

Towards efficient, robust and privacy-preserving machine learning

Joint work with G. Pavliotis, P. Parpas, N. Kantas, H. Haddadi, M. Malekzadeh, F. Mo, S. Demetriou, D. Gündüz, D. Kalise, A. Laignelet, D. Lengyel

Anastasia Borovykh

Imperial College London

November 21, 2022



Machine learning: intro


- We have some dataset $(x_i, y_i)_{i=1}^N$ where x_i 's are the input datapoints and y_i the 'to-predict' values.
- Our goal is to learn a mapping $x \rightarrow \hat{y} \approx y$.
- A three-choice menu:
 - The parametrisation of the model, i.e. the function $\hat{y}(x|\theta)$ with θ some parameters.
 - Some loss function $f(x, y|\theta)$ that measures how close \hat{y} is to the true value y .
 - The learning algorithm, i.e. how we find the θ that minimize the loss.



What we would want

- The standard goal is to find θ^* given by,

$$\theta^* = \arg \min_{\theta \in \Theta} \sum_{i=1}^N f(x^i, y^i | \theta).$$

- We want our algorithm to:
 - Converge to θ^* with **minimal computational resources** e.g. few iterations of an iterative algorithm.
 - Robustness: i) to noise, ii) in terms of out of sample performance.
 - Preserve the privacy of the individuals whose data we train over. Specific notion of privacy to be defined later.
 - Learn from small datasets.
- 

What we would want

- We will work with variants of gradient descent algorithms:

$$d\theta_t = \underbrace{\nabla f(\theta_t)}_{\text{gradient step}} dt + \underbrace{\Sigma dW_t}_{\text{Brownian noise}}$$


- Note: the noise can come from the stochastic gradient (SGD) or can be added into the algorithm.



The setting.

- Notation change: we think of x being the parameters of some model, f being some loss function of the model output $y(x)$.
- Consider some objective function f of $x \in \mathbb{R}^d$. We will work with f being one of:
 - Or μ_f -strongly convex: $(\nabla f(x) - \nabla f(x'))^T(x - x') \geq \mu_f \|x - x'\|_2^2$,
 - Or the log-Sobolev inequality holds; i.e. for $g : \mathbb{R}^n \rightarrow \mathbb{R}$ with $\mathbb{E}_\nu[g^2] \leq \infty$,

$$\mathbb{E}_\nu[g^2 \log g^2] - \mathbb{E}_\nu[g^2] \ln \mathbb{E}_\nu[g^2] \leq \frac{2}{\alpha} \mathbb{E}_\nu[\|\nabla g\|^2]. \quad (1)$$

- Or we take a quadratic: $f(x) = \frac{1}{2}x^T A x$ with $A \in \mathbb{R}^{m \times d}$. Typically we take A with some high or low condition number.
- 

i. Convergence speed



The setting.

- We consider two (related) goals:
 - the goal being to sample from the invariant measure,

$$\pi := \frac{1}{Z} e^{-f(x)} dx, \quad Z := \int e^{-f(x)} d, \quad (2)$$

- the goal being to find,

$$x^* = \arg \min_{x \in \mathcal{X}} f(x). \quad (3)$$

For this talk \mathcal{X} is the whole of \mathbb{R}^d .

- The relation between these goals: the mode of $x \sim \pi$ is x^* .



Background: Langevin dynamics

- Consider the Langevin dynamics for $x_t \in \mathbb{R}^d$

$$dx_t = -\nabla f(x_t)dt + \sigma dW_t. \quad (4)$$

- Denote with π_t the probability density function of x_t at time t . These dynamics satisfy the Fokker-Plank equation,

$$\partial_t \pi_t = \nabla \cdot (\nabla f \pi_t + \nabla \pi_t). \quad (5)$$

- Under appropriate assumptions, a stationary measure exists $\pi \sim e^{-f(x)}$.
- If f is μ_f -strongly convex Convergence to optimum x^* ,

$$\|x_t - x^*\|_2^2 \leq e^{-\mu_f t} \|x_0 - x^*\|_2^2 + \frac{d\sigma^2}{\mu_f}. \quad (6)$$

- If α -LSI holds Convergence to the invariant measure π ,

$$KL(\pi_t || \pi) \leq e^{-2\alpha t} KL(\pi_0 || \pi). \quad (7)$$



Background: speeding up Langevin dynamics

A subset of the ideas that have been proposed to speed up convergence to the optimum or to the invariant measure:

- Precondition the dynamics [Some Remarks on Preconditioning Molecular Dynamics; H. AlRachid, L. Mones, C. Ortner]
 - Let K be a preconditioner,

$$dx_t = -(K\nabla f)(x_t)dt + (\nabla \cdot K)(x_t)dt + \sigma\sqrt{K(x_t)}dW_t. \quad (8)$$

- The preconditioner can be taken to be the Hessian of the objective function. But: how to compute it? Approximations do not always lead to satisfactory results.



Background: speeding up Langevin dynamics

A subset of the ideas that have been proposed to speed up convergence to the optimum or to the invariant measure:


- Precondition with covariance matrix [[Affine Invariant Interacting Langevin Dynamics for Bayesian Inference](#) A. Garbuno-Inigo, N. Nüsken, S. Reich] [[Interacting Langevin Diffusions: Gradient Structure And Ensemble Kalman Sampler](#), A. Garbuno-Inigo, F. Hoffmann, W. Li and A. M. Stuart]:

$$dx_t^i = -C(\mathbf{x})\nabla f(x^i)dt + (\nabla \cdot C)(\mathbf{x})dt + \sqrt{2C(\mathbf{x})}dW_t, \quad (9)$$

where

$$C(\mathbf{x}) = \frac{1}{N} \sum_{k=1}^N (x^k - \bar{x}) \otimes (x^k - \bar{x}), \quad (10)$$

where $\bar{x} = \frac{1}{N} \sum_{i=1}^N x^i$.




Background: speeding up Langevin dynamics

A subset of the ideas that have been proposed to speed up convergence to the optimum or to the invariant measure:

- Optimal drift [Optimal non-reversible linear drift for the convergence to equilibrium of a diffusion; T. Lelièvre, F. Nier G.A. Pavliotis] [Optimal non-symmetric Fokker-Planck equation for the convergence to a given equilibrium, A. Arnold and B. Signarello] [Optimal linear drift for the speed of convergence of an hypoelliptic diffusion A. Guillin, P. Monmarché] [Variance reduction using nonreversible Langevin samplers AB Duncan, T Lelièvre, GA Pavliotis] :

$$dx_t = -\nabla f(x_t)dt + J\nabla f(x_t)dt + \sigma dW_t. \quad (11)$$

- The dynamics are non-reversible (generator is not self adjoint).
 - The convergence speed / asymptotic variance can be improved with the right choice of J .
- 

Background: speeding up Langevin dynamics

A subset of the ideas that have been proposed to speed up convergence to the optimum or to the invariant measure:

- N copies of optimal drift dynamics [[Accelerating the diffusion-based ensemble sampling by non-reversible dynamics](#), F. Futami, I. Sato, M. Sugiyama],

$$dx_t^i = -\nabla f(x_t^i)dt + J^i \nabla f(x_t^i)dt + \sigma^i dW_t^i. \quad (12)$$

- These dynamics do not seem faster than the single-particle dynamics...



Our prior work

Optimizing interacting Langevin dynamics using spectral gaps

Anastasia Borovykh¹ Nikolas Kantas² Panos Parpas¹ Greg Pavliotis²

This work considers $f_i = \nabla f$, quadratic interaction, σdW_t^i noise and shows how the spectrum of the interaction matrix can be used to optimize the convergence of the interacting dynamics - [we will discuss more about this.](#)




Our idea: the dynamics considered.

- A general form of the dynamics:

$$dx_t^i = \underbrace{f_i(x_t^i)}_{\text{local dynamics}} dt + \underbrace{\sum_{j=1}^N \phi(x_t^i, x_t^j)}_{\text{interaction with other agents}} dt + \underbrace{\Sigma^i dW_t^i}_{\text{Brownian noise}} \quad (13)$$

- Here we will consider a specific form:
 - Local dynamics: $-\nabla f(x_t^i)$ (a gradient over f),
 - Interaction is quadratic and interaction strength is governed by θ :
 $\theta A_{ij}(x_t^i - x_t^j)$,
 - Noise: σdW_t^i , $\sigma \in \mathbb{R}$.
- This leads to

$$dx_t^i = -\nabla f(x_t^i) dt + \theta \sum_{j=1}^N A_{ij}(x_t^j - x_t^i) dt + \sigma dB_t^i. \quad (14)$$


On to the results.

- Remember our dynamics,

$$dx_t^i = -\nabla f(x_t^i)dt + \sum_{j=1}^N A_{ij}(x_t^i - x_t^j)dt + \sigma^i dW_t^i. \quad (15)$$

- In vectorized form we write,

$$d\mathbf{x}_t = -\nabla f(\mathbf{x})dt + \mathcal{L}\mathbf{x}_t dt + \sigma dW_t, \quad (16)$$

where $\nabla f(\mathbf{x}) = [\nabla f(x^1), \dots, \nabla f(x^N)]^T$ and where $\mathcal{L} = L \otimes 1$ with L the graph Laplacian of A .



Convergence speed

- Note that the invariant measure of the particle system is given by,

$$\pi(d\mathbf{x}) = \frac{1}{Z_N} \exp\left(-\frac{2}{\sigma^2} \left(\frac{1}{2}\mathbf{x}^T \mathcal{L}\mathbf{x} + \sum_{i=1}^N f(x^i)\right)\right) d\mathbf{x} =: \frac{1}{Z_N} \exp(-W(\mathbf{x})).$$

- It is not the same as our original $\pi \sim e^{-f}$.
- In the case of optimization, we care about the mode of this distribution.

Lemma (Interaction preserves the minimum)

Let $x^* := \arg \min_x f(x)$, and let $W(\mathbf{x})$ be as above. Then

$\mathbf{x}^* = (x^{*T}, \dots, x^{*T})^T$ is a minimizer of $W(\mathbf{x})$.

Proof.

Follows from interaction term vanishing at consensus. □



Convergence speed

- Background: if the logarithmic Sobolev inequality holds, exponential convergence to the invariant measure in e.g. KL-divergence holds with speed governed by LSI constant (in our notation α).
- Spectral gap is the smallest eigenvalue of the generator other than zero. If generator has a spectral gap, then the dynamics can satisfy LSI or Poincare.
- What is the spectrum of our system?
- Consider the quadratic system with $f(\mathbf{x}) = \mathbf{x}^T \mathbf{A} \mathbf{x}$ where $\mathbf{A} = \text{diag}([A, \dots, A])$.



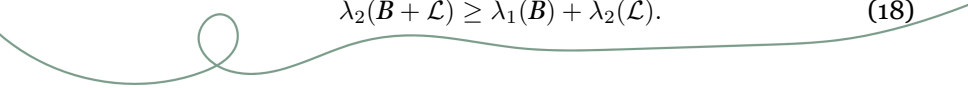
Convergence speed

Lemma (Spectrum for a quadratic potential)

Assume that $f(\mathbf{x}) = \mathbf{x}^T \mathbf{A} \mathbf{x}$, where $\mathbf{A} \in \mathbb{R}^{dN}$. Let the spectrum of \mathbf{A} be given by $\lambda_1^A \leq \dots \leq \lambda_k^A$. Denote with \mathcal{A} the generator of our interacting dynamics. Then, the spectrum of \mathcal{A} can be related to the spectrum of the drift as follows:

$$\sigma(\mathcal{A}) = \left\{ \sum_{j=1}^r -n_j \lambda_j^A, n_j \subset \mathbb{N} \right\}. \quad (17)$$

- From this result, we are thus interested in the eigenvalues of $\mathbf{A} + \mathcal{L}$, and specifically if \mathcal{L} can improve the spectral gap (thus improving convergence speed).
- When is $\min \sigma(\mathbf{A} + \mathcal{L}) \geq \min \sigma(\mathbf{A}) = \min \sigma(\mathcal{A})$?
- A result based on [Eigenvalues of sums of Hermitian matrices, W. Fulton]

$$\lambda_2(B + \mathcal{L}) \geq \lambda_1(B) + \lambda_2(\mathcal{L}). \quad (18)$$


Convergence speed: numerics

- Consider $f(x) = x^T A x$ or for all particles $f(\mathbf{x}) = \mathbf{x}^T \mathbf{A} \mathbf{x}$.
- Consider A with different condition numbers.
- Consider \mathcal{L} to be o (i.i.d.), a circle graph (each x_i is connected to x_{i+1} and a fully-connected graph.
- We make the distinction between graph Laplacian L being doubly stochastic (all rows and columns sum to one) and non-doubly stochastic.



Convergence speed: numerics

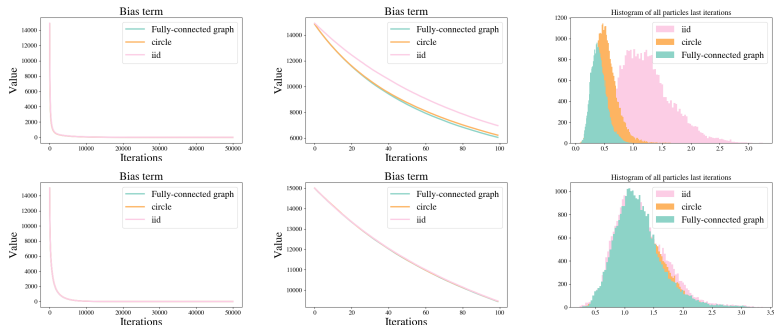


Figure 1: The quadratic setting with condition number 10; $N = 10$ and λ_2 is 1 and 0.049, respectively. (T) uses an interaction strength of 100, while (B) uses 1. Interactions help and a fully-connected graph is slightly better

Convergence speed: numerics

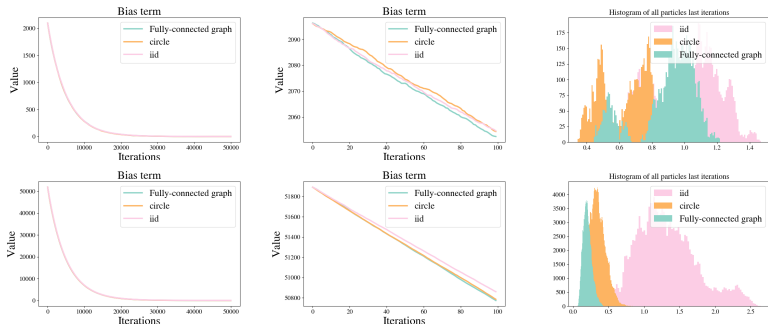


Figure 2: The quadratic setting with condition number 1 and interaction strength 10, (T) $N = 2$ with λ_2 is 1 and 1 and (B) $N = 50$ with λ_2 is 1 and 0.002, respectively.

Convergence speed: numerics

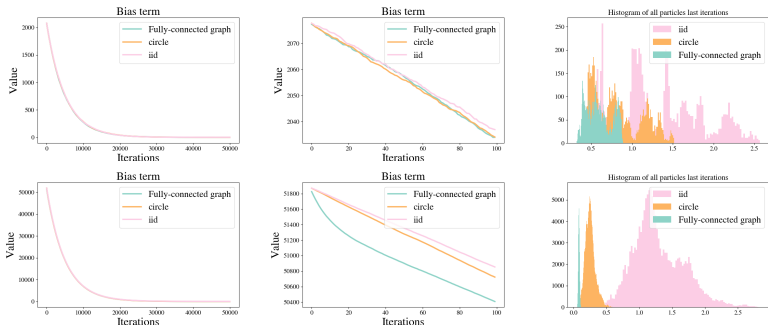


Figure 3: The quadratic setting with condition number 1, a non-doubly stochastic graph Laplacian and interaction strength 10, (T) $N = 2$ with λ_2 is 1 and 1 and (B) $N = 50$ with λ_2 is 50 and 0.004, respectively.

ii. Robustness to noise



Our prior work

To interact or not? The convergence properties of interacting stochastic mirror descent

Anastasia Borovykh¹ Nikolas Kantas² Panos Parpas¹ Grigorios A. Pavliotis²

On stochastic mirror descent with interacting particles: Convergence properties and variance reduction

A. Borovykh^{a,*}, N. Kantas^b, P. Parpas^a, G.A. Pavliotis^b

^a Department of Computing, Imperial College London, United Kingdom

^b Department of Mathematics, Imperial College London, United Kingdom

These works consider $f_i = \nabla f$, quadratic interaction, σdW_t^i noise and shows **variance reduction** effects of interactions for the general mirror descent setup - **we will discuss more about this.**



Variance reduction

- We have already seen it a bit in the last numerics, but it seems like the spread of the parameter values is decreased with interactions.
- Can we make this precise?

Background:

- noise in the algorithm can come from e.g. gradient subsampling or corruptions in the graph communication structure.
- To reduce the effect of noise, in practice one can decrease the learning rate over time.
- What if we do not want to decrease the learning rate? **Interactions can help.**



Variance reduction

- Let f be a μ_f -strongly convex function. Define $\tilde{x}_t^i = x_t^i - \frac{1}{N} \sum_{i=1}^N x_t^i$ (a fluctuation term). Then we have,


$$\begin{aligned} \frac{1}{T} \int_0^T \mathbb{E}[(f(x_t^i) - f(x^*))] dt &\leq \frac{1}{2T} \|\mathbf{x}_0 - \mathbf{x}_T\|_2^2 + \frac{\sigma^2}{2N} \\ &+ \int_0^T \frac{L}{\mu T} \mathbb{E} [\|\tilde{x}_t^i\|_*] dt + \int_0^T \frac{2L}{\mu NT} \sum_{i=1}^N \mathbb{E} [\|\tilde{x}_t^i\|_*] dt. \end{aligned}$$



Variance reduction

- How do we derive this? Define $\bar{x}_t^N = \frac{1}{N} \sum_{i=1}^N x_t^i$. Observe that

$$\begin{aligned} \int_0^T (f(x_t^i) - f(x^*)) dt &= \int_0^T (f(x_t^N) - f(x^*)) dt + \int_0^T (f(x_t^i) - f(x_t^N)) dt \\ &\leq \int_0^T (f(x_t^N) - f(x^*)) dt + \int_0^T L \|x_t^i - x_t^N\| dt \end{aligned}$$

- Bound the first term by a Lyapunov argument and note that the second term is the fluctuation.
- 

Variance reduction


- $\frac{1}{2T} \|\mathbf{x}_0 - \mathbf{x}_T\|^2$: the standard optimization error giving linear in time convergence **mirror map also appears here, so its choice can influence the convergence speed**,
- $\frac{\sigma^2}{2N}$: **variance is decreased by a factor of N** ;
- $\int_0^T \frac{L}{\mu T} \mathbb{E} [\|\tilde{\mathbf{x}}_t\|_*]$ and $\int_0^T \frac{2L}{\mu NT} \sum_{i=1}^N \mathbb{E} [\|\tilde{\mathbf{x}}_t^i\|_*] dt$ measure deviation from the particle average.
- **If**, fluctuation term $\tilde{\mathbf{x}}_t^i$ is bounded and non-increasing with N , variance is reduced!



Variance reduction: Bounding the fluctuation term

- Luckily, fluctuation is bounded under certain assumptions.
- We have, for a μ_f -strongly convex function f ,


$$\mathbb{E} \left[\frac{1}{N} \sum_{i=1}^N \|\tilde{x}_t^i\|_*^2 \right] \leq e^{-\theta(\mu_f + \underline{\lambda})t} \mathbf{C} + \frac{dK}{\theta(\mu_f + \underline{\lambda})} \sigma^2 \frac{N-1}{N}.$$

- For a sufficiently large $\theta(\kappa + \underline{\lambda})$ the interaction is controlled.
 - Strong convexity μ_f plays a role.
 - Interaction strength θ plays a role.
 - Connectivity of the particles plays a role through $\underline{\lambda}$.
- 

Numerics for variance reduction

- Consider again our quadratic function.
- Consider,

$$f(x) = \frac{1}{m} \sum_{i=1}^m f_i(x) := \frac{1}{m} \sum_{i=1}^m \|W_{i,\cdot}x - b_i\|_2^2. \quad (19)$$

- In every iteration, the gradient is computed over a subset of the data, $f_S(x) = \frac{1}{|S|} \sum_{i \in S} f_i(x)$, where $|S|$ refers to the size of the batch.
 - The noise is thus implicit in the gradient ∇f_S .
- 

Numerics for variance reduction

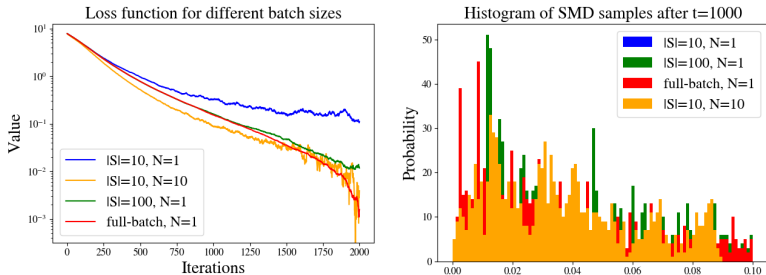


Figure 4: The loss function (L) and the histogram (R) for ISMD with different batch sizes with $\kappa(W) = 200$. Using interacting particles allows to use a smaller batch size while still attaining convergence. The presented results are averaged over 10 runs.

iii. Robustness in terms of generalisation.



Nonconvex case

- So far we have seen some benefits of interactions in the convex case. What about the **nonconvex** setting?

Background:

- In deep learning a common goal is to achieve good out-of-sample performance, i.e. to have our model configuration generalize well to unseen data.
- Remember: a neural network loss surface is very nonconvex (saddle points, local minima, sharp minima, flat minima).



More on robustness.

- x^* minimises the *train* loss i.e. on the train dataset the performance will be good.
- But what about the test loss i.e. the performance on unseen data?
- A classical example [[Explaining and harnessing adversarial examples, I.J. Goodfellow et al.](#)]:

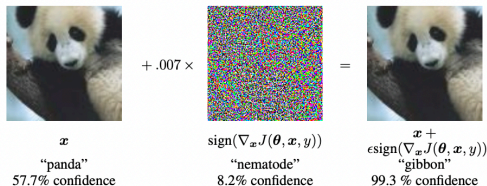


Figure 1: A demonstration of fast adversarial example generation applied to GoogLeNet (Szegedy et al., 2014a) on ImageNet. By adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, we can change GoogLeNet’s classification of the image. Here our ϵ of .007 corresponds to the magnitude of the smallest bit of an 8 bit image encoding after GoogLeNet’s conversion to real numbers.

- The model does not *generalize* to unseen data.

Nonconvex case

- Generalization can be related to ‘robustness’ to changes in the dataset.
- The intuition goes that [Flat minima S Hochreiter, J Schmidhuber] flat minima generalize better since these can handle more perturbations in data without affecting performance.

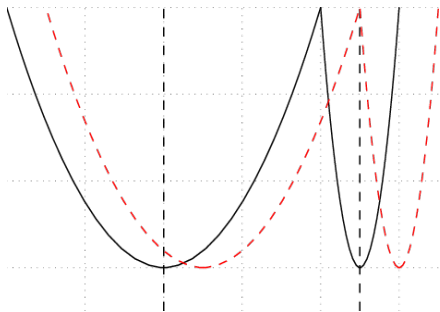


Figure 5: Picture from [On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima, N.S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, P. Tak, P. Tang]

Nonconvex case: toy example

- Consider the Müller-Brown (MB) objective function,

$$f(x, y) = \sum_{i=1}^4 A_i \exp(a_i(x - \bar{x}_i)^2 + b_i(x - \bar{x}_i)(y - \bar{y}_i) + c_i(y - \bar{y}_i)^2).$$

- It has several saddle points and local minima.
- Deep learning: flat minima are good. Can interaction help?



Nonconvex case: toy example

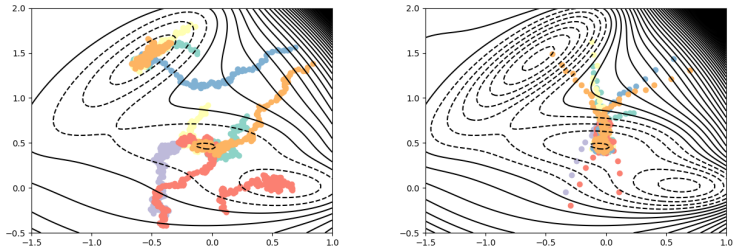


Figure 6: Starting from a saddle point with 10 particles using interacting SGD with a low and high interaction strength in a no-noise setting. Interactions seem to help converge to flatter minima!

Nonconvex case: neural nets

- But the benefit is not that clear in neural networks...
- We distinguish two different configurations:
 - Deep neural networks: are considered to be more nonconvex
 - Wide neural networks: are considered to be more convex
- Note: EASGD is elastic averaging SGD [Deep learning with Elastic Averaging SGD, S. Zhang, A. E. Choromanska, Y. LeCun]
- FC is fully connected graph; 2B is a 2-Barbell graph

10 nodes/layer			100 nodes/layer		
method	train	test	method	train	test
i.i.d.	0.153	0.753	i.i.d.	0.044	0.748
ISGD FC	0.113	0.686	ISGD FC	0.039	0.710
ISGD 2B	0.123	0.684	ISGD 2B	0.039	0.704
EASGD	0.273	0.618	EASGD	0.117	0.623

- To be continued...
- 

iv. Privacy risks



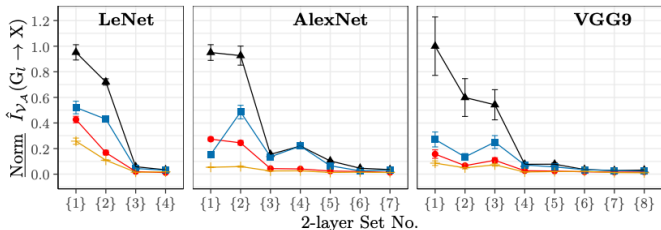
On the privacy

- Remember: we train our models over potentially sensitive datasets.
- **Issue:** There exist ‘attacks’ that can extract sensitive information from parameters or gradients.
- **How?** some attacker is able to get his hands on some parameters or gradients from the model and she performs some kind of computation to extract private information from this.
- The literature on such attacks is huge [[Do Gradient Inversion Attacks Make Federated Learning Unsafe?](#), Ali Hatamizadeh et al.], [[Inverting gradients—how easy is it to break privacy in federated learning](#), J. Geiping et al.], [[Quantifying and Localizing Private Information Leakage from Neural Network Gradients](#), Fan Mo, et al.] and very many more.



Attacks.

- How does such an attack work? Suppose the attacker obtains gradients.
 1. Randomly initialize dummy data
 2. Feed into the neural network model to get dummy gradients
 3. Optimize dummy data such that dummy gradient is close to real gradient (obtained from other devices).
- From: [\[Quantifying and Localizing Private Information Leakage from Neural Network Gradients, Fan Mo, et al.\]](#):



Attacks.

- How does such an attack work? Suppose we share gradients.
 1. Randomly initialize dummy data
 2. Feed into the neural network model to get dummy gradients
 3. Optimize dummy data such that dummy gradient is close to real gradient (obtained from other devices).
- From: [\[Inverting gradients—how easy is it to break privacy in federated learning, J. Geiping et al.\]](#)

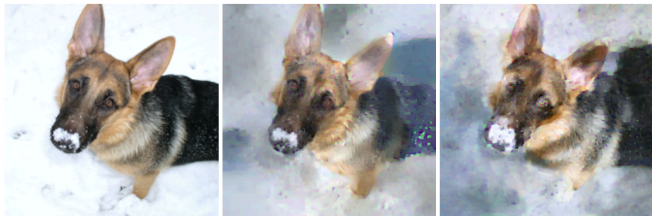


Figure 1: Reconstruction of an input image x from the gradient $\nabla_{\theta} \mathcal{L}_{\theta}(x, y)$. Left: Image from the validation dataset. Middle: Reconstruction from a trained ResNet-18 trained on ImageNet. Right: Reconstruction from a trained ResNet-152. In both cases, the intended privacy of the image is broken. Also note that previous attacks cannot recover ImageNet-sized data [38].

The algorithm here

- Stochastic gradient descent can be related to the Langevin dynamics (the continuous-time limit):

$$d\mathbf{x}_t = -\nabla f(\mathbf{x}_t)dt + \Sigma(\mathbf{x}_t)dW_t. \quad (20)$$

- Of interest is to understand the privacy risk of these dynamics.



How do we measure privacy?

- So how can we measure if a model is privacy-proof or not?
- One way of measuring the extent to which a learning algorithm preserves privacy is **differential privacy**.
- Informally:
 - Consider a dataset over which we compute a loss function $f(x)$
 - Consider a second dataset which is **almost identical** to the first **but one datapoint differs** $f'(x)$
 - Differential privacy means: if we optimize the parameters over the two datasets, the parameter distribution should be indistinguishable.



A formal definition

- The classic definition of differential privacy is based on a worst-case privacy guarantee:

Definition ((ϵ, δ)-differential privacy Dwork (2014))

Consider two adjacent datasets $D, D' \in \mathcal{D}^n$. The randomized estimator $\nu : \mathcal{D}^n \rightarrow \mathcal{P}(\mathcal{X})$ is said to be (ϵ, δ)-differentially private if for all measurable sets $A \in \mathcal{X}$ and for all adjacent datasets D, D' it holds,

$$\nu(A) \leq \exp(\epsilon)\nu'(A) + \delta. \quad (21)$$

with ν, ν' are computed D, D' , respectively.



An alternative formulation

- Consider a measure $\nu = p(\mathbf{x})d\mathbf{x}$ computed over dataset D , and similarly $p'(\mathbf{x})$ computed over dataset D' .
- An alternative formulation: (ϵ, δ) -differential privacy is satisfied if,

$$\nu \left(\ln \frac{p(\mathbf{x})}{p'(\mathbf{x})} \geq \epsilon \right) \leq \delta, \quad (22)$$

for *every* adjacent pair of datasets D, D' .



How to compute this? One option.

- Of interest is thus the quantity $\ln \frac{p(\mathbf{x})}{p'(\mathbf{x})}$.
- Let the measure $\nu \in \mathcal{P}(\mathbb{R}^d)$ satisfy a log Sobolev inequality with constant α . Then we have the following concentration inequality:


$$\nu(F \geq r + \mathbb{E}_\nu[F]) \leq \exp\left(-\frac{\alpha r^2}{2 \|F\|_{Lip}^2}\right) \quad (23)$$

for any Lipschitz F where $\|F\|_{Lip}$ denotes the Lipschitz constant of F .

- For $F = \ln \frac{p(\mathbf{x})}{p'(\mathbf{x})}$ this directly implies (for the right choice of ϵ and δ):

$$\nu\left(\ln \frac{p(\mathbf{x})}{p'(\mathbf{x})} \geq \epsilon\right) \leq \delta. \quad (24)$$

- It thus requires a bound on the KL-divergence:

$$\mathbb{E}_\nu \left[\ln \frac{p(\mathbf{x})}{p'(\mathbf{x})} \right] = KL(p||p').$$


Prior work: Gibbs distribution privacy risk

- Typically, the assumption is made that the noise covariance is isotropic $\Sigma(\mathbf{x}_t) = \sigma I_d$.
- Prior work [Differential privacy without sensitivity, Minami et al.] considers two Gibbs distributions,

$$p(\mathbf{x}) = \frac{1}{Z} e^{-\frac{1}{2\sigma^2} f(\mathbf{x})}, \quad (25)$$

$$p'(\mathbf{x}) = \frac{1}{Z'} e^{-\frac{1}{2\sigma^2} f'(\mathbf{x})}, \quad (26)$$

- Suppose that f is L -Lipschitz in the data so that $|\nabla f(\mathbf{x})| \leq L$. Assume also that $\nu := p_t(\mathbf{x}) d\mathbf{x}$ satisfies the log-Sobolev inequality with constant α . Then,

$$KL(p|p') \leq \frac{2L^2}{\alpha\sigma^4}. \quad (27)$$


The (ϵ, δ) -differential privacy guarantee is then given for an r such that,

$$r + \frac{2L^2}{\alpha\sigma^4} \leq \epsilon, \quad \exp\left(-\frac{r^2\sigma^4}{2\alpha L^2}\right) \leq \delta, \quad (28)$$

Prior work

- Using a derivation similar to the work [Differential privacy dynamics of langevin diffusion and noisy gradient descent, Shokri et al.] one can show that the privacy risk is bounded as follows,

$$\frac{d}{dt}KL(p_t||p'_t) \leq \sigma^{-1}||(\nabla f(\mathbf{x}) - \nabla f(\mathbf{x}'))||_2^2.$$

- This only holds for *isotropic* noise.
- 

Langevin dynamics with anisotropic noise

- Our work:

$$\frac{d}{dt}KL(p_t||p'_t) \leq \left\| \Sigma^{-1/2}(\nabla f(\mathbf{x}) - \nabla f(\mathbf{x}')) \right\|_2^2. \quad (29)$$

- Why is this of interest? The impact of anisotropic noise on the privacy risks is made explicit.
- It can further show the impact of the objective function structures.



v. Learn from small datasets.



The challenges that arise: dataset size

- Deep neural networks are able to achieve excellent performance when trained on big datasets, even if the input data size is very large (e.g. 28×28 pixels for the MNIST dataset).
- System constraints such as on-device computing or simply the fact that large datasets are not always available has led to a renewed interest in learning over small datasets.
- [How to choose the datapoints optimally?](#)



Our prior work

Efficient regression with deep neural networks: how many datapoints do we need?

Daniel Lengyel


Department of Computing
Imperial College London
London, United Kingdom
d.lengyel@imperial.ac.uk

Anastasia Borovykh

Department of Mathematics
Imperial College London
London, United Kingdom
a.borovykh@imperial.ac.uk



The method

- We consider the goal of learning a function representation \hat{f} from datapoints $(\mathbf{x}_i, f(\mathbf{x}_i))_{i=1}^N$.
 - In one dimension: we use the curvature to determine **good** points.
 - In two dimensions: we exploit literature on **mesh optimization** from computer graphics:
 - This method defines a triangulation: a mesh that ‘covers’ a surface using simplices.
 - It assumes an initial triangulation is given which consists of an arbitrary number of vertices.
 - The algorithm simplifies this triangulation to obtain a triangulation with a given number of vertices.
 - It does so by iteratively contracting sets of vertices if the cost associated to their contraction (i.e. how much accuracy is lost by removing one of the vertices in the approximated surface) is not too high.
- 

Some results

- We consider a modified version of the Rastrigin function

$$f(\mathbf{x}) = -25e^{-\frac{\|\mathbf{x}\|^2}{1.5}} - 10e^{\frac{\|\mathbf{x}\|^2}{0.5}} + 10d + \sum_{i=1}^d -10 \cos(2\pi x_i). \quad (30)$$

- We present the results for $N = 12$, $d = 1$ and a neural network architecture of width=23 and nr of layers=2 trained with Adam.

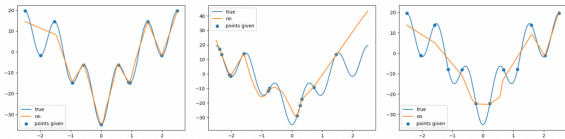


Figure 1: Rastrigin function for $d = 1$. Left to right: curvature-based sampling (MSE is 16.85), uniform random sampling (MSE is 137.88), uniform grid (MSE is 44.53)

Some results

- We consider a modified version of the Rastrigin function

$$f(\mathbf{x}) = -25e^{-\frac{\|\mathbf{x}\|^2}{1.5}} - 10e^{\frac{\|\mathbf{x}\|^2}{0.5}} + 10d + \sum_{i=1}^d -10 \cos(2\pi x_i). \quad (31)$$

- Here we show the results for different number of datapoints and model architectures:

Table 1: Rastrigin function. (L) Effect of model architecture: (train, test) performance averaged over 5 runs. (R) Effect of number of datapoints for arch. (32,2): test performance.

(N_W, N_L)	Uniform Grid	Curvature	Nr of datapoints	Uniform Grid	Curvature
(8,2)	(1.97,48.2)	(1.87,41.6)	10	52.1	59.0
(32,2)	(1.26,38.6)	(0.88,24.8)	12	36.0	48.9
(8,4)	(1.59,39.2)	(0.65,32.6)	20	30.8	18.1



THE END!

- Directions of interest in the theory:
 - Robustness of machine learning algorithms,
 - Better understanding deep neural networks and the optimisation algorithms' performance there,
 - Defining privacy-preserving algorithms to safeguard sensitive data over which models are trained.
- I didn't mention the applications of interest but some current lines of work:
 - Smart cities: optimize logistics or mobility flows in cities.
 - Deep brain stimulation: analyse data from the brain to understand how to stimulate the brain to help patients with Parkinson.
 - Robotics and optimal control: we apply neural networks to learn the 'optimal control'.

