Deep Relaxation:
PDEs for optimizing Deep Neural Networks

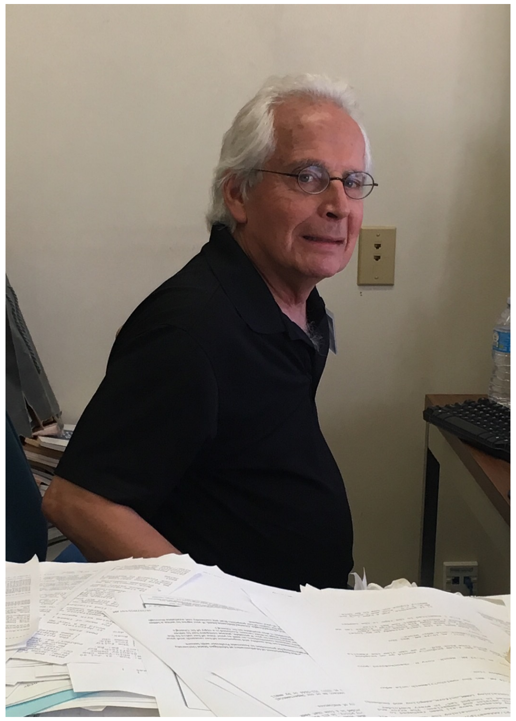
IPAM Mean Field Games

August 30, 2017

Adam Oberman (McGill)

Thanks to the Simons Foundation (grant 395980) and the hospitality of the UCLA math department

Coauthors



Pratik Chaudhari,
UCLA Comp Sci.

Stanley Osher,
UCLA Math

Stefano Soatto
UCLA Comp Sci.

Guillaume Carlier,
CEREMADE, U.
Parix IX Dauphine

Thanks to the Simons Foundation (grant 395980) and the hospitality of the UCLA math department

Introduction

Deep Learning

Machine Learning vs. Deep Learning

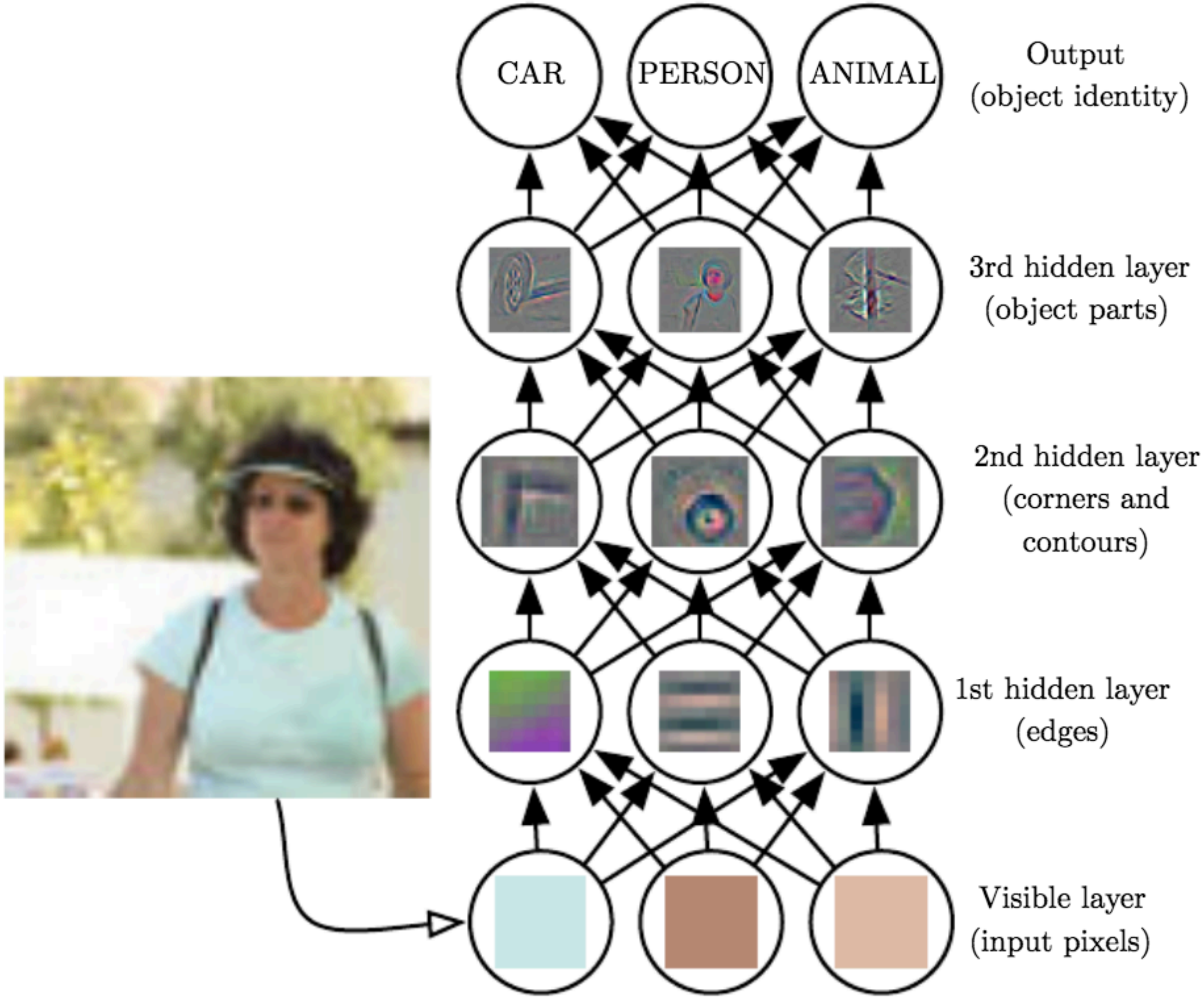
- Typical Machine Learning models
  - better understood mathematically,
  - don’t scale as well to very large problems.

$$\min J(x_1, \ldots, x_K) = \sum_{i=1}^{K} \sum_{j=1}^{N} \min_i \|x_i - y_j\|^2$$

K-means Cluster Energy

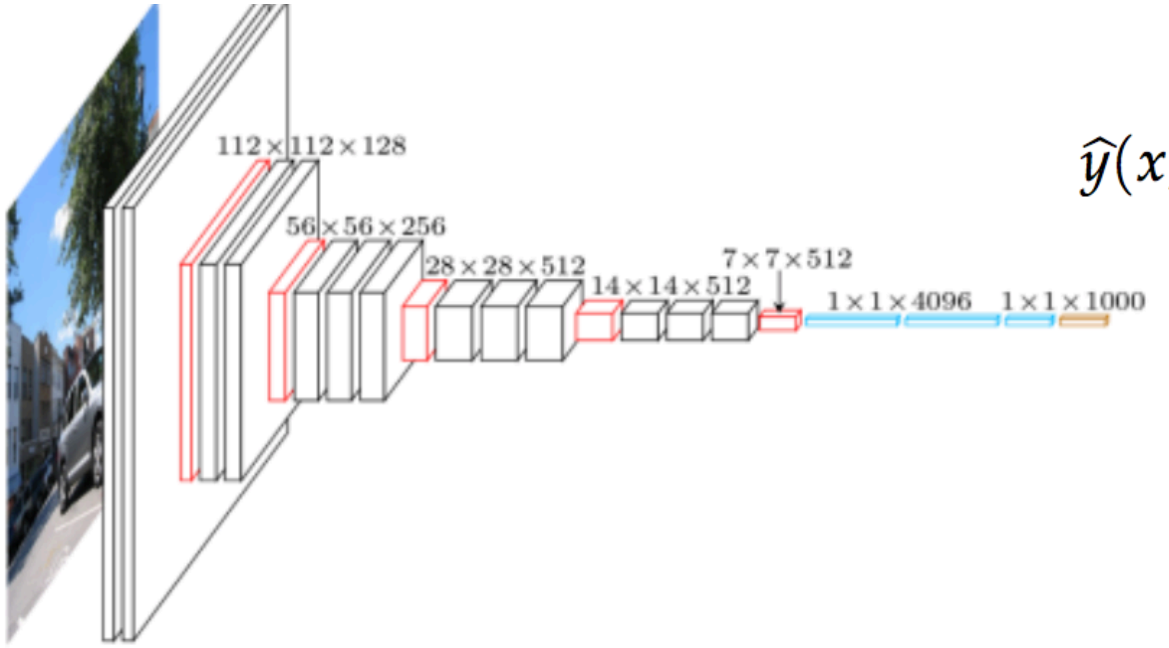
- Deep Learning
  - Very effective for large scale problems (e.g. identifying images).
  - Major open problem: understand *generalization* (why training on a large data set works so well on real problems).

Deep Network



Nested hierarchy of concepts, each defined in relation to simpler concepts [Goodfellow *Deep Learning*]

Deep Learning Background



$$\widehat{y}(x;\, w) = \sigma(w^p\, \sigma(w^{p-1}\, (\ldots \sigma(w^1\, x)) \ldots))$$

- Deep Learning:
  - recognize face in picture,
  - translate voice recording into text.
- Training: optimizing the parameters of the model, the weights, to best fit the data.

FIGURE 2. MNIST

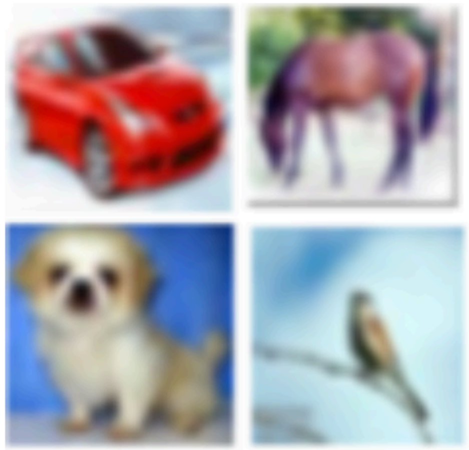


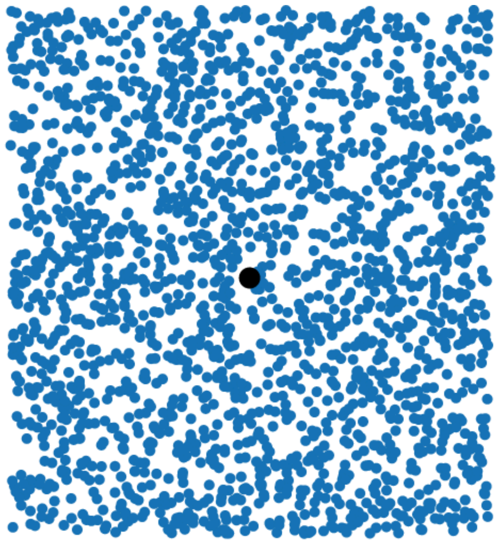
FIGURE 3. CIFAR-10

Derivation of Stochastic Gradient from mini-batch

Motivating Example: k-means clustering, (take k = 1)

$$f(x) = \frac{1}{2N}\sum_{i=1}^{N}(x-y_i)^2$$

$$\nabla f(x) = x - \bar{y}, \qquad \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

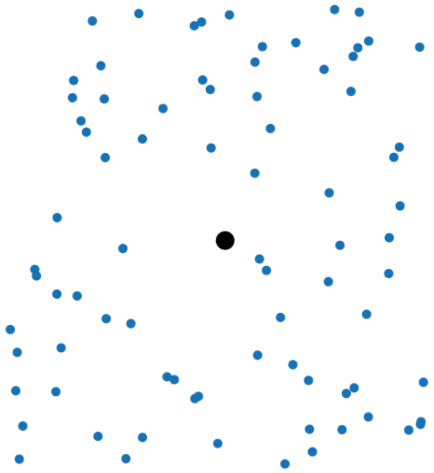


Cost is N

mini-batch: randomly choose a smaller set I of points from the data.

$$f_{mb}(x) = \frac{1}{2|I|}\sum_{i\in I}(x-y_i)^2$$

$$\nabla f_{mb}(x) = x - \bar{y}_{mb}, \qquad \bar{y}_{mb} = \frac{1}{|I|}\sum_{i\in I} y_i$$



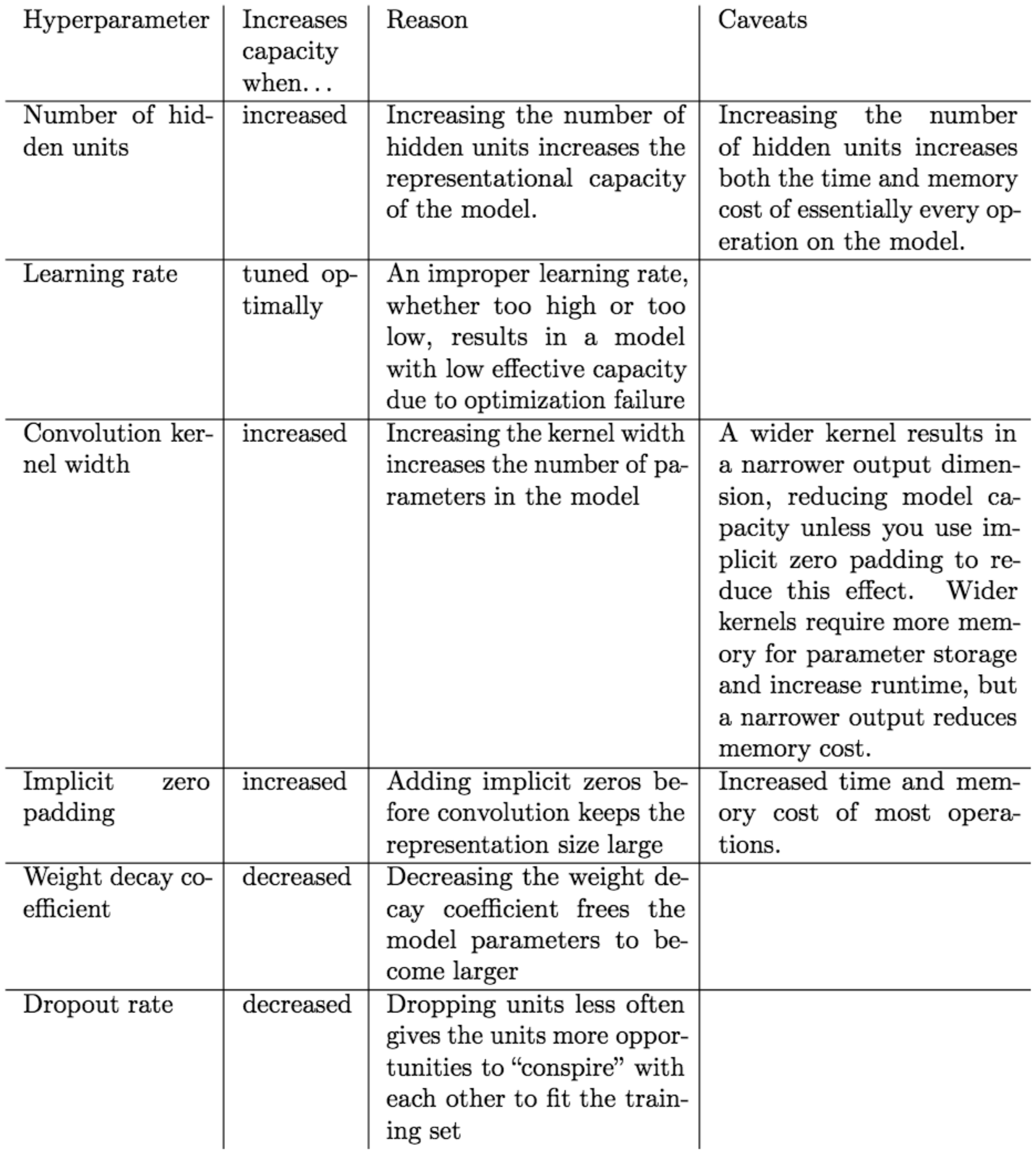
Cost is |I|

If the points are IID, then by the Central Limit Theorem

$$\nabla f(x) - \nabla f_{mb}(x) \sim N\left(0, \frac{1}{|I|}\right)$$

Training a DNN

- Tuning hyperparameters is labor intensive.
- Training is performed, simply and effectively, by *Stochastic Gradient Descent* (SDG).
- SGD is so effective, some popular programming languages [*TensorFlow*] do not allow modification.



Hyperparameter	Increases capacity when...	Reason	Caveats
Number of hidden units	increased	Increasing the number of hidden units increases the representational capacity of the model.	Increasing the number of hidden units increases both the time and memory cost of essentially every operation on the model.
Learning rate	tuned optimally	An improper learning rate, whether too high or too low, results in a model with low effective capacity due to optimization failure	
Convolution kernel width	increased	Increasing the kernel width increases the number of parameters in the model	A wider kernel results in a narrower output dimension, reducing model capacity unless you use implicit zero padding to reduce this effect. Wider kernels require more memory for parameter storage and increase runtime, but a narrower output reduces memory cost.
Implicit zero padding	increased	Adding implicit zeros before convolution keeps the representation size large	Increased time and memory cost of most operations.
Weight decay coefficient	decreased	Decreasing the weight decay coefficient frees the model parameters to become larger	
Dropout rate	decreased	Dropping units less often gives the units more opportunities to "conspire" with each other to fit the training set	

Table 11.1: The effect of various hyperparameters on model capacity.

Tuning hyperparameters

Local Entropy:
from Spin Glasses
to Deep Networks
to Hamilton-Jacobi PDEs

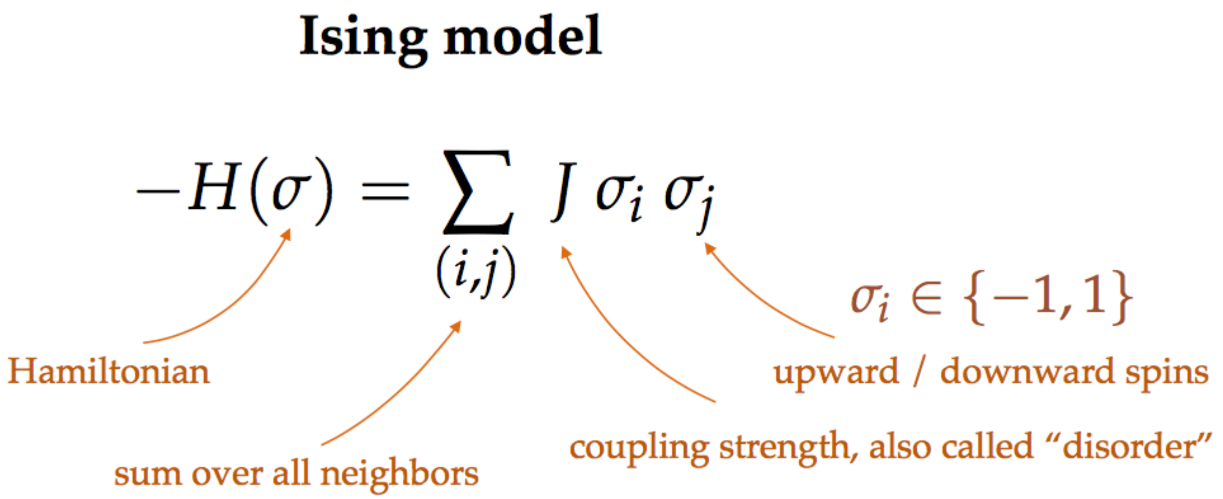
Motivation: Local Entropy in Spin Glasses
seeking to improve generalization

Local Entropy
(statistical physics)

$$E_\gamma(\sigma) = -\log \sum_{\sigma'} \exp(-\beta E(\sigma') - d(\sigma, \sigma')^2)$$

$$\sigma \in \{-1, +1\}^N \text{ spin}$$

[*Local Entropy … in Constraint Satisfaction Problems*, Baldassi 2016]
[*Unreasonable Effectiveness of Deep Learning*, B.,…,Zecchina PNAS 2016]



Ising model

$$-H(\sigma) = \sum_{(i,j)} J\, \sigma_i\, \sigma_j$$

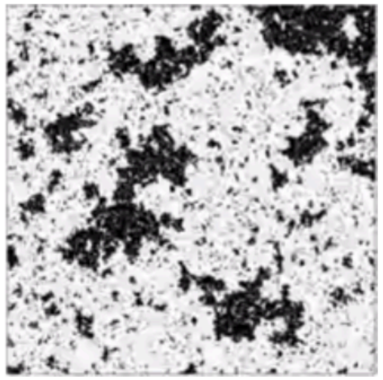
$$\sigma_i \in \{-1, 1\}$$

Hamiltonian

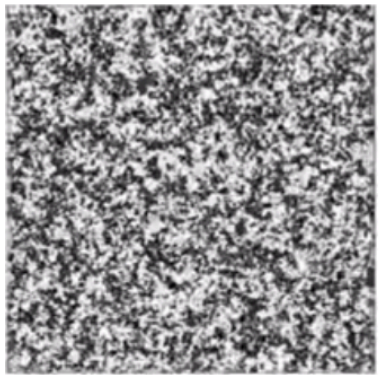
sum over all neighbors

coupling strength, also called “disorder”

upward / downward spins



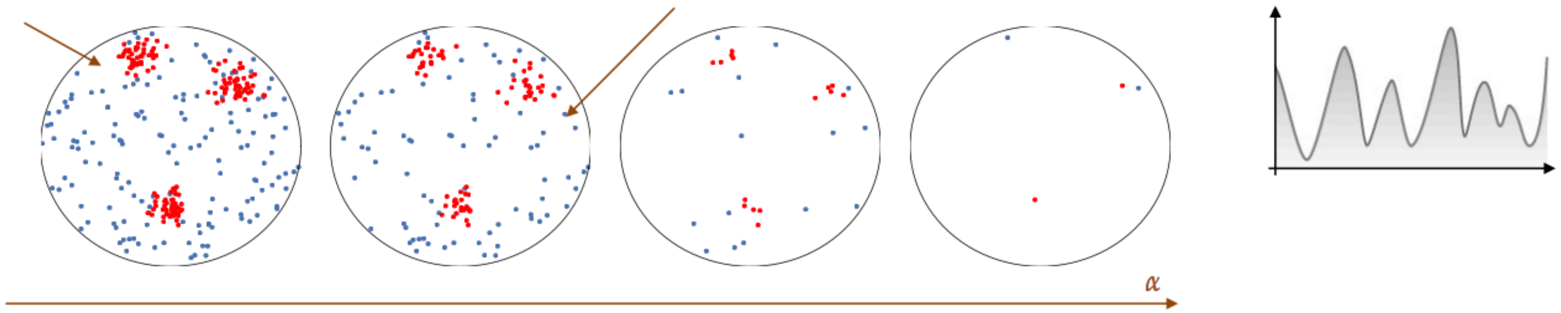
large correlations at
low temperature

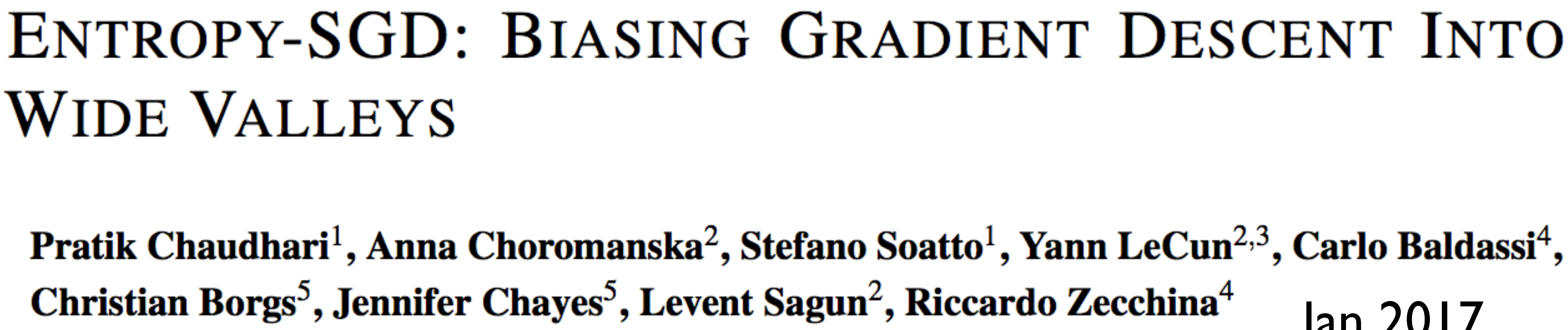


complete disorder at
high temperature

dense clusters do not show up in the
standard replica analysis of the
Gibbs distribution

only blue points = golf course-like landscape





Entropy-SGD: Biasing Gradient Descent Into Wide Valleys

Pratik Chaudhari[1], Anna Choromanska[2], Stefano Soatto[1], Yann LeCun[2,3], Carlo Baldassi[4], Christian Borgs[5], Jennifer Chayes[5], Levent Sagun[2], Riccardo Zecchina[4]

Jan 2017

- Similar formula to Local Entropy in Spin Glasses, but now in continuous variables.

$$f_\gamma(x) = -\log\left[G_\gamma * e^{-f(x)}\right]$$

- *Algorithmic*: can evaluate grad f efficiently by an auxiliary SGD dynamics.

$$G_\gamma(x) = Ce^{-\frac{\|x\|^2}{2\gamma}},$$

- No PDEs in this paper!

Entropy-SGD improves training and generalization

- Entropy-SGD: a modification of SGD, which results in shorter training time, and weights with better generalization.

  - Training time is important: large networks may take weeks

  - Generalization is very important. Gains from training are "free" compared gains from model/data

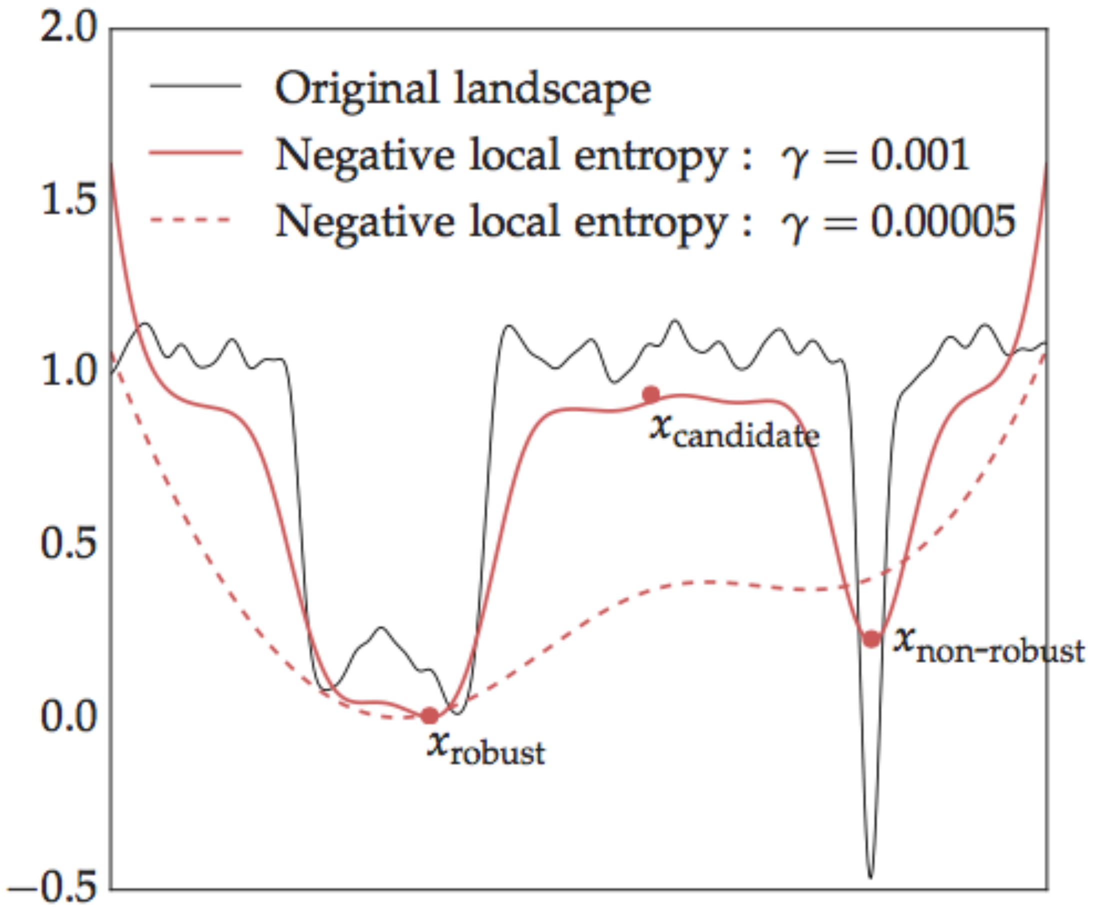


Figure 2: Local entropy concentrates on wide valleys in the energy landscape.

HJB-PDEs and Local-Entropy

[*Deep Relaxation* C. O. O. S. C. 2017/05] started by identifying the Local Entropy function as the solution of a Hamilton-Jacobi PDE. This observation led to:

- Proof that the method trains faster
- Proof of wider minima (believed to be related to generalization).

and eventually, improvements to the algorithm.

$f_\gamma(x) = u(x,\gamma)$ where u is the solution of

$$u_t(x,t) = -\frac{1}{2}|\nabla u|^2 + \frac{1}{2}\Delta u$$

$$u(x,0) = f(x)$$

Parallel SGD

EASGD [*LeCun … Elastic Averaging SGD*] effective parallel training

Very recently new algorithm, PARLE [Chaudry 2017/07], giving best results to date on CIFAR-10, CIFAR-100, SVHN

- ESGD on each processor
- Elastic forcing term between each particle.
- JKO gradient flow interpretation for PARLE:

$$J(\rho) = \int f_\gamma(x)\rho \, dx + \frac{1}{2\gamma} \int \int |x-y|^2 \rho(x) \, \rho(y) \, dx \, dy;$$

PDE interpretation of local entropy and equation for the gradient

Hopf-Cole Transformation for HJB

Define

$$f_\gamma(x) := u(x,\gamma) = -\frac{1}{\beta}\log\left(G_{\beta^{-1}\gamma} * \ \exp(-\beta f(x))\right);$$

where $G_\gamma(x)$ is the heat kernel. Then $u(x,\gamma)$ is the solution of

$$\begin{aligned}\frac{\partial u}{\partial t} &= -\frac{1}{2}|\nabla u|^2 + \frac{\beta^{-1}}{2}\Delta u, \qquad \text{for } 0 < t \leq \gamma\\ u(x,0) &= f(x)\end{aligned}$$

Moreover

$$\nabla u(x,t) = \int_{\mathbb{R}^n} \frac{y-x}{t}\rho_1^\infty(dy;\ x)$$

$$\rho_1^\infty(y;\ x) = Z_1^{-1}\exp\left(-\beta f(y) - \frac{\beta}{2t}|x-y|^2\right)$$

This is well-known result, see [Evans PDE]

Hopf-Cole Proof

Proof. Define $u(x,t) = -\beta^{-1} \log v(x,t)$. So $v = \exp(-\beta u)$ solves the heat equation

$$v_t = \beta^{-1} \Delta v$$

with initial data $v(x,0) = \exp(-\beta f(x))$. Taking partial derivatives gives

$$v_t = -\beta \ v \ u_t, \qquad \nabla v = -\beta \ v \ \nabla u, \qquad \Delta v = -\beta \ v \ \Delta u + \beta^2 \ v \ |\nabla u|^2.$$

Combining these expressions results in (viscous-HJ).

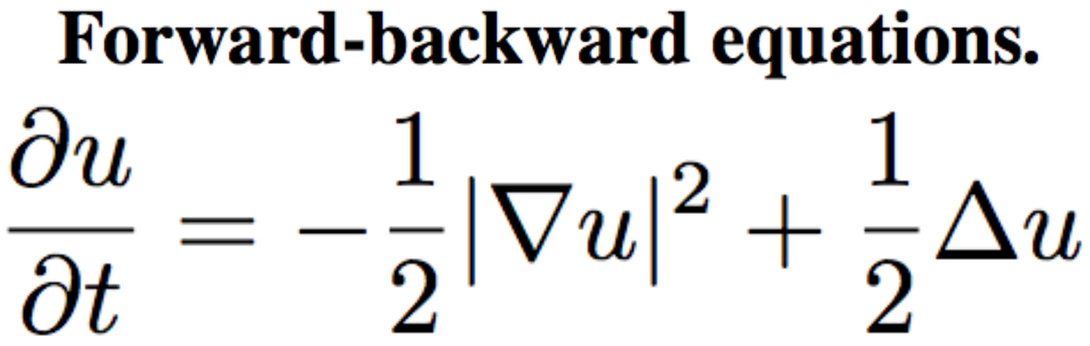
Differentiating $v(x,t) = \exp(-\beta u(x,t))$ gives up to positive constants which can be absorbed into the density,

$$\nabla u(x,t) = C \ \nabla_x \left(G_t \ * \ e^{-\beta f(x)} \right) = C \ \nabla_x \int \ G_t \ (y) \ e^{-\beta f(x-y)} \ dy$$

Local Entropy:

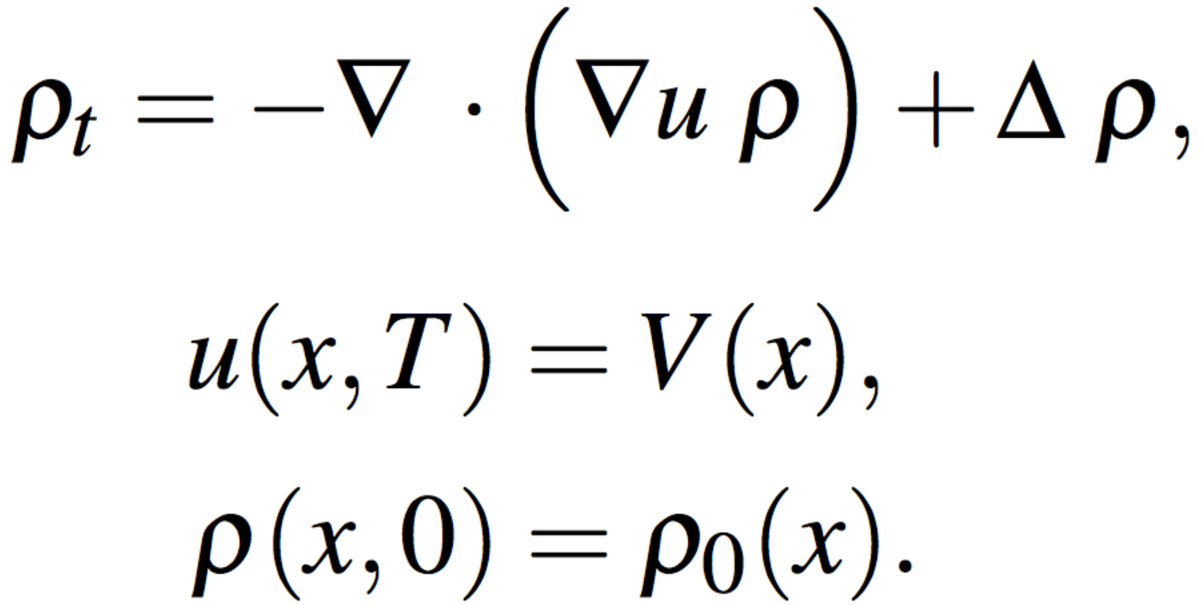
Visualization

Stochastic Optimal Control Interpretation



Forward-backward equations.

$$\frac{\partial u}{\partial t} = -\frac{1}{2}|\nabla u|^2 + \frac{1}{2}\Delta u$$



$$\rho_t = -\nabla \cdot \Big(\nabla u \, \rho \Big) + \Delta \, \rho,$$

$$u(x,T) = V(x),$$

$$\rho(x,0) = \rho_0(x).$$

Visualization of Improvement: dimension 1, PDE simulation.

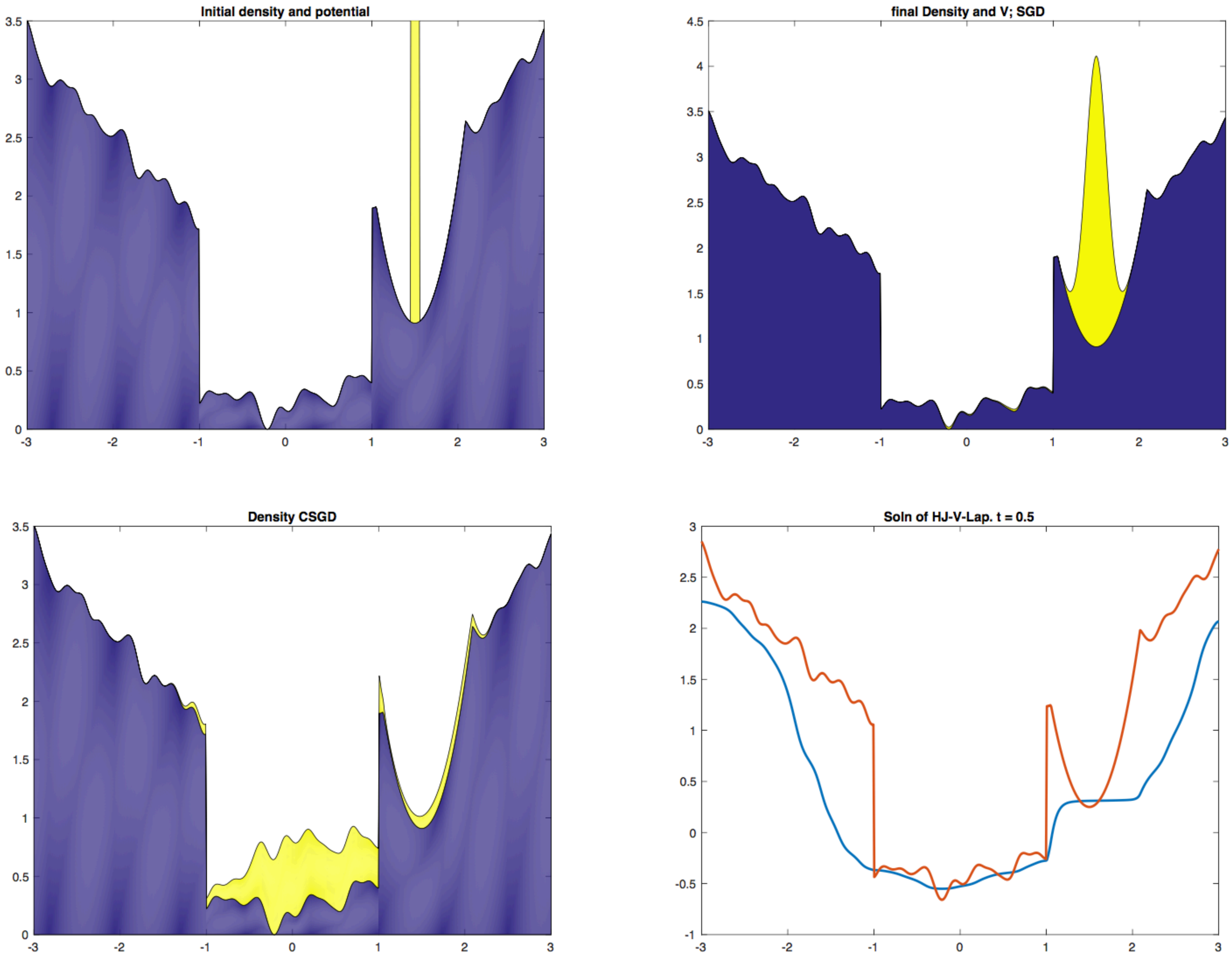
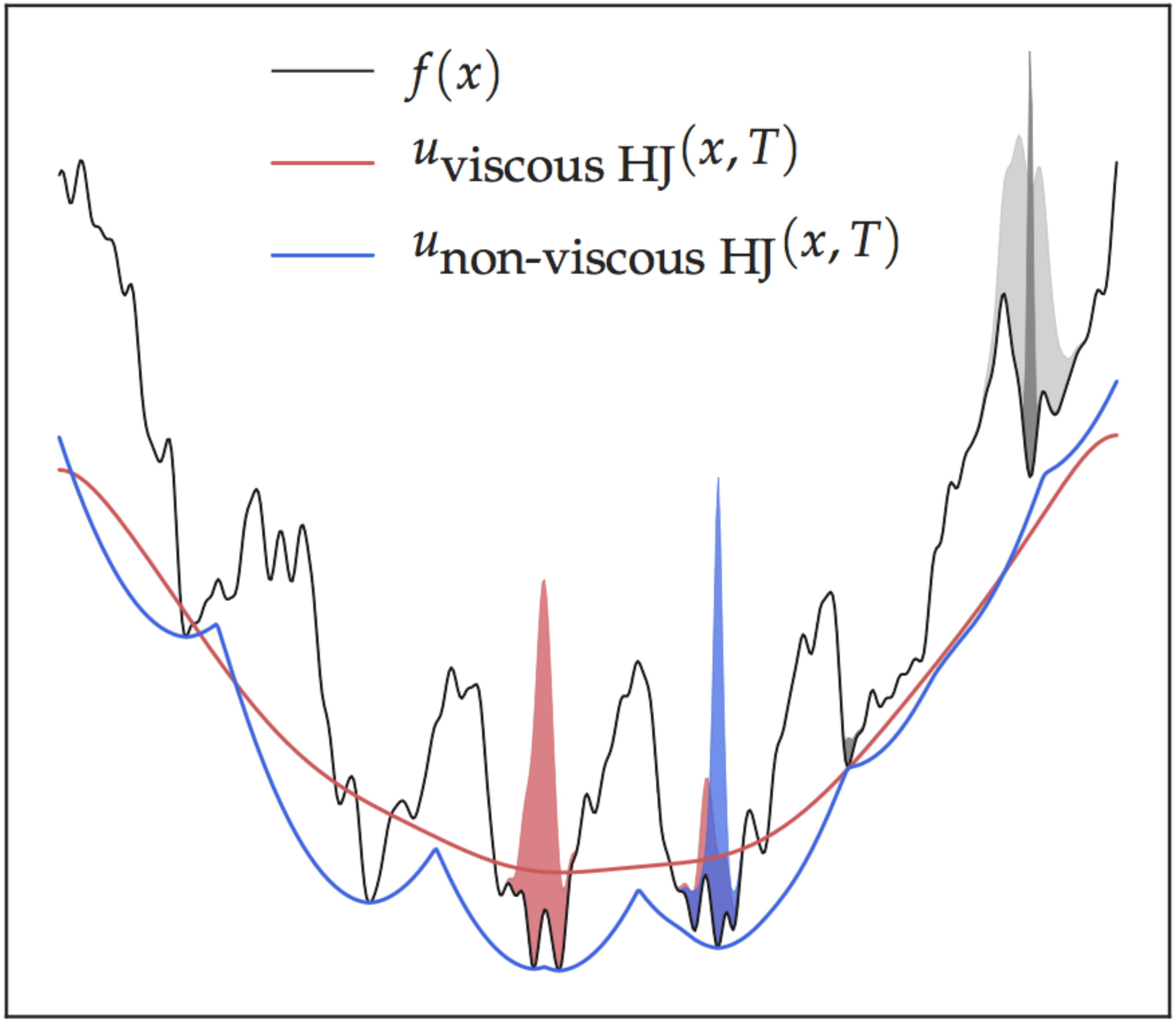


FIGURE 1. Initial density, final density SGD, final density CSGD, Solution of HJB **Forward-backward equations.**

Local Entropy is Regularization using Viscous Hamilton-Jacobi PDE



- True solution in one dimension. (Cartoon in high dimensions, because algorithm only works for shorter times.)

Proof of Improvement for Modified dynamics

Modified System

Consider the following controlled SDE

$$dx(s) = -\nabla f(x(s))\, ds - \alpha(s)\, ds + \beta^{-1/2}\, dW(s), \quad \text{for } t \leq s \leq T,$$

$$\mathscr{C}(x(\cdot),\ \alpha(\cdot)) = \mathbb{E}\left[V(x(T)) + \frac{1}{2}\int_0^T |\alpha(s)|^2\, ds\right].$$

Using stochastic control theory [Fleming] obtain HJB equation for the value function …

$$-u_t = -\nabla f \cdot \nabla u - \frac{1}{2}\, |\nabla u|^2 + \frac{\beta^{-1}}{2}\, \Delta u,$$

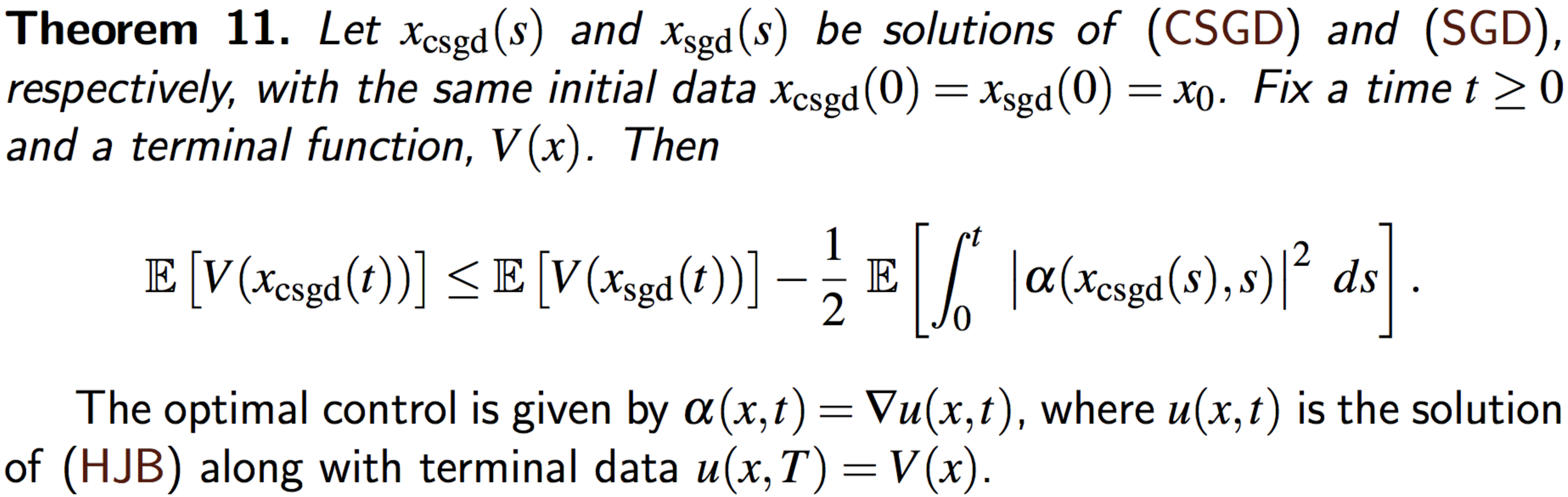
$$\rho_t = -\nabla\ \cdot \Big((\nabla u + \nabla f)\rho \Big) + \Delta\rho,$$

$$\text{for } 0 \leq\ s \leq\ T$$

$$u(x,T) = V(x), \qquad \rho(x,0) = \rho_0(x)$$

Note: the zero control corresponds to SGD

Expected Improvement Theorem



Theorem 11. *Let $x_{\text{csgd}}(s)$ and $x_{\text{sgd}}(s)$ be solutions of* (CSGD) *and* (SGD)*, respectively, with the same initial data $x_{\text{csgd}}(0) = x_{\text{sgd}}(0) = x_0$. Fix a time $t \geq 0$ and a terminal function, $V(x)$. Then*

$$\mathbb{E}\left[V(x_{\text{csgd}}(t))\right] \leq \mathbb{E}\left[V(x_{\text{sgd}}(t))\right] - \frac{1}{2}\,\mathbb{E}\left[\int_0^t \left|\alpha(x_{\text{csgd}}(s),s)\right|^2\, ds\right].$$

The optimal control is given by $\alpha(x,t) = \nabla u(x,t)$, where $u(x,t)$ is the solution of (HJB) along with terminal data $u(x,T) = V(x)$.

- Note this is the modified (HJB) from the previous slide.

- Alternately, if we go back to the original HJB, we have the implicit gradient descent interpretation.

- Or, same theorem, comparing LE-SGD to random walk (no gradient)

Solving PDEs in high dimensions?

not quite, just need gradient at one point.

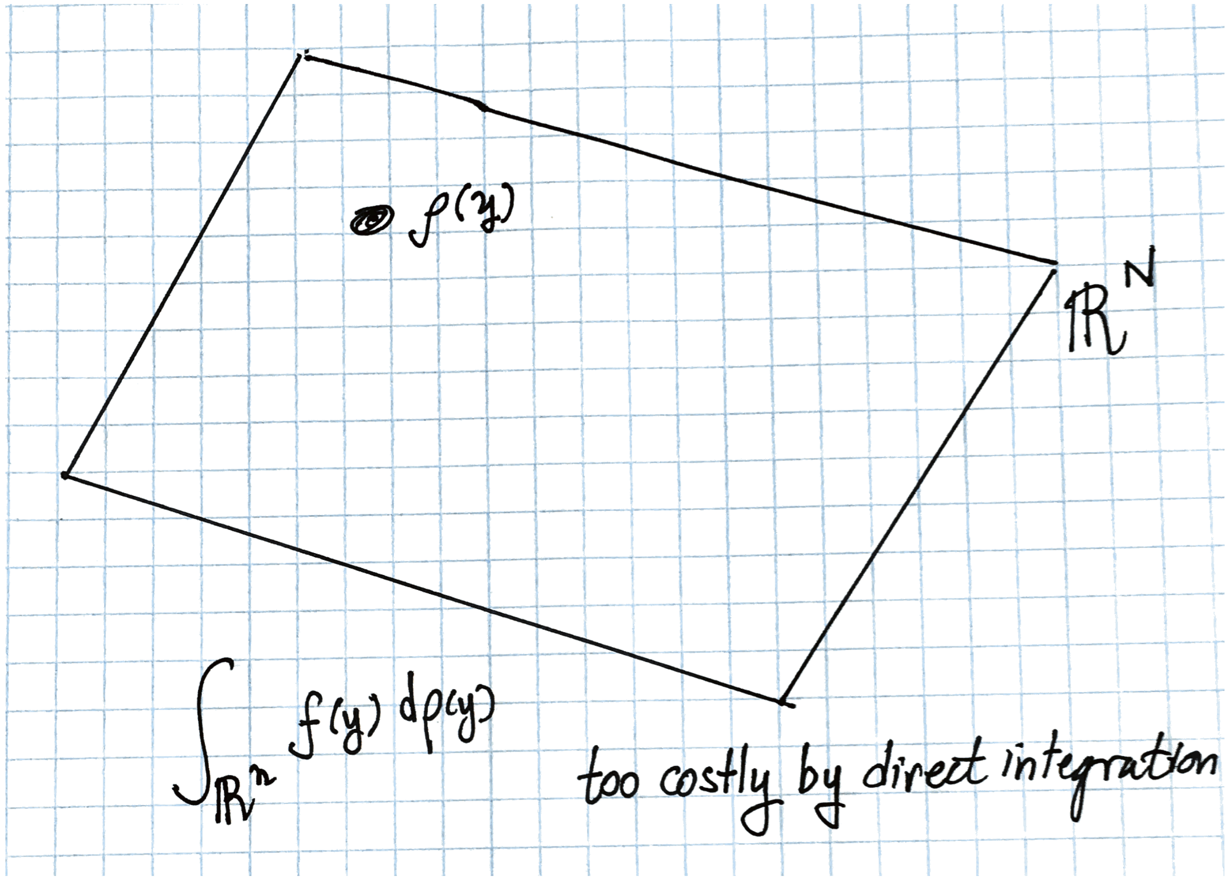
Will integration work?

no! curse of dimensionality.

Require a method which overcomes the curse of dimensionality:

Langevin Markov-Chain Monte Carlo (MCMC)

Langevin MCMC



Want to compute:

$$\nabla u(x,t) = \int_{\mathbb{R}^n} \frac{y-x}{t} \rho_1^\infty(dy;\ x)$$

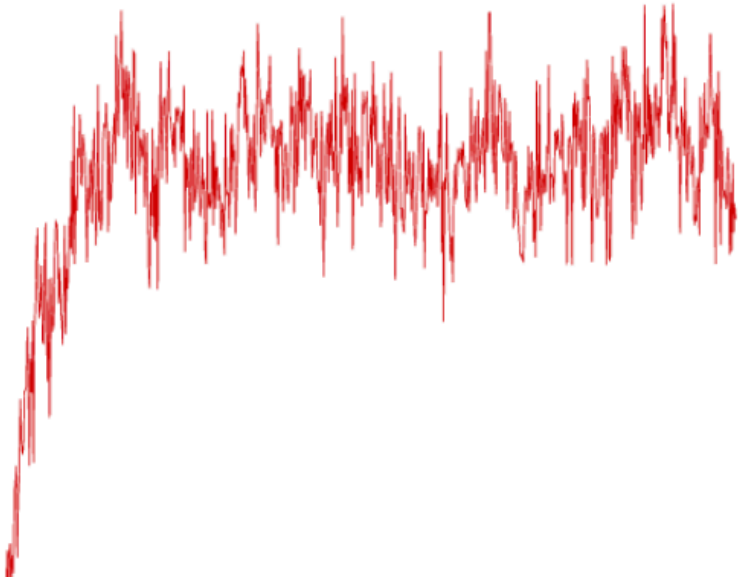
Langevin MCMC:
sample the measure using a dynamical system, and average expression against the measure by a time average, using ergodicity.

Find dynamics with invariant measure $\rho(y)$:

$$dy(s) = -(x-y)ds + dW,$$

take expectation of f via dynamics

$$\int f(y)\rho(y)dy = \lim_{T\to\infty} \frac{1}{T} \int_0^T f(y(s))ds$$



Stochastic Differential Equations and Fokker Planck PDE

$$dx(t) = -\nabla f(x(t))\, dt + \sqrt{2\beta^{-1}}\, dW(t);$$

$$\mathscr{L}\phi = -\nabla f \;\cdot \nabla\; \phi + \beta^{-1}\Delta\, \phi.$$

$$\frac{\partial u}{\partial t} = \mathscr{L}u,$$

$$u(x,t) = \mathbb{E}\Big[V(x(T))\;\Big|\; x(t) = x\Big]$$

$$\mathscr{L}^{*}\rho = \nabla \cdot (\nabla f \rho) + \beta^{-1}\Delta\, \rho$$

$$\frac{\partial}{\partial t}\, \rho(x,t) = \;\; \mathscr{L}^{*}\rho$$

$$\boxed{\rho^{\infty}(x;\; \beta) = Z(\beta)^{-1} e^{-\beta f(x)}}$$

Background: Homogenization of SDEs

Pavliotis and Stuart (2008, Chap. 10, 17)

- Two scale dynamics

$$dx(s) = h(x,\, y)\, ds$$

$$dy(s) = \frac{1}{\varepsilon}\, g(x,\, y)\, ds + \frac{1}{\sqrt{\varepsilon}}\, dW(s);$$

- Unique invariant measure of the fast dynamics

$$\mathscr{L}_0^*\, \rho^\infty(y;\, x) = 0;$$

- In the limit, obtain homogenized dynamics

$$d\overline{x}(s) = \overline{h}(x)\, ds$$

- given by averaging against the invariant measure

$$\overline{h}(x) = \int h(x,y)\, \rho^\infty(dy;\, x).$$

- Equivalent by ergodicity to a time average.

$$= \lim_{T\to\infty} \frac{1}{T} \int_0^T h(x,y(s))\, ds$$

Langevin MCMC for the Gradient

Consider the following auxiliary system of SDEs

$$\begin{aligned} dx(s) &= -\gamma^{-1}\,(x-y)\,ds \\ dy(s) &= -\frac{1}{\varepsilon}\left[\nabla f(y) + \frac{1}{\gamma}\,(y-x)\right]\,ds + \frac{\beta^{-1/2}}{\sqrt{\varepsilon}}\,dW(s). \end{aligned} \qquad \text{(Entropy-SGD)}$$

Theorem 4. *As $\varepsilon \to 0$, the system (Entropy-SGD) converges to the homogenized dynamics given by*

$$dX(s) = -\nabla f_\gamma(X)\,ds.$$

Moreover, $-\nabla f_\gamma(x) = -\gamma^{-1}\,(x - \langle y\rangle)$ *where*

$$\langle y\rangle = \int y\,\rho_1^\infty(dy;\,X) = \lim_{T\to\infty}\frac{1}{T}\int_0^T y(s)\,ds$$

Proof of MCMC for the Gradient

Proof. Write

$$H(x,y;\gamma) = f(y) + \frac{1}{2\gamma}|x-y|^2 .$$

The Fokker-Planck equation for the density of $y(s)$ is given by

$$\rho_t = \mathscr{L}_0^* \rho = \nabla_y \cdot (\nabla_y H\rho) + \frac{\beta^{-1}}{2} \Delta_y \rho;$$

The invariant measure for this Fokker-Planck equation is thus

$$\rho_1^\infty(y;\ x) = Z^{-1} \exp(-\beta H(x,y;\gamma))$$

which agrees with the expression for the gradient from the Hopf-Cole formula.

The conclusion then follows homogenization of SDEs

$$\overline{h}(X) = -\gamma^{-1} \int (X-y)\ \rho_1^\infty(y;\ X)$$

Exponential Convergence in Wasserstein for Fokker-Planck in convex case

Fokker Planck is Gradient descent in Wasserstein of

$$J(\rho) = \int f(x)\,\rho\,dx + \beta^{-1}\int \rho\,\log\rho\,dx;$$

The convergence rate for a λ-convex function $f(x)$ (meaning $D^2 f(x) \geq \lambda I$) is exponential with rate λ.

$$d_{W_2}\left(\rho(x,t),\, \rho^\infty\right) \leq d_{W_2}\left(\rho(x,0),\, \rho^\infty\right)\, e^{-\lambda t}.$$

Langevin dynamics, the λ-convexity of f is improved by a factor of $1/\gamma$.

So the MCMC step is exponentially convergent, for small enough values of time. This explains why the algorithm converges with a relatively small (100) time steps. (Accurate enough with 25 steps).

Algorithm and Results in Deep Networks

Algorithm for Local Entropy

- *Scoping: for the control problem. Gamma decreases linearly with time. (at leads near final time).*

$$\gamma(t) = T - t$$

- *Outer Loop: Implicit Gradient descent*

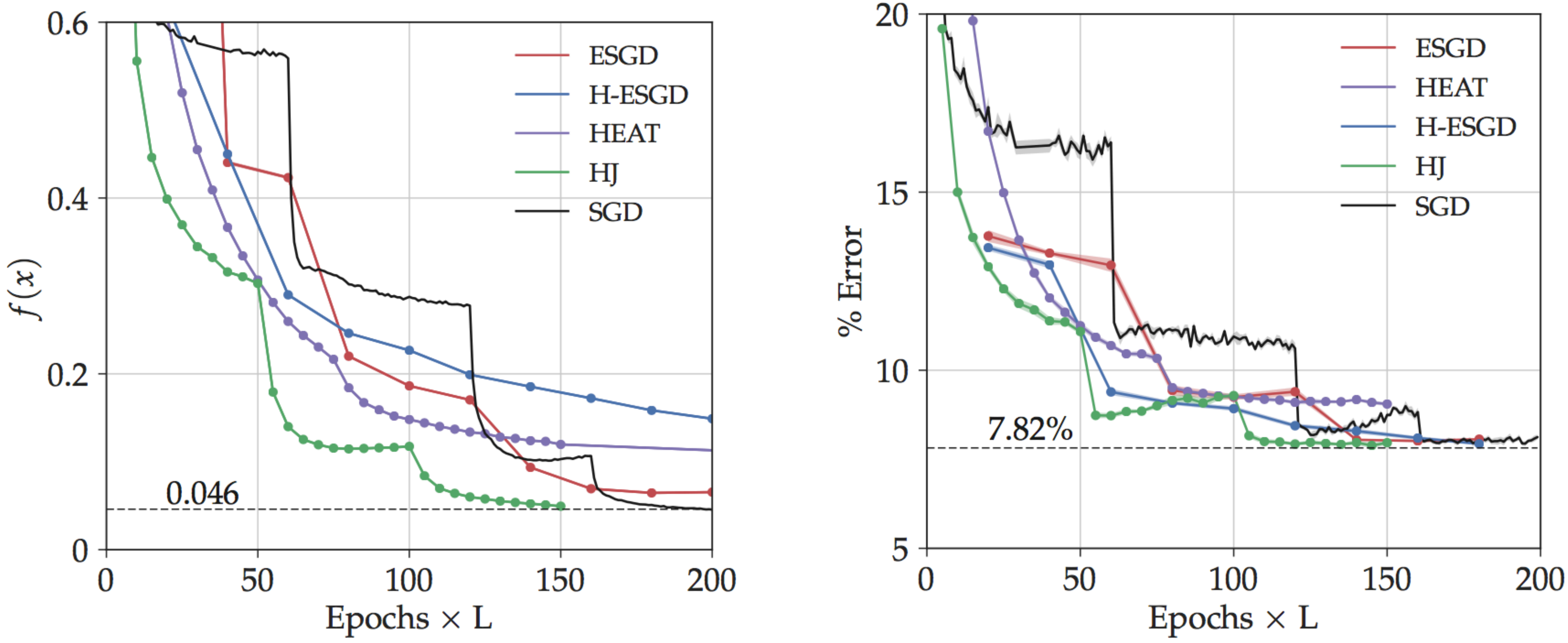
$$\frac{x_{k+1} - x_k}{\gamma(t)} = -\nabla u(x_k, \gamma(t))$$

- *Inner Loop: Estimate gradient by Langevin MCMC*

$$\nabla u(x_k, \gamma(t)) = \int_{\mathbb{R}^n} \frac{y - x_k}{\gamma(t)} \rho_1^{\infty}(dy;\ x)$$

Numerical Results

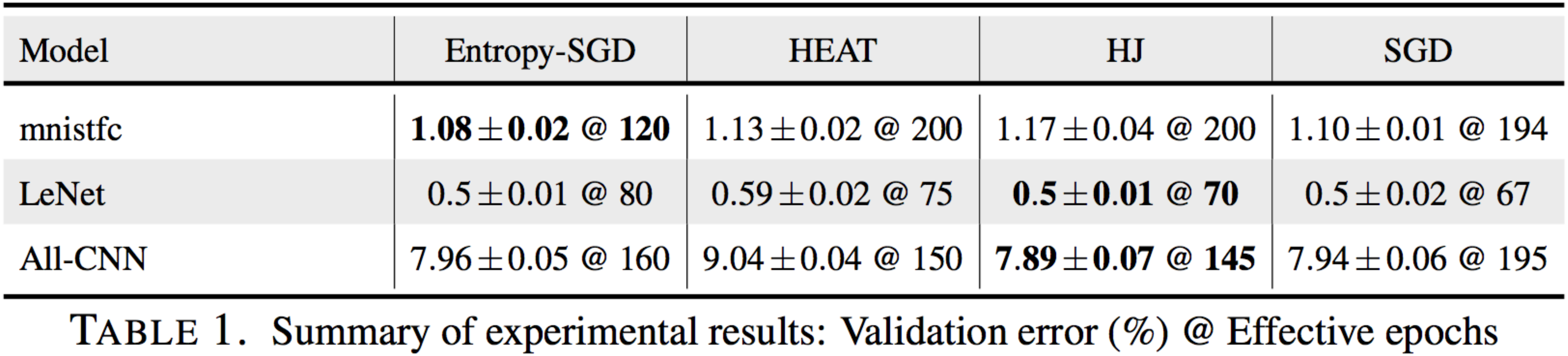
Visualization of Improvement in training loss (left)
Improve in Validation Error (right)
dimension = 1.67 million



(A) All-CNN: Training loss

(B) All-CNN: Validation error

Numerical Results



Model	Entropy-SGD	HEAT	HJ	SGD
mnistfc	**1.08** ± **0.02 @ 120**	1.13 ± 0.02 @ 200	1.17 ± 0.04 @ 200	1.10 ± 0.01 @ 194
LeNet	0.5 ± 0.01 @ 80	0.59 ± 0.02 @ 75	**0.5** ± **0.01 @ 70**	0.5 ± 0.02 @ 67
All-CNN	7.96 ± 0.05 @ 160	9.04 ± 0.04 @ 150	**7.89** ± **0.07 @ 145**	7.94 ± 0.06 @ 195

TABLE 1. Summary of experimental results: Validation error (%) @ Effective epochs

E-SGD: previous algorithm,
HJ improved algorithm

SGD well tuned, i.e. best results previously obtained.
HJ improves both the training time and the Validation error.
These fractions of a percent are significant.

PARLE-SGD

$$J(\rho) = \int f_\gamma(x)\rho \, dx + \frac{1}{2\gamma} \int \int |x-y|^2 \rho(x) \, \rho(y) \, dx \, dy;$$

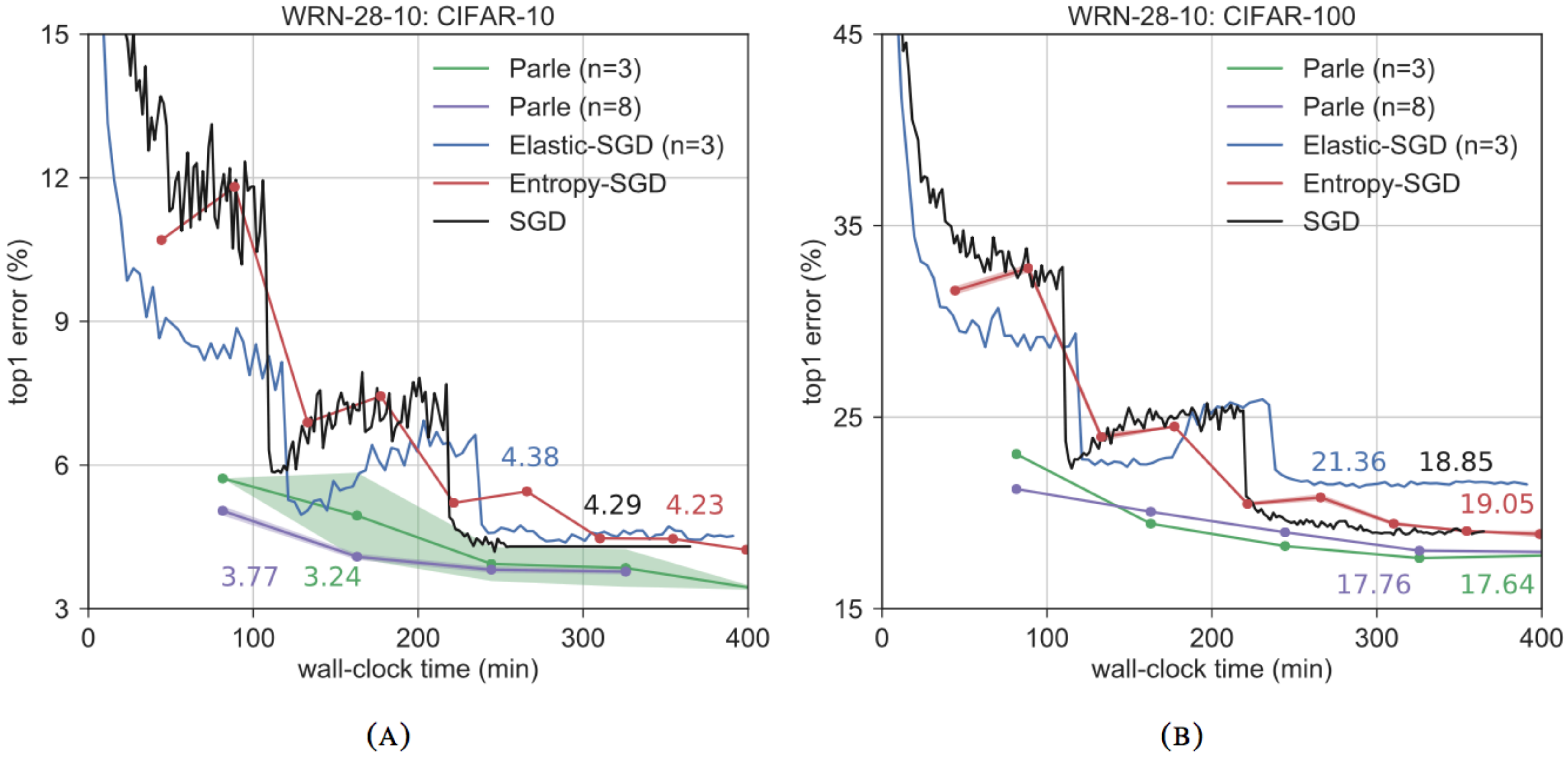
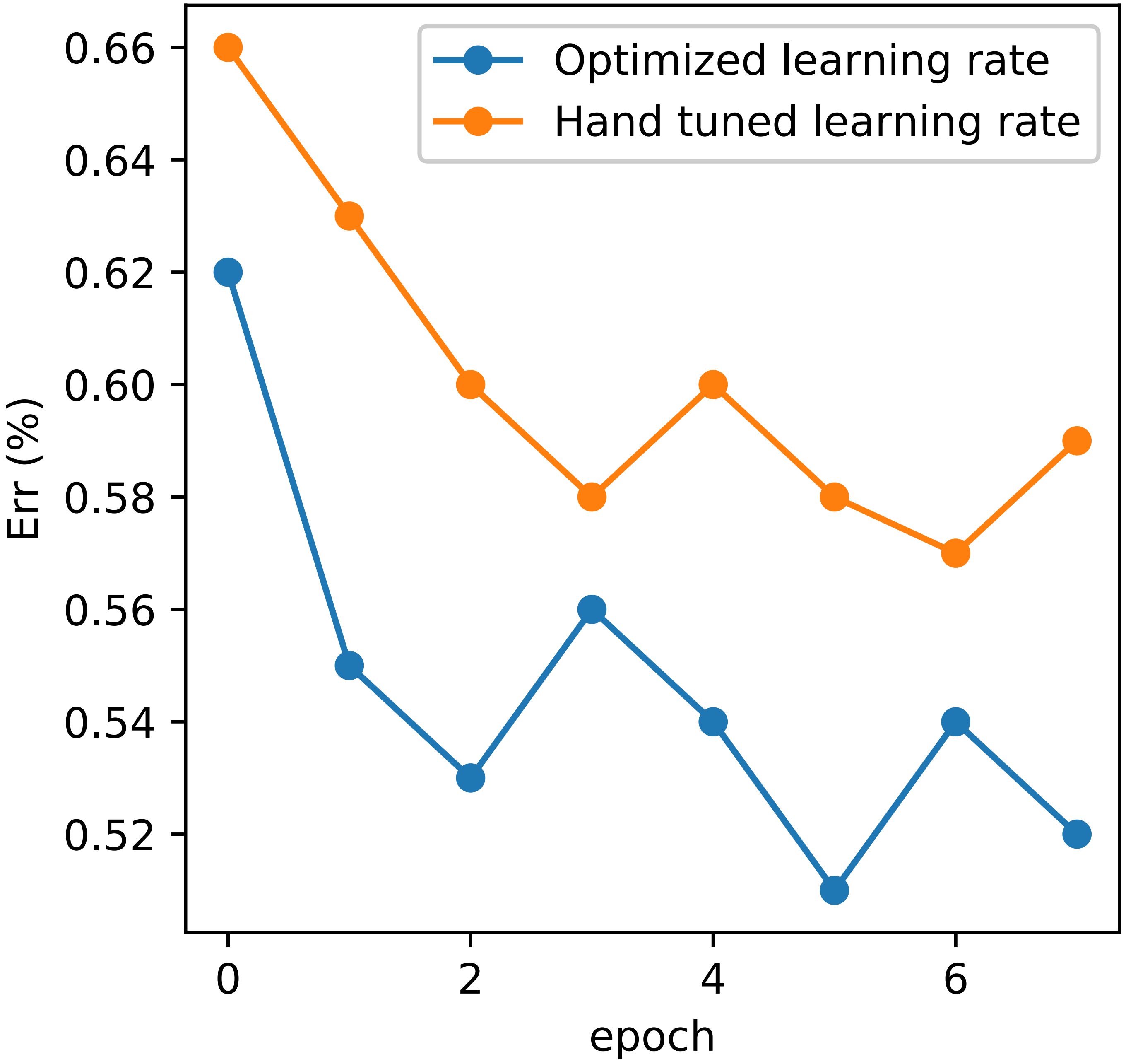


FIGURE 3. Validation error of WRN-28-10 on CIFAR-10 (Fig. 3a) and CIFAR-100 (Fig. 3b)

Improvements using PDE optimized learning rate

MNIST



with Chris Finlay PhD student McGill

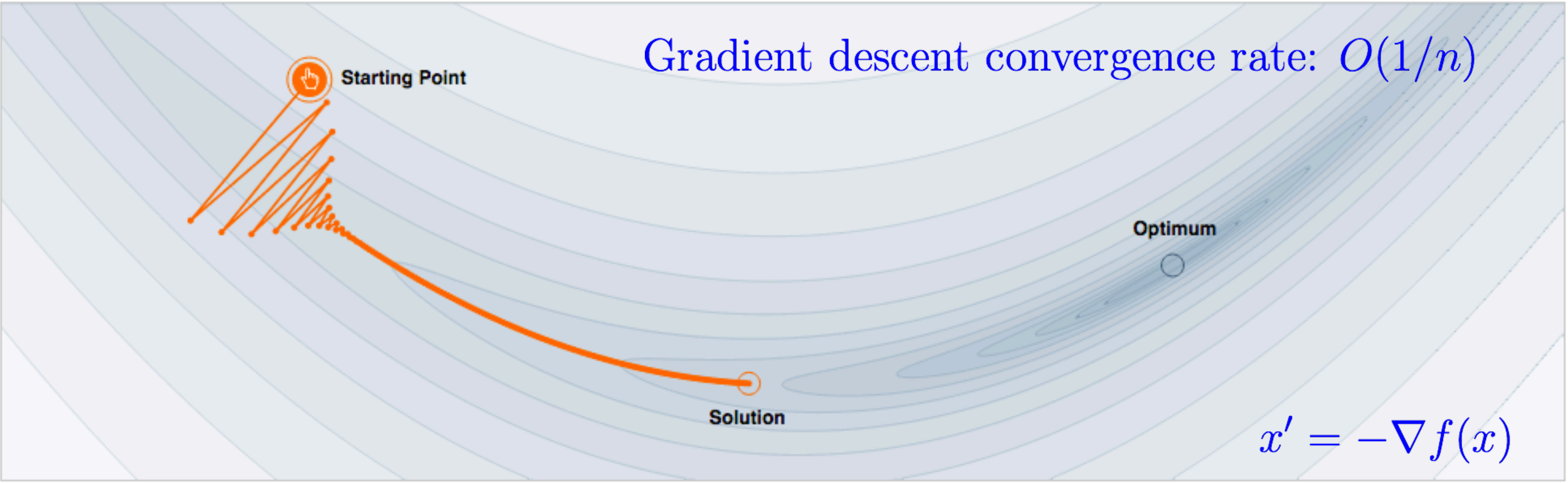
Optimization:

Acceleration methods
Deterministic and Stochastic

HJB gradient as implicit gradient descent

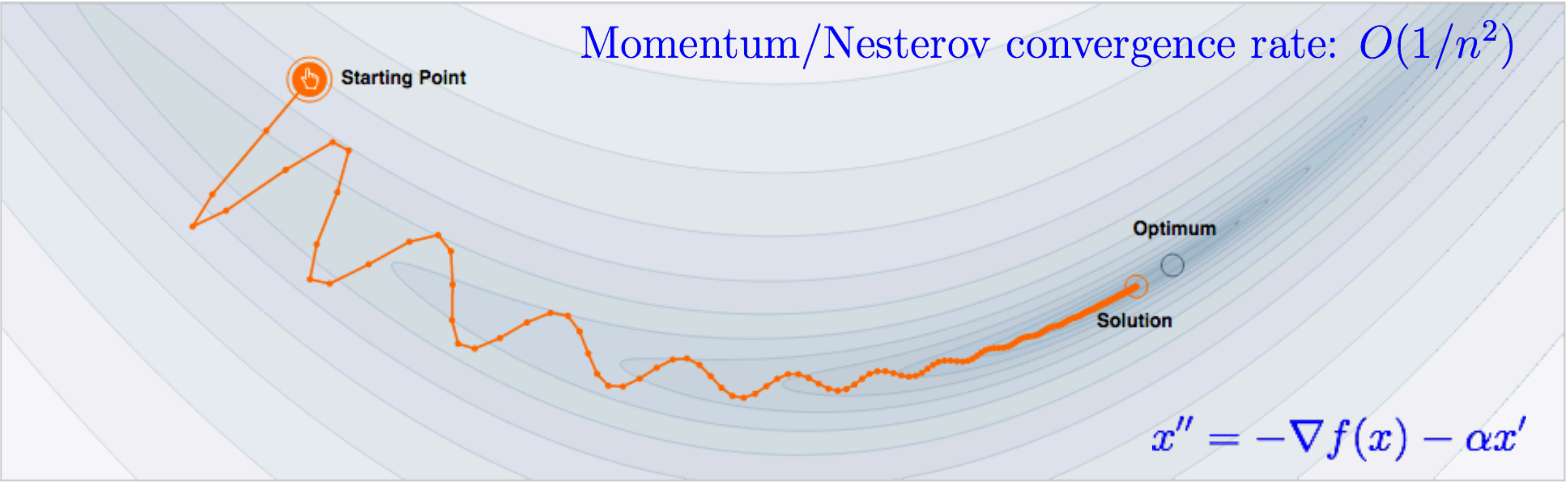
(Most of our analysis is for continuous time
in practice, take discrete time steps)

Accelerated Gradient Methods for (non-strictly) convex functions



Gradient descent convergence rate: $O(1/n)$

$$x' = -\nabla f(x)$$



Momentum/Nesterov convergence rate: $O(1/n^2)$

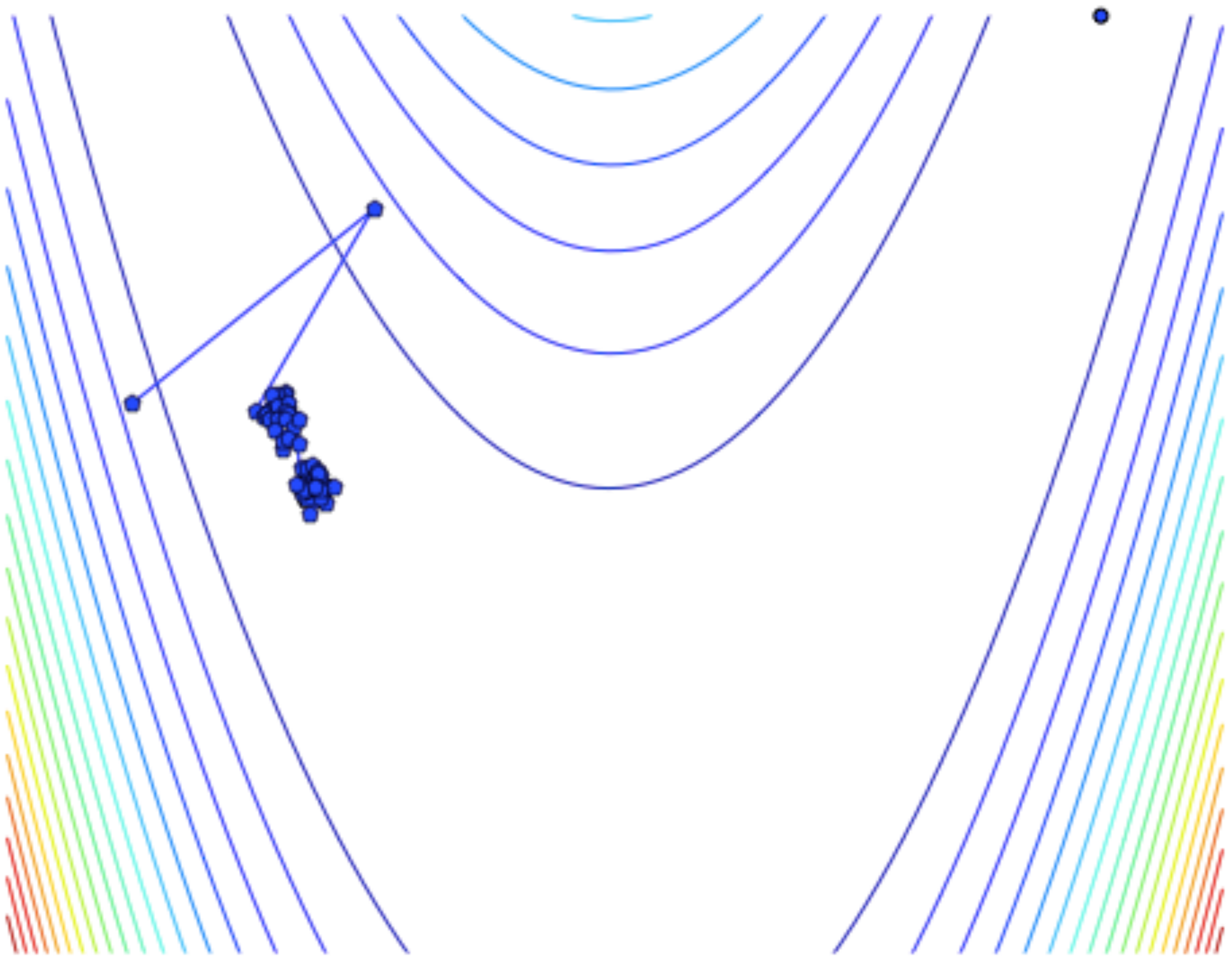
$$x'' = -\nabla f(x) - \alpha x'$$

SGD versus Entropy-SGD

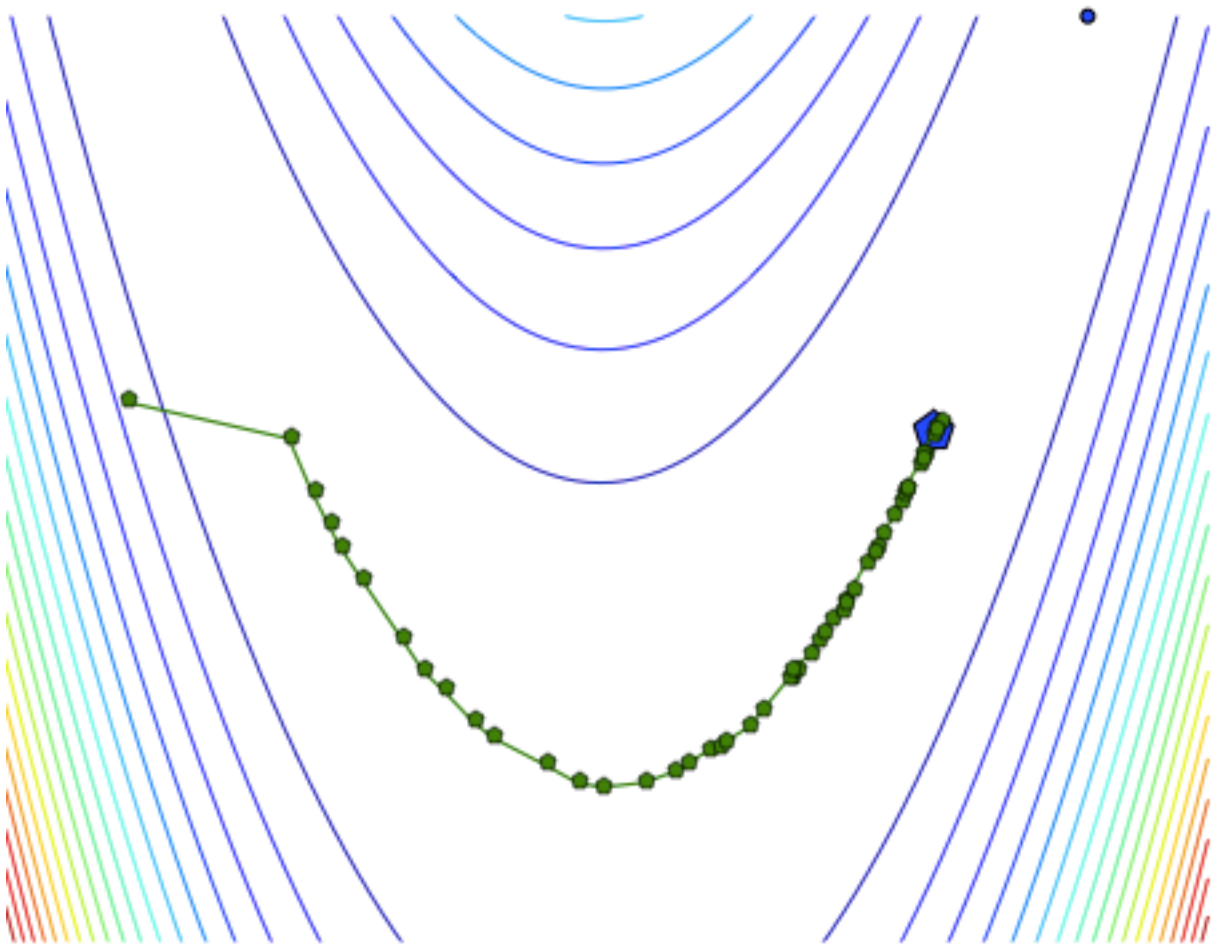
$$dx(t) = -\nabla f(x(t))dt + dW_t$$

Stochastic Gradient descent convergence rate: $O(1/\sqrt{n})$

$$\frac{x_{k+1} - x_k}{\tau} = -\nabla u(x_k, \tau)$$



50 steps of SGD



50 outer steps LESGD,
(25 steps in each inner loop)

figures: PhD student Bilal Abbasi

Implicit/Proximal gradient descent

Implicit methods: more stable, allow longer time step.
Not practical: requires a (local) minimization/equation solve at each step.

$$x_{k+1} \in \arg\min_x \left\{ f(x) + \frac{1}{2\tau}|x - x_k|^2 \right\}$$

Advantages: stable, guaranteed descent, even in *nonconvex* case

$$f(x_{k+1}) \leq f(x_k) - \frac{1}{\tau}|x_{k+1} - x_k|^2$$

Method is equivalent to backward Euler method for gradient descent.

$$\frac{x_{k+1} - x_k}{\tau} = -\nabla f(x_{k+1})$$

Gradient can be evaluated from the solution of Hamilton-Jacobi PDE

$$u(x,\tau) = \min_y \left\{ f(y) + \frac{1}{\tau}|y - x|^2 \right\}$$

$$u_t = -\frac{1}{2}|\nabla u|^2, \quad u(x,0) = f(x)$$

The corresponding update is exactly

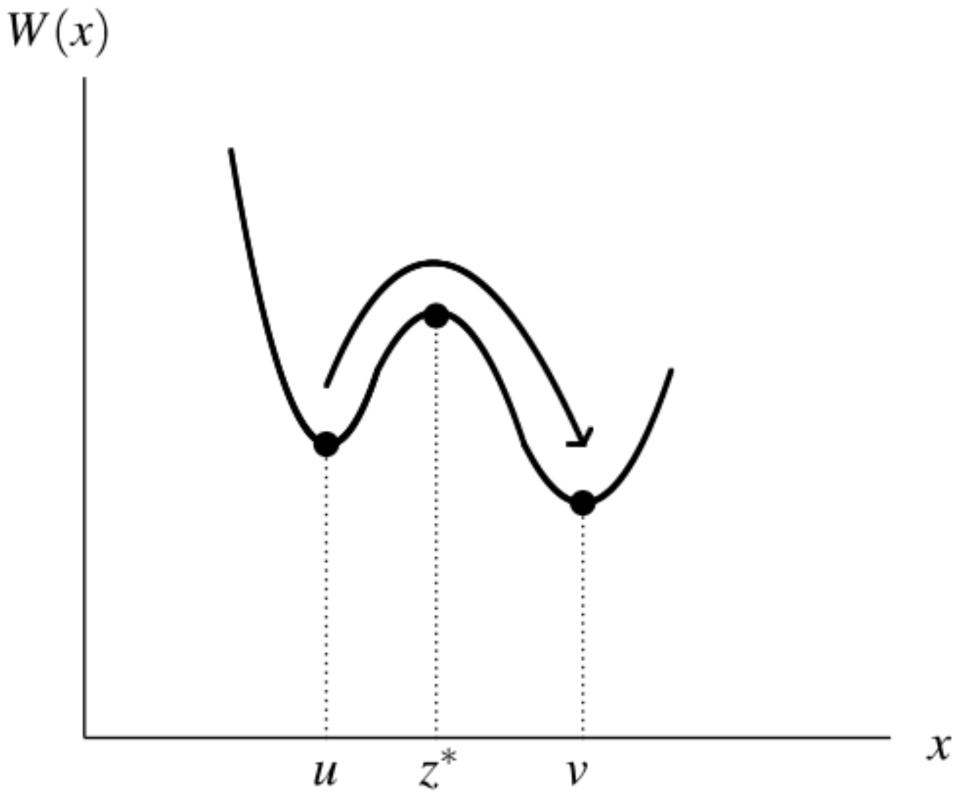
$$\frac{x_{k+1} - x_k}{\tau} = -\nabla u(x_k, \tau)$$

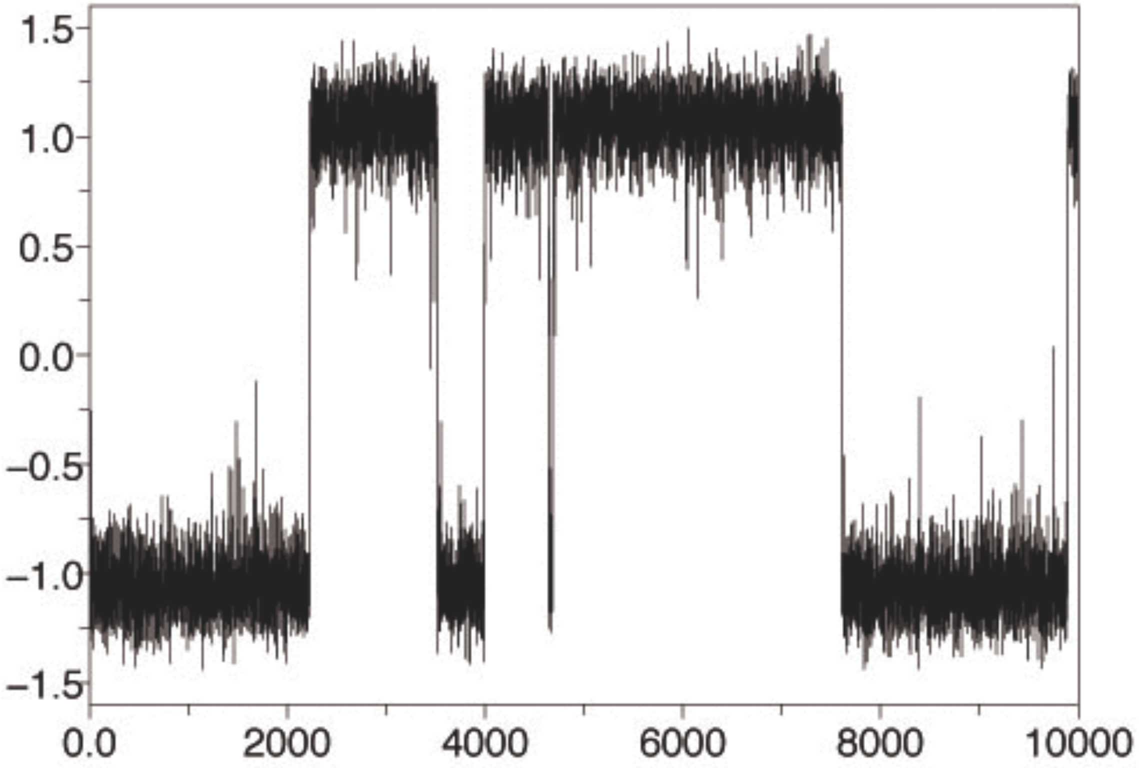
So PDE solution gives a formula for implicit GD

Fokker-Planck with nonconvex Potentials

Challenges, and insights from Computational Molecular Dynamics

Metastability in one dimension:
Exponential time to discover nearby minima





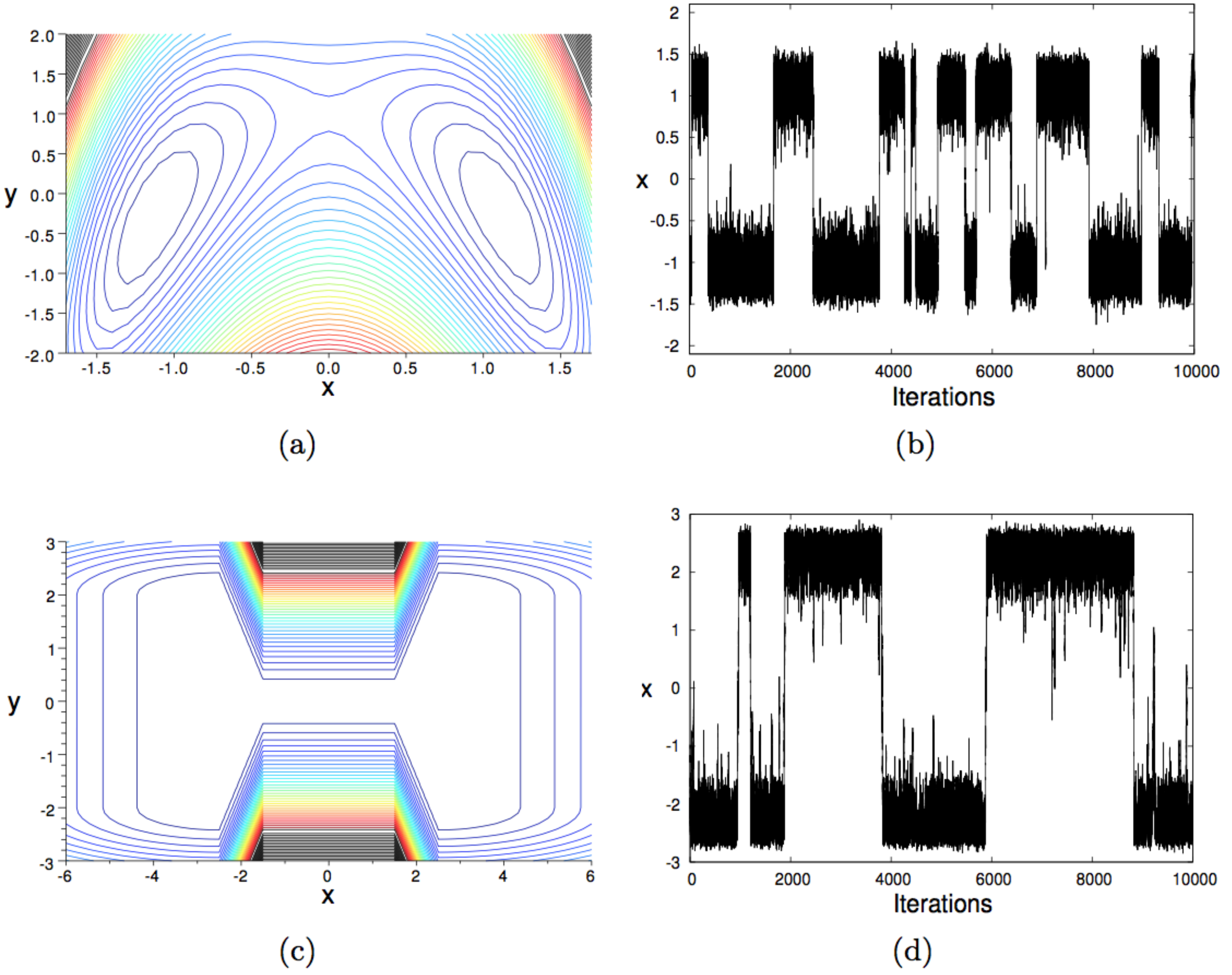
$$dX_t = -W'(X_t)\,dt + \sqrt{2\epsilon}dW_t$$

$$\mathbb{E}_u[\tau_v] = [1+o(1)]\frac{2\pi}{\sqrt{[-W''(z^*)]W''(u)}}\exp\big[\big(W(z^*) - W(u)\big)/\varepsilon\big].$$

Ref: [Bovier, *Metastability*] for [Kramer's 1940] formula

Video

Metastability in higher dimensions



(a) (b)

(c) (d)

Figure 1.1. (a, c) Level sets of the two-dimensional potentials. (b, d) Time evolution of the x-coordinate of the stochastic process solution to overdamped Langevin dynamics (1.8) in these potentials. (a, b) Energetic barrier ($\beta = 4$), (c, d) entropic barrier ($\beta = 10$).

Energetic Barrier

climb mountain pass
between valleys

Entropic Barrier

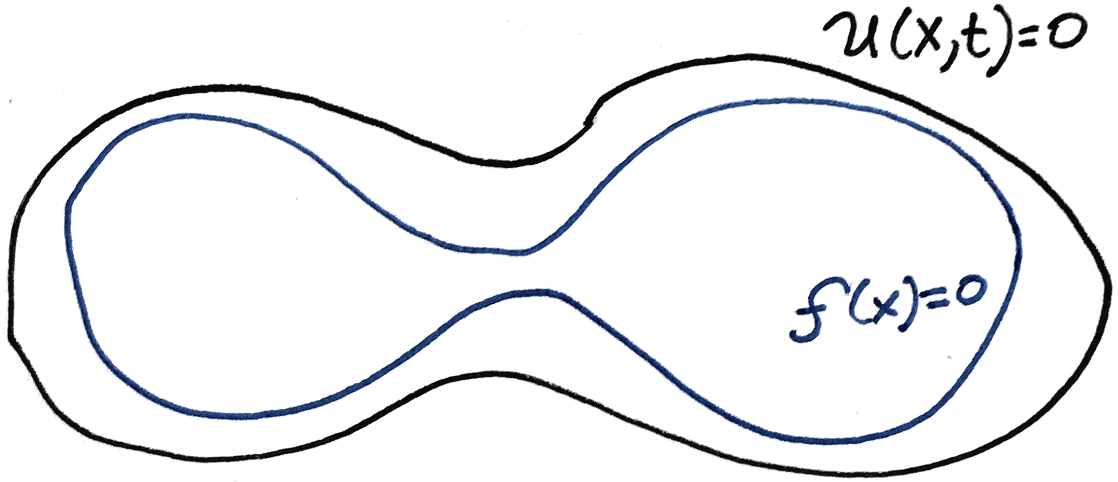
lakes connected
by narrow rivers

ref: [*PDEs … in Molecular Dynamics*, Lelievre and Stolz]

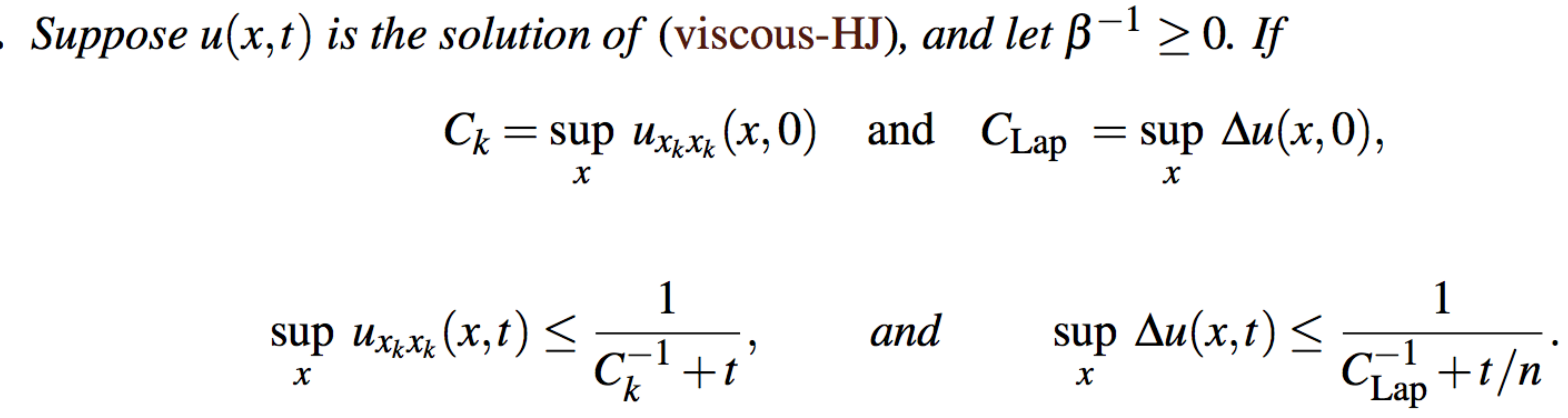
Metastability and widening in DNNs

Entropic Barriers are believed to be significant in DNNs.

- Conjecture: Local Entropy improves the entropic barriers, by “widening” local minima.
- PDE proof of second conjecture. using standard semi-concavity estimates.



Thm: HJB widens the narrow rivers



. *Suppose* $u(x,t)$ *is the solution of* (viscous-HJ), *and let* $\beta^{-1} \geq 0$. *If*

$$C_k = \sup_x \, u_{x_k x_k}(x,0) \quad \text{and} \quad C_{\text{Lap}} = \sup_x \, \Delta u(x,0),$$

$$\sup_x \, u_{x_k x_k}(x,t) \leq \frac{1}{C_k^{-1} + t}, \qquad \textit{and} \qquad \sup_x \, \Delta u(x,t) \leq \frac{1}{C_{\text{Lap}}^{-1} + t/n}.$$

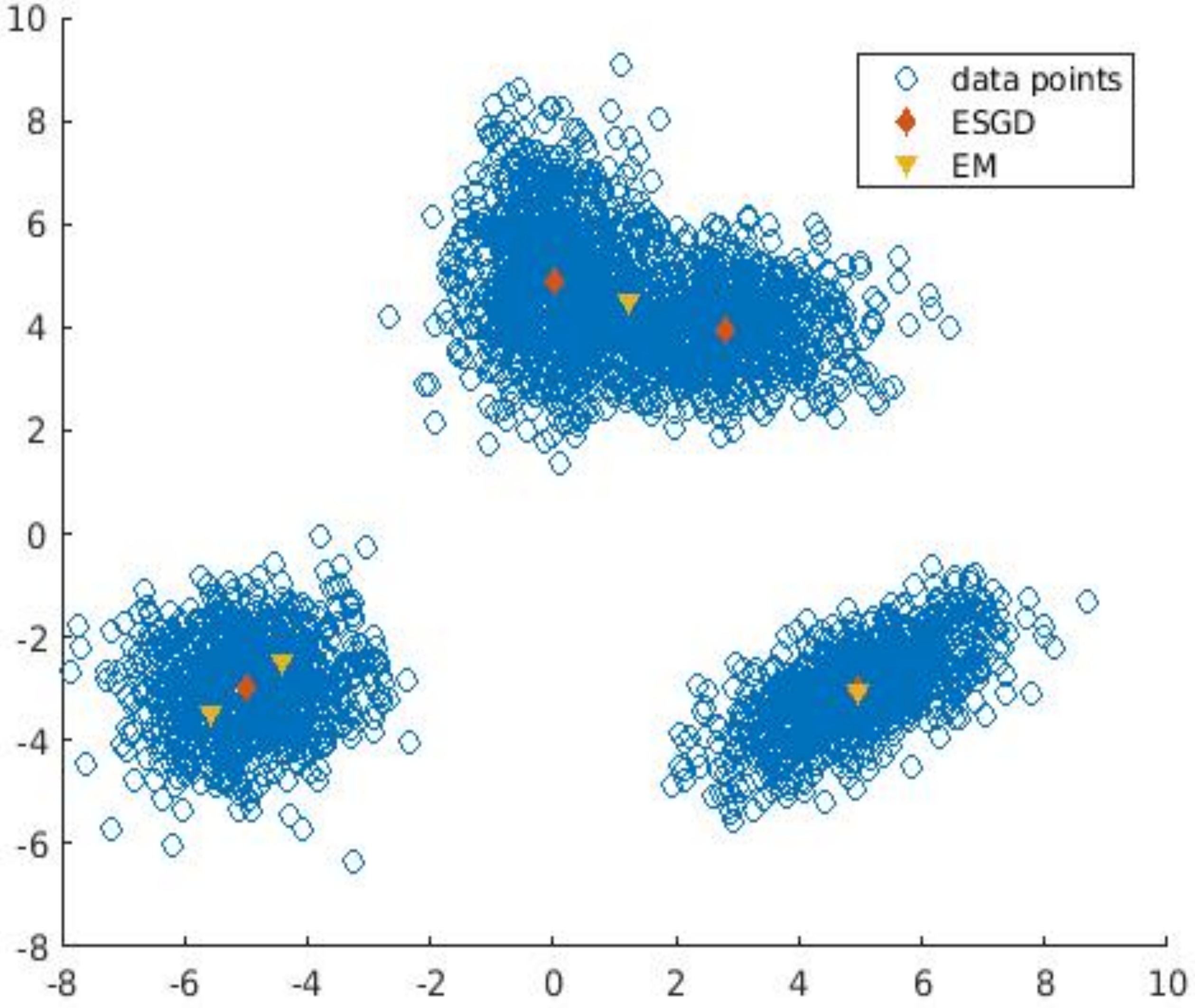
Algorithm Test: K-Means

work in progress with:
S. Osher, Mihn Pham, Penghang Yin, UCLA

Algorithm Applied to k-means clustering

$$\min f(x_1, \ldots, x_K) = \sum_{i=1}^{K} \sum_{j=1}^{N} \min_i \|x_i - y_j\|^2$$

- Standard algorithm, Lloyd's/EM can get stuck in a local minimum.
- Our algorithm, in comparable test case, finds global minimum
- Example on Right:
  - 4 means, 3 clusters
  - Optimal solution puts two means in the double cluster



Algorithm Applied to k-means clustering

1. ESGD vs. EM

   100 trials, K = 8 (ground truth), ESGD batch size = 1000

Method	Min	Max	Mean	Variance	% global min found
mb-EM	15.6800	27.2828	20.0203	6.0030	10%
ESGD	15.6808	15.6808	15.6808	1.49×10^{-10}	100%

2. ESGD vs. mini-batch EM (mb-EM)

   100 trials, K = 8 (ground truth), both batch size = 500

Method	Min	Max	Mean	Variance	% global min found
mb-EM	15.9148	18.1848	16.4009	0.7646	77%
ESGD	15.6816	15.6821	15.6820	1.18×10^{-9}	100%

Conclusions

- Discovered a HJB-PDE connection with Entropy-SGD algorithm, which has very good performance in Deep Networks.

- Exploited this connection to better understand the algorithm, giving proofs to empirical results about training.

- Improvements to algorithm using PDE insights and numerical PDE ideas.