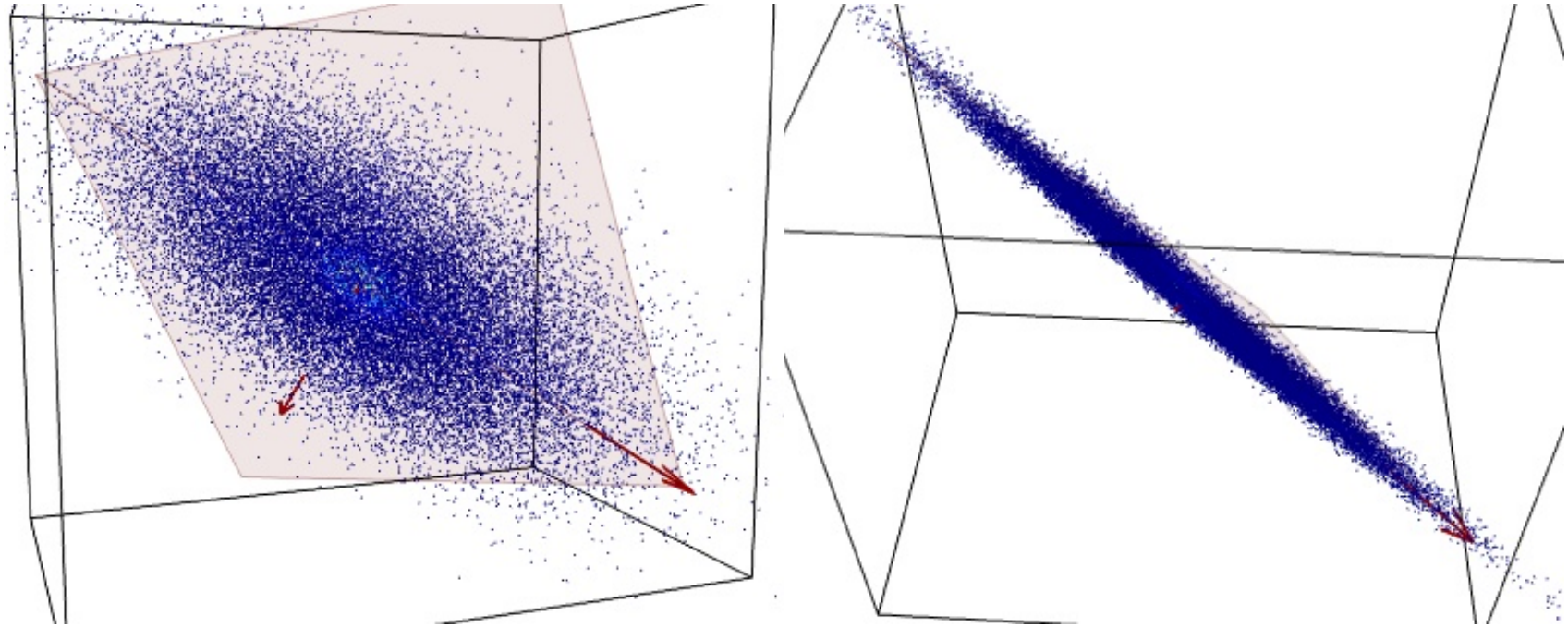
Particularly interesting regime: p large and n large.

2 Matrix factorization and its variants

- For signals
- For discrete objects

Principal Component Analysis

Find the directions of largest variance



Computation

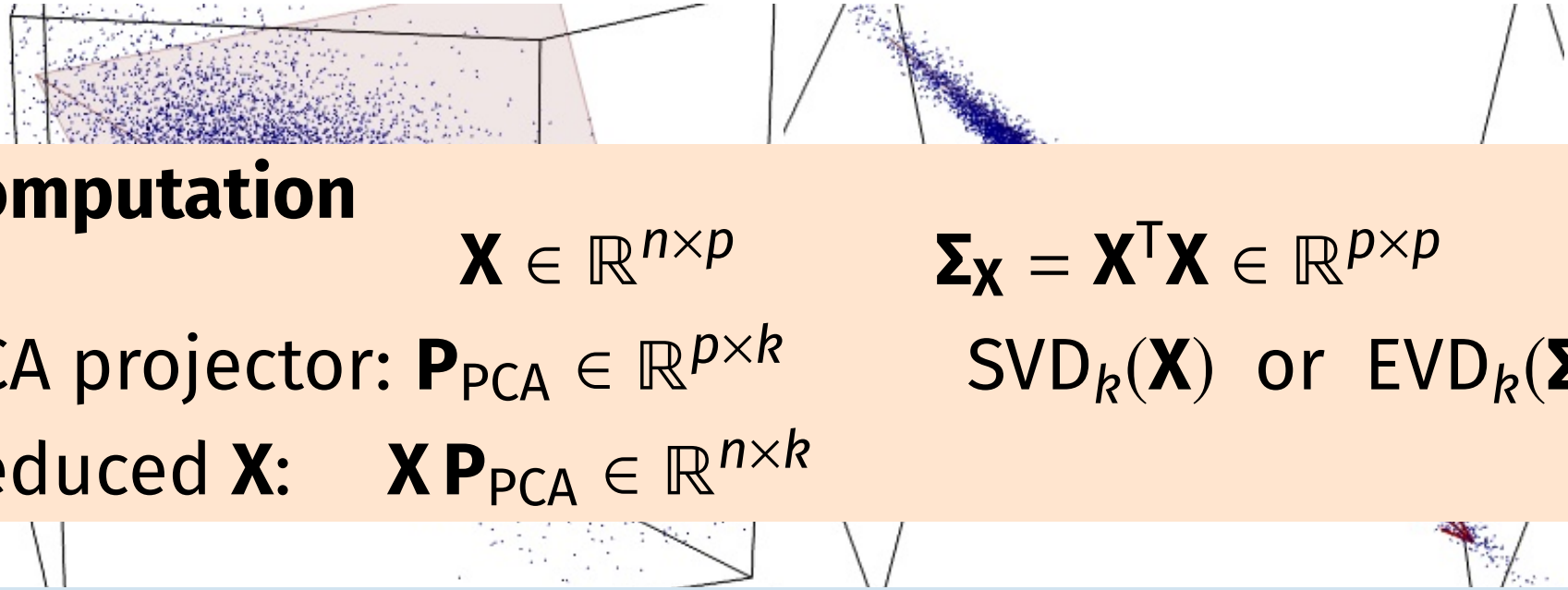
$\mathbf{X} \in \mathbb{R}^{n \times p}$ $\quad$ $\mathbf{\Sigma_X} = \mathbf{X}^\mathsf{T}\mathbf{X} \in \mathbb{R}^{p \times p}$

PCA projector: $\mathbf{P}_{\text{PCA}} \in \mathbb{R}^{p \times k}$ $\quad$ $\text{SVD}_k(\mathbf{X})$ or $\text{EVD}_k(\mathbf{\Sigma_X})$

Reduced $\mathbf{X}$: $\quad \mathbf{X}\,\mathbf{P}_{\text{PCA}} \in \mathbb{R}^{n \times k}$

Principal Component Analysis

Find the directions of largest variance



Computation

$\mathbf{X} \in \mathbb{R}^{n \times p}$ $\quad \mathbf{\Sigma_X} = \mathbf{X}^\mathsf{T}\mathbf{X} \in \mathbb{R}^{p \times p}$

PCA projector: $\mathbf{P}_{\text{PCA}} \in \mathbb{R}^{p \times k}$ $\quad \text{SVD}_k(\mathbf{X})$ or $\text{EVD}_k(\mathbf{\Sigma_X})$

Reduced $\mathbf{X}$: $\quad \mathbf{X}\,\mathbf{P}_{\text{PCA}} \in \mathbb{R}^{n \times k}$

Model: low-rank Gaussian latent factors

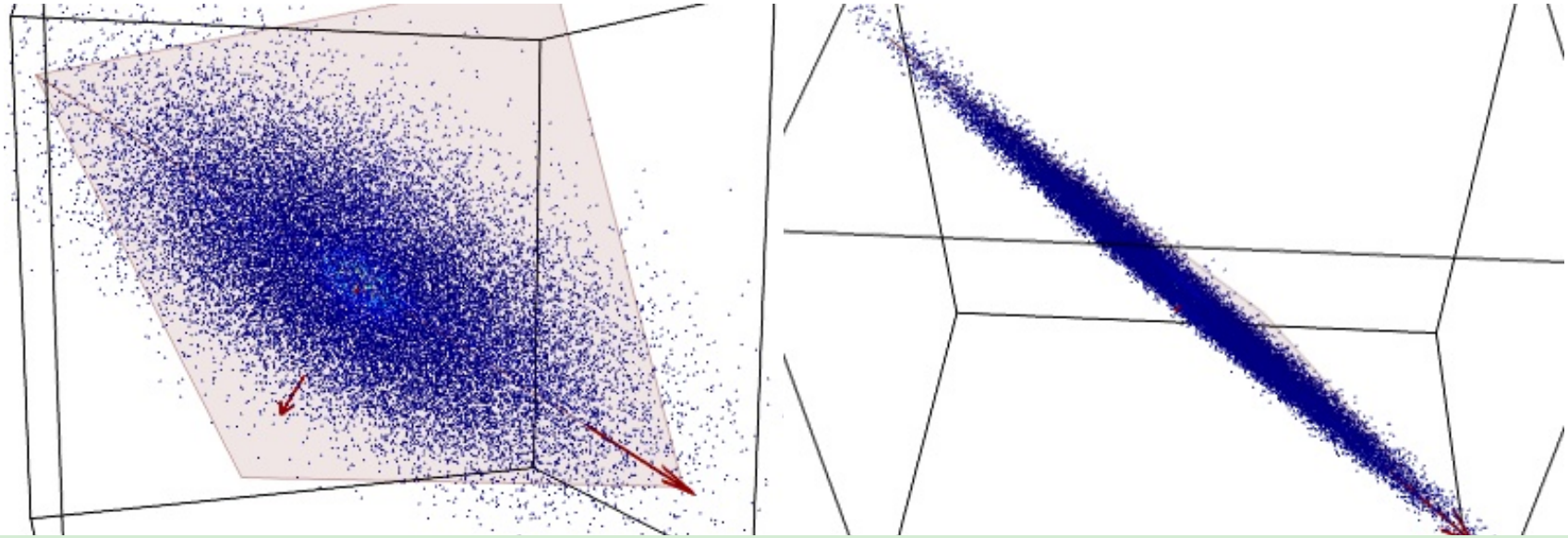
$$\mathbf{X} \approx \mathbf{U}\,\mathbf{V} + \mathbf{E}, \qquad \mathbf{E} \sim \mathcal{N}(0, \mathbf{I}_p),\ \mathbf{U} \in \mathbb{R}^{n \times k},\ \mathbf{V} \in \mathbb{R}^{k \times p}$$

$$\hat{\mathbf{U}}, \hat{\mathbf{V}} = \underset{\mathbf{U},\mathbf{V}}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{U}\,\mathbf{V}\|^2_{\text{Fro}}$$

Rotationally invariant: $\mathbf{U}' = \mathbf{U}\,\mathbf{O}, \mathbf{O}^\mathsf{T}\,\mathbf{V}$ also solution for $\mathbf{O}$ s.t. $\mathbf{O}^\mathsf{T}\mathbf{O} = \mathbf{I}$

Principal Component Analysis

Find the directions of largest variance

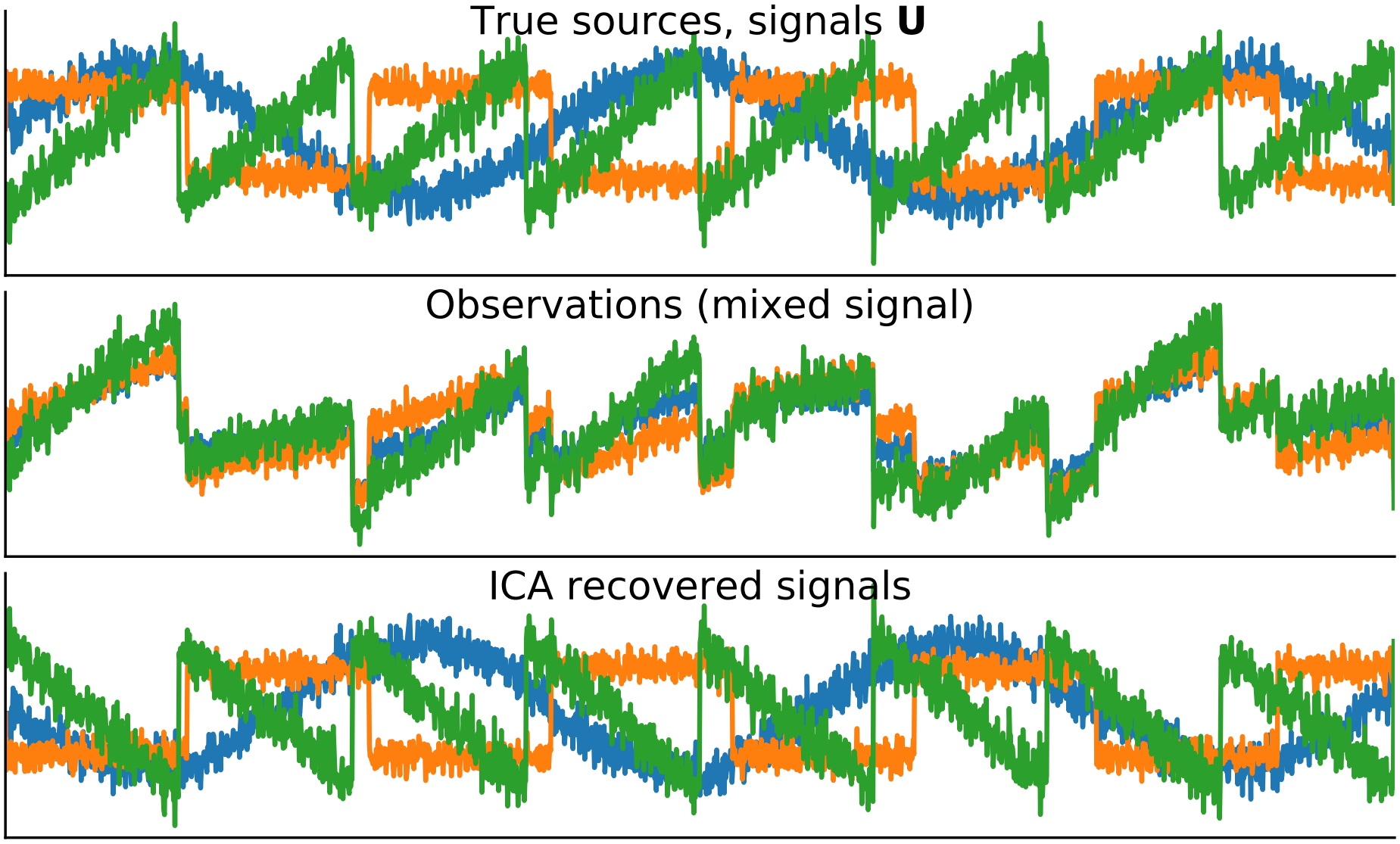


In a learning pipeline

- Useful for dimensionality reduction (eg p is large)
  Eases statistics and computations
- Generalization error of PCA + OLS
  within a factor of 4 of ridge
  [Dhillon... 2013]

Beyond variance: Independent Component Analysis

Separate out signals **U** observed mixed[1]



[1] Classic ICA has no noise model: it does not do dimension reduction

Beyond variance: Independent Component Analysis

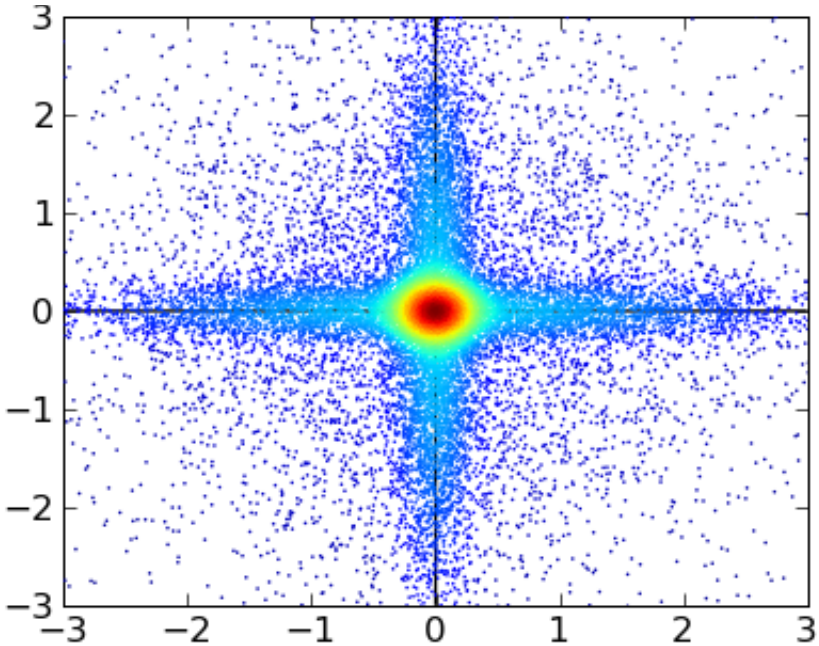
Separate out signals **U** observed mixed[1]

Model: $\mathbf{X} = \mathbf{U}\,\mathbf{V}$ $\quad \mathbf{V} \in \mathbb{R}^{p \times p}$, $\mathbf{V}^T\mathbf{V} = \mathbf{I}_p$

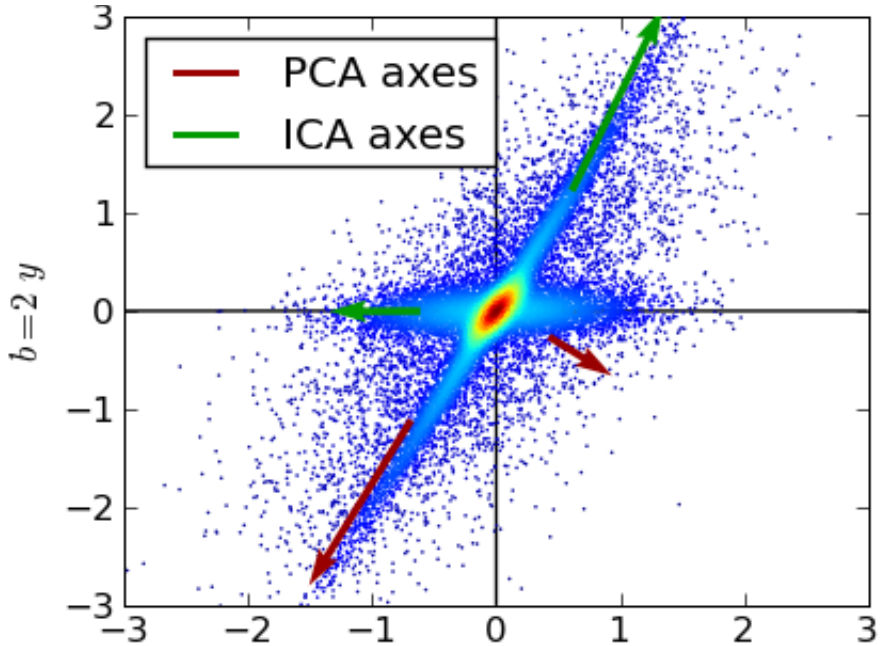
If **V** is Gaussian, the model is not identifiable

Seek low mutual information across $\{\mathbf{u}_j\}$

$\Rightarrow$ Maximally non-Gaussian marginals [Cardoso 2003]



Latent signals **V**



Observed data **U V**

[1] Classic ICA has no noise model: it does not do dimension reduction

Beyond variance: Independent Component Analysis

Separate out signals $\mathbf{U}$ observed mixed[1]

Model: $\mathbf{X} = \mathbf{U}\mathbf{V}$ $\quad \mathbf{V} \in \mathbb{R}^{p \times p}$, $\mathbf{V}^T\mathbf{V} = \mathbf{I}_p$

If $\mathbf{V}$ is Gaussian, the model is not identifiable

Seek low mutual information across $\{\mathbf{u}_j\}$

$\Rightarrow$ Maximally non-Gaussian marginals [Cardoso 2003]

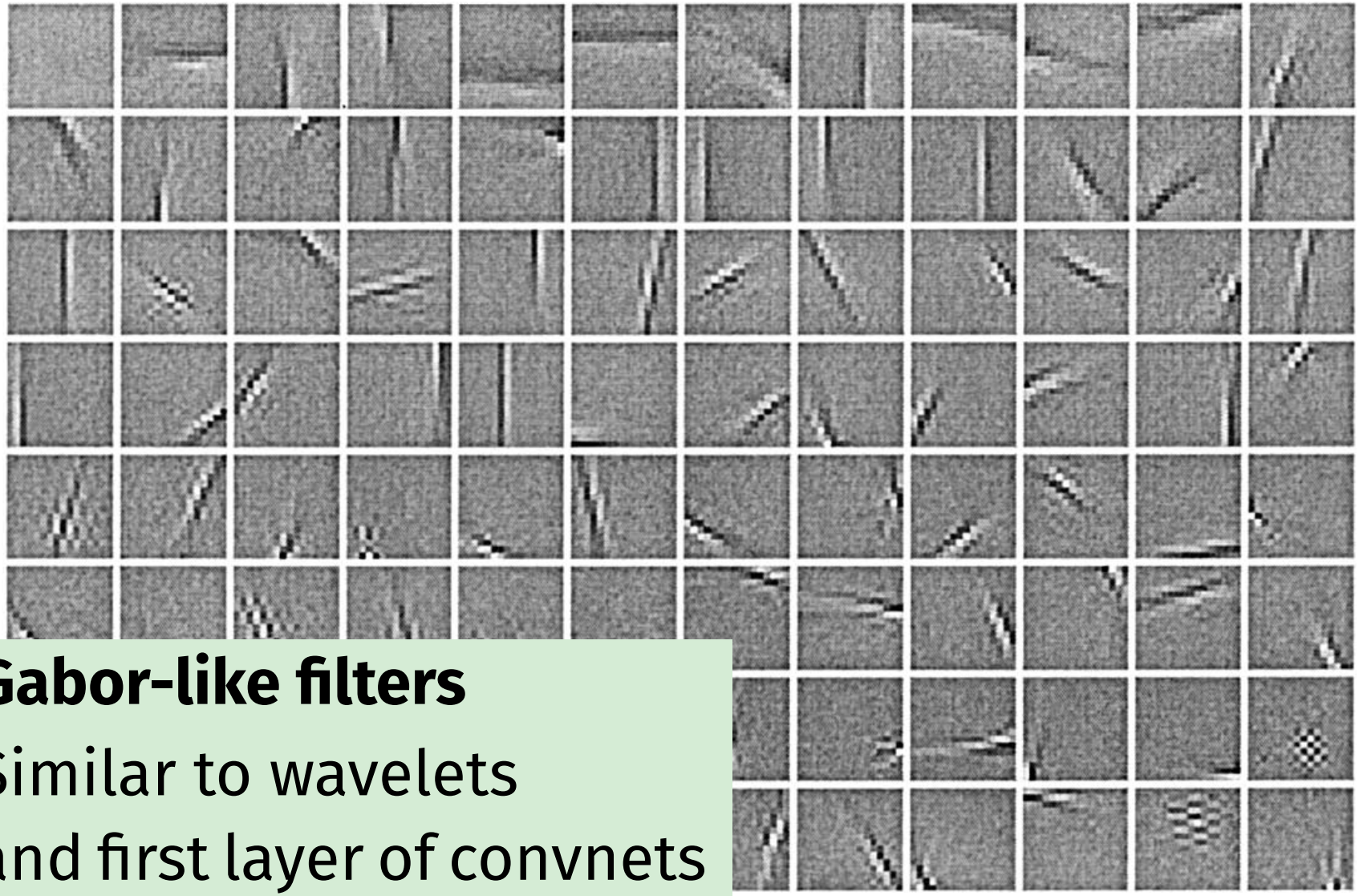
Computation: FastICA [Hyvärinen and Oja 2000]

- Power iterations on $\mathbf{V}$
- Each time:
  - apply a smooth increasing non-linearity on $\{\mathbf{u}_j\}$
  - decorrelate
- Preprocessing: whiten the data *eg* with PCA

[1] Classic ICA has no noise model: it does not do dimension reduction

ICA to learn representations

Across patches of natural images:



Gabor-like filters

Similar to wavelets

and first layer of convnets

[Hyvärinen and Oja 2000]

Dictionary learning

Find vectors $\mathbf{V}$ that represents well the signal with sparse combinations $\mathbf{U}$

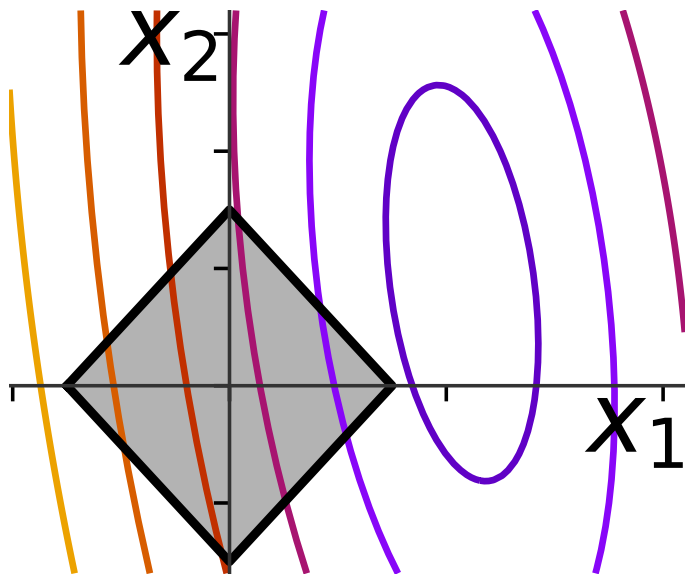
Model: $\mathbf{X} = \mathbf{U}\mathbf{V}$ s.t. $\mathbf{U}$ is sparse

$$\mathbf{U} \in \mathbb{R}^{n \times k},\ \mathbf{V} \in \mathbb{R}^{k \times p}$$

k can be $> p$ (overcomplete dictionary)

Estimation: $$\hat{\mathbf{U}}, \hat{\mathbf{V}} = \underset{\substack{\mathbf{U},\mathbf{V}, \\ \text{s.t. } \|\mathbf{v}_i\|_2^2 \leq 1}}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{U}\mathbf{V}\|_{\text{Fro}}^2 + \lambda \|\mathbf{U}\|_1$$

- Combining squared loss and ℓ_1 penalty creates sparsity
- Constraint on $\|\mathbf{v}_i\|_2^2$ required to avoid cancelling out penalty with $\mathbf{V} \to \infty$ and $\mathbf{U} \to 0$



Dictionary learning

Find vectors $\mathbf{V}$ that represents well the signal with sparse combinations $\mathbf{U}$

Model: $\mathbf{X} = \mathbf{U}\,\mathbf{V}$ s.t. $\mathbf{U}$ is sparse

$$\mathbf{U} \in \mathbb{R}^{n \times k},\ \mathbf{V} \in \mathbb{R}^{k \times p}$$

k can be $> p$ (overcomplete dictionary)

Estimation: $$\hat{\mathbf{U}}, \hat{\mathbf{V}} = \underset{\substack{\mathbf{U},\mathbf{V}, \\ \text{s.t. } \mathbf{V} \in \mathcal{C}}}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{U}\,\mathbf{V}\|_{\text{Fro}}^2 + \lambda \Omega(\mathbf{U})$$

Constraint set and penalty can be varied[1]

Typically, ℓ_2, ℓ_1, and positivity[2] on $\mathbf{U}$ or $\mathbf{V}$.

[1] Fast when $\mathcal{C}$ and Ω lead to simple projections and penalized regression.

[2] Recovers a form of NMF (non-negative matrix factorization)

Sparse dictionary learning to learn representations

Across patches of natural images:

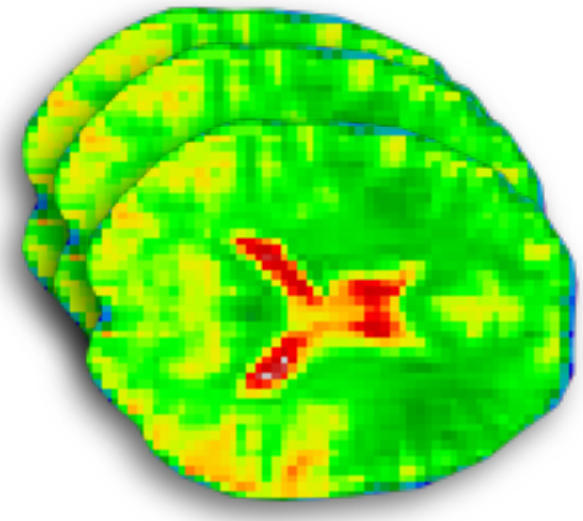
Also learns Gabor-like filters[a]

Good for sparse models,
eg for denoising

[a] as ICA, K-Means, *etc* on images patches

[Mairal... 2014]

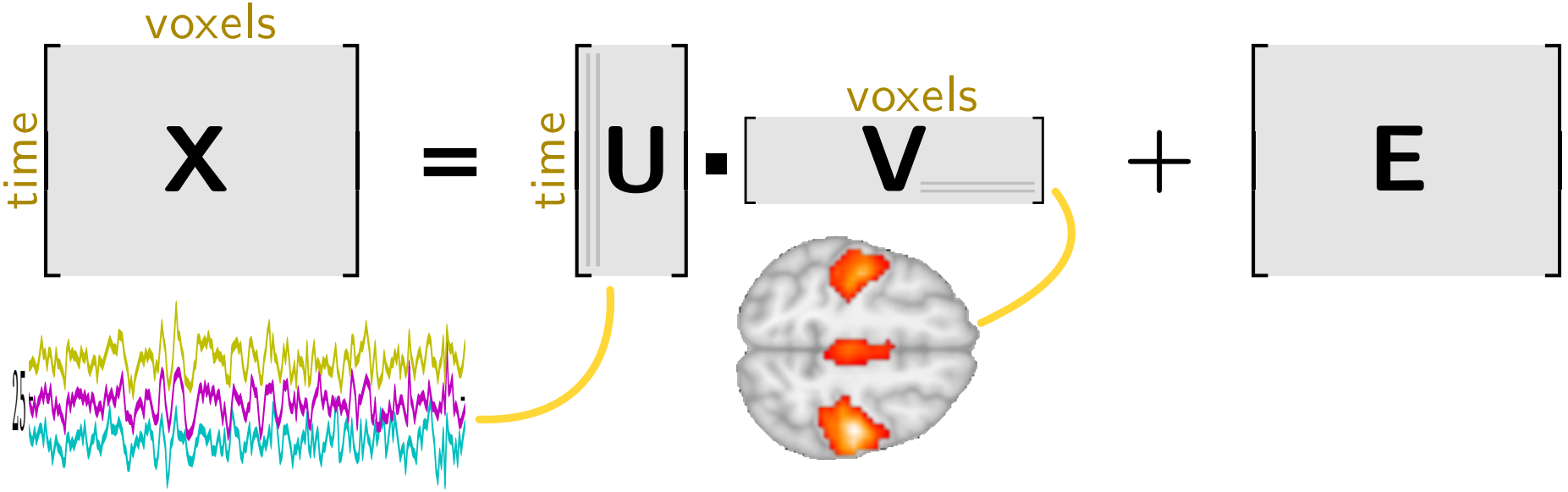
Large *n* large *p*: brain imaging



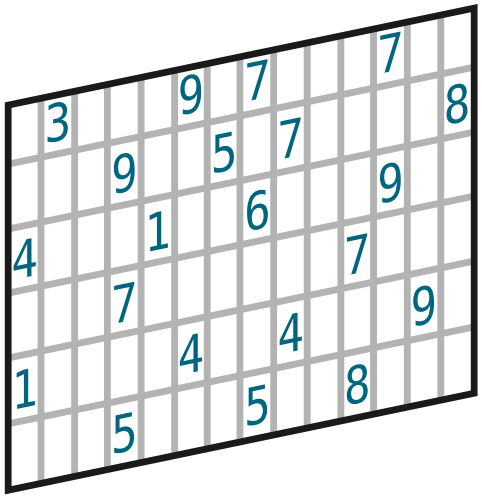
Brain activity at rest

- 1000 subjects with
  ~ 100–10 000 samples
- Images of dimensionality
  > 100 000

Dense matrix, large both ways



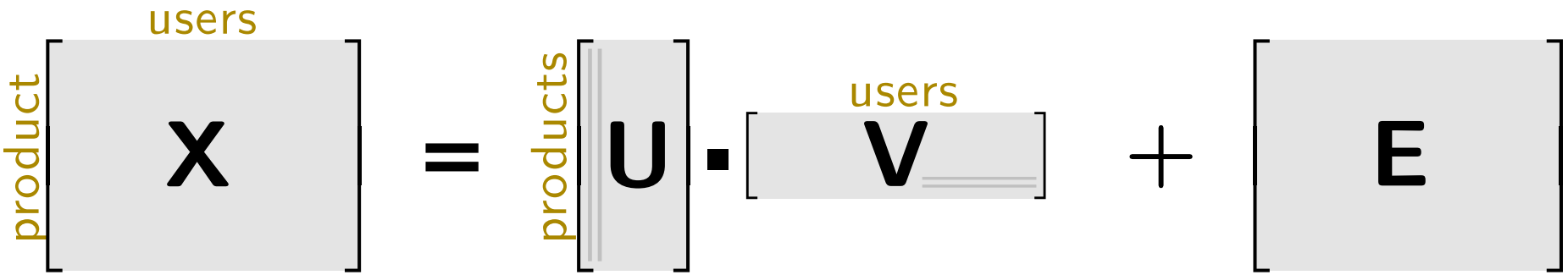
Large *n* large *p*: recommender systems



Product ratings

- Millions of entries
- Hundreds of thousands of products and users

Large sparse matrix



Online estimation: stochastic optimization

$$\min_{\mathbf{w}} \sum_i l(\mathbf{x}_i, \mathbf{w})$$

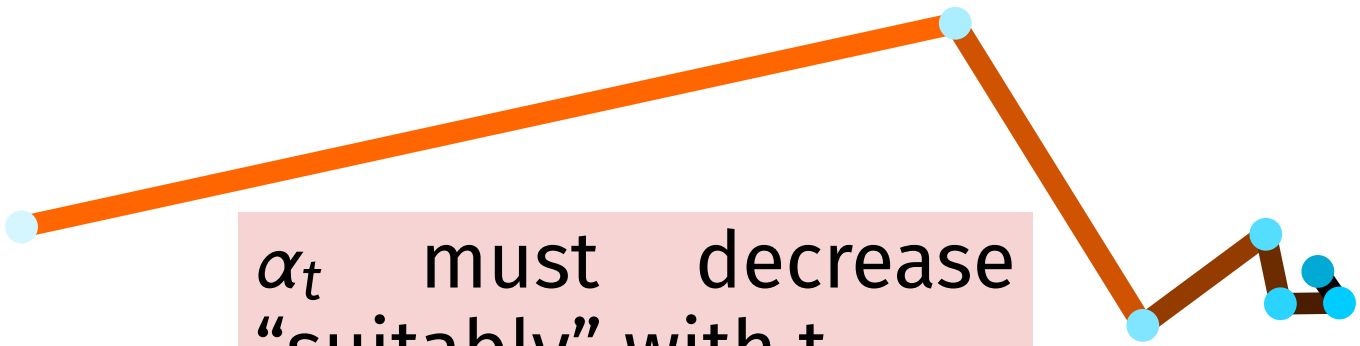
Many samples

$$\min_{\mathbf{w}} \mathbb{E}[l(y, \mathbf{x}\,\mathbf{w})]$$

Gradient descent: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_t \nabla_{\mathbf{w}} l$

Stochastic gradient descent: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_t \mathbb{E}[\nabla_{\mathbf{w}} l]$

Use a cheap estimate of $\mathbb{E}[\nabla_{\mathbf{w}} l]$ (*e.g.* subsampling)



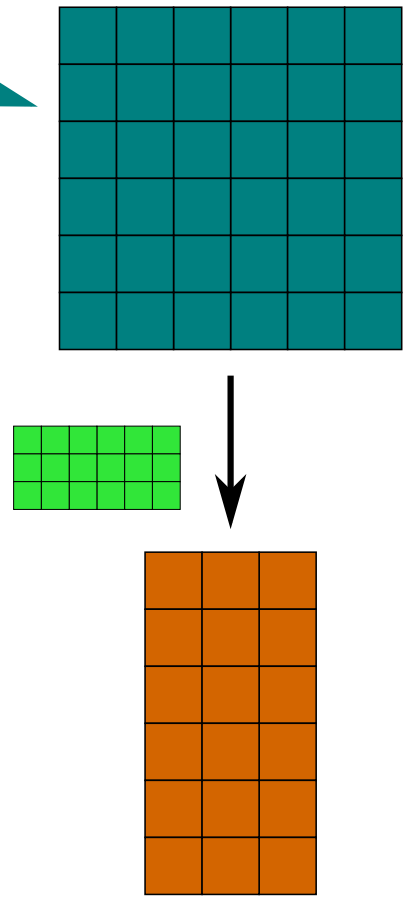
α_t must decrease "suitably" with t.

Those pesky learning rate

Online estimation for matrix factorization

Data matrix

Alternating minimization



- Data access
- Code computation
- Dictionary update

Large matrices = terabytes of data

$$\underset{\mathbf{U},\mathbf{V}}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{U}\,\mathbf{V}\|_{\text{Fro}}^2 + \lambda\Omega(\mathbf{U})$$

[Mairal... 2010]

Online estimation for matrix factorization

Large matrices
= terabytes of data

$$\underset{\mathbf{U},\mathbf{V}}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{U}\,\mathbf{V}\|_{\text{Fro}}^2 + \lambda\Omega(\mathbf{U})$$

Rewrite as an expectation:

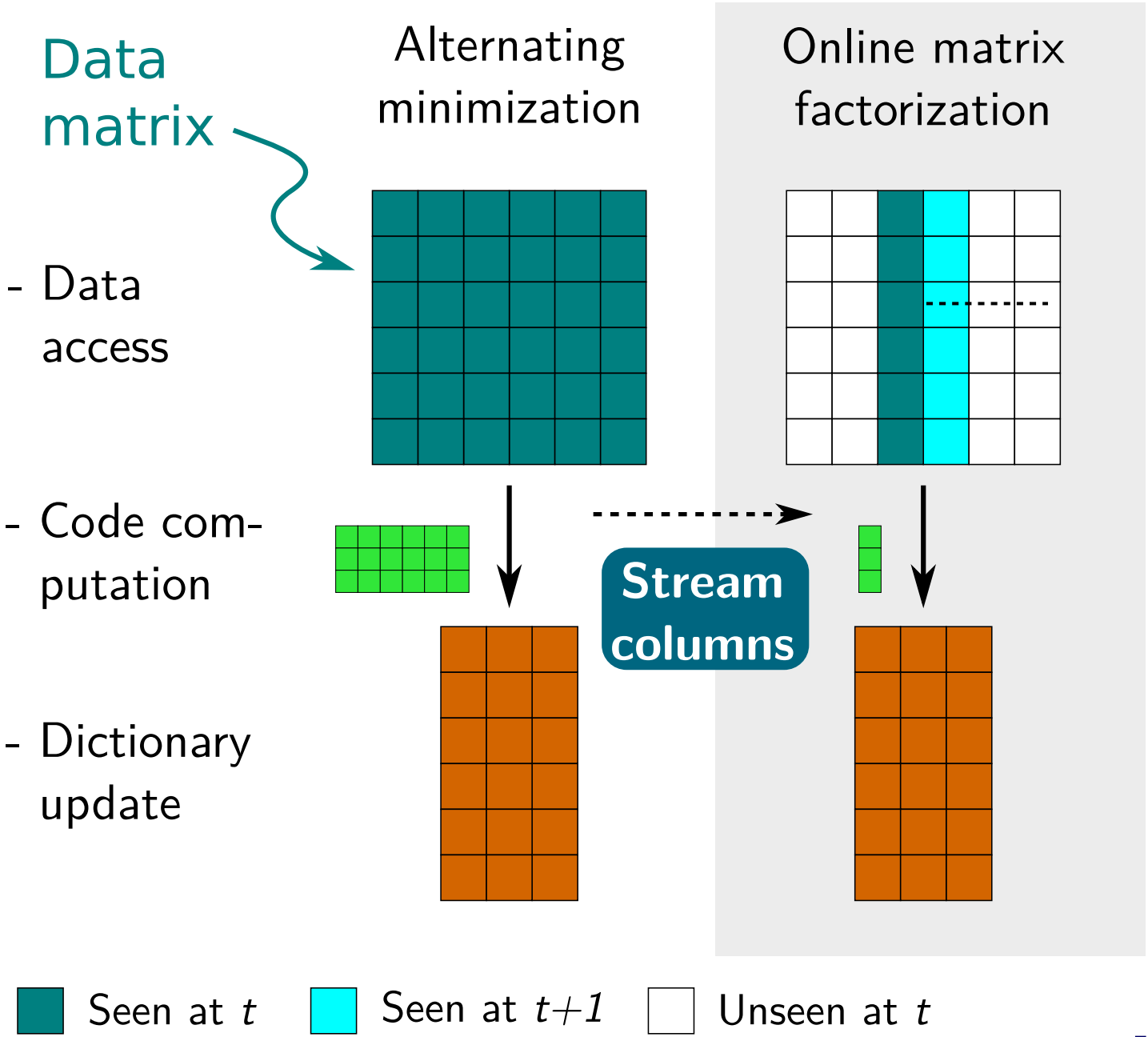
$$\underset{\mathbf{V}}{\operatorname{argmin}} \sum_i \left(\min_{\mathbf{u}} \|\mathbf{X}_i - \mathbf{V}\,\mathbf{u}\|_{\text{Fro}}^2 + \lambda\Omega(\mathbf{u}) \right)$$

$$\underset{\mathbf{E}}{\operatorname{argmin}}\, \mathbb{V}\big[f(\mathbf{V})\big]$$

$\Longrightarrow$ Optimize on approximations (sub-samples)

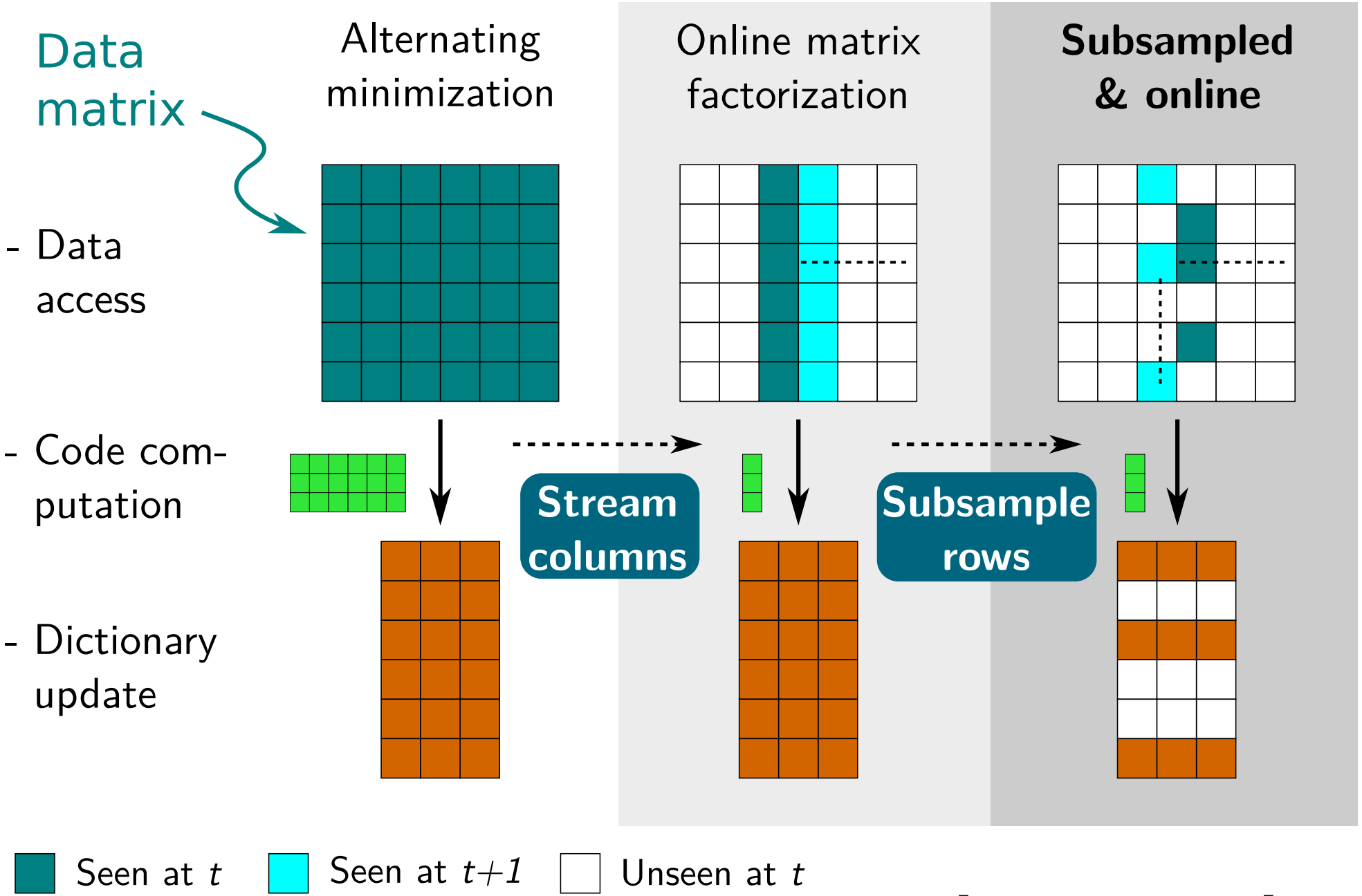
[Mairal... 2010]

Online estimation for matrix factorization



[Mairal... 2010]

Online estimation for matrix factorization



[Mensch... 2017]

Online matrix factorization algorithm [Mairal... 2010]

Stream samples $\mathbf{x}_t$:

1. **Compute code**

$$\mathbf{u}_t = \underset{\mathbf{u} \in \mathbb{R}^k}{\operatorname{argmin}} \|\mathbf{x}_t - \mathbf{V}_{t-1}\mathbf{u}\|_2^2 + \lambda\Omega(\mathbf{u})$$

Online matrix factorization algorithm [Mairal... 2010]

Stream samples $\mathbf{x}_t$:

1. **Compute code**

$$\mathbf{u}_t = \underset{\mathbf{u} \in \mathbb{R}^k}{\operatorname{argmin}} \|\mathbf{x}_t - \mathbf{V}_{t-1}\mathbf{u}\|_2^2 + \lambda\Omega(\mathbf{u})$$

2. **Update the surrogate function**

$$g_t(\mathbf{V}) = \frac{1}{t}\sum_{i=1}^{t} \|\mathbf{x}_i - \mathbf{V}\,\mathbf{u}_i\|_2^2$$

$$g_t(\mathbf{V}) \overset{\text{surrogate}}{=} \sum_{\mathbf{x}} l(\mathbf{x}, \mathbf{V}) \qquad \mathbf{u}_i \text{ is used, and not } \mathbf{u}^\star$$

Online matrix factorization algorithm [Mairal... 2010]

Stream samples $\mathbf{x}_t$:

1. **Compute code**

$$\mathbf{u}_t = \underset{\mathbf{u}\in\mathbb{R}^k}{\operatorname{argmin}} \|\mathbf{x}_t - \mathbf{V}_{t-1}\mathbf{u}\|_2^2 + \lambda\Omega(\mathbf{u})$$

2. **Update the surrogate function**

$$g_t(\mathbf{V}) = \frac{1}{t}\sum_{i=1}^{t} \|\mathbf{x}_i - \mathbf{V}\,\mathbf{u}_i\|_2^2 = \operatorname{tr}\left(\frac{1}{2}\mathbf{V}^\top\mathbf{V}\mathbf{A}_t - \mathbf{V}^\top\mathbf{B}_t\right)$$

$$\mathbf{A}_t \overset{def}{=} (1-\frac{1}{t})\mathbf{A}_{t-1} + \frac{1}{t}\mathbf{u}_t\mathbf{u}_t^\top \qquad \mathbf{B}_t \overset{def}{=} (1-\frac{1}{t})\mathbf{B}_{t-1} + \frac{1}{t}\mathbf{x}_t\mathbf{u}_t^\top$$

$\mathbf{A}_t$ and $\mathbf{B}_t$ are sufficient statistics of the loss accumulated over the data

Online matrix factorization algorithm [Mairal... 2010]

Stream samples $\mathbf{x}_t$:

1. **Compute code**

$$\mathbf{u}_t = \underset{\mathbf{u}\in\mathbb{R}^k}{\operatorname{argmin}} \|\mathbf{x}_t - \mathbf{V}_{t-1}\mathbf{u}\|_2^2 + \lambda\Omega(\mathbf{u})$$

2. **Update the surrogate function**

$$g_t(\mathbf{V}) = \frac{1}{t}\sum_{i=1}^{t} \|\mathbf{x}_i - \mathbf{V}\,\mathbf{u}_i\|_2^2 = \operatorname{tr}\left(\frac{1}{2}\mathbf{V}^\top\mathbf{V}\mathbf{A}_t - \mathbf{V}^\top\mathbf{B}_t\right)$$

$$\mathbf{A}_t \overset{def}{=} (1-\frac{1}{t})\mathbf{A}_{t-1} + \frac{1}{t}\mathbf{u}_t\mathbf{u}_t^\top \qquad \mathbf{B}_t \overset{def}{=} (1-\frac{1}{t})\mathbf{B}_{t-1} + \frac{1}{t}\mathbf{x}_t\mathbf{u}_t^\top$$

3. **Minimize surrogate**

$$\mathbf{V}_t = \underset{\mathbf{V}\in\mathcal{C}}{\operatorname{argmin}}\, g_t(\mathbf{V}) \qquad \nabla g_t = \mathbf{V}\mathbf{A}_t - \mathbf{B}_t$$

Stochastic Majorization-Minimization [Mairal 2013]

$$\mathbf{V} = \underset{\mathbf{V} \in C}{\operatorname{argmin}} \sum_{\mathbf{x}} l(\mathbf{x}, \mathbf{V}) \quad \text{where } l(\mathbf{x}, \mathbf{V}) = \min_{\mathbf{u}} f(\mathbf{x}, \mathbf{V}, \mathbf{u})$$

Algorithm:

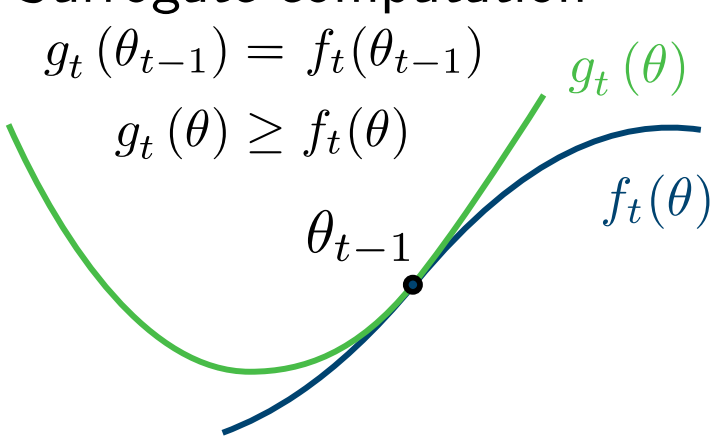
$$g_t(\mathbf{V}) \overset{\text{majorant}}{=} \sum_{\mathbf{x}} l(\mathbf{x}, \mathbf{V}) \qquad \mathbf{u}_i \text{ is used, and not } \mathbf{u}^\star$$

$\Rightarrow$ Majorization-Minimization scheme[1]

Surrogate computation

$g_t(\theta_{t-1}) = f_t(\theta_{t-1})$

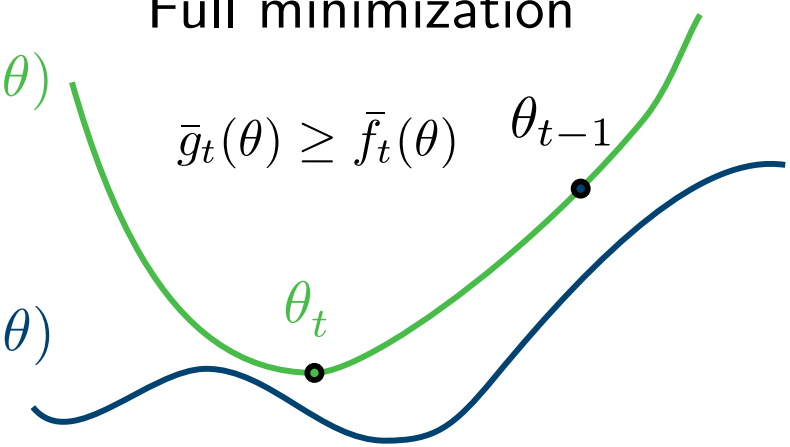
$g_t(\theta) \geq f_t(\theta)$



SMM

Full minimization

$\bar{g}_t(\theta) \geq \bar{f}_t(\theta)$

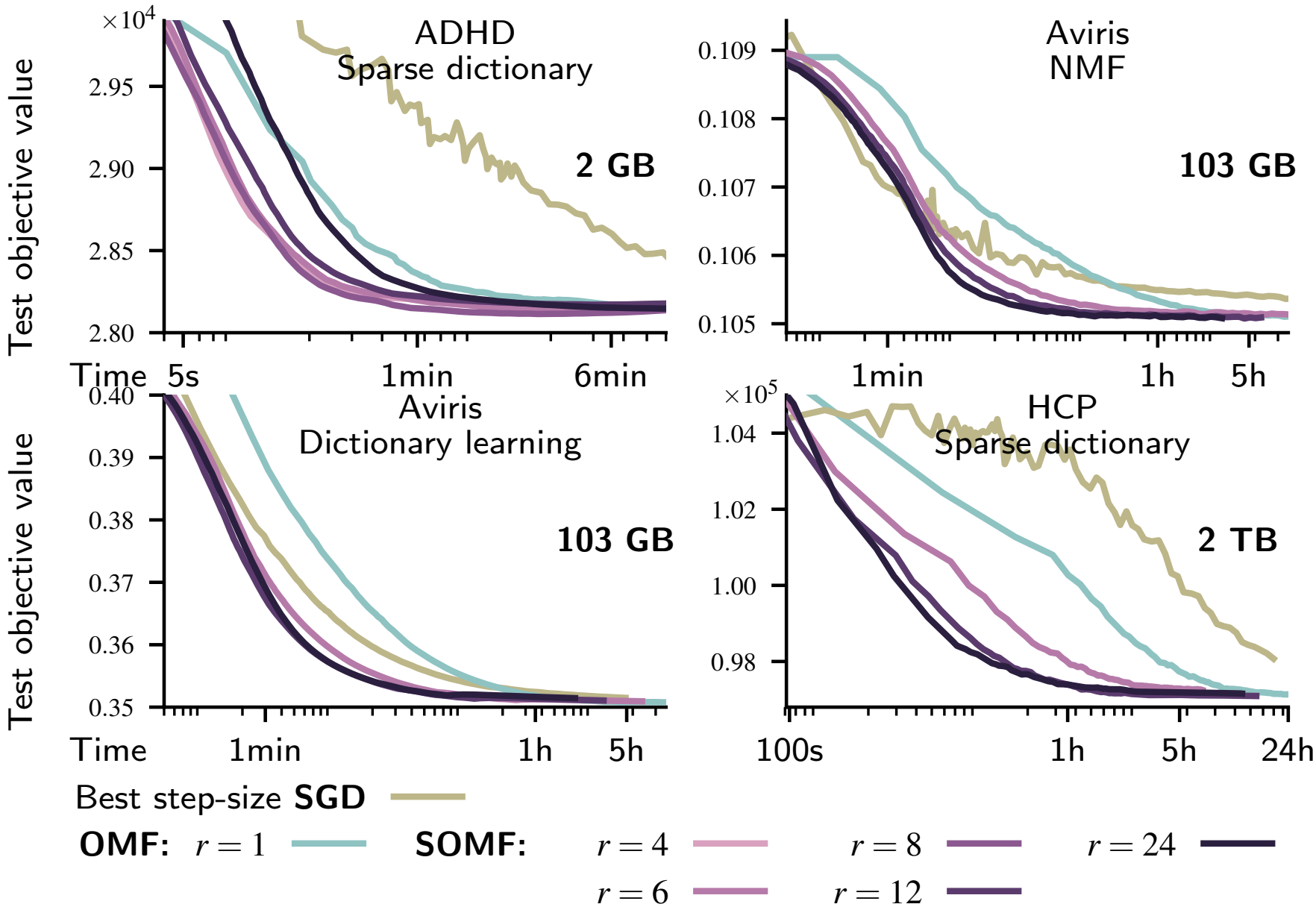


2nd order information

No learning rate

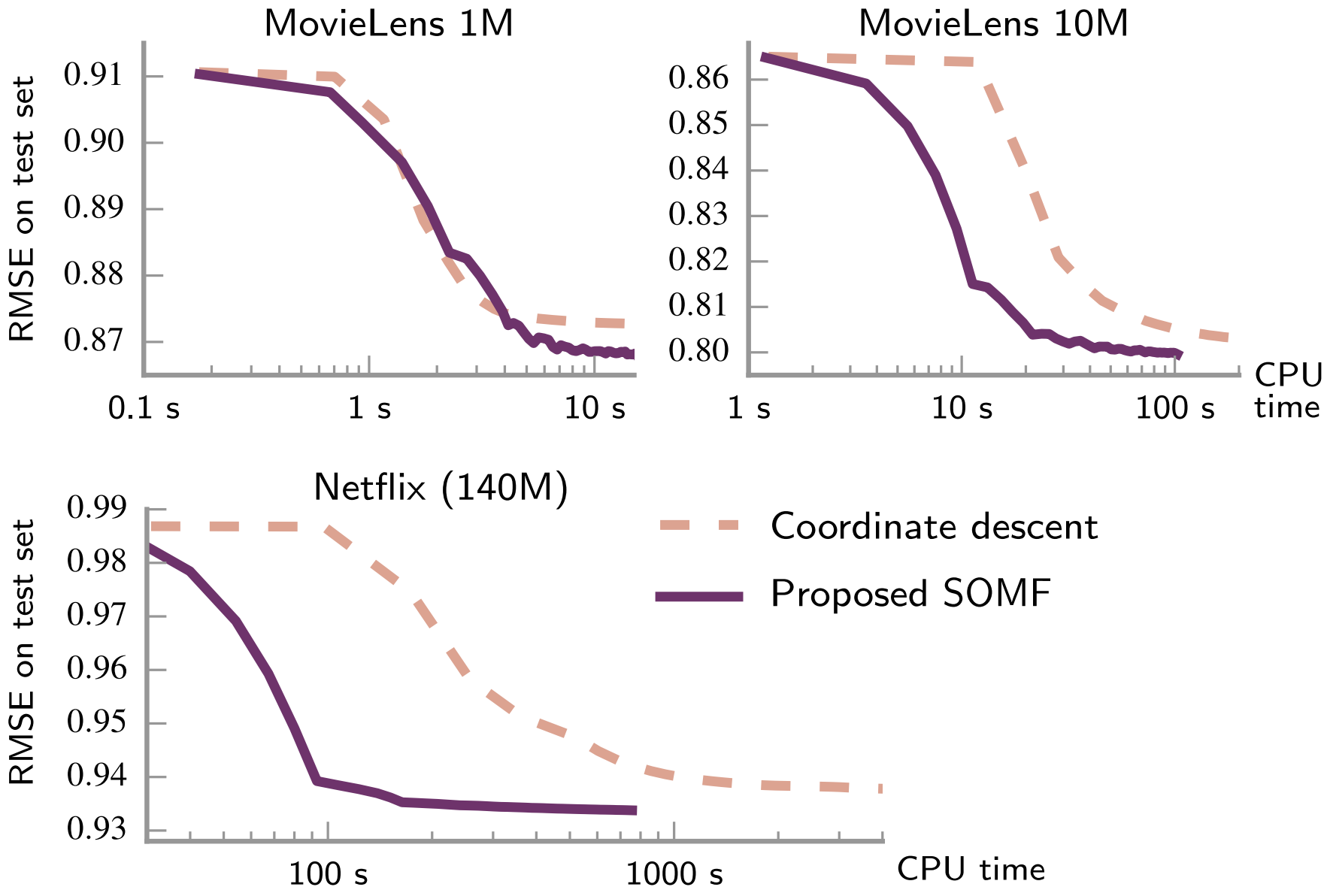
[1] SOMF uses a approximate majorant and minimization [Mensch... 2017]

Experimental convergence: large images



SOMF = Subsampled Online Matrix Factorization

Experimental convergence: recommender system



SOMF = Subsampled Online Matrix Factorization

Summary

Versatile matrix-factorization formulation[1]

$$\underset{\mathbf{U}\in\mathbb{R}^{n\times k},\mathbf{V}\in\mathcal{C}}{\operatorname{argmin}} \; \|\mathbf{X}-\mathbf{U}\,\mathbf{V}\|_{\text{Fro}}^2 \; + \; \lambda\Omega(\mathbf{U})$$

Estimation

Stochastic majorization miniminization[2]

$\Rightarrow$ an online alternated optimization

Example use of learned representations

Biomakers of autism on brain images:

$p \sim 100\,000$, $n \sim 1\,000$ [Abraham... 2017]

[1] 1-layer linear autoencoder

[2] Common case algorithm readily usable in scikit-learn: `MiniBatchDictionaryLearning`

2 Matrix factorization and its variants

- For signals
- For discrete objects

Gamma-Poisson for factorizing counts [Canny 2004]

When $\mathbf{X}$ is a matrix of counts

- Recommenders systems [Gopalan... 2014]
- Database string entries [Cerda and Varoquaux 2019]

$\Longrightarrow$ Poisson loss, instead of squared loss

$$\mathbb{P}(\mathbf{x}_j|\mathbf{u},\mathbf{V}) = \text{Poisson}\big((\mathbf{u}\,\mathbf{V})_j\big) = {}^1\!/_{x_j!}\,(\mathbf{u}\,\mathbf{V})_j^{x_j} e^{-(\mathbf{u}\,\mathbf{V})_j}$$

$\mathbf{u}$ are loadings, modeled as random with a Gamma prior[3]

$$\mathbb{P}(u_i) = \frac{u_i^{\alpha_i - 1} e^{-u_i/\beta_i}}{\beta_i^{\alpha_i}\,\Gamma(\alpha_i)}$$

[3] Because it is the conjugate prior of the Poisson, it imposes soft sparsity, and it raises rotational invariance

Gamma-Poisson for factorizing counts [Canny 2004]

When $\mathbf{X}$ is a matrix of counts

- Recommenders systems [Gopalan... 2014]
- Database string entries [Cerda and Varoquaux 2019]

$\Longrightarrow$ Poisson loss, instead of squared loss

$$\mathbb{P}(\mathbf{x}_j|\mathbf{u},\mathbf{V}) = \text{Poisson}\big((\mathbf{u}\,\mathbf{V})_j\big) = {}^1\!/_{x_j!}\,(\mathbf{u}\,\mathbf{V})_j^{x_j} e^{-(\mathbf{u}\,\mathbf{V})_j}$$

$\mathbf{u}$ are loadings, modeled as random with a Gamma prior[3]

$$\mathbb{P}(u_i) = \frac{u_i^{\alpha_i - 1} e^{-u_i/\beta_i}}{\beta_i^{\alpha_i}\,\Gamma(\alpha_i)}$$

Maximum a posteriori estimation:

$$\hat{\mathbf{U}}, \hat{\mathbf{V}} = \underset{\mathbf{U},\mathbf{V}}{\text{argmin}} -\sum_j \Big(\log \mathbb{P}(\mathbf{x}_j|\mathbf{u},\mathbf{V}) + \sum_i \log \mathbb{P}(u_i)\Big)$$

[3] Because it is the conjugate prior of the Poisson, it imposes soft sparsity, and it raises rotational invariance

Gamma-Poisson estimation

Full log-likelihood expression:

$$\log \mathcal{L} = \sum_{j=1}^{p} x_j \log((\mathbf{u}\,\mathbf{V})_j) - (\mathbf{u}\,\mathbf{V})_j - \log(x_j!)$$

$$+ \sum_{i=1}^{k} (\alpha_i - 1)\log(u_i) - \frac{u_i}{\beta_i} - \alpha_i \log \beta_i - \log \Gamma(\alpha_i)$$

Gradients:

$$\frac{\partial}{\partial V_{ij}} \log \mathcal{L} = \frac{x_j}{(\mathbf{u}\,\mathbf{V})_j} u_i - u_i$$

$$\frac{\partial}{\partial u_i} \log \mathcal{L} = \sum_{j=1}^{p} \frac{x_j}{(\mathbf{u}\,\mathbf{V})_j} V_{ij} - V_{ij} + \frac{\alpha_i - 1}{u_i} - \frac{1}{\beta_i}$$

Gamma-Poisson estimation

Gradients:

$$\frac{\partial}{\partial V_{ij}} \log \mathcal{L} = \frac{x_j}{(\mathbf{u}\,\mathbf{V})_j} u_i - u_i$$

$$\frac{\partial}{\partial u_i} \log \mathcal{L} = \sum_{j=1}^{p} \frac{x_j}{(\mathbf{u}\,\mathbf{V})_j} V_{ij} - V_{ij} + \frac{\alpha_i - 1}{u_i} - \frac{1}{\beta_i}$$

Equivalent to some NMF formulation: multiplicative updates[1]

$$V_{ij} \leftarrow V_{ij} \left(\sum_{\ell=1}^{n} \frac{x_{\ell j}}{(\mathbf{UV})_{\ell j}} u_{\ell i} \right) \left(\sum_{\ell=1}^{n} u_{\ell i} \right)^{-1}$$

$$u_{\ell i} \leftarrow u_{\ell i} \left(\sum_{j=1}^{p} \frac{x_{\ell j}}{(\mathbf{UV})_{\ell j}} V_{ij} + \frac{\alpha_i - 1}{u_{\ell i}} \right) \left(\sum_{j=1}^{p} V_{ij} + \beta_i^{-1} \right)^{-1}$$

[1] Efficient implementation with sparse matrices: the summations can be done only on non-zero entries of $\mathbf{X}$.

Adapt the majorization minimization algorithm

while $\|\mathbf{V}^{(t)} - \mathbf{V}^{(t-1)}\|_F > \eta$ **do**

draw $\mathbf{x}_t$ from the training set.

while $\|\mathbf{u}_t - \mathbf{u}_t^{old}\|_2 > \epsilon$ **do**

$$\mathbf{u}_t \leftarrow \mathbf{u}_t . \left[\left(\frac{\mathbf{x}_t}{\mathbf{u}_t \mathbf{V}^{(t)}} \right) \mathbf{V}^{(t)\mathsf{T}} + \frac{\mathbf{a}-1}{\mathbf{u}_t} \right] . \left[\mathbf{1}\mathbf{V}^{(t)\mathsf{T}} + \mathbf{b}^{-1} \right]^{\cdot -1}$$

$$\mathbf{A}_t \leftarrow \mathbf{V}^{(t)} . \left[\mathbf{u}_t^{\mathsf{T}} \left(\frac{\mathbf{x}_t}{\mathbf{u}_t \mathbf{V}^{(t)}} \right) \right]$$

$$\mathbf{B}_t \leftarrow \mathbf{u}_t^{\mathsf{T}} \mathbf{1}$$

$$\mathbf{A}^{(t)} \leftarrow \rho \, \mathbf{A}^{(t-1)} + \mathbf{A}^{(t)}$$

$$\mathbf{B}^{(t)} \leftarrow \rho \, \mathbf{B}^{(t-1)} + \mathbf{B}^{(t)}$$

$$\mathbf{V}^{(t)} \leftarrow \mathbf{A}^{(t)} ./ \mathbf{B}^{(t)}$$

$$t \leftarrow t + 1$$

[Lefevre... 2011, Cerda and Varoquaux 2019]

Application: sub-string representation

Problem: representing non-normalized categories

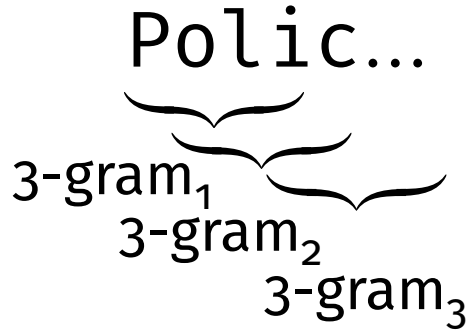
Drug Name
alcohol
ethyl alcohol
isopropyl alcohol
polyvinyl alcohol
isopropyl alcohol swab
62% ethyl alcohol
alcohol 68%
alcohol denat
benzyl alcohol
dehydrated alcohol

Employee Position Title
Police Aide
Master Police Officer
Mechanic Technician II
Police Officer III
Senior Architect
Senior Engineer Technician
Social Worker III

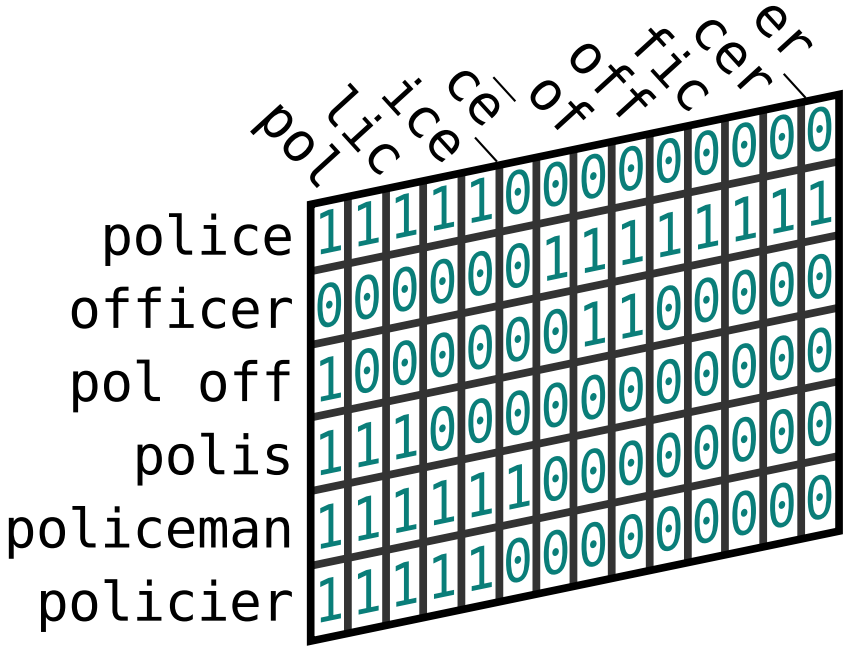
[Cerda and Varoquaux 2019]

Application: sub-string representation

Gamma-Poisson factorization on sub-strings counts



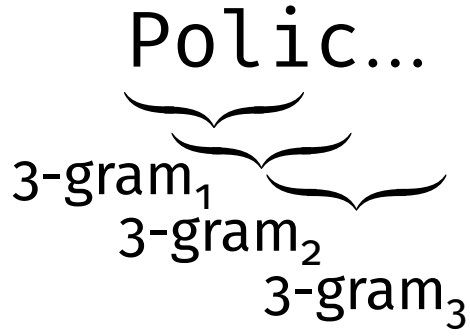
Models strings as a linear combination of substrings



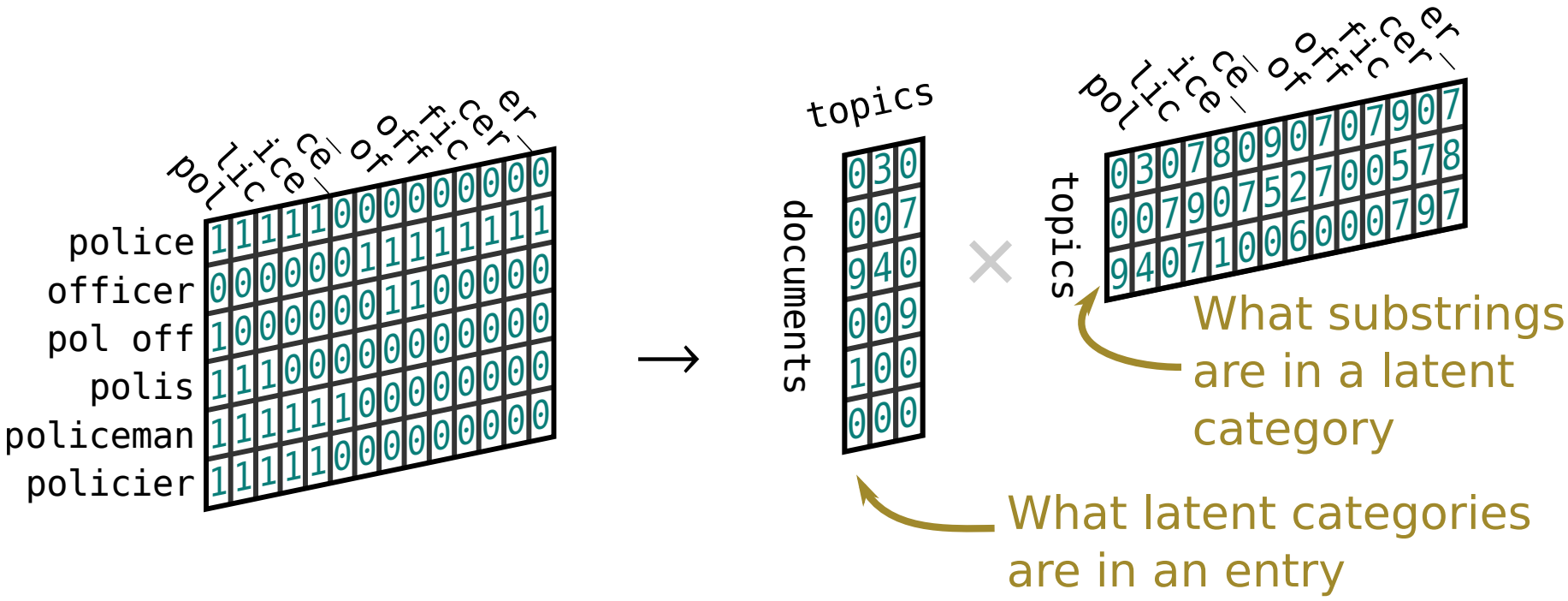
[Cerda and Varoquaux 2019]

Application: sub-string representation

Gamma-Poisson factorization on sub-strings counts



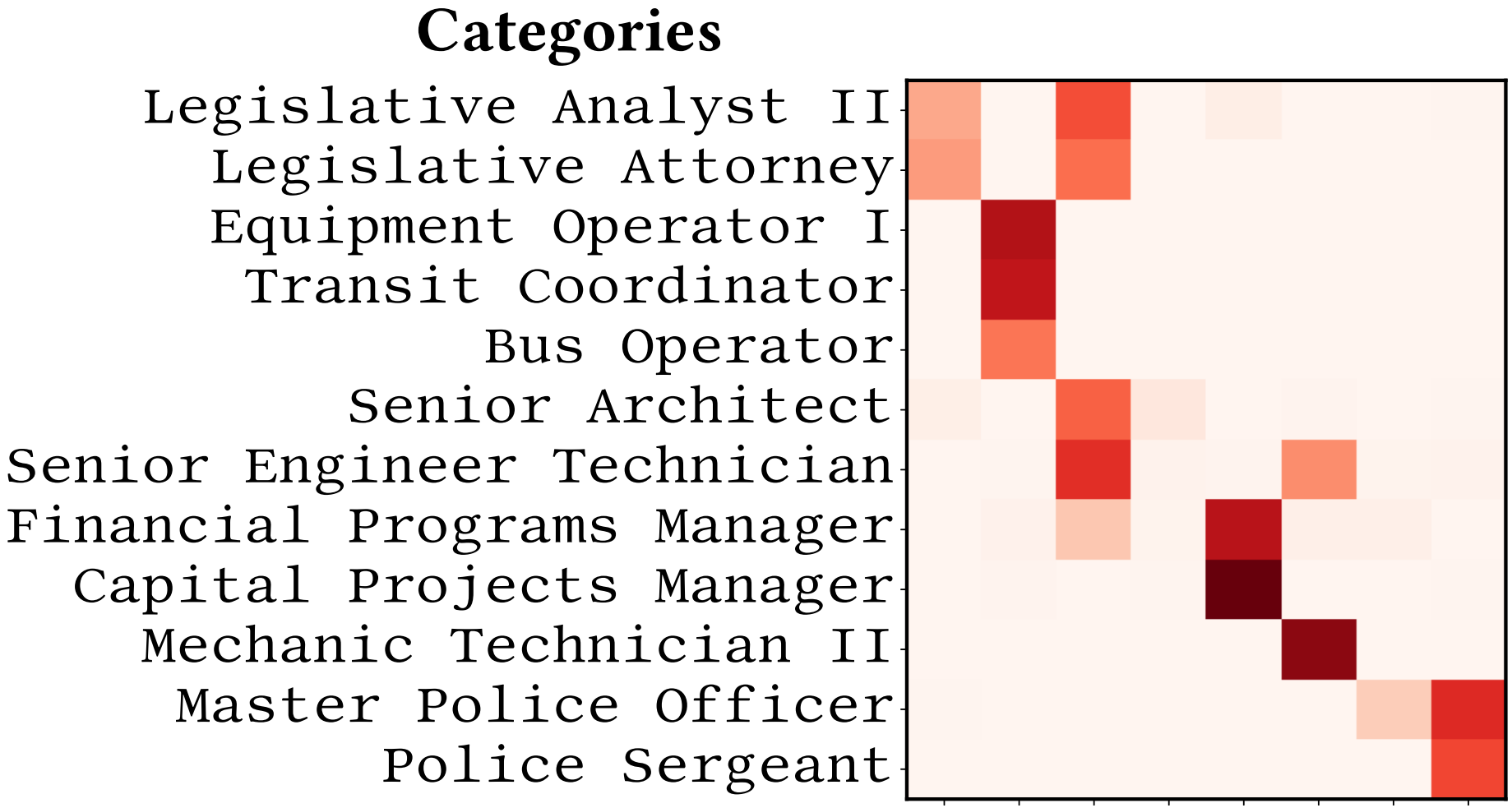
Models strings as a linear combination of substrings



[Cerda and Varoquaux 2019]

Application: sub-string representation

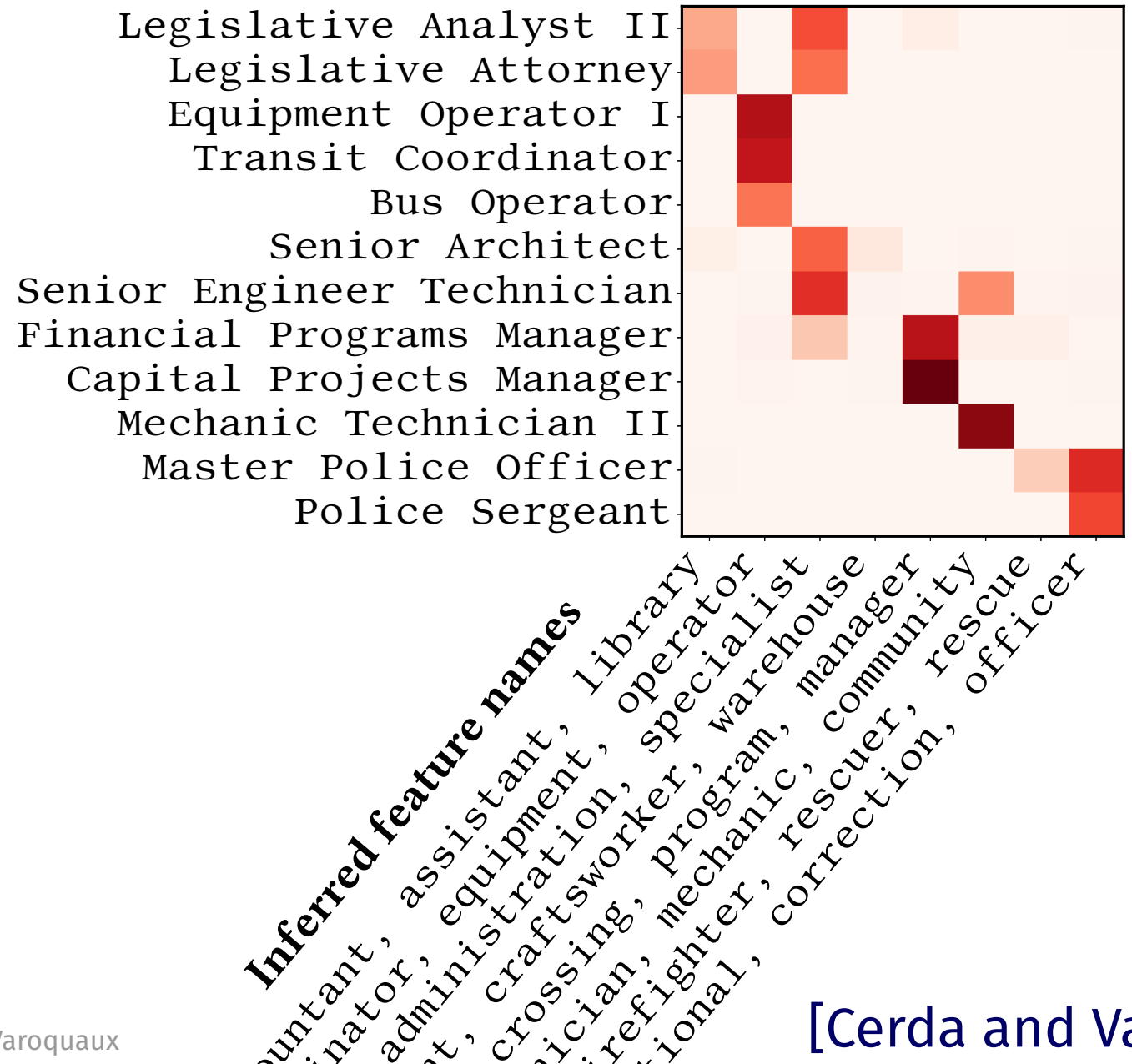
Representations that extract latent categories



[Cerda and Varoquaux 2019]

Application: sub-string representation

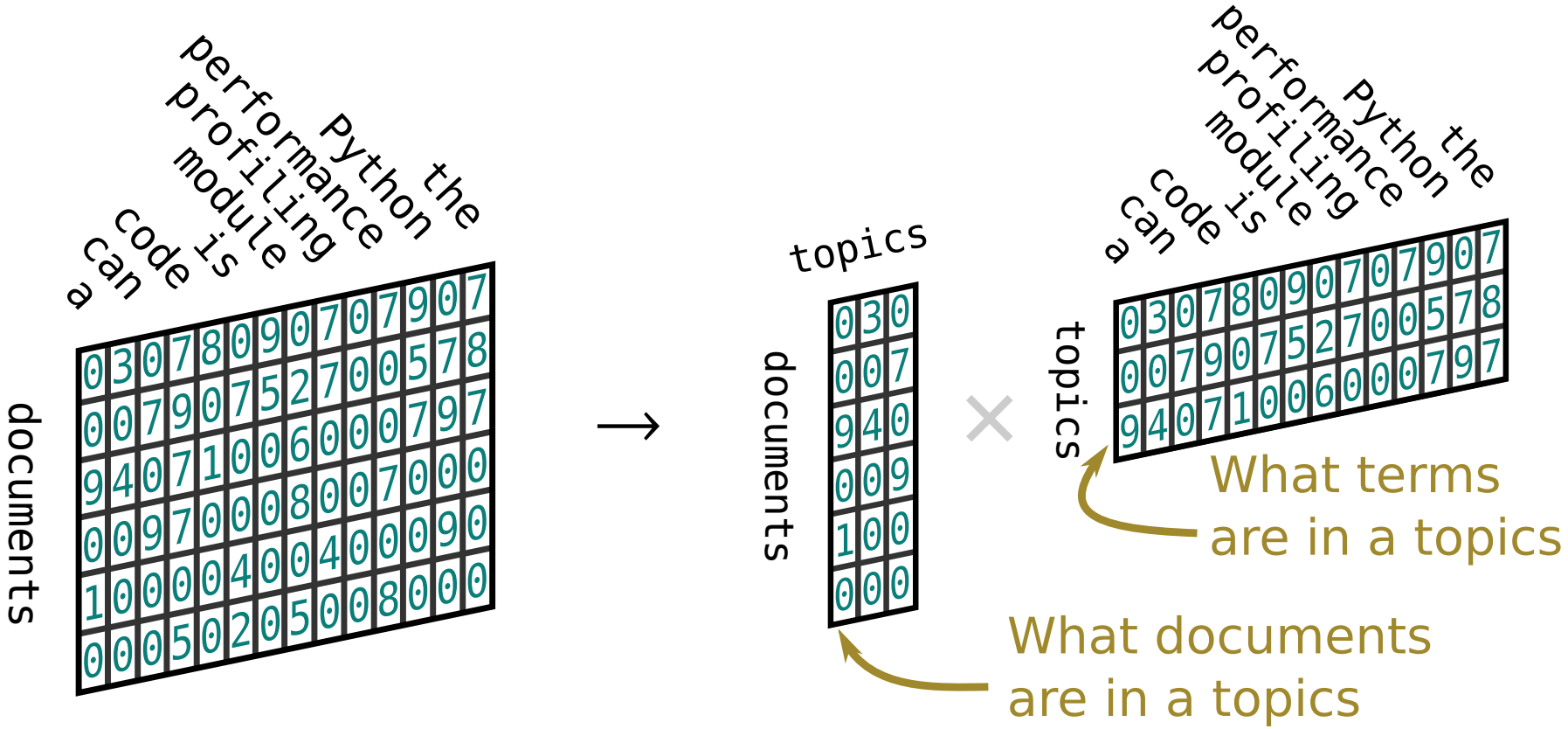
Inferring plausible feature names



[Cerda and Varoquaux 2019]

Natural language processing: topic-modeling history

Topic modeling: embedding documents[1]



- LSA (Latent Semantic Analysis) [Landauer... 1998]

  SVD[2] of the terms×documents matrix

[1] Typically for information retrieval purpose, *aka* search engines

[2] Later: refinements for more complex loss: LDA (Latent Dirichlet Allocation) [Blei... 2003] and Gamma Poisson [Canny 2004].

Word embeddings

Distributional semantics: meaning of words

"You shall know a word by the company it keeps"

Firth, 1957

Example: *A glass of red _ _ _, please*

Could be *wine* maybe *juice*?

wine and *juice* have related meanings

Factorization of the word×context matrix

- What choice of context?
- What loss?

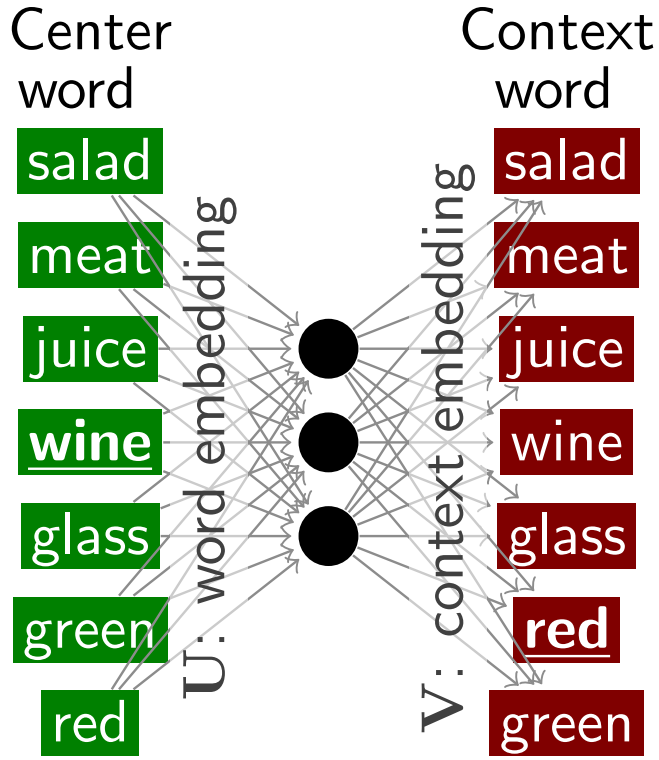
word2vec [Mikolov... 2013a] glove [Pennington... 2014]

Word2vec: skip-gram sampling [Mikolov... 2013b]

$$\{\hat{\mathbf{u}}_w, \hat{\mathbf{v}}_c\} = \underset{\{\mathbf{u}_w, \mathbf{v}_c\}}{\operatorname{argmax}} \sum_{\text{pairs of words } (w,c) \text{ in the same window}^1} \log \operatorname{softmax}(\mathbf{V}\,\mathbf{u}_w^\top)_c$$

$$\operatorname{softmax}(\mathbf{z})_i = \frac{\exp \mathbf{z}_i}{\sum_j \exp \mathbf{z}_j}$$

- $\mathbf{u}_w \in \mathbb{R}^k$: embedding of word w
- $\mathbf{V} \in \mathbb{R}^{\text{card(voc)} \times k}$: $[\mathbf{v}_c, c \in \text{voc}]$ all context words



Big sum on contexts
$\Rightarrow$ solved by SGD[2]

Other view:
Language models
Prediction of words

[1] Efficient: never build the matrix, stream directly from text.

[2] These windows are called skip gram

Word2vec: negative sampling

[Mikolov... 2013a]

Costly loss: $\log \text{softmax}(\mathbf{z})_i = \log \frac{\exp \mathbf{z}_i}{\sum_j \exp \mathbf{z}_j}$

Approximate[1] Huge sum in softmax (all vocabulary)

Downsample it by drawing the positive (numerator) and a few negative examples (denominator)

Negative sampling loss[2]:

[Goldberg and Levy 2014]

$$\log \sigma(\mathbf{v}_c \mathbf{u}_w^\mathsf{T}) + \sum_{\substack{n_{\text{neg}} \text{ words } w \\ \text{not in window}}} \log \sigma(-\mathbf{v}_c \mathbf{u}_{w'})$$

σ: sigmoid ($\log \sigma(z) = -1 - \exp -z$)

[1] Related to noise contrastive estimate, that avoid computing costly normalizations in likelihoods [Gutmann and Hyvärinen 2010]

[2] Related to a matrix factorization of mutual information inword occurence [Levy and Goldberg 2014]

Beyond natural language: metric learning

Triplet loss

For a "anchor", b close to a, c far from a:

$$\log \sigma(\mathbf{v}_a^\top \mathbf{u}_b) - \log \sigma(\mathbf{v}_a^\top \mathbf{u}_c)$$

Quadruplet loss [Chen... 2017]

For a and b close by, c and d far appart:

$$\log \sigma(\mathbf{v}_a^\top \mathbf{u}_b) - \log \sigma(\mathbf{v}_c^\top \mathbf{u}_d)$$

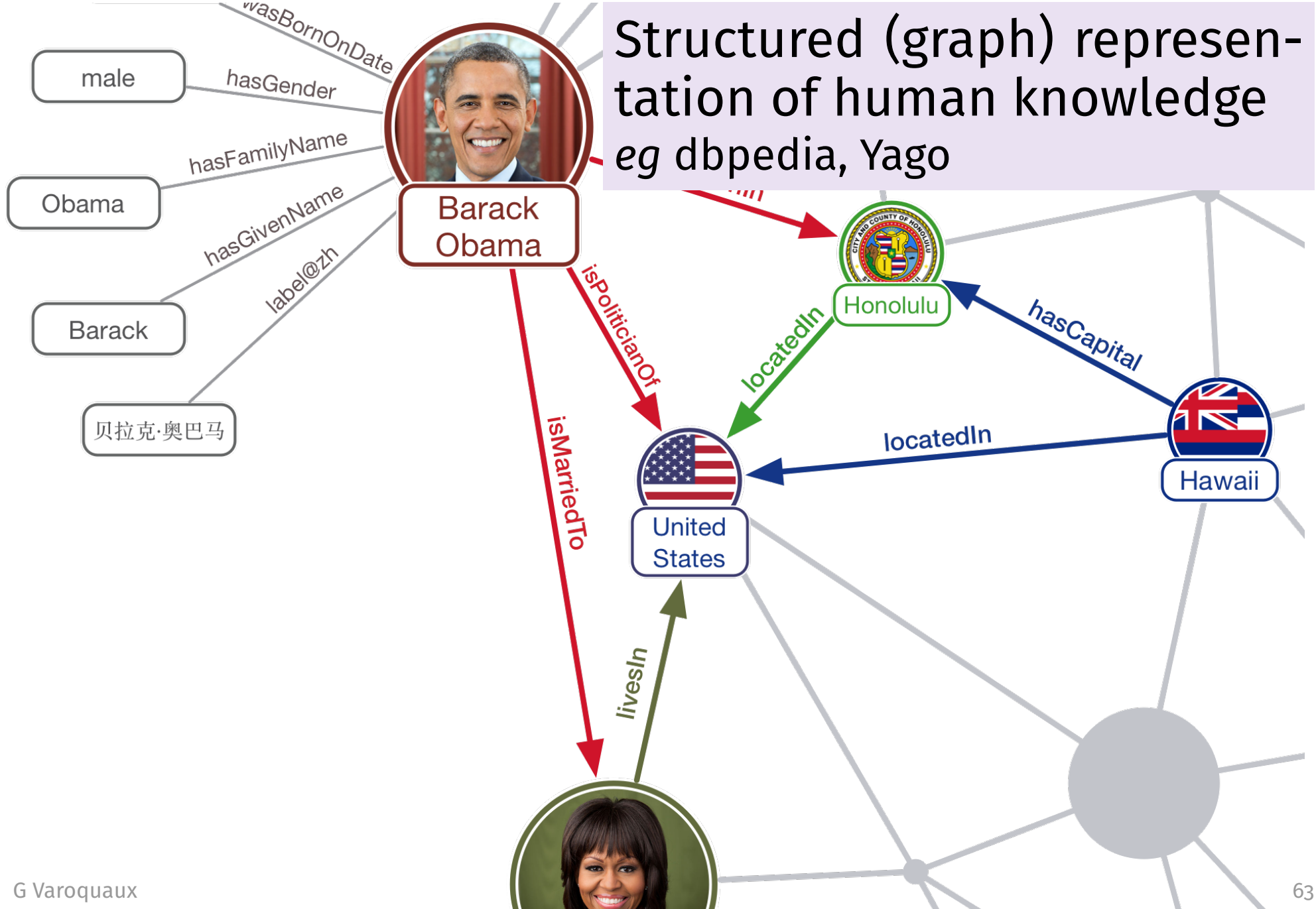
In practice: draw[1] randomly (a, b, c) or (a, b, c, d)

Metric learning: [Bellet... 2013]

Learning embeddings with weak supervision

[1] Many strategies, *eg* "hard negative mining", requires a good test set and metric to set, as with SGD hyperparameters.

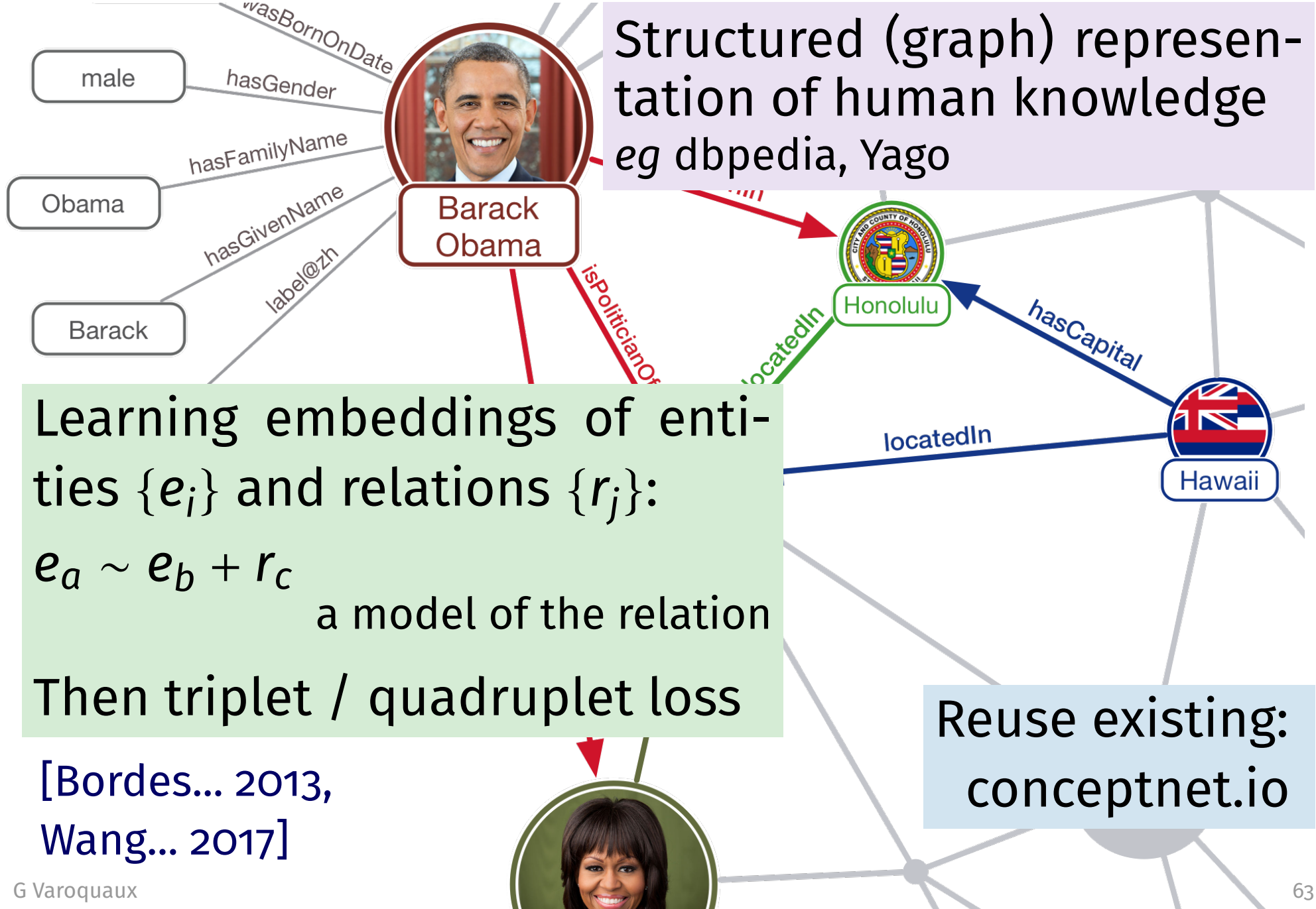
Embedding entities in knowledge graphs



Structured (graph) representation of human knowledge

eg dbpedia, Yago

Embedding entities in knowledge graphs



Structured (graph) representation of human knowledge
eg dbpedia, Yago

Learning embeddings of entities $\{e_i\}$ and relations $\{r_j\}$:

$e_a \sim e_b + r_c$

a model of the relation

Then triplet / quadruplet loss

[Bordes... 2013, Wang... 2017]

Reuse existing: conceptnet.io

The value of simple models

Risk of invisible overfit during search for hyperparameters and models

Complex models call for a clear utility measure with low measurement error

Many reliable labels

The value of simple models

Risk of invisible overfit during search for hyperparameters and models

Complex models call for a clear utility measure with low measurement error

Many reliable labels

Matrix factorization models[1]: 2 hyper parameters:

- Dimensionality k
- Regularization λ

Set them to optimize representations for supervised problems

[1] Using majorization-minimization approaches to avoid learning rate

Summary

Discrete entities lead to counting occurences

$\Rightarrow$ Poisson and logistic loss (ugly logs in equations)

Word & entity embeddings

Factorization of coocurrences in a notion of context

more generally: metric learning

Limited-data settings:

Avoid negative-sampling models (⚠hyper-parameters)

Try to reuse representations (fastext, conceptnet.io)

3 Fisher kernels

What if the objects studied do not naturally live in a vector space?

eg graphs of varying number of nodes

3 Fisher kernels

Kernels feature maps

From likelihoods to Kernels

Learning with Kernels

[Scholkopf and Smola 2001]

Kernels

- A kernel K is a function: $\mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}^+$
  positive symmetric
- It captures similarity between observations

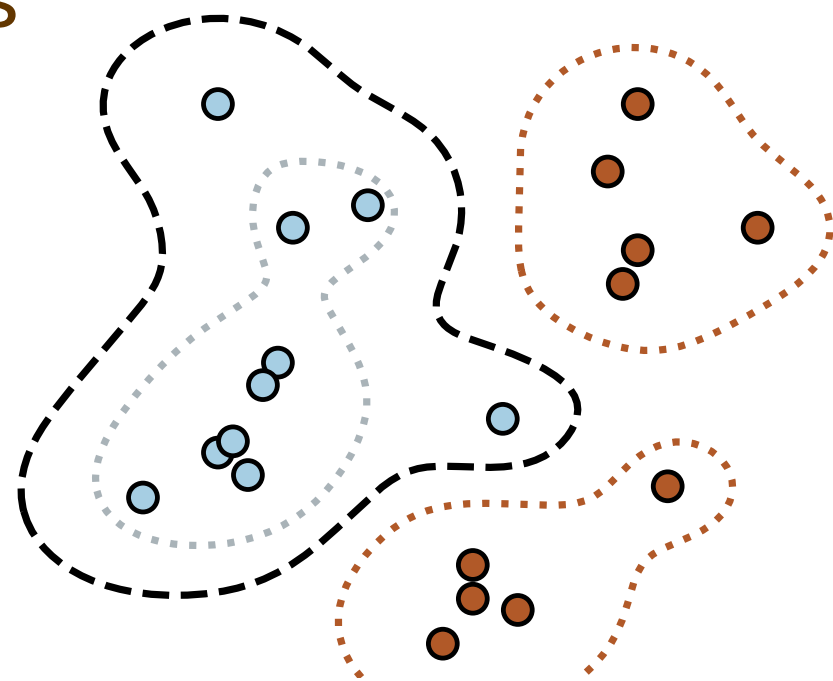
Building functions with kernels

- on the training data:

$$K_i \stackrel{\text{def}}{=} K(x_i, \cdot) \quad \forall i \in \text{train}$$

- prediction function[2]:

$$f(x) = \sum_{i \in train} w_i K_i(x)$$



[2] Benefits of this formulation: *i)* non-linear predictor trained with linear problem; *ii)* expressiveness that increases with amount of training data

Feature maps

[Scholkopf and Smola 2001]

$$f(x) = \sum_{i \in train} w_i K_i(x)$$

Drawbacks of kernels

- Compute cost $O(n^2)$
- Representations not explicit

- As K is symmetric positive[1],

  $\exists \phi : \mathcal{X} \to \mathbb{R}^d$, such that $\forall x, x' \; K(x, x') = \phi(x)^\mathsf{T} \phi(x')$

  ϕ is a "feature map"

- $f(x)$ is a linear function of $\phi(x)$

  but d can be ∞

Approximate ϕ

[1] Think of it as a generalization of the Cholesky decomposition

Nyström approximate feature maps

[Drineas and Mahoney 2005]

On a random subset of the training data:

$$\mathbf{G} \stackrel{\text{def}}{=} \begin{bmatrix} K(x_1, x_1) & \dots & K(x_1, x_m) \\ \vdots & . & \vdots \\ K(x_m, x_1) & \dots & K(x_m, x_m) \end{bmatrix} \in \mathbf{R}^{m \times m}$$

Let $\mathbf{L} \in \mathbb{R}^{k \times m}$ rank-k approximation $\mathbf{L}^\mathsf{T}\mathbf{L} \overset{\text{rank}-k}{\approx} \mathbf{G}^{-1}$

Feature map[1] $\phi_{\text{Nystrom}}(x) = \begin{bmatrix} K(x_1, x) \\ \vdots \\ K(x_m, x) \end{bmatrix} \mathbf{L}^\mathsf{T}$

`sklearn.kernel_approximation.Nystroem`

[1] Exercise: check that $\phi_{\text{Nystrom}}(x)^\mathsf{T}\phi_{\text{Nystrom}}(x) \approx \phi(x)^\mathsf{T}\phi(x)$ for x in our subset.

Nyström approximate feature maps

[Drineas and Mahoney 2005]

On a random subset of the training data:

$$\mathbf{G} \stackrel{\text{def}}{=} \begin{bmatrix} K(x_1, x_1) & \dots & K(x_1, x_m) \\ \vdots & . & \vdots \\ K(x_m, x_1) & \dots & K(x_m, x_m) \end{bmatrix} \in \mathbf{R}^{m \times m}$$

Let $\mathbf{L} \in \mathbb{R}^{k \times m}$ rank-k approximation $\mathbf{L}^\mathsf{T}\mathbf{L} \overset{\text{rank}-k}{\approx} \mathbf{G}^{-1}$

Feature map $\phi_{\text{Nystrom}}(x) = \begin{bmatrix} K(x_1, x) \\ \vdots \\ K(x_m, x) \end{bmatrix} \mathbf{L}^\mathsf{T}$

`sklearn.kernel_approximation.Nystroem`

See also: Random features [Rahimi and Recht 2008]

`sklearn.kernel_approximation.RBFSampler`

3 Fisher kernels

Kernels feature maps

From likelihoods to Kernels

Parametric generative model

Consider a model of x parametrized by $\mathbf{w} \in \mathbb{R}^k$:

$\mathbb{P}(x) = \mathcal{P}_{\mathbf{w}}(x)$ log-likelihood $\mathcal{L}_{\mathcal{P}} \overset{\text{def}}{=} \log \mathcal{P}_{\mathbf{w}}$

Maximum likelihood estimates: $\hat{\mathbf{w}} = \text{argmax}_{\mathbf{w}} \mathcal{L}_{\mathcal{P}}(x)$

Kullback-Leibler divergence

Natural distance[1] to another distribution

$$KL(\mathcal{P}|\mathcal{Q}) = \mathbb{E}_{\mathcal{P}}[\mathcal{L}_{\mathcal{P}} - \mathcal{L}_{\mathcal{Q}}]$$

Goal:
Benefit from our model to build a representation

All models are wrong but some are useful

[1] Not a distance, technically, as not symmetric.

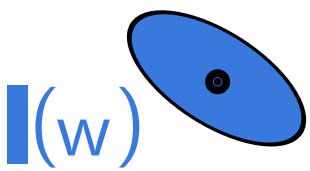
Local behavior of parametric models

Fisher information matrix

Expectation of Hessian of $\mathcal{L}$ given $\mathbf{w}$:

$$\mathbf{I}(\mathbf{w}) \stackrel{\text{def}}{=} \mathbb{E}\left[\frac{\partial^2}{\partial^2 \mathbf{w}} \mathcal{L}(x|\mathbf{w}) \,\middle|\, \mathbf{w}\right] \in \mathbb{R}^{k \times k}$$

- Order-2 approximation of *KL* divergence:



$$KL(\mathcal{P}_{\mathbf{w}}|\mathcal{P}_{\mathbf{w}+\epsilon}) = \epsilon^{\mathsf{T}} \mathbf{I}_{\mathbf{w}} \epsilon$$

- $\mathbf{I}_{\mathbf{w}}$ also scales the covariance of the estimation error on maximum-likelihood estimates of $\mathbf{w}$ (Cramer-Rao bounds)

Fisher-Rao manifold (information geometry)

Order-2 approximation of $KL(\mathcal{P}_{\mathbf{w}}|\mathcal{P}_{\mathbf{w}+\epsilon}) = \epsilon^{\mathsf{T}}\mathbf{I}_{\mathbf{w}}\epsilon$

KL close to $\mathbf{w}_1$

$\neq$ KL close to $\mathbf{w}_2$

Fisher-Rao manifold (information geometry)

Order-2 approximation of $KL(\mathcal{P}_{\mathbf{w}}|\mathcal{P}_{\mathbf{w}+\epsilon}) = \epsilon^{\mathsf{T}}\mathbf{I}_{\mathbf{w}}\epsilon$

KL close to $\mathbf{w}_1$

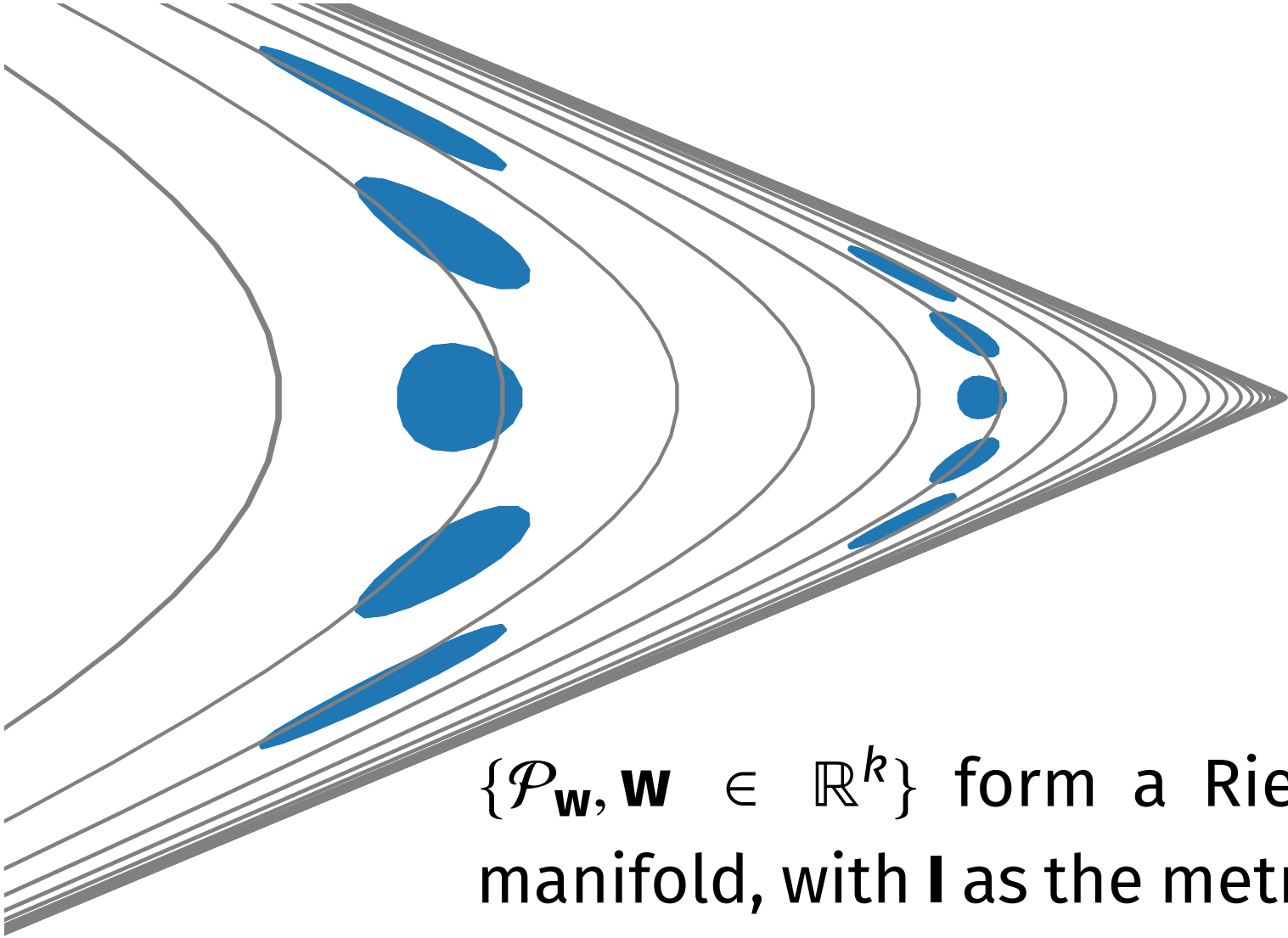
$\neq$ KL close to $\mathbf{w}_2$

Non constant across the family of distributions

$\{\mathcal{P}_{\mathbf{w}}, \mathbf{w} \in \mathbb{R}^k\}$

Fisher-Rao manifold (information geometry)

Order-2 approximation of $KL(\mathcal{P}_{\mathbf{w}}|\mathcal{P}_{\mathbf{w}+\epsilon}) = \epsilon^{\mathsf{T}}\mathbf{I}_{\mathbf{w}}\epsilon$



$\{\mathcal{P}_{\mathbf{w}}, \mathbf{w} \in \mathbb{R}^k\}$ form a Riemannian manifold, with $\mathbf{I}$ as the metric tensor

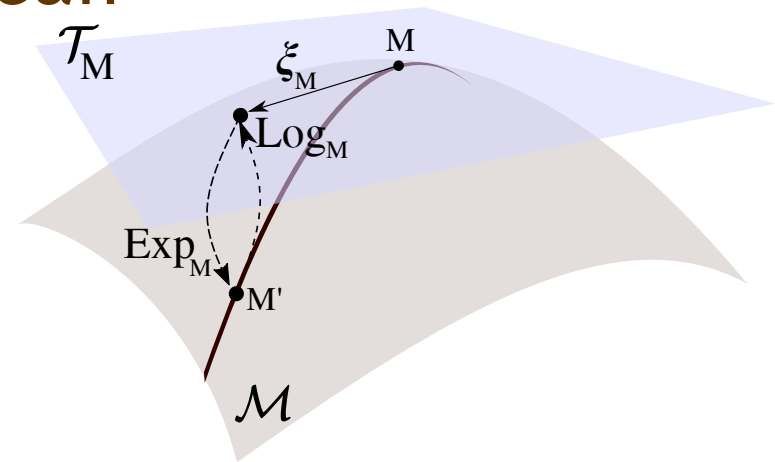
[Rao 1945]

Remannian manifolds

Continuous geometry on curved spaces (*eg* the Earth)

Locally, but not globally, Euclidean

A Riemmannian manifold $\mathcal{M}$ is a differentiable space endowed with a metric d that is locally equivalent to a Euclidean vector space:



for M and $M' \in \mathcal{M}$, if $d(M, M') \to 0$, M and M' can be mapped to elements of a vector space $\mathbf{m}, \mathbf{m}'$ such that $d(M, M') \sim \mathbf{m}^\top \mathbf{m}'$

Global structure: geodesic distance

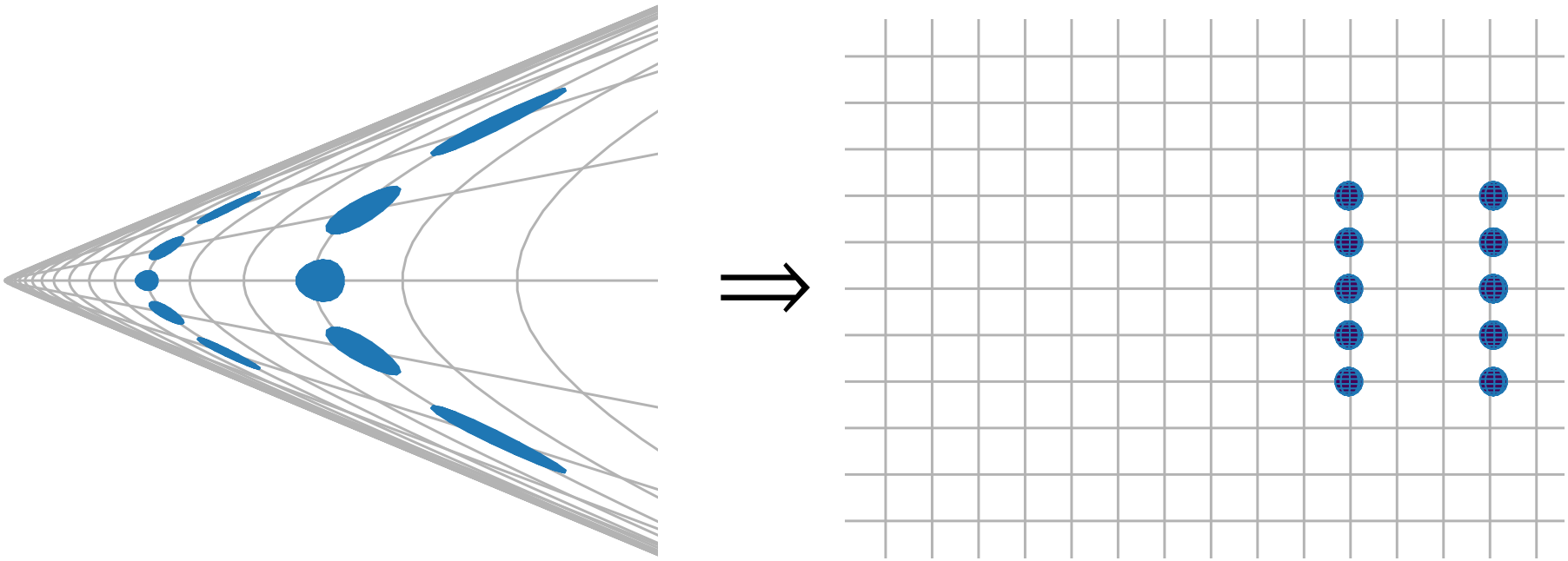
Fisher Kernel

[Jaakkola and Haussler 1999]

A Kernel locally equivalent to the *KL* divergence

- Build upon the Fisher matrix
- Create a feature map

Vector space where Euclidean distance ≈ KL



Fisher Kernel

[Jaakkola and Haussler 1999]

A Kernel locally equivalent to the *KL* divergence

- Build upon the Fisher matrix
- Create a feature map

Vector space where Euclidean distance $\approx$ KL

In practice:

1. Fit model $\mathcal{P}_{\mathbf{w}}$ on train data:

$$\hat{\mathbf{w}} \leftarrow \operatorname*{argmax}_{\mathbf{w}} \sum_{i \in \text{train}} \mathcal{L}(x_i, \mathbf{w})$$

2. Compute gradient on $\mathbf{w}$ of likelihood for $\hat{\mathbf{w}}$:

$$\mathbf{z}_{\text{Fisher}}(x) = \nabla_{\mathbf{w}} \mathcal{L}(x, \hat{\mathbf{w}}) \in \mathbb{R}^k$$

Fisher Kernel applications

Text: TF-IDF [Elkan 2005]

- Multinomial model of word appearance

Genomics [Jaakkola and Haussler 1999]

- Hidden Markov model of DNA sequences
  (variable-length sequences $\Rightarrow$ encoding difficult)

Tree-structured data [Nicotra... 2004]

- A transition model on the tree

Brain connectivity [Varoquaux... 2010]

- Multivariate Gaussian model (covariances)

Summary

Kernels build prediction functions on similarities

Features maps / kernel approximation captures the corresponding representation

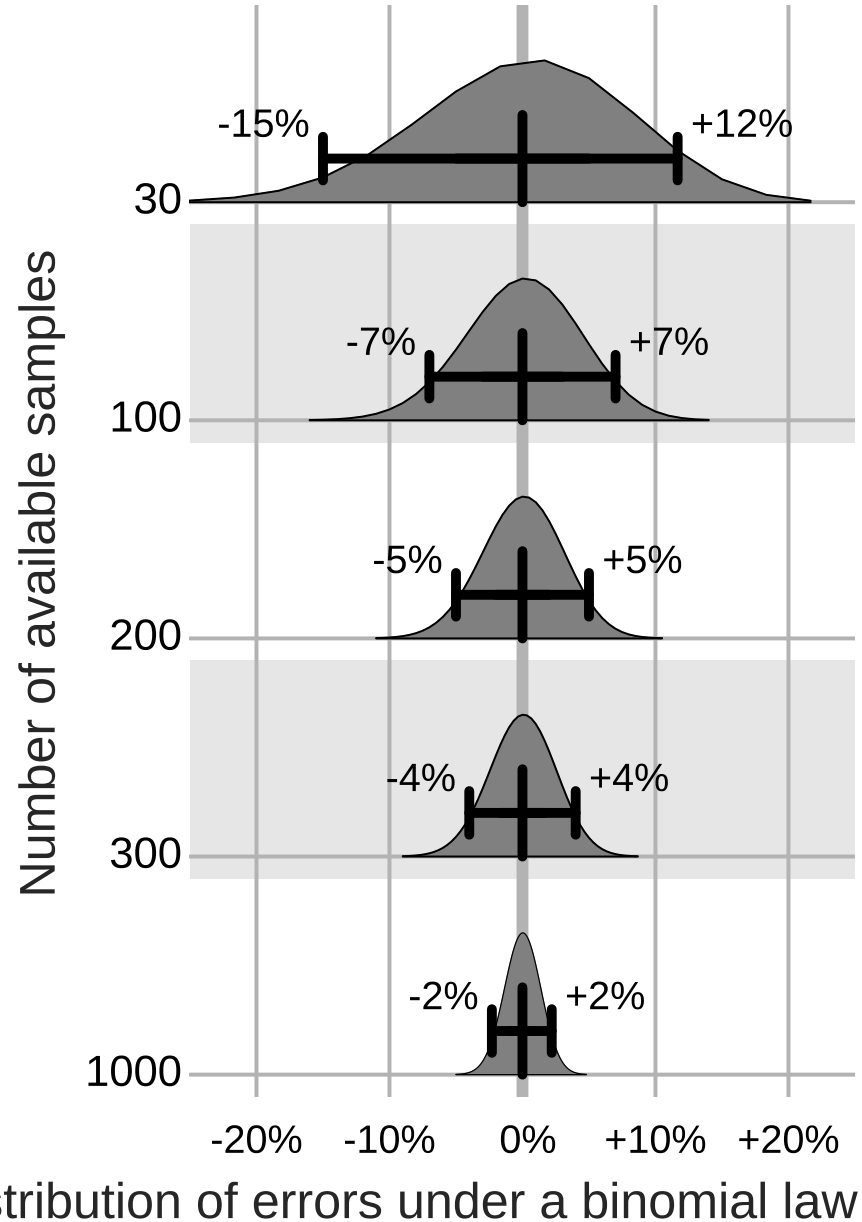
Fisher Kernels can go from likelihood to vector space

Very useful for non numeric objects

Limited-data settings

Reminder: Your validation measure is intrinsically unreliable
(sampling noise)

- Get more data
  For instance acquiring data on a related task, to learn representations
- Use simple models
  Do not spend too much time tweaking



[Varoquaux 2018]

References I

A. Abraham, M. P. Milham, A. Di Martino, R. C. Craddock, D. Samaras, B. Thirion, and G. Varoquaux. Deriving reproducible biomarkers from multi-site resting-state data: an autism-based example. *NeuroImage*, 147:736, 2017.

A. Achille and S. Soatto. Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 19(1):1947–1980, 2018.

A. Bellet, A. Habrard, and M. Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, 2013.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

References II

A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795, 2013.

D. Bzdok, M. Eickenberg, O. Grisel, B. Thirion, and G. Varoquaux. Semi-supervised factored logistic regression for high-dimensional neuroimaging data. In *Advances in Neural Information Processing Systems*, page 3348, 2015.

J. Canny. Gap: A factor model for discrete data. In *SIGIR*, page 122, 2004.

J.-F. Cardoso. Dependence, correlation and gaussianity in independent component analysis. *Journal of Machine Learning Research*, 4:1177, 2003.

References III

P. Cerda and G. Varoquaux. Encoding high-cardinality string categorical variables. *arXiv:1907.01860*, 2019.

W. Chen, X. Chen, J. Zhang, and K. Huang. Beyond triplet loss: a deep quadruplet network for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, page 403, 2017.

P. S. Dhillon, D. P. Foster, S. M. Kakade, and L. H. Ungar. A risk comparison of ordinary least squares vs ridge regression. *The Journal of Machine Learning Research*, 14:1505, 2013.

P. Drineas and M. W. Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *journal of machine learning research*, 6:2153, 2005.

References IV

C. Elkan. Deriving tf-idf as a fisher kernel. In *International Symposium on String Processing and Information Retrieval*, page 295, 2005.

Y. Goldberg and O. Levy. word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv:1402.3722*, 2014.

P. K. Gopalan, L. Charlin, and D. Blei. Content-based recommendations with poisson factorization. In *Advances in Neural Information Processing Systems*, page 3176, 2014.

M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, page 297, 2010.

References V

D. Hsu, S. Kakade, and T. Zhang. Random design analysis of ridge regression. *Foundations of Computational Mathematics*, 14, 2014.

A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4):411, 2000.

A. J. Izenman. Reduced-rank regression for the multivariate linear model. *Journal of multivariate analysis*, 5:248, 1975.

T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in neural information processing systems*, pages 487–493, 1999.

T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse processes*, 25:259, 1998.

References VI

A. Lefevre, F. Bach, and C. Févotte. Online algorithms for nonnegative matrix factorization with the itakura-saito divergence. In *Applications of Signal Processing to Audio and Acoustics (WASPAA)*, page 313. IEEE, 2011.

O. Levy and Y. Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, page 2177, 2014.

J. Mairal. Stochastic majorization-minimization algorithms for large-scale optimization. In *Advances in Neural Information Processing Systems*, 2013.

J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, 2010.

References VII

J. Mairal, F. Bach, and J. Ponce. Sparse modeling for image and vision processing. *Foundations and Trends® in Computer Graphics and Vision*, 8(2-3):85–283, 2014.

S. Mallat. Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A*, 374: 20150203, 2016.

A. Mensch, J. Mairal, B. Thirion, and G. Varoquaux. Stochastic subsampling for factorizing huge matrices. *IEEE Transactions on Signal Processing*, 66:113, 2017.

A. Mensch, J. Mairal, B. Thirion, and G. Varoquaux. Extracting universal representations of cognition across brain-imaging studies. *arXiv preprint arXiv:1809.06035*, 2018.

References VIII

D. Micci-Barreca. A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. *ACM SIGKDD Explorations Newsletter*, 3:27, 2001.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *ICLR Workshop Papers*. 2013a.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, page 3111, 2013b.

References IX

G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, page 2924, 2014.

L. Nicotra, A. Micheli, and A. Starita. Fisher kernel for tree structured data. In *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)*, volume 3, pages 1917–1922. IEEE, 2004.

E. Oyallon, E. Belilovsky, and S. Zagoruyko. Scaling the scattering transform: Deep hybrid networks. In *Proceedings of the IEEE international conference on computer vision*, page 5618, 2017.

References X

J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, page 1532, 2014.

M. Rahim, B. Thirion, D. Bzdok, I. Buvat, and G. Varoquaux. Joint prediction of multiple scores captures better individual traits from brain images. *Neuroimage*, 158:145–154, 2017a.

M. Rahim, B. Thirion, and G. Varoquaux. Multi-output predictions from neuroimaging: assessing reduced-rank linear models. In *2017 International Workshop on Pattern Recognition in Neuroimaging (PRNI)*, pages 1–4. IEEE, 2017b.

References XI

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

C. Rao. Information and accuracy attainable in the estimation of statistical parameters. *Bull Calcutta. Math. Soc.*, 37:81, 1945.

S. Rosset and R. J. Tibshirani. From fixed-x to random-x regression: Bias-variance decompositions, covariance penalties, and prediction error estimation. *Journal of the American Statistical Association*, pages 1–14, 2018.

B. Scholkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

References XII

G. Varoquaux. Cross-validation failure: small sample sizes lead to large error bars. *Neuroimage*, 180:68–77, 2018.

G. Varoquaux, F. Baronnet, A. Kleinschmidt, P. Fillard, and B. Thirion. Detection of brain functional-connectivity difference in post-stroke patients using group-level covariance modeling. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 200–208. Springer, 2010.

Q. Wang, Z. Mao, B. Wang, and L. Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12): 2724–2743, 2017.