The quest for provably efficient ML algorithms

Lorenzo Rosasco
MaLGa, Università degli Studi di Genova, MIT, IIT

Joint with:
Daniele Calandriello, Raffaello Camoriano, Luigi Carratino, Giacomo Meanti, Francesca Odone, Paolo Alfano, Vito Paolo Pastore (MaLGa), Alessandro Rudi (INRIA Paris), Bharath Sriperumbudur, Nick Sterge (UPenn), Alessandro Lazaric (Facebook), Michal Valko (DeepMind)

Data

⟺

Computations

Outline

Learning theory: statistics and computations

A case study: scalable kernel methods

Empirical results

Supervised learning

From data

$$(x_i, y_i)_{i=1}^n$$

to predictions

$$f(x_{\text{new}}) \approx y_{\text{new}}$$

Statistical supervised learning

Given $(x_i, y_i)_{i=1}^n \sim P(x, y)^n$ find $\widehat{f}$ with small

$$L(\widehat{f}) = \mathbb{E}[\ell(y, \widehat{f}(x))]$$

Empirical risk minimization

Minimize

$$\widehat{L}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(x_i))$$

over some space $\mathcal{H}$, e.g.

- $f(x) = \theta^\top x$
- $f(x) = \theta^\top \Phi(x)$ (kernels)
- $f(x) = w^\top \sigma(Wx)$, $\theta = (w, W)$, (neural nets)

+weights constraint:

Error analysis

How good is the ERM solution $\widehat{f}_R$?

$$\mathbb{E}[L(\widehat{f}_R)] - \min L$$

Bias and variance

Idealized ERM: f_R minimize L (**with** weights constraint).

$$
\begin{aligned}
&\mathbb{E}[L(\widehat{f}_R)] - \min L \\
&= \mathbb{E}[L(\widehat{f}_R)] - L(f_R) + L(f_R) - \min L
\end{aligned}
$$

A typical bound

$$
\begin{aligned}
&\mathbb{E}[L(\widehat{f}_R)] - \min L \\
&= \mathbb{E}[L(\widehat{f}_R)] - L(f_R) + L(f_R) - \min L \\
&\lesssim \frac{R}{n} + \frac{1}{R}
\end{aligned}
$$

$$
R = \sqrt{n} \quad \Longrightarrow \quad \mathbb{E}[L(\widehat{f}_R)] - \min L \lesssim \frac{1}{\sqrt{n}}
$$

A grain of salt

It's just an example of bound!

$$\mathbb{E}[L(\widehat{f}_R)] - \min L \lesssim \frac{R}{n} + \frac{1}{R}$$

$$R = \sqrt{n} \quad \Longrightarrow \quad \mathbb{E}[L(\widehat{f}_R)] - \min L \lesssim \frac{1}{\sqrt{n}}$$

- bounds depend on P! Much better/worse bounds possible.
- bounds depend on setting! e.g. classification, $n \ll d$ (interpolation)...

What about computations?

Optimization

ERM

$$\min_{\theta} \widehat{L}(f_{\theta})$$

by gradient descent

$$\widehat{\theta}_{t+1} = \widehat{\theta}_t + \gamma_t \nabla \widehat{L}(f_{\widehat{\theta}_t})$$

Another error decomposition

$$\mathbb{E}[L(f_{\widehat{\theta}_t})] - \min L$$
$$= \mathbb{E}[L(f_{\widehat{\theta}_t})] - \mathbb{E}[L(\widehat{f}_R)] + \mathbb{E}[L(\widehat{f}_R)] - \min L$$
$$= \mathbb{E}[L(f_{\widehat{\theta}_t})] - \mathbb{E}[L(\widehat{f}_R)] + \mathbb{E}[L(\widehat{f}_R)] - L(f_R) + L(f_R) - \min L$$

Outline

Learning theory: statistics and computations

A case study: scalable kernel methods

Empirical results

ERM with square loss

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} (y_i - f_\theta(x_i)))^2 + \lambda \|\theta\|^2$$

- $f_\theta(x) = \theta^\top x$
- $f_\theta(x) = \theta^\top \Phi(x), \quad \Phi(x)^\top \Phi(x') = k(x, x')$

Optimization: solving a linear system!

A typical bound and its cost

$$\mathbb{E}[L(\widehat{f}_\lambda)] - \min L \lesssim \frac{1}{\lambda n} + \lambda$$

$$\lambda = 1/\sqrt{n} \quad \Longrightarrow \quad \mathbb{E}[L(\widehat{f}_\lambda)] - \min L \lesssim \frac{1}{\sqrt{n}}$$

For large feature space/kernels the computational cost is

- time $O(n^3)$
- memory $O(n^2)$.

Can we do better? Statistically no but computationally...

An efficient approach: FALKON

- Conjugate gradient+
- subsampling/sketching (Nyström)+
- sketched preconditioning.

Statistical and computational error

$$\mathbb{E}[L(f_{\hat{\theta}_t})] - \min L$$
$$= \mathbb{E}[L(f_{\hat{\theta}_t})] - \mathbb{E}[L(\widehat{f}_\lambda)] + \mathbb{E}[L(\widehat{f}_\lambda)] - \min L$$
$$\lesssim e^{-t/\lambda} + \frac{1}{M^2} + \frac{1}{\lambda n} + \lambda$$

$$\lambda = 1/\sqrt{n}, \quad t = \log n, \quad M = \sqrt{n} \quad \Longrightarrow \quad \mathbb{E}[L(f_{\hat{\theta}_t})] - \min L \lesssim \frac{1}{\sqrt{n}}$$

Just theory?

Outline

Learning theory: statistics and computations

A case study: scalable kernel methods

Empirical results

Falkon 1.0: some experiments

	MillionSongs ($n \sim 10^6$)			YELP ($n \sim 10^6$)		TIMIT ($n \sim 10^6$)	
	MSE	Relative error	Time(s)	RMSE	Time(m)	c-err	Time(h)
FALKON	**80.30**	$\mathbf{4.51 \times 10^{-3}}$	**55**	**0.833**	**20**	32.3%	**1.5**
Prec. KRR	-	4.58×10^{-3}	$289^{\dagger}$	-	-	-	-
Hierarchical	-	4.56×10^{-3}	$293^{\star}$	-	-	-	-
D&C	80.35	-	737^{*}	-	-	-	-
Rand. Feat.	80.93	-	772^{*}	-	-	-	-
Nyström	80.38	-	876^{*}	-	-	-	-
ADMM R. F.	-	5.01×10^{-3}	$958^{\dagger}$	-	-	-	-
BCD R. F.	-	-	-	0.949	$42^{\ddagger}$	34.0%	$1.7^{\ddagger}$
BCD Nyström	-	-	-	0.861	$60^{\ddagger}$	33.7%	$1.7^{\ddagger}$
KRR	-	4.55×10^{-3}	-	0.854	$500^{\ddagger}$	33.5%	$8.3^{\ddagger}$
EigenPro	-	-	-	-	-	32.6%	$3.9^{\wr}$
Deep NN	-	-	-	-	-	32.4%	-
Sparse Kernels	-	-	-	-	-	**30.9%**	-
Ensemble	-	-	-	-	-	33.5%	-

Table: MillionSongs, YELP and TIMIT Datasets. Times obtained on: $\ddagger$ = cluster of 128 EC2 r3.2xlarge machines, $\dagger$ = cluster of 8 EC2 r3.8xlarge machines, $\wr$ = single machine with two Intel Xeon E5-2620, one Nvidia GTX Titan X GPU and 128GB of RAM, $\star$ = cluster with 512 GB of RAM and IBM POWER8 12-core processor, $*$ = unknown platform.

Falkon 1.0: some more experiments

	SUSY ($n \sim 10^6$)			HIGGS ($n \sim 10^7$)		IMAGENET ($n \sim 10^6$)	
	c-err	AUC	Time(m)	AUC	Time(h)	c-err	Time(h)
FALKON	**19.6%**	0.877	**4**	0.833	**3**	20.7%	**4**
EigenPro	19.8%	-	$6^{\wr}$	-	-	-	-
Hierarchical	20.1%	-	$40^{\dagger}$	-	-	-	-
Boosted Decision Tree	-	0.863	-	0.810	-	-	-
Neural Network	-	0.875	-	0.816	-	-	-
Deep Neural Network	-	**0.879**	$4680^{\ddagger}$	**0.885**	$78^{\ddagger}$	-	-
Inception-V4	-	-	-	-	-	**20.0%**	-

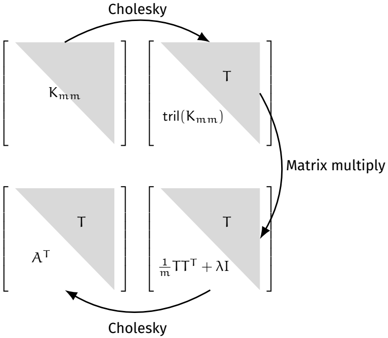
Table: Architectures: $\dagger$ = cluster with IBM POWER8 12-core cpu, 512 GB RAM, $\wr$ = single machine with two Intel Xeon E5-2620, one Nvidia GTX Titan X GPU, 128GB RAM, $\ddagger$ = single machine.

Implementing Falkon 2.0

[Meanti, Carratino, R., Rudi '20]

Function Falkon($X \in \mathbb{R}^{n \times d}$, $y \in \mathbb{R}^n$, λ, m, t):

- $X_m \leftarrow$ RamdomSubsample(X, m);
- $T, A \leftarrow$ Preconditioner(X_m, λ);
- **Function** LinOp(β):
  - $v \leftarrow A^{-1}\beta$;
  - $c \leftarrow k(X_m, X)k(X, X_m)T^{-1}v$;
  - **return** $A^{-\top}T^{-\top}c + \lambda n v$;
- $\text{rhs} \leftarrow A^{-\top}T^{-\top}k(X, X_m)y$;
- $\beta \leftarrow$ ConjugateGradient(LinOp, rhs, t);
- **return** $T^{-1}A^{-1}\beta$;



UniGe | MaLGa

Falkon2.0

- Least squares and logisitic loss [Marteau Ferey, Bach, Rudi '18,'19]
- Multi-GPU
- Mixed precision
- Optimized matrix-vector product
- Optimized kernel computation
- Out of core modules

Falkon2.0

Table: Relative performance improvement wrt Falkon 1.0

Experiment	Preconditioner		Iterations	
	Time	Improvement	Time	Improvement
Falkon1.0	2337 s	—	4565 s	—
Float32 precision	1306 s	1.8×	1496 s	3×
GPU preconditioner	179 s	7.3×	1344 s	1.1×
2 GPUs	118 s	1.5×	693 s	1.9×
KeOps	119 s	1×	232 s	3×
Overall improvement		19.7×		18.8×

Falkon2.0: thousands of points in seconds

	MNIST $n = 6 \cdot 10^4, d = 780$	CIFAR10 $n = 6 \cdot 10^4, d = 1024$	SVHN $n = 7 \cdot 10^4, d = 1024$
Falkon	10.9 s	13.7 s	17.2 s
ThunderSVM	19.6 s	82.9 s	166.4 s

Table: Comparing running times of FALKON and ThunderSVM. Parameters were tuned to have approximately the same accuracy.

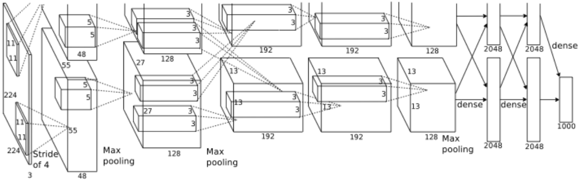
Falkon2.0: millions/billions (!) of points in minutes

	TAXI $n \approx 10^9$		HIGGS $n \approx 10^7$		YELP $n \approx 10^6$, $d \approx 10^7$		TIMIT $n \approx 10^6$	
	RMSE	time(h)	AUC	time(m)	RMSE	time(m)	c-err	time(m)
FALKON	**311.7**	**1**	0.8196	**7.4**	**0.810**	**16.8**	32.27%	**4.8**
LogFALKON	–	–	**0.8213**	37.8	–	–	–	–
EigenPro2	FAIL		FAIL		FAIL		**31.91%**	29
GPyTorch	322.5	10.8	0.8005	52.9	FAIL		–	–
GPflow	313.2	8.5	0.8042	24.3	FAIL		33.78%	44.5

	AIRLINE-CLS $n \approx 10^6$		AIRLINE $n \approx 10^6$		MSD $n \approx 10^5$		SUSY $n \approx 10^6$	
	c-err	time(m)	MSE	time(m)	relative error	time(m)	c-err	time(m)
FALKON	31.5%	**3.1**	**0.758**	**4.1**	4.4834×10^{-3}	**1**	19.67%	**0.4**
LogFALKON	**31.3%**	21.5	–	–	–	–	**19.58%**	1.4
EigenPro2	32.5%	27.2	0.785	24.5	4.4778×10^{-3}	6.3	20.08%	1.5
GPyTorch	33.0%	24.2	0.803	31	4.5344×10^{-3}	15.5	19.71%	16.5
GPflow	32.6%	10.5	0.790	28.7	4.4986×10^{-3}	8.8	19.65%	9.3

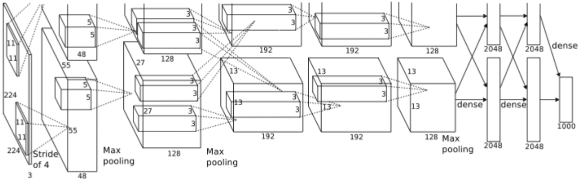
Deep neural networks (DNN)

$$f(x) = \langle w, \Phi(x) \rangle \,, \qquad \underbrace{x \mapsto \Phi_L \circ \cdots \circ \Phi_1(x)}_{\text{compositional representation}}$$



Convolutional and fully connected DNN

$$f(x) = \langle w, \Phi(x) \rangle \,, \qquad x \mapsto \underbrace{\Phi_L \circ \ldots \Phi_K}_{\text{Fully connected}} \circ \underbrace{\Phi_{K-1} \cdots \circ \Phi_1(x)}_{\text{Convolutional}}$$



- Convolutional layers, thousands parameters.
- Fully connected layers, million parameters.

$\rightarrow$ End to end learning.

(LeCun et al. ’98)

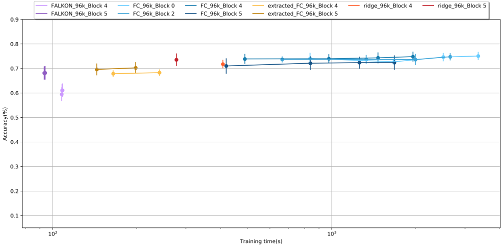
Trading DNN for kernel methods

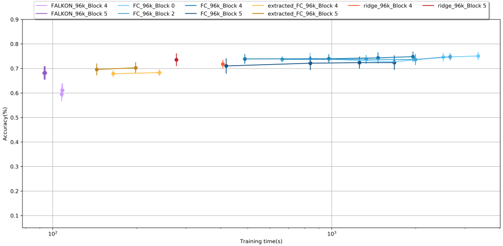
$$f(x) = \langle w, \Phi(x) \rangle , \quad x \mapsto \underbrace{\Phi_L}_{\text{Kernel representation}} \circ \underbrace{\Phi_{L-1} \cdots \circ \Phi_1(x)}_{\text{Convolutional}}$$

- not about data representation…
- …but scaling kernel methods to millions of points.

Don’t fine tune, use kernel methods

[Alfano, Pastore, R., Odone ’20]





UniGe | MaLGa

Wrapping up

- Efficiency: don't separate optimization and statistics.
- Beyond supervised learning: PCA, K-means...
- Beyond statistica learning: online learning and bandits.

TBD

- Develop theory for special settings: classification/Interpolation/
- Combine Nyström and multiscale approaches [Chen, Avron, Sindawhani '16].

UniGe | MaLGa

PhD/Postdoc positions available!

Relevant papers

Papers

Less is More: Nyström Computational Regularization
A. Rudi, R. Camoriano and L. Rosasco · NIPS15

FALKON: An Optimal Large Scale Kernel Method
A. Rudi, L. Carratino and L. Rosasco · NIPS17

Gaussian Process Optimization with Adaptive Sketching: Scalable and No Regret
D. Calandriello, L. Carratino, A. Lazaric, M. Valko and L. Rosasco · COLT19

Statistical and computational trade-offs in kernel k-means
D. Calandriello, L. Rosasco · NeurIPS18

Gain with no Pain: Efficient Kernel-PCA by Nyström Sampling
N. Sterge, B. Sriperumbur, L. Rosasco, A. Rudi · AISTATS20

Code

FALKON
G. Meanti,L. Carratino, L. Rosasco and A. Rudi · `http://lcsl.mit.edu`

BKB
D. Calandriello, L. Carratino, A. Lazaric, M. Valko and L. Rosasco · `http://lcsl.mit.edu`

More relevant papers

Papers

Learning with SGD and Random Features
L. Carratino, A. Rudi and L. Rosasco · NeurIPS18

On Fast Leverage Score Sampling and Optimal Learning
A. Rudi, D. Calandriello, L. Carratino and L. Rosasco · NeurIPS18

Exact sampling of determinantal point processes with sublinear time preprocessing
M. Dereziński , D. Calandriello, M. Valko · NeurIPS19

Near-linear Time GP Optimization with Adaptive Batching and Resparsification
D. Calandriello, L. Carratino, A. Lazaric, M. Valko and L. Rosasco · Preprint 2020

Code

BLESS: leverage score sampling
A. Rudi, D. Calandriello, L. Carratino and L. Rosasco · `http://lcsl.mit.edu`

DPP sampling

M. Dereziński , D. Calandriello, M. Valko · `http://lcsl.mit.edu`