Machine Learning and Science?

Michael W. Mahoney

ICSI and Dept of Statistics, UC Berkeley

(For more info, see:
http://www.stat.berkeley.edu/~mmahoney

February 2020

Outline

General thoughts

Randomized matrix algorithms

Matrix algorithms and scientific data

Matrix computations in really large-scale machine learning

Incorporating physical insight into matrix models

Thinking about large-scale data

Data generation is modern version of microscope/telescope*:

- See things couldn't see before: e.g., fine-scale movement of people, fine-scale clicks and interests; fine-scale tracking of packages; fine-scale measurements of temperature, chemicals, etc.
- Those inventions ushered new scientific eras and new understanding of the world and new technologies to do stuff

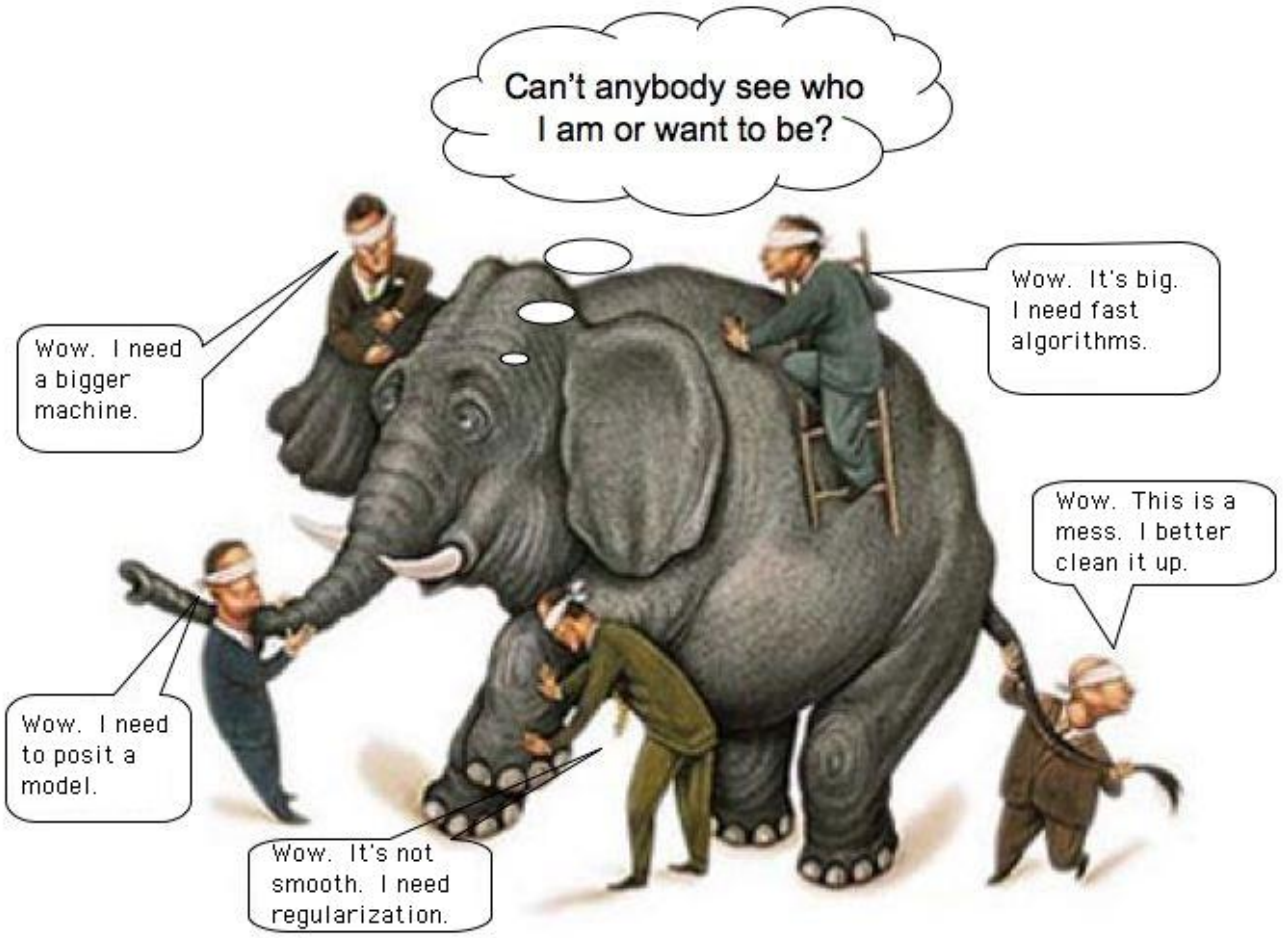
**But algorithms are also a "microscope/telescope" to probe data.*

Easy things become hard and hard things become easy:

- Easier to see the other side of universe than bottom of ocean
- Means, sums, medians, correlations is easy with small data

Our ability to generate data far exceeds our ability to extract insight from data.

How do we view BIG data?



Outline

General thoughts

Randomized matrix algorithms

Matrix algorithms and scientific data

Matrix computations in really large-scale machine learning

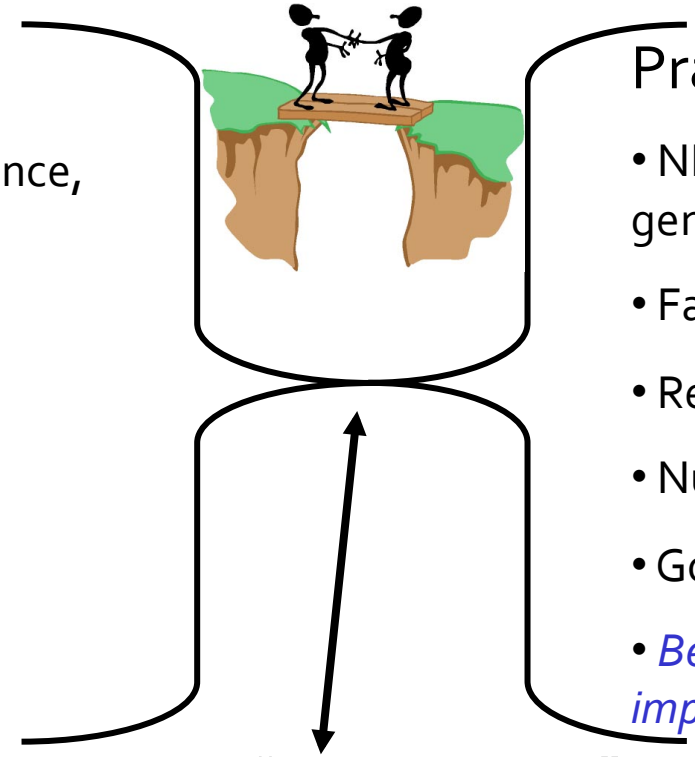
Incorporating physical insight into matrix models

Anecdote 1: Randomized Matrix Algorithms

Mahoney "Algorithmic and Statistical Perspectives on Large-Scale Data Analysis" (2010)
Mahoney "Randomized Algorithms for Matrices and Data" (2011)

Theoretical origins

- theoretical computer science, convex analysis, etc.
- Johnson-Lindenstrauss
- Additive-error algs
- Good worst-case analysis
- No statistical analysis
- No implementations



Practical applications

- NLA, ML, statistics, data analysis, genetics, etc
- Fast JL transform
- Relative-error algs
- Numerically-stable algs
- Good statistical properties
- *Beats LAPACK & parallel-distributed implementations on terabytes of data*

How to "bridge the gap"?

- decouple (implicitly or explicitly) randomization from linear algebra
- importance of statistical leverage scores!

Why RandNLA?

- **RandNLA** = **R**andomized **N**umerical **L**inear **A**lgebra
- Randomized methods for linear algebra can quickly produce approximate answers to familiar problems, such as computing the SVD, PCA or CUR.
- *Sacrifice some amount of accuracy for scalability.*

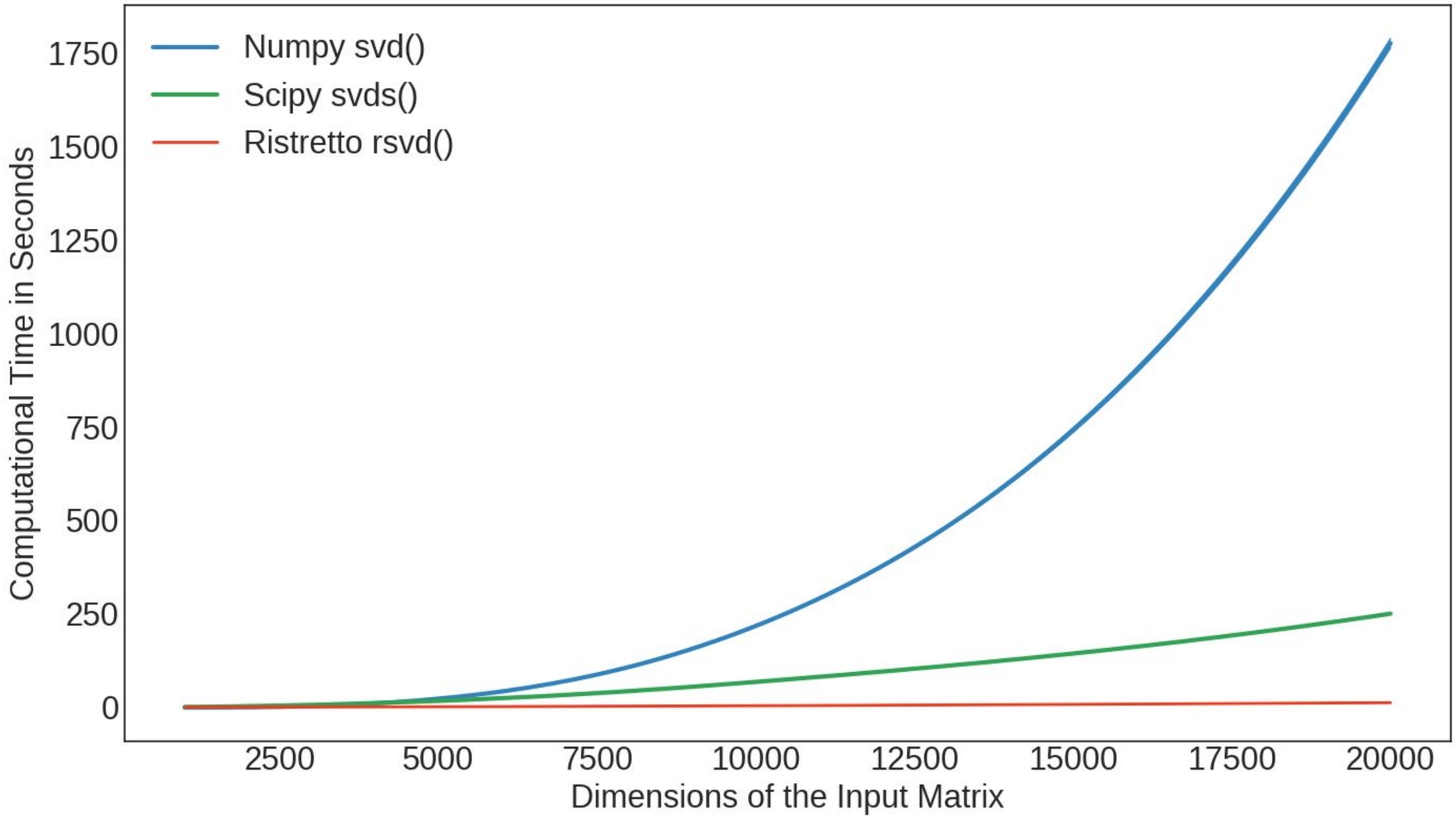
Mobile Computing

Bridge the Gap

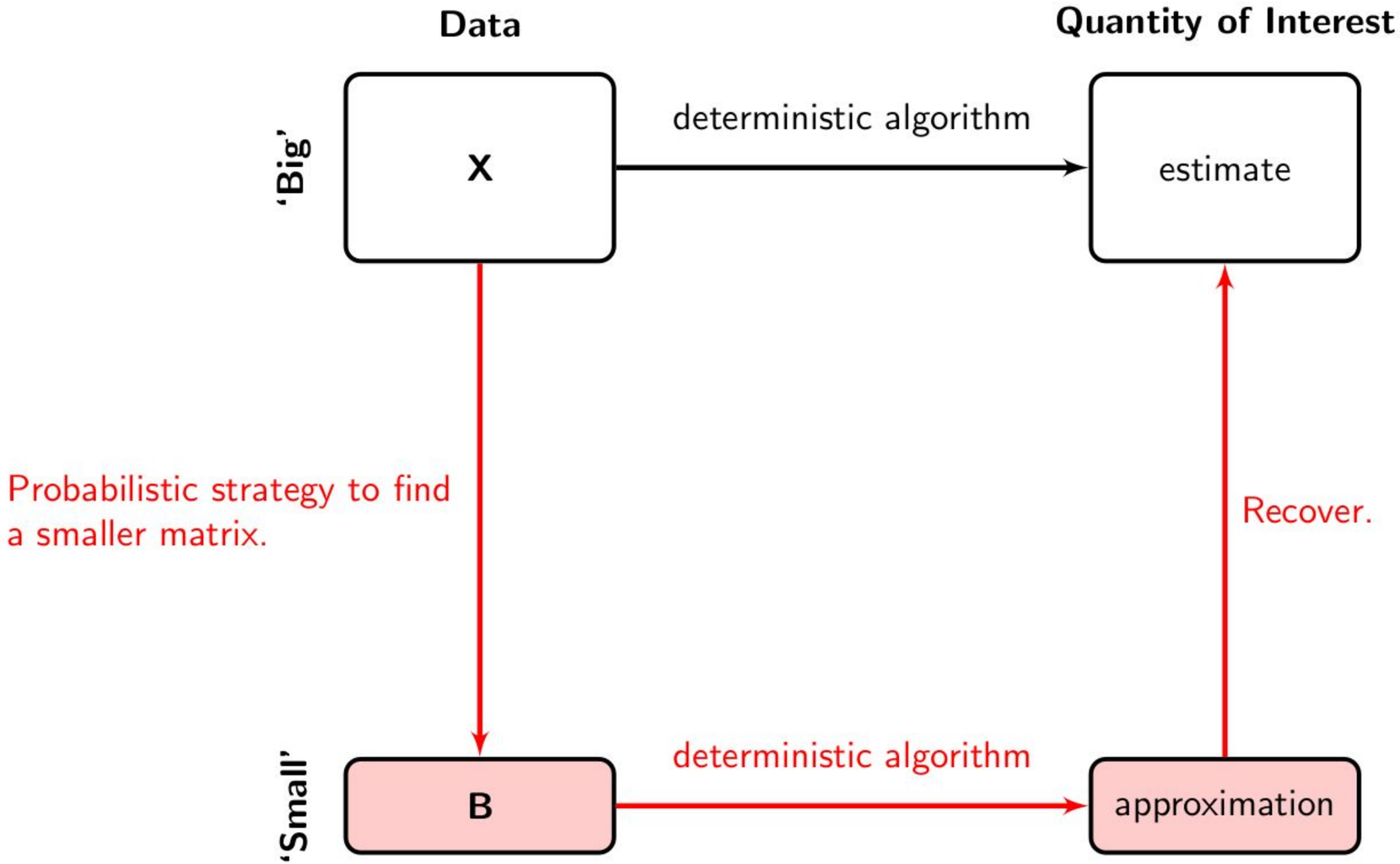
Super Computing

Deterministic SVD vs Randomized SVD

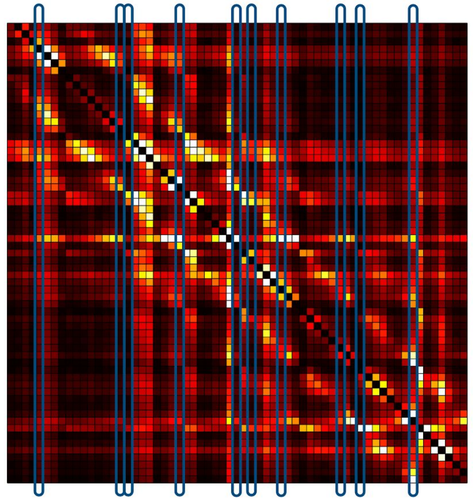
Using a AWS EC instance with 36 cores



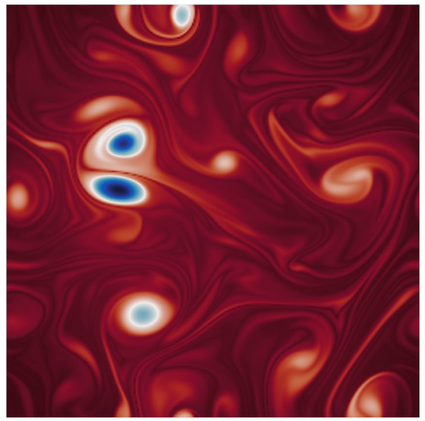
Turn a Big Matrix into a Small Matrix: Sketch to Solve



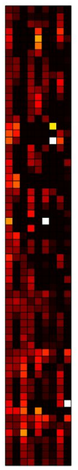
Motivating Example



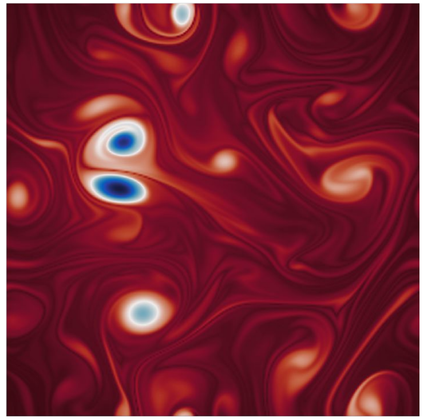
Compute
domiant eigenvector



Sketch



Approximated
domiant eigenvector



Outline

General thoughts

Randomized matrix algorithms

Matrix algorithms and scientific data

Matrix computations in really large-scale machine learning

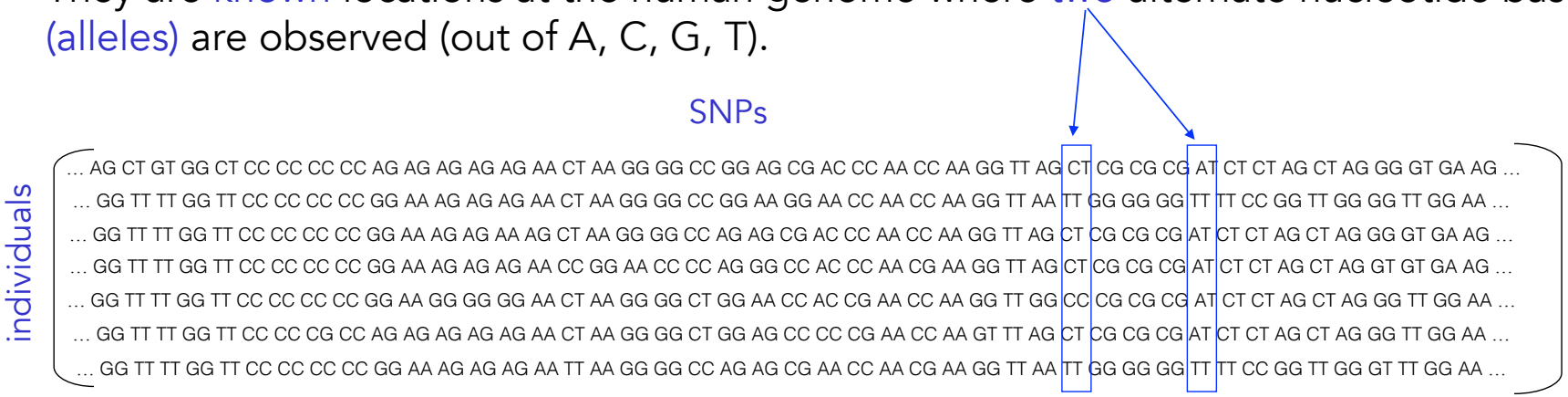
Incorporating physical insight into matrix models

Scientific data and choosing good columns as features

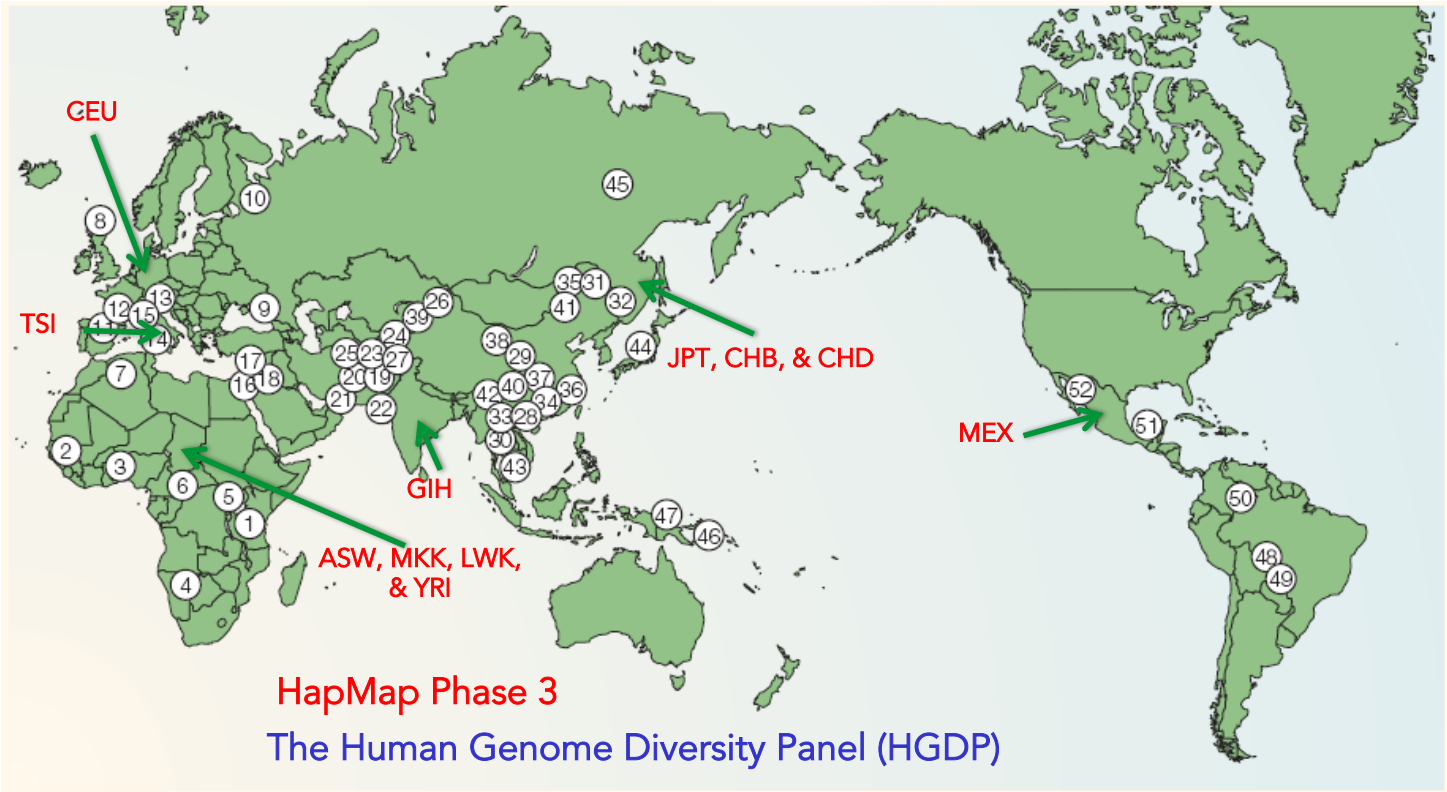
E.g., application in: Human Genetics

Single Nucleotide Polymorphisms: the most common type of genetic variation in the genome across different individuals.

They are known locations at the human genome where two alternate nucleotide bases (alleles) are observed (out of A, C, G, T).



Matrices including thousands of individuals and hundreds of thousands (large for some people, small for other people) if SNPs are available.



Africans
1 Bantu
2 Mandenka
3 Yoruba
4 San
5 Mbuti pygmy
6 Biaka
7 Mozabite

Europeans
8 Orcadian
9 Adygei
10 Russian
11 Basque
12 French
13 North Italian
14 Sardinian
15 Tuscan

Western Asians
16 Bedouin
17 Druze
18 Palestinian

Central and Southern Asians
19 Balochi
20 Brahui
21 Makrani
22 Sindhi
23 Pathan
24 Burusho
25 Hazara
26 Uygur
27 Kalash

Eastern Asians
28 Han (S. China)
29 Han (N. China)
30 Dai
31 Daur
32 Hezhen
33 Lahu
34 Miao
35 Oroqen
36 She
37 Tujia
38 Tu
39 Xibo
40 Yi
41 Mongola
42 Naxi
43 Cambodian
44 Japanese
45 Yakut

Oceanians
46 Melanesian
47 Papuan

Native Americans
48 Karitiana
49 Surui
50 Colombian
51 Maya
52 Pima

Cavalli-Sforza (2005) *Nat Genet Rev*

Rosenberg et al. (2002) *Science*

Li et al. (2008) *Science*

The International HapMap Consortium (2003, 2005, 2007) *Nature*

HGDP data

- 1,033 samples
- 7 geographic regions
- 52 populations

HapMap Phase 3 data

- 1,207 samples
- 11 populations

Apply SVD/PCA on the (joint) HGDP and HapMap Phase 3 data.

Matrix dimensions:

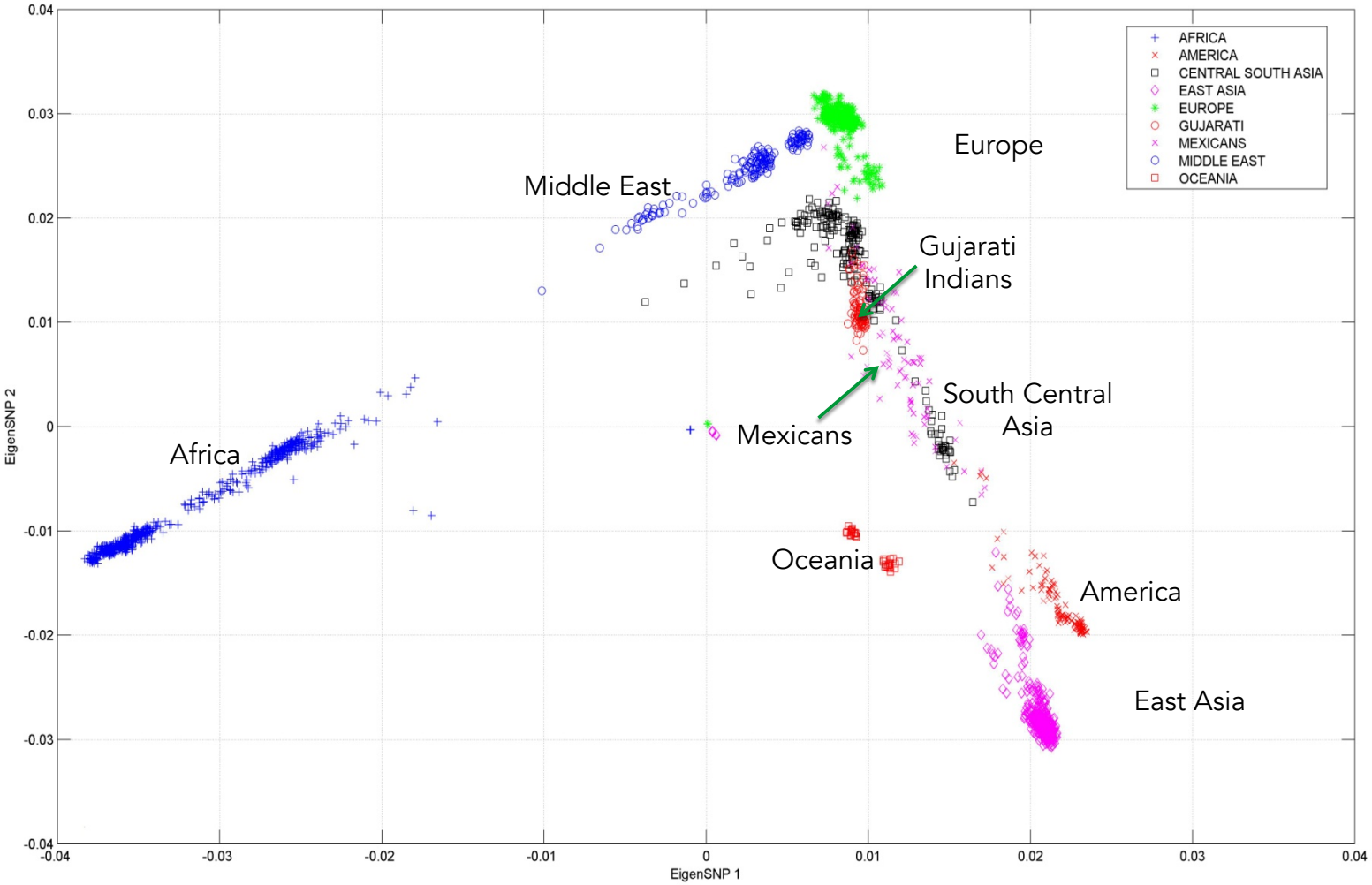
2,240 subjects (rows)

447,143 SNPs (columns)

Dense matrix:

over one billion entries

Paschou, et al (2010) J Med Genet



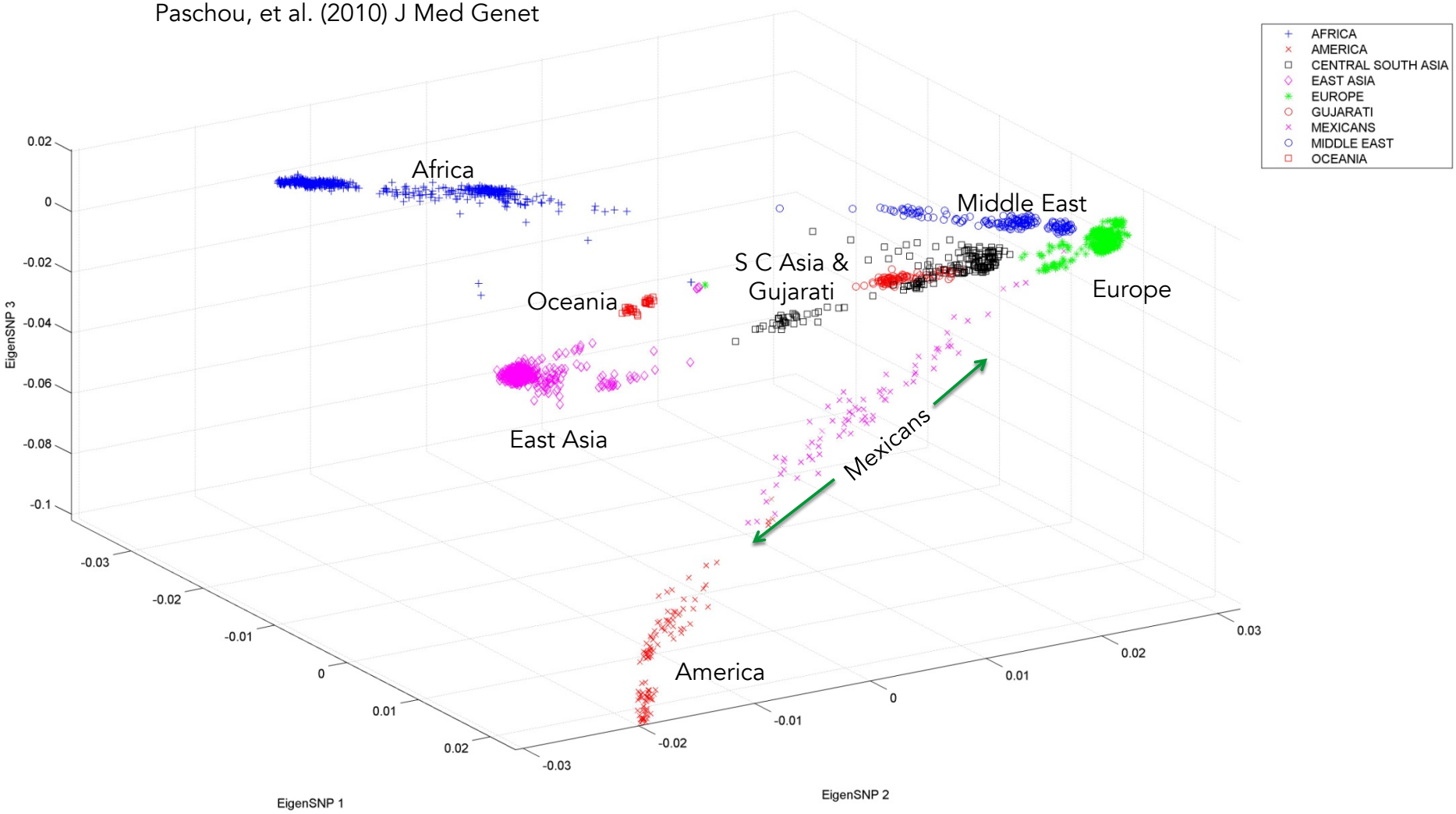
• Top two Principal Components (PCs or eigenSNPs)

(Lin and Altman (2005) *Am J Hum Genet*)

• The figure renders visual support to the "out-of-Africa" hypothesis.

• Mexican population seems out of place: we move to the top three PCs.

Paschou, et al. (2010) J Med Genet



- Not altogether satisfactory: the principal components are linear combinations of all SNPs, and – of course – can not be assayed!
- Can we find actual SNPs that capture the information in the singular vectors?
- Relatedly, can we compute them and/or the truncated SVD "efficiently."

Two related issues with eigen-analysis

Computing large SVDs: computational time

- In commodity hardware (e.g., a 4GB RAM, dual-core laptop), using MatLab 7.0 (R14), the computation of the SVD of the dense 2,240-by-447,143 matrix A takes ca 20 minutes.
- Computing this SVD is not a one-liner, since we can not load the whole matrix in RAM (runs out-of-memory in MatLab).
- Instead, compute the SVD of AA^T.
- In a similar experiment, compute **1,200 SVDs** on matrices of dimensions (approx.) 1,200-by-450,000 (roughly, a full leave-one-out cross-validation experiment) (DLP2010)

Selecting *actual columns* that "capture the structure" of the top PCs

- Combinatorial optimization problem; hard even for small matrices.
- Often called the Column Subset Selection Problem (CSSP).
- Not clear that such "good" columns even exist.
- Avoid "reification" problem of "interpreting" singular vectors!
- (Solvable in "random projection time" with CX/CUR decompositions! (PNAS, MD09))

Outline

General thoughts

Randomized matrix algorithms

Matrix algorithms and scientific data

Matrix computations in really large-scale machine learning

Incorporating physical insight into matrix models

Where do you run your linear algebra?

Single machine

- Think about RAM, call LAPACK, etc.
- Someone else thought about numerical issues, memory hierarchies, etc.
- This is the 99%

Supercomputer

- High end, compute-intensive.
- Big emphasis on HPC (High Performance Computing)
- C+MPI, etc.

Distributed data center

- High end, data-intensive
- BIG emphasis on HPC (High Productivity Computing)
- Databases, MapReduce/Hadoop, Spark, etc.

Apache Spark

- Cluster computing system
- Interoperable with Apache Hadoop, much faster
- Improved efficiency:
  - In-memory computing primitives
  - Computation graphs
- Rich and easy-to-use API
- Allows for iterative algorithms (important for ML and linear algebra)
- Fault tolerant

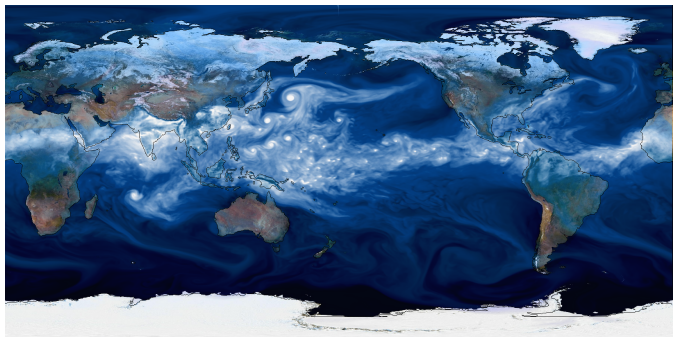
MPI

- MPI = Message Passing Interface
- A *specification* for the developers and users of message passing libraries
- *Message-Passing Parallel Programming Model*: cooperative operations between processes, data moved from address space of one process to that of another
- Popular *implementations*: MPICH, Open MPI

Our Goals

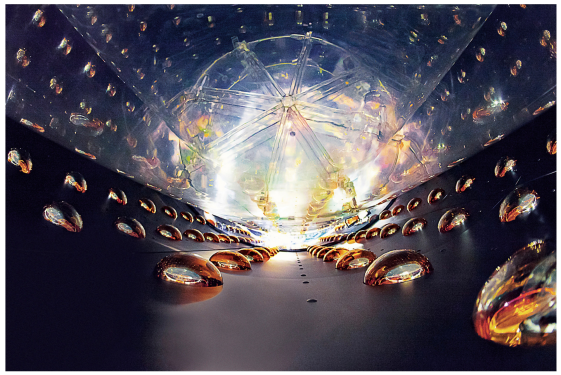
- ***Provide implementations*** of low-rank factorizations (PCA, NMF, and randomized CX) in Spark
- Apply low-rank matrix factorization methods ***to TB-scale scientific datasets*** in Spark
- Understand Spark performance on ***commodity clusters vs HPC platforms***
- ***Quantify the scalability gaps*** between highly-tuned C/MPI and current Spark-based implementations
- ***Provide a general-purpose interface*** for matrix-based algorithms between Spark and traditional MPI codes

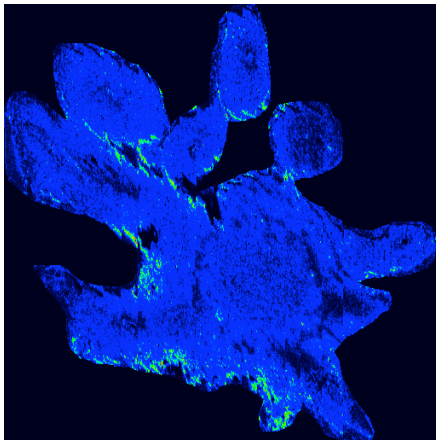
Three Science Drivers



Climate Science:
extract trends in variations of oceanic and atmospheric variables (**PCA**)

Nuclear Physics:
learn useful patterns for classification of subatomic particles (**NMF**)





Mass Spectrometry:
location of chemically important ions (**CX**)

Datasets

Science Area	Format/Files	Dimensions	Size
MSI	Parquet/2880	$8,258,911 \times 131,048$	1.1TB
Daya Bay	HDF5/1	$1,099,413,914 \times 192$	1.6TB
Ocean	HDF5/1	$6,349,676 \times 46,715$	2.2TB
Atmosphere	HDF5/1	$26,542,080 \times 81,600$	16TB

MSI — a sparse matrix from measurements of drift times and mass charge ratios at each pixel of a sample of *Peltatum*; used for CX decomposition

Daya Bay — neutrino sensor array measurements; used for NMF

Ocean and Atmosphere — climate variables (ocean temperature, atmospheric humidity) measured on a 3D grid at 3 or 6 hour intervals over about 30 years; used for PCA

Our first (of by now several) results

Matrix Factorizations at Scale: a Comparison of Scientific Data Analytics in Spark and C+MPI Using Three Case Studies

Alex Gittens [*] Aditya Devarakonda[†] Evan Racah[‡] Michael Ringenburg[§]
Lisa Gerhardt[‡] Jey Kottaalam[†] Jialin Liu[‡] Kristyn Maschhoff[§] Shane Canon[‡]
Jatin Chhugani[¶] Pramod Sharma[§] Jiyan Yang[‖] James Demmel[**] Jim Harrell[§]
Venkat Krishnamurthy[§] Michael W. Mahoney[*] Prabhat [‡]

July 1, 2016

arXiv:1607.01335v1 [cs.DC] 5 Jul 2016

[*]ICSI and Department of Statistics, UC Berkeley
[†]EECS, UC Berkeley
[‡]NERSC, Lawrence Berkeley National Laboratory
[§]Cray, Inc.
[¶]Hiperform Consulting LLC
[‖]ICME, Stanford University
[**]EECS and Math, UC Berkeley

BIG thanks in particular to:
- Mike Ringenberg
- Kristi Maschoff
- Pramod Sharma
- Jim Harrell
- Venkat Krishnamurthy

(and Cray for funding!)

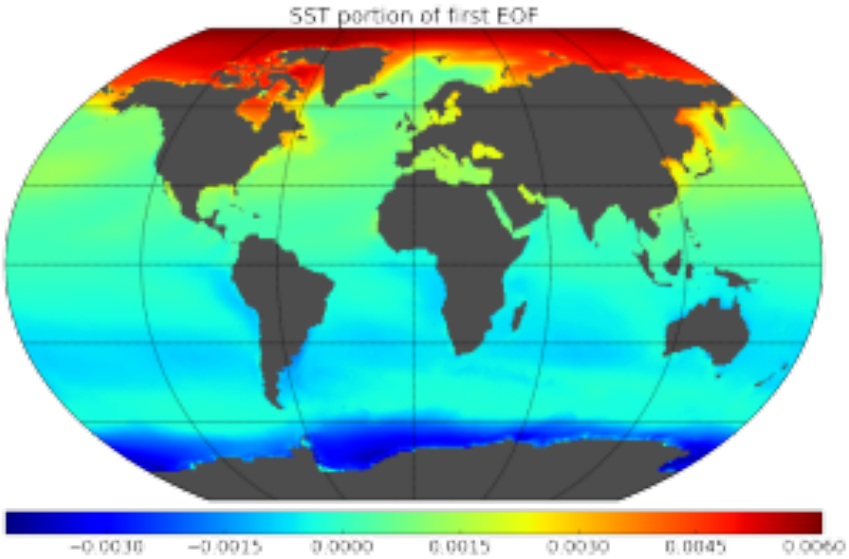
Randomized CX/CUR Decomposition

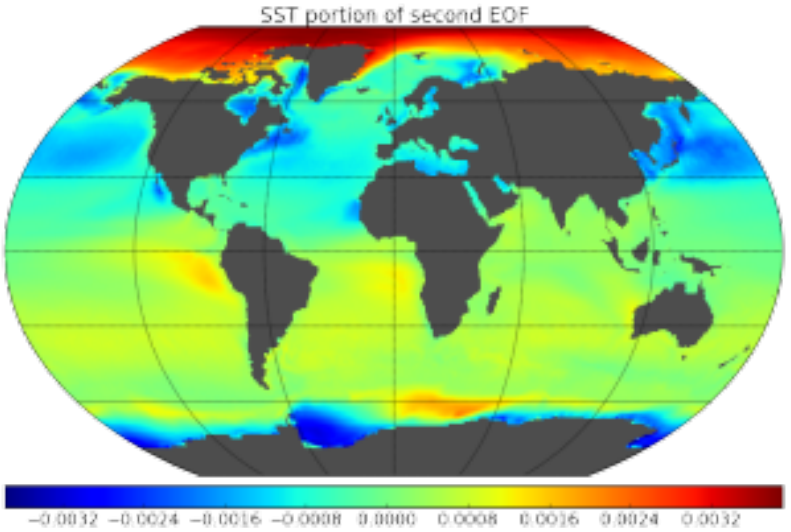
- Dimensionality reduction is a ubiquitous tool in science (bio-imaging, neuro-imaging, genetics, chemistry, climatology, …), typical approaches include PCA and NMF which give approximations that rely on non-interpretable combinations of the data points in A

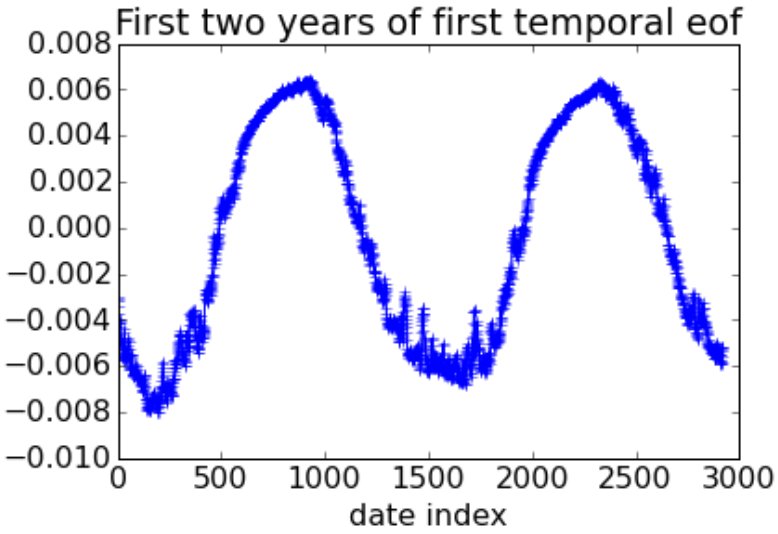
- PCA, NMF lack reifiability. Instead, CX matrix decompositions identify **exemplar** data points (columns of A) that capture the same information as the top singular vectors, and give approximations of the form

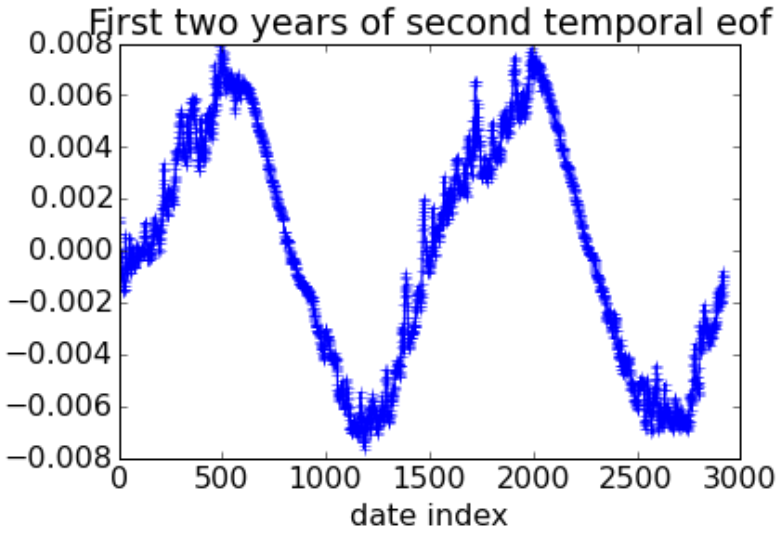
$$\mathbf{A} \approx \mathbf{CX}$$

CFSR Ocean Temperature Dataset (II)









Running times for NMF and PCA

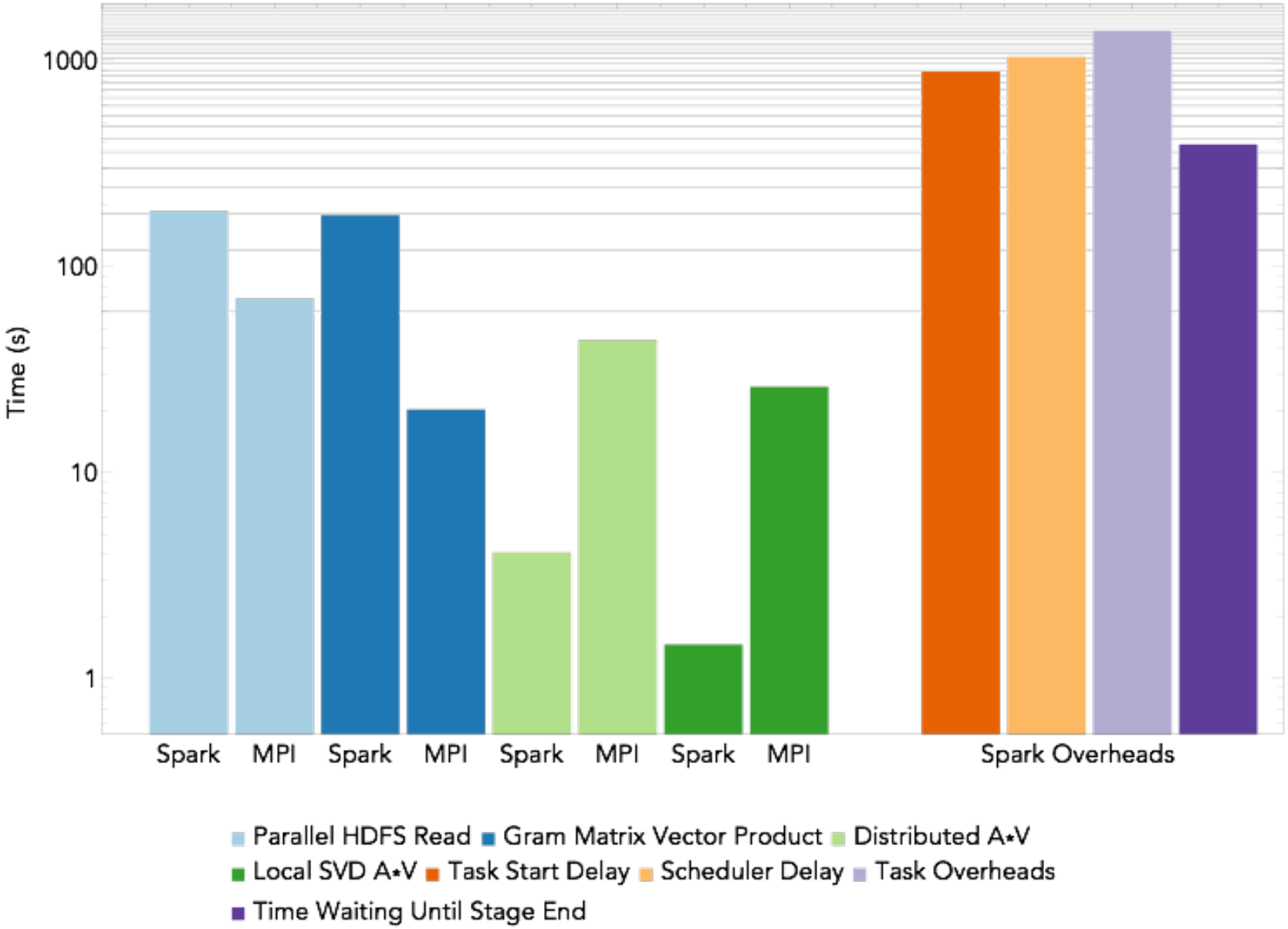
Cori's specs:

- 1630 compute nodes,
- 128 GB/node,
- 32 2.3GHz Haswell cores/node

	Nodes / cores	MPI Time	Spark Time	Gap
NMF	50 / 1,600	1 min 6 s	4 min 38 s	4.2x
	100 / 3,200	45 s	3 min 27 s	4.6x
	300 / 9,600	30 s	70 s	2.3x
PCA (2.2TB)	100 / 3,200	1 min 34 s	15 min 34 s	9.9x
	300 / 9,600	1 min	13 min 47 s	13.8x
	500 / 16,000	56 s	19 min 20 s	20.7x
PCA (16TB)	MPI: 1,600 / 51,200 Spark: 1,522 / 48,704	2 min 40 s	69 min 35 s	26x

- Anti-scaling!
- And it worsens both with concurrency and data size.

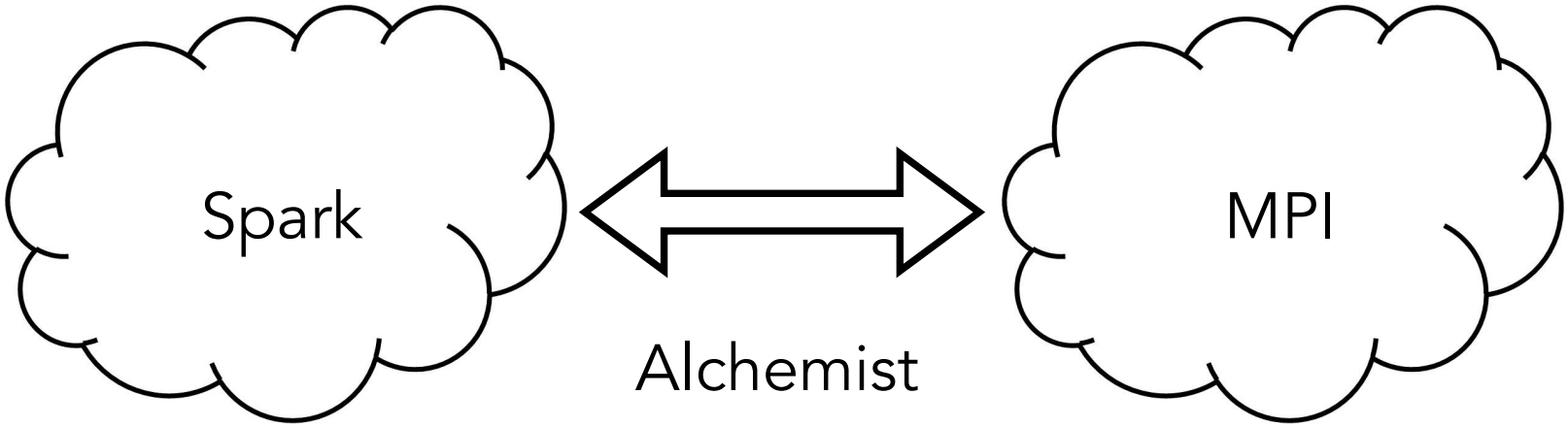
Rank 20 PCA of 16 TB Climate using 48K+ cores



MPI vs Spark: Lessons Learned

- With favorable data (tall and skinny) and well-adapted algorithms, ***Spark LA is 4x-26x slower than MPI when IO is included***
- ***Spark overheads are orders of magnitude higher than the computations*** in PCA (time till stage end, scheduler delay, task start delay, executor deserialize time), ***and it anti-scales***
- The large gaps mean it is worthwhile to ***investigate efficiently interfacing MPI-based codes with Spark***

Using Alchemist



Spark:

1) Sends the metadata for input and output matrices to Alchemist

2) Sends the matrix to Alchemist using sockets

3) Sends commands to the Alchemist driver

Alchemist:

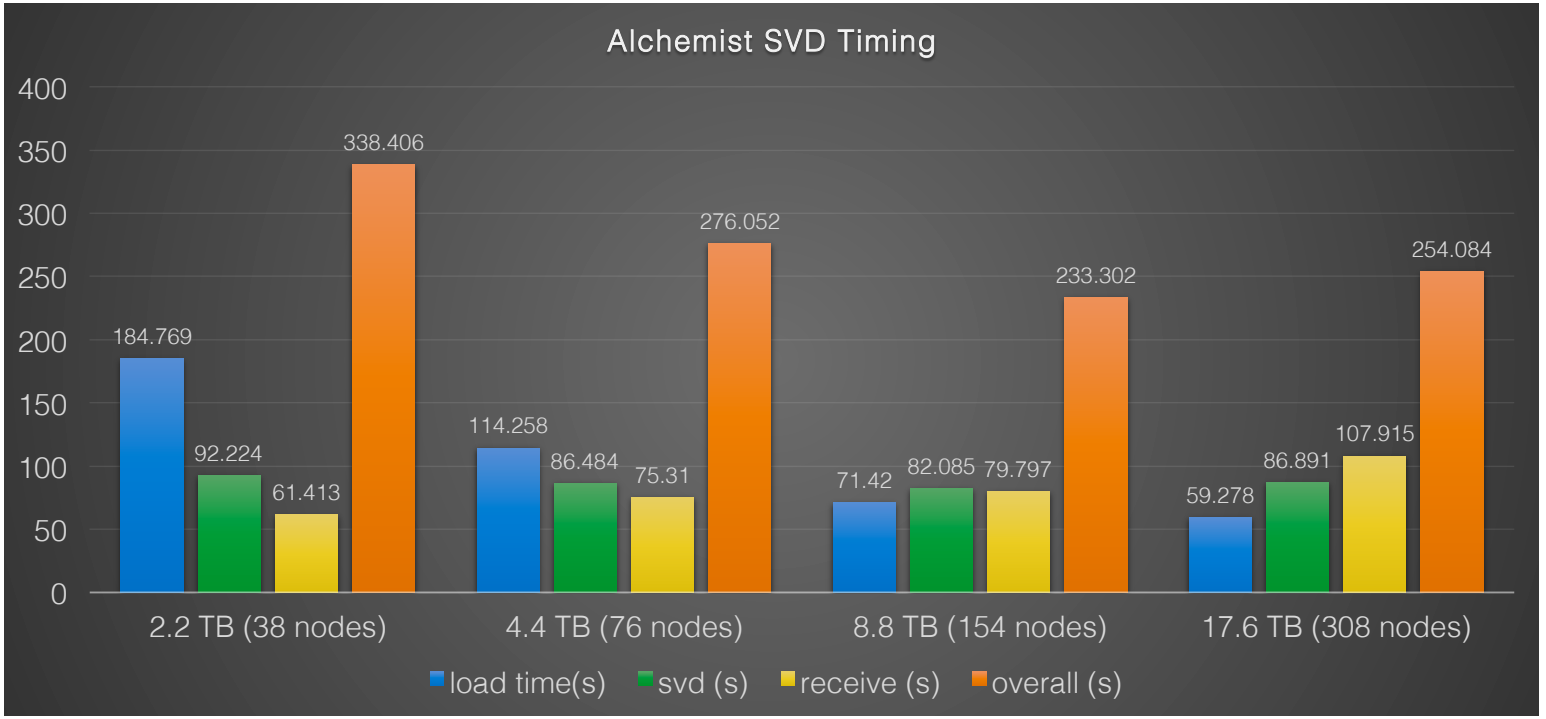
1) Repartitions the matrix for MPI using Elemental

2) Driver coordinates workers in executing the MPI codes

3) Returns outputs to Spark

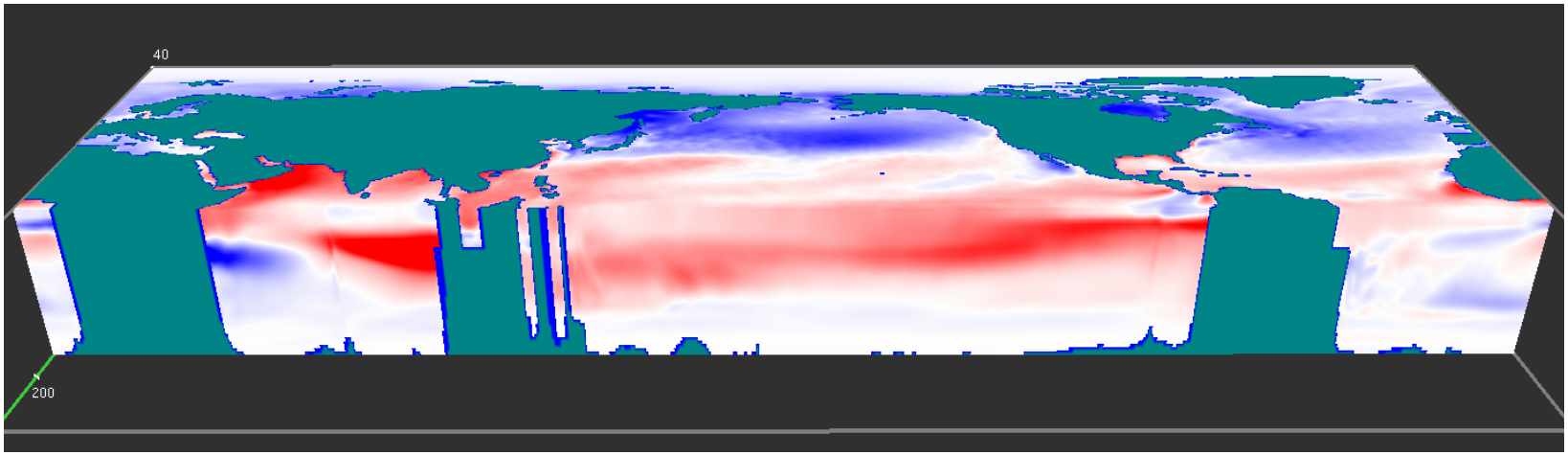
Scaling of Truncated SVD

- Each node of Cori has 128GB RAM and 32 cores
- Replicated the 2.2 TB Climate data set (row-wise)
- Use Alchemist to get rank 20 truncated SVD
- *Prior work shows Spark requires 934s on 100 nodes for just 2.2 TB*



A Climate Science Application

- Problem: extract the top principal components of a 3D ocean temperature data set collected over 30 years at 3 hour increments on a 360-by-720-by-40 lat-long-depth grid
- Yields a **2.2TB matrix, 6M-by-46K and dense**



Visualization of the 5th principal component

Outline

General thoughts

Randomized matrix algorithms

Matrix algorithms and scientific data

Matrix computations in really large-scale machine learning

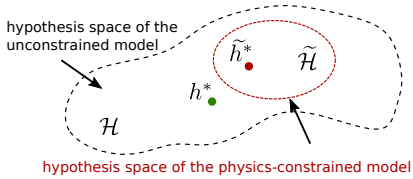
Incorporating physical insight into matrix models

Physics-constrained learning (PCL)

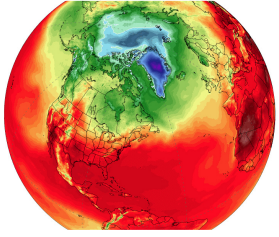
- Deep learning aims to learn a model $\mathcal{H}$ that best maps a set of inputs $\mathcal{X}$ to a set of outputs $\mathcal{Y}$:

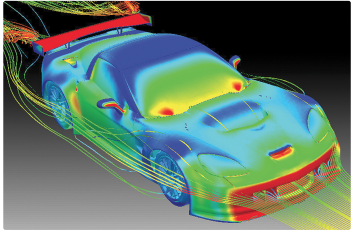
$$\mathcal{H} : \mathcal{X} \to \mathcal{Y}$$

- We hope that this model also works on new inputs.



PCL aims to introduce prior knowledge about the problem into the learning process.





Problem setup: Fluid flow prediction

- We assume that the dynamical system of interest can be modeled as

$$\mathbf{x}_{t+1} = \mathcal{A}(\mathbf{x}_t) + \eta_t, \quad t = 0, 1, 2, \ldots, T.$$

- In a data-driven setting we might only have access to (high-dimensional) observations

$$\mathbf{y}_t = \mathcal{G}(\mathbf{x}_t) + \xi_t, \quad t = 0, 1, 2, \ldots, T.$$

- Given a sequence of observations $\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_T \in \mathbb{R}^m$ for training, the objective of this work is to learn a model which maps the snapshot $\mathbf{y}_t$ to $\mathbf{y}_{t+1}$.

Problem setup: Fluid flow prediction

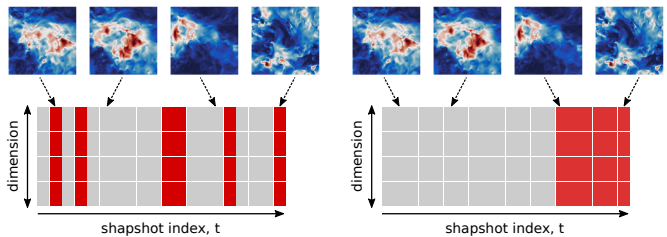
- We assume that the dynamical system of interest can be modeled as

$$\mathbf{x}_{t+1} = \mathcal{A}(\mathbf{x}_t) + \eta_t, \quad t = 0, 1, 2, \ldots, T.$$

- In a data-driven setting we might only have access to (high-dimensional) observations

$$\mathbf{y}_t = \mathcal{G}(\mathbf{x}_t) + \xi_t, \quad t = 0, 1, 2, \ldots, T.$$

- Given a sequence of observations $\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_T \in \mathbb{R}^m$ for training, the objective of this work is to learn a model which maps the snapshot $\mathbf{y}_t$ to $\mathbf{y}_{t+1}$.



Physics-agnostic model

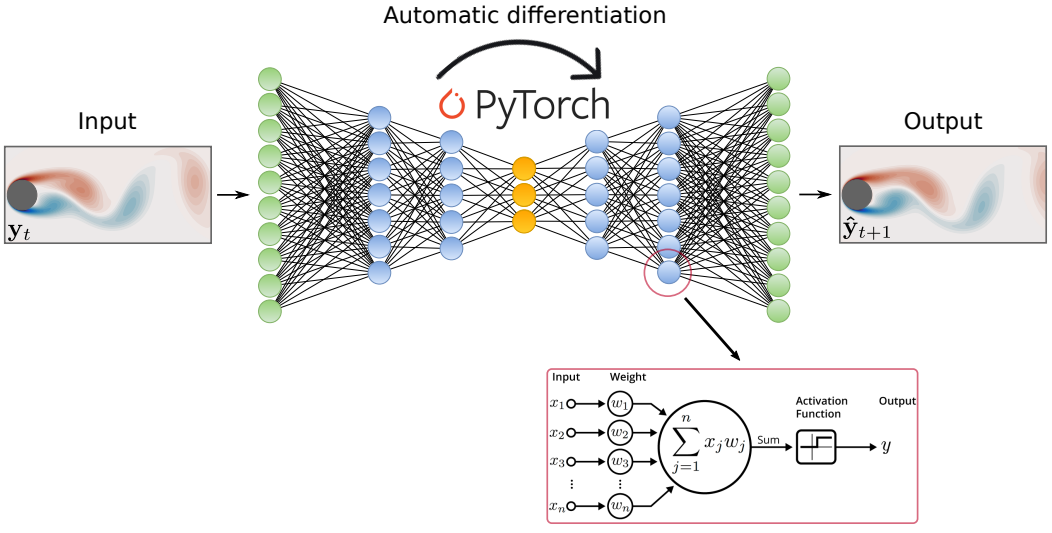
- Given the pairs $\{\mathbf{y}_t, \mathbf{y}_{t+1}\}_{t=1,2,...T}$, we train a model by minimizing the MSE

$$\min \frac{1}{T} \sum_{t=0}^{T} \|\mathbf{y}_{t+1} - \mathcal{F}(\mathbf{y}_t)\|_2^2.$$

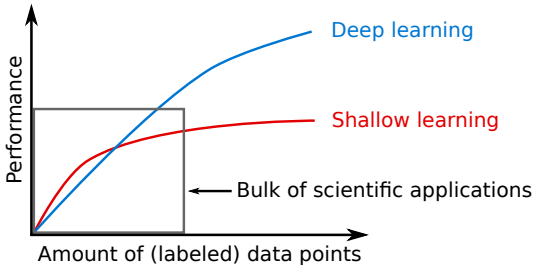
- During inference time, we can obtain predictions by composing the learned model k-times

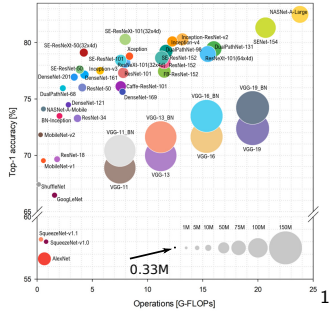
$$\hat{\mathbf{y}}_k = \mathcal{F} \circ \mathcal{F} \circ \mathcal{F} \circ \ ... \ \circ \mathcal{F}(\mathbf{y}_0).$$

Dynamic Autoencoder



We use shallow networks...





	very shallow	(our) shallow	deeper
Computational demands:	low	☺☺☺	high
Time for hyper-parameter tuning:	low	☺☺☺	high
Complexity of architecture design:	low	☺☺☺	high
Inference time:	low	☺☺☺	high
Carbon footprint:	low	☺☺☺	high

[1]Adapted from `https://arxiv.org/pdf/1810.00736.pdf`

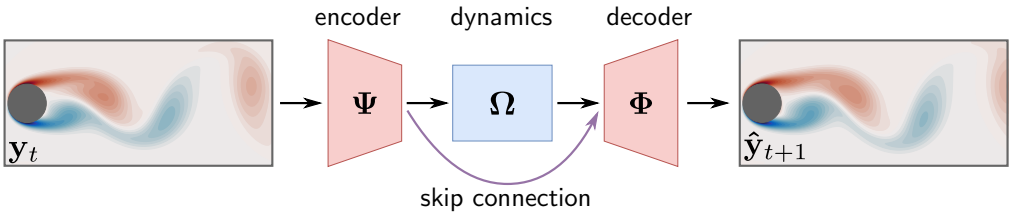
Omega Autoencoder

- A new loss function:

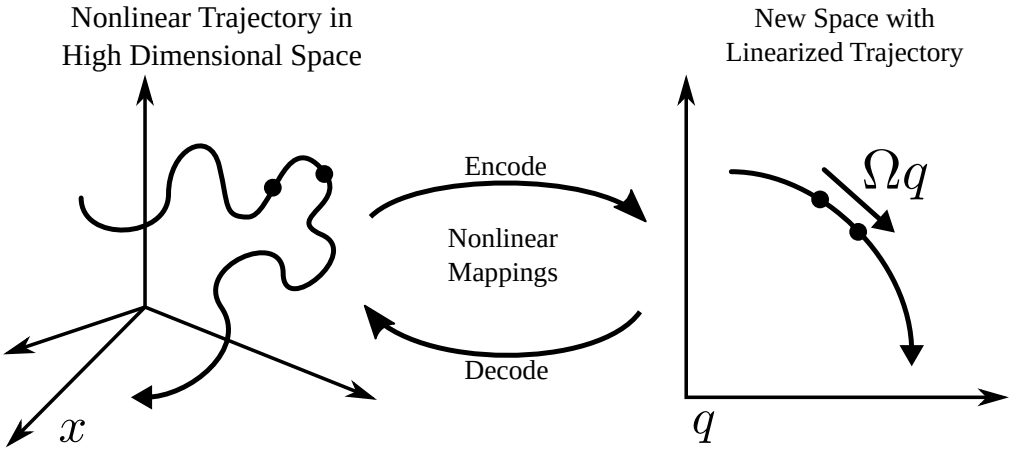
$$\min \frac{1}{T} \sum_{t=0}^{T} \|\mathbf{y}_{t+1} - \mathbf{\Phi} \circ \mathbf{\Omega} \circ \mathbf{\Psi}(\mathbf{y}_t)\|_2^2 + \lambda \|\mathbf{y}_t - \mathbf{\Phi} \circ \mathbf{\Psi}(\mathbf{y}_t)\|_2^2.$$

- If the model obeys the assumption that $\mathbf{\Psi}$ approximates $\mathcal{G}^{-1}$, then we have that

$$\hat{\mathbf{y}}_k \approx \mathbf{\Phi} \circ \mathbf{\Omega}^k \circ \mathbf{\Psi}(\mathbf{y}_0).$$



Coordinate transform



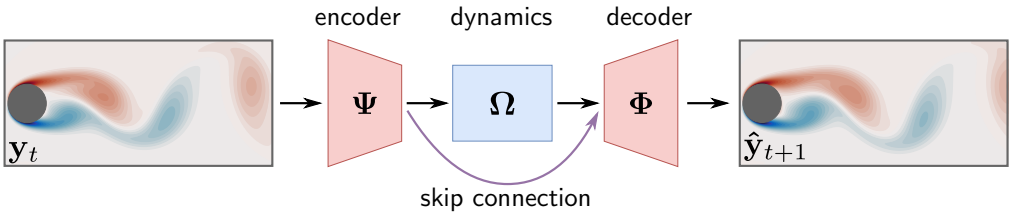
From black box to gray box models

- We start by adding a meaningful constraint to our model:

$$\min \frac{1}{T} \sum_{t=0}^{T} \|\mathbf{y}_{t+1} - \mathbf{\Phi} \circ \mathbf{\Omega} \circ \mathbf{\Psi}(\mathbf{y}_t)\|_2^2 + \lambda \|\mathbf{y}_t - \mathbf{\Phi} \circ \mathbf{\Psi}(\mathbf{y}_t)\|_2^2 + \kappa \ \rho(\mathbf{\Omega}).$$

- If the model obeys the assumption that $\mathbf{\Psi}$ approximates $\mathcal{G}^{-1}$, then we have that

$$\hat{\mathbf{y}}_k \approx \mathbf{\Phi} \circ \mathbf{\Omega}^k \circ \mathbf{\Psi}(\mathbf{y}_0).$$

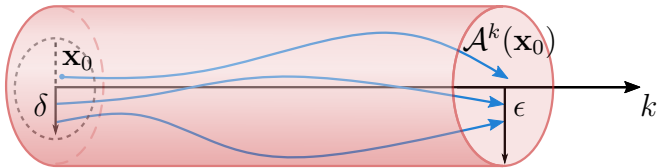


Lyapunov stability in a nutshell

- The origin of a dynamic system

$$\mathbf{x}_{t+1} = \mathcal{A}(\mathbf{x}_t) + \eta_t \quad t = 0, 1, 2, \ldots, T.$$

  is stable if all trajectories that starting arbitrarily close to the origin (in a ball of radius δ) remain arbitrarily close (in a ball of radius ϵ).



- If the dynamics $\mathcal{A}$ are linear, stability can be checked with an eigenvalue analysis.

Lyapunov's method... an idea from over 120 years ago[2]

- For linear systems, Lyapunov's second method states that a dynamic system

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \eta_t \quad t = 0, 1, 2, \ldots, T$$

is stable if and only if for any (symmetric) positive definite matrix $\mathbf{Q} \in \mathbb{R}^{n \times n}$ there exists a (symmetric) positive definite matrix $\mathbf{P} \in \mathbb{R}^{n \times n}$ satisfying

$$\mathbf{A}^\top \mathbf{P}\mathbf{A} - \mathbf{P} = -\mathbf{Q}.$$

[2] https://stanford.edu/~boyd/papers/pdf/springer_15_colloquium.pdf

Lyapunov's method... an idea from over 120 years ago[2]

- For linear systems, Lyapunov's second method states that a dynamic system

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \eta_t \quad t = 0, 1, 2, \ldots, T$$

  is stable if and only if for any (symmetric) positive definite matrix $\mathbf{Q} \in \mathbb{R}^{n \times n}$ there exists a (symmetric) positive definite matrix $\mathbf{P} \in \mathbb{R}^{n \times n}$ satisfying

$$\mathbf{A}^\top \mathbf{P} \mathbf{A} - \mathbf{P} = -\mathbf{Q}.$$

- Using this idea, we impose that the symmetric matrix $\mathbf{P}$, defined by

$$\mathbf{\Omega}^\top \mathbf{P} \mathbf{\Omega} - \mathbf{P} = -\mathbf{I},$$

  is positive definite.

[2] https://stanford.edu/~boyd/papers/pdf/springer_15_colloquium.pdf

To gain some intuition...

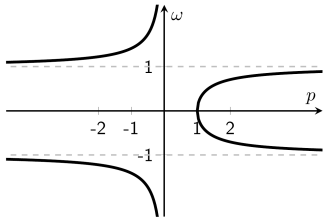
- ... we consider the case where $\mathbf{\Omega}$ is diagonalizable and $\mathbf{Q}$ chosen appropriately.
- Then, for a particular choice of coordinates the following problem

$$\mathbf{\Omega}^\top \mathbf{P} \mathbf{\Omega} - \mathbf{P} = -\mathbf{I}, \tag{1}$$

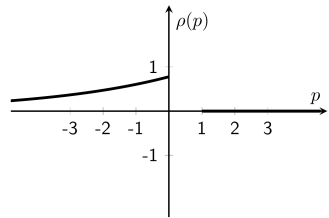
reduces to the system of linear equations

$$\omega_i p_i \omega_i - p_i = -1, \tag{2}$$

where ω_i, p_i, for $i = 1, 2, \ldots, n$, denote the eigenvalues of $\mathbf{\Omega}$ and $\mathbf{P}$, respectively.



(a) Discrete-time Lyapunov function.



(b) Stability promoting prior.

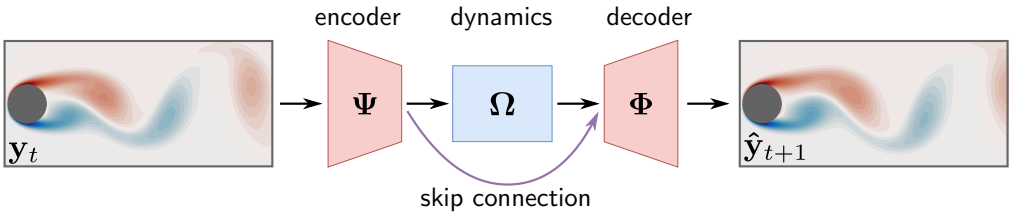
Physics-aware model that preserves stability

- The physics-informed autoencoder is trained by minimizing the following objective

$$\min \frac{1}{T}\sum_{t=0}^{T} \|\mathbf{y}_{t+1} - \mathbf{\Phi}\circ\mathbf{\Omega}\circ\mathbf{\Psi}(\mathbf{y}_t)\|_2^2 \; + \; \lambda\|\mathbf{y}_t - \mathbf{\Phi}\circ\mathbf{\Psi}(\mathbf{y}_t)\|_2^2 + \; \kappa\sum_i \rho(p_i).$$

- The prior p can take various forms. We use the following in our experiments:

$$\rho(p) := \begin{cases} \exp\left(-\frac{|p-1|}{\gamma}\right) & \text{if } p < 0 \\ 0 & \text{otherwise.} \end{cases}$$



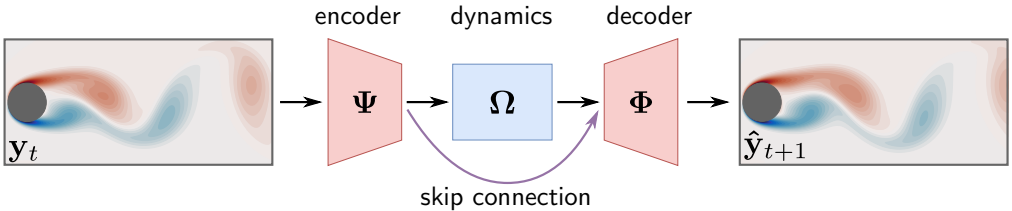
Physics-aware model that preserves stability

- The physics-informed autoencoder is trained by minimizing the following objective

$$\min \frac{1}{T}\sum_{t=0}^{T} \|\mathbf{y}_{t+1}-\mathbf{\Phi}\circ\mathbf{\Omega}\circ\mathbf{\Psi}(\mathbf{y}_t)\|_2^2+\|\mathbf{y}_{t+2}-\mathbf{\Phi}\circ\mathbf{\Omega}\circ\mathbf{\Omega}\circ\mathbf{\Psi}(\mathbf{y}_t)\|_2^2+\lambda\|\mathbf{y}_t-\mathbf{\Phi}\circ\mathbf{\Psi}(\mathbf{y}_t)\|_2^2+\kappa\sum_i \rho(p_i).$$

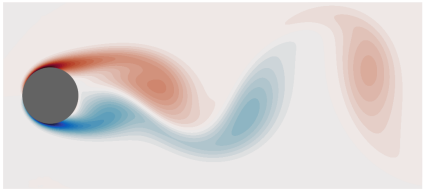
- The prior p can take various forms. We use the following in our experiments:

$$\rho(p) := \begin{cases} \exp\left(-\frac{|p-1|}{\gamma}\right) & \text{if } p < 0 \\ 0 & \text{otherwise.} \end{cases}$$

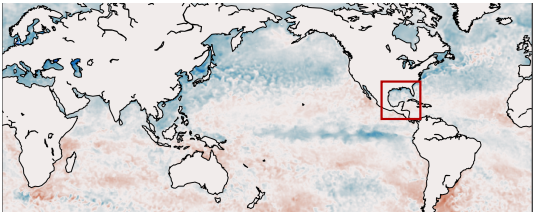


Examples that we consider

- Flow past the cylinder.

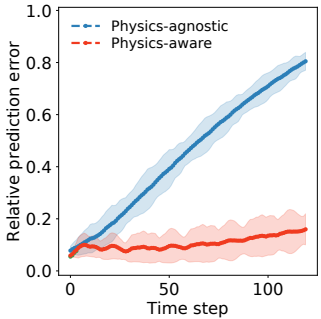


- Daily sea surface temperature data of the gulf of Mexico over a period of 6 years.

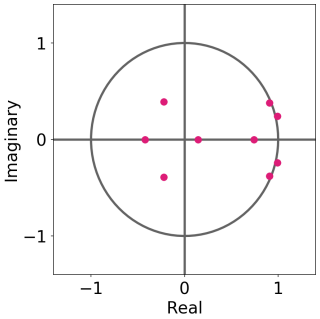


Prediction performance for flow past the cylinder (without weight decay)

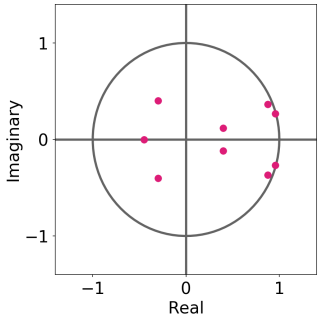
$$\min \frac{1}{T}\sum_{t=0}^{T} \|\mathbf{y}_{t+1}-\mathbf{\Phi}\circ\mathbf{\Omega}\circ\mathbf{\Psi}(\mathbf{y}_t)\|_2^2+\|\mathbf{y}_{t+2}-\mathbf{\Phi}\circ\mathbf{\Omega}\circ\mathbf{\Omega}\circ\mathbf{\Psi}(\mathbf{y}_t)\|_2^2+\lambda\|\mathbf{y}_t-\mathbf{\Phi}\circ\mathbf{\Psi}(\mathbf{y}_t)\|_2^2+\kappa\sum_i \rho(p_i).$$



(a) With LR $1e{-}2$.

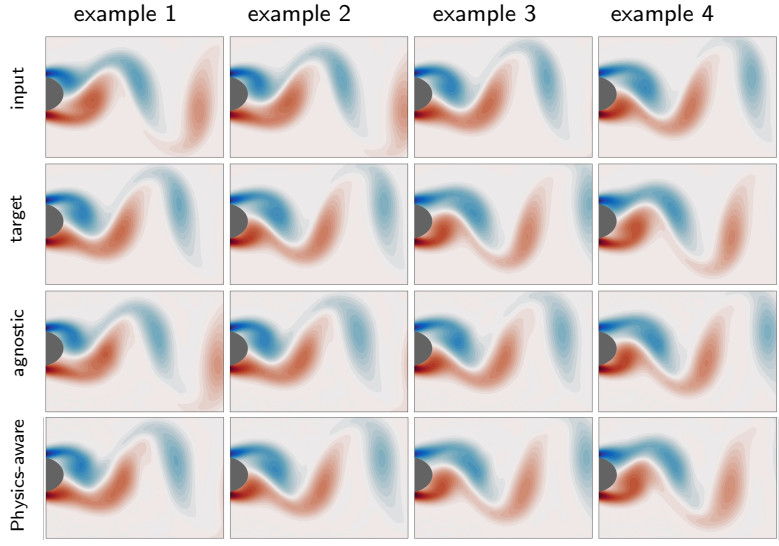


(b) Physics-agnostic (blue).

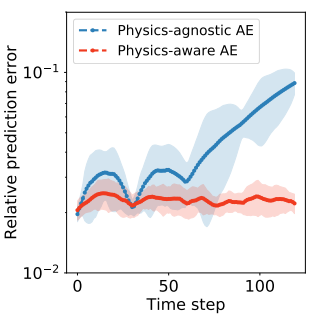


(c) Physics-aware (red).

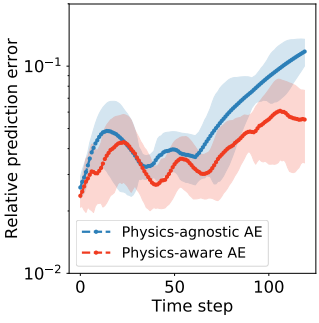
Visual results for flow past the cylinder (100 time steps)



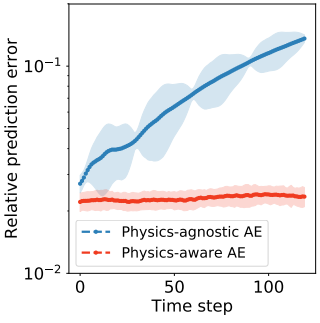
More results for the flow past the cylinder (with weight decay)



(a) With LR $1e{-}2$ and WD $1e{-}6$.



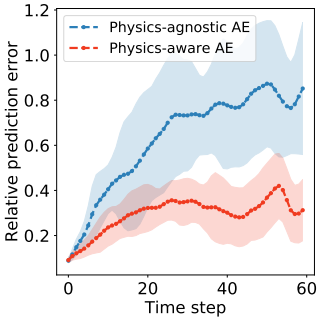
(b) With LR $1e{-}2$ and WD $1e{-}8$.



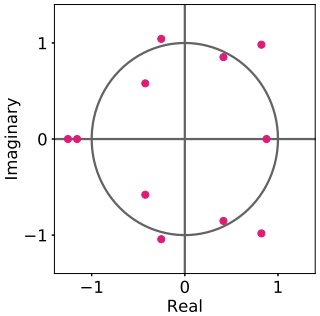
(c) With LR $5e{-}3$ and WD $1e{-}6$.

Results for the sea surface temperature data

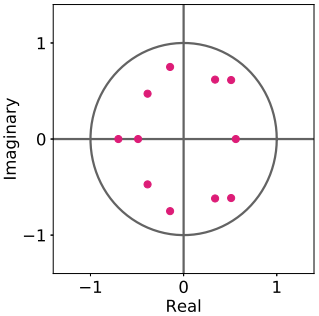
$$\min \frac{1}{T}\sum_{t=0}^{T} \|\mathbf{y}_{t+1} - \mathbf{\Phi}\circ\mathbf{\Omega}\circ\mathbf{\Psi}(\mathbf{y}_t)\|_2^2 + \lambda\|\mathbf{y}_t - \mathbf{\Phi}\circ\mathbf{\Psi}(\mathbf{y}_t)\|_2^2 + \kappa\sum_i \rho(p_i).$$



(a) With LR $1\mathrm{e}{-2}$.

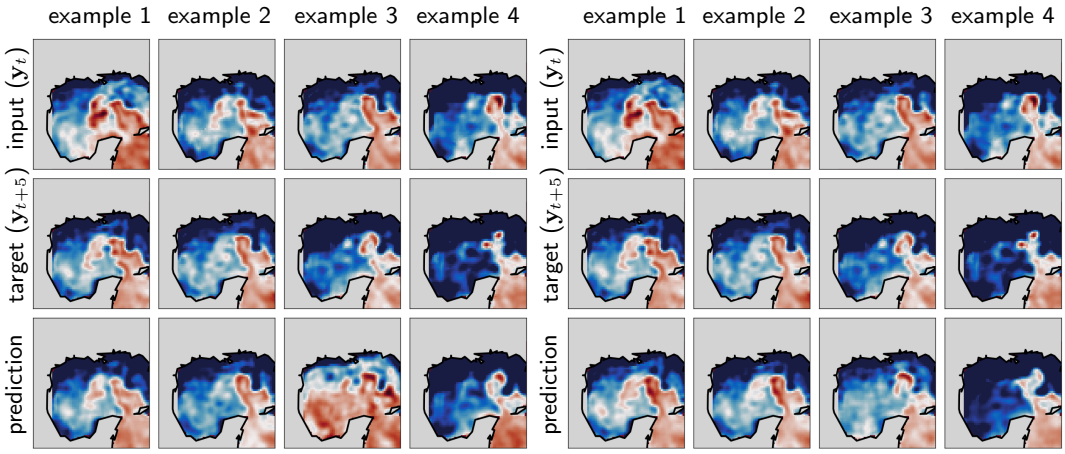


(b) Physics-agnostic (blue).



(c) Physics-aware (red).

Visual results for the sea surface temperature data

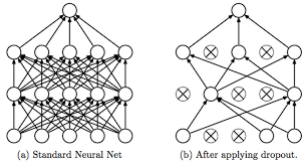


(a) Physics-agnostic model.

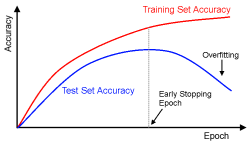
(b) Physics-aware model.

Have we just proposed a new regularizer?

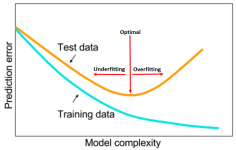
Every adjustable knob and switch — and there are many[3] — is regularization.



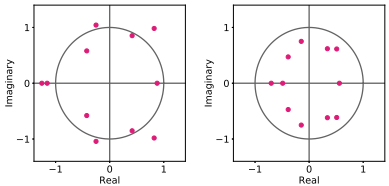
(a) Dropout.



(b) Early stopping.



(c) Bottleneck.

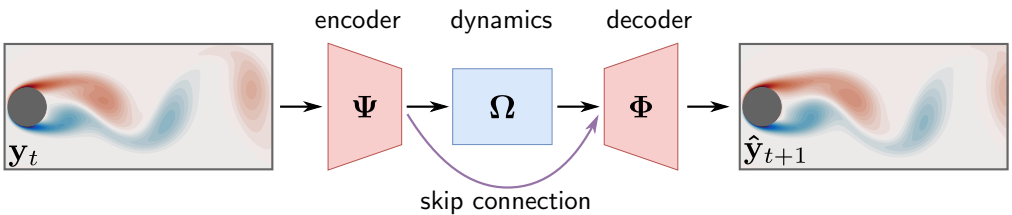


(d) Stability.

[3] https://arxiv.org/pdf/1710.10686.pd

Summary

- *Physics-informed* autoencoders can help to improve the generalization performance.
- Caveat of physics-informed learning are complicated loss functions: $\mathcal{L}_1 + \gamma\mathcal{L}_2 + \kappa\mathcal{L}_3 + ...$
- Next steps: non-linear dynamics, recurrent networks and parameterized layers.



For more details: https://arxiv.org/abs/1905.10866

PHYSICS-INFORMED AUTOENCODERS FOR LYAPUNOV-STABLE FLUID FLOW PREDICTION

N. BENJAMIN ERICHSON[♥], MICHAEL MUEHLEBACH[♦] AND MICHAEL W. MAHONEY[♥]

ABSTRACT. In addition to providing high-profile successes in computer vision and natural language processing, neural networks also provide an emerging set of techniques for scientific problems. Such data-driven models, however, typically ignore physical insights from the scientific system under consideration. Among other things, a "physics-informed" model formulation should encode some degree of *stability* or *robustness* or *well-conditioning* (in that a small change of the input will not lead to drastic changes in the output), characteristic of the underlying scientific problem. We investigate whether it is possible to include physics-informed prior knowledge for improving the model quality (*e.g.*, generalization performance, sensitivity to parameter tuning, or robustness in the presence of noisy data). To that extent, we focus on the stability of an equilibrium, one of the most basic properties a dynamic system can have, via the lens of Lyapunov analysis. For the prototypical problem of fluid flow prediction, we show that models preserving Lyapunov stability improve the generalization error and reduce the prediction uncertainty.

Conclusions

Transport phenomena and other scientific challenges:

- want to use machine learning to do discover new science
- provide forcing functions different than common machine learning applications

Bridging the gap is hard:

- *using* machine learning is not *doing* machine learning (any more than using a scientific data set is doing science)
- hiring mechanisms, funding mechanisms, etc., etc., etc., all conspire against it
- (but it's certainly not impossible)

Question: How to go forward?