Probing the energy landscape of Artificial Neural Networks

European Research Council

With the contributions of
ERC Grant n. 267915
OptInf

Carlo Lucibello
Research Fellow
Politecnico di Torino

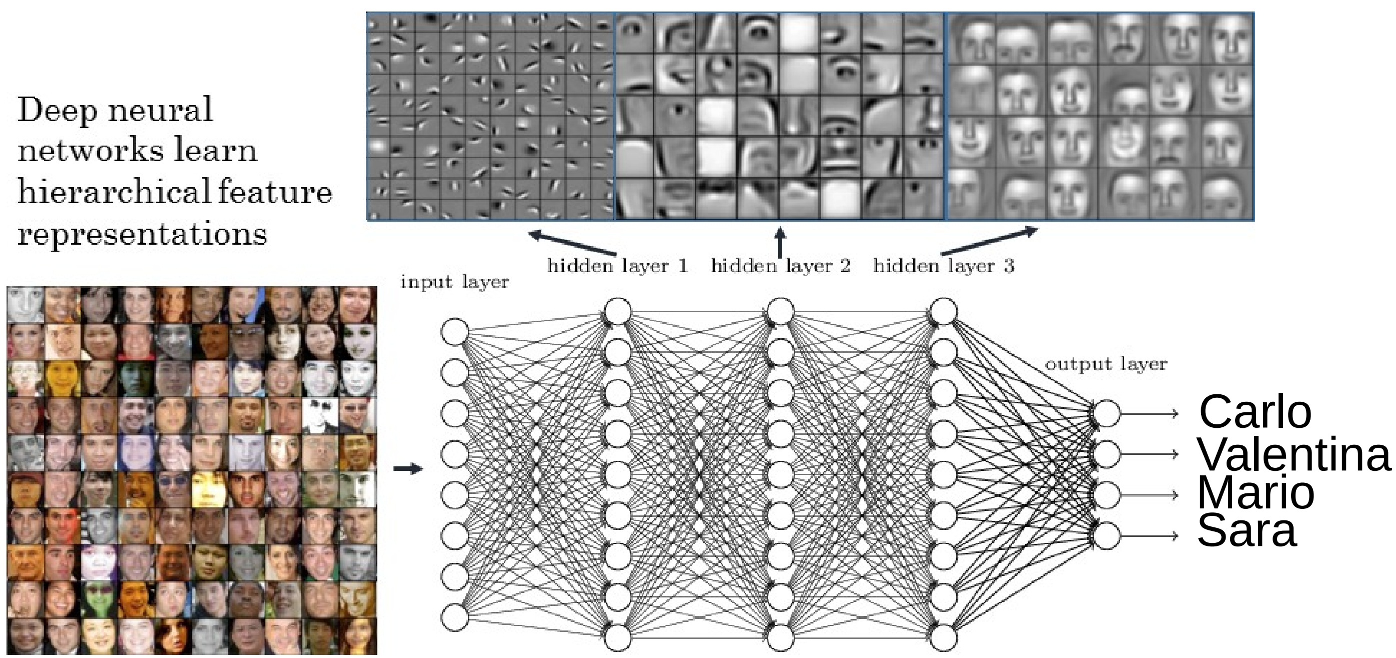
— HUMAN GENETICS FOUNDATION · TORINO —

In collaboration with
Carlo Baldassi
Alessandro Ingrosso
Luca Saglietti
Riccardo Zecchina

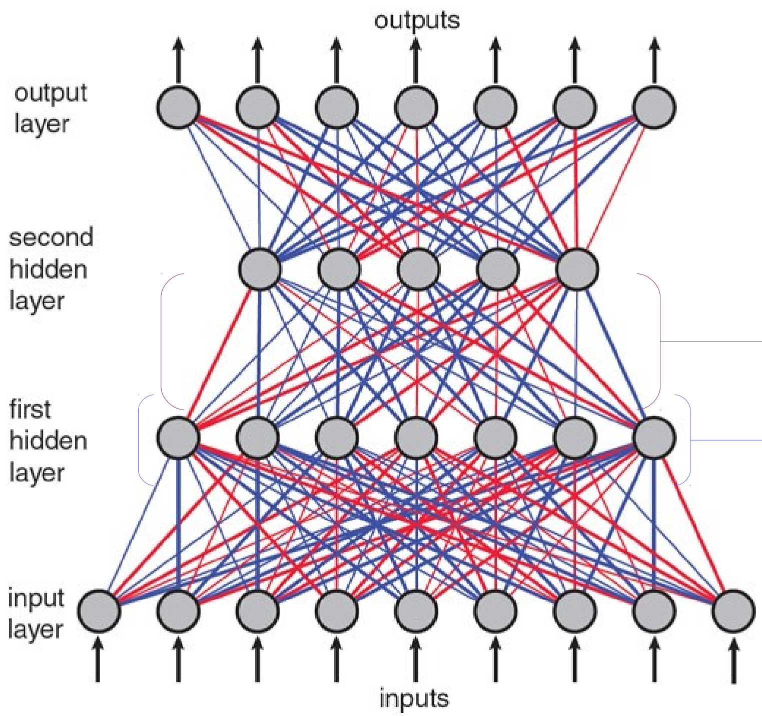
and
Christian Borgs
Jennifer Chayes
(Microsoft Research)

Deep Learning

- In the last decade huge leap forward of machine’s performance in many cognitive tasks thanks to **deep Artificial Neural Networks** (Hinton, LeCun, Bengio, ...)
- Loosely inspired from real neural networks (visual system) → stacks of simple artificial neurons (perceptrons; few basic variants)
- Very versatile (image classification, game playing, speech recognition, emerging applications in physics…)
- Impressive results (super-human in some cases, e.g. AlphaGo)



Deep Neural Networks



Their input-output relation is highly non-linear

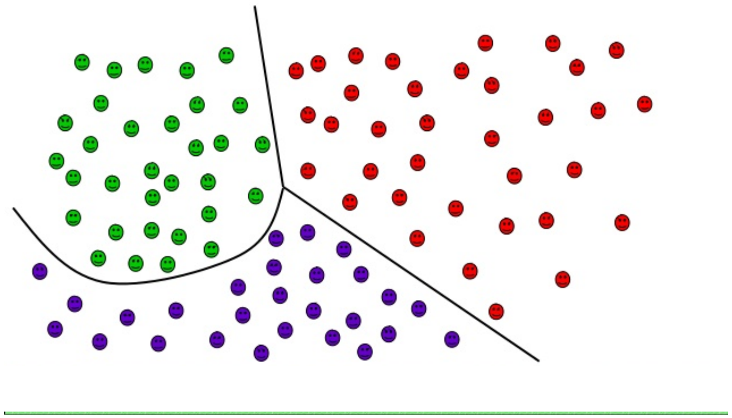
$$f(\mathbf{x};\mathbf{W}) = g_3(\mathbf{W}_3 \cdot g_2(\mathbf{W}_2 \cdot g_1(\mathbf{W}_1 \cdot \mathbf{x})))$$

Synaptic Weights

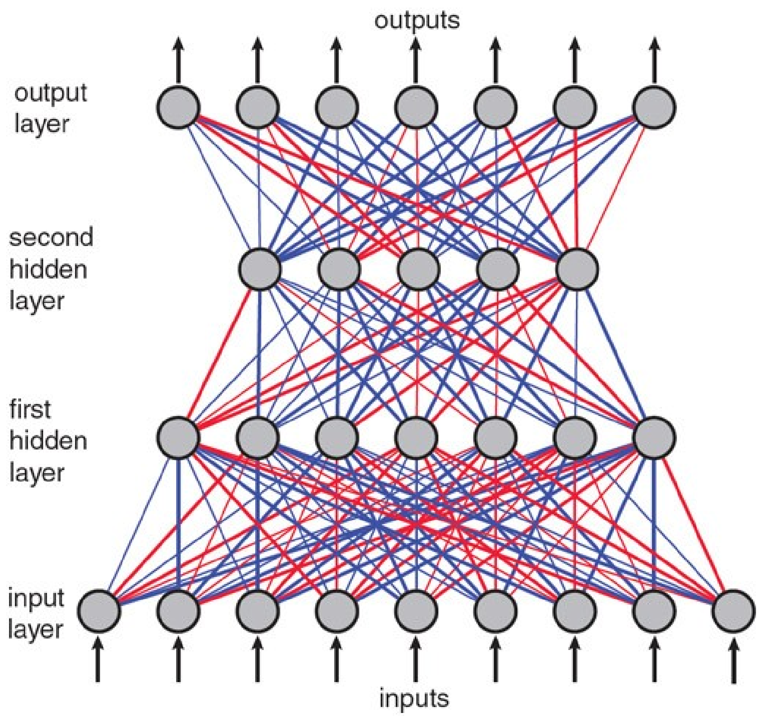
Neurons / Activation functions

Typically ReLU neurons: $g(z) = \max(0, z)$

Multi-class classification



Deep Neural Networks



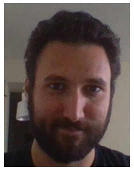
Their input-output relation is highly non-linear

$$f(\mathbf{x};\mathbf{W}) = g_3(\mathbf{W}_3 \cdot g_2(\mathbf{W}_2 \cdot g_1(\mathbf{W}_1 \cdot \mathbf{x})))$$

Supervised learning :

- Training Set

$$\{(y^\mu, \mathbf{x}^\mu)\}_{\mu=1}^{P}$$

e.g. Carlo, 

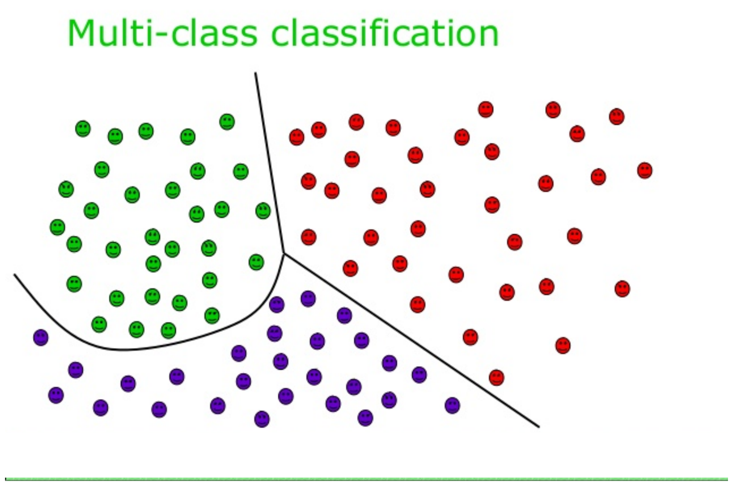
- Loss function (non-convex)**:**

$$\mathcal{L}(\mathbf{W}) = -\sum_\mu \log \mathcal{P}(y^\mu | f(\mathbf{x}^\mu; \mathbf{W}))$$

- Stochastic Gradient Descent

$$\mathbf{W}^{t+1} = \mathbf{W}^t - \eta \frac{\partial \mathcal{L}_t}{\partial \mathbf{W}^t}$$

random minibatch



Robustly achieve low error on the training set (doesn't get stuck in bad local minima) and good generalization properties.

Poor theoretical understanding!

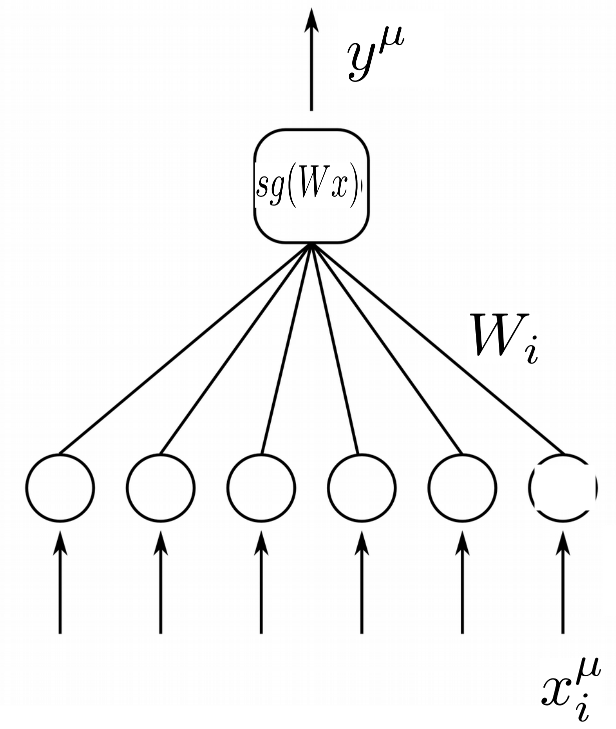
Undeep learning: the perceptron

The perceptron is the building block of multi-layer networks.

We want to learn a binary linear classifier $(y^\mu = \pm 1)$,

i.e. find synaptic weights W such that

$$y^\mu = \mathrm{sign}\left(\sum_{i=1}^{N} W_i\, x_i^\mu\right) \qquad \forall \mu \in \{1, \ldots, P\}$$



For **binary** synapses the perceptron problem is NP-hard and also standard heuristics such as Simulated Annealing take exponential time to solve it.

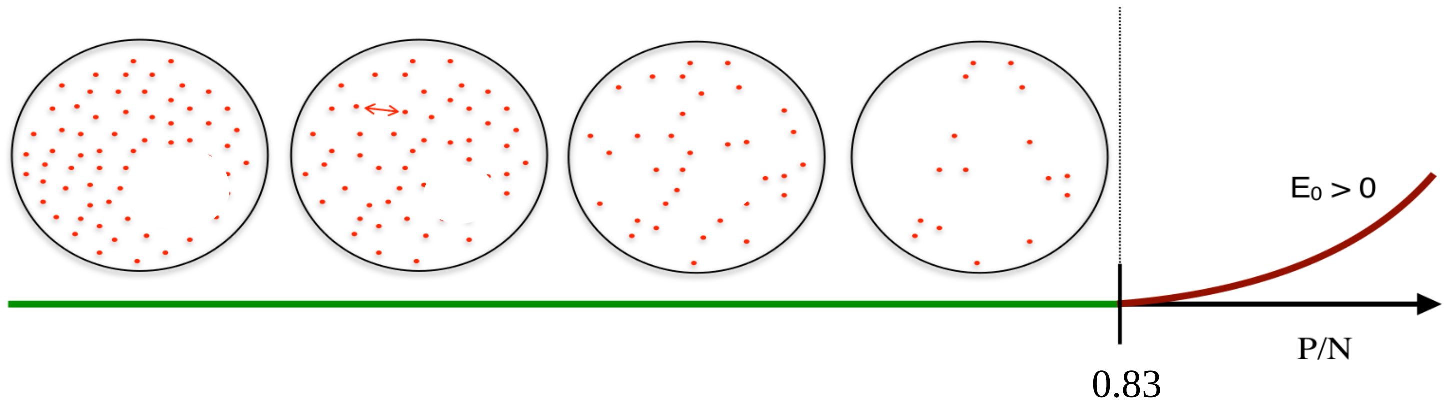
Average case theorethical analysis can be done with Replica and Cavity method from statistical physics.

$$Z = \sum_W e^{-\beta\, \mathcal{H}(W)}$$

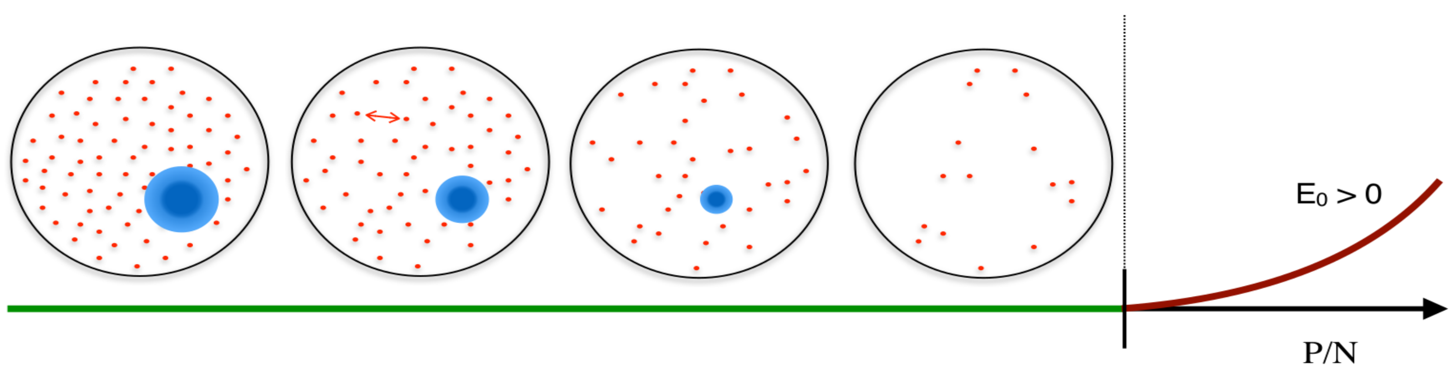
One possible choice for H(W) is the error counting Hamiltonian.

Mostly isolated solutions but...

The origin of the computational hardness is due to the fact that **typical solution are isolated** (Kabashima and Huang ‘14), anologue of deep holes for non-discrete networks.



Still efficient algorithms do exist (Braunstein and Zecchina ‘07) and they can almost reach the theoretical SAT/UNSAT threshold. **The solutions found are not isolated** at all!

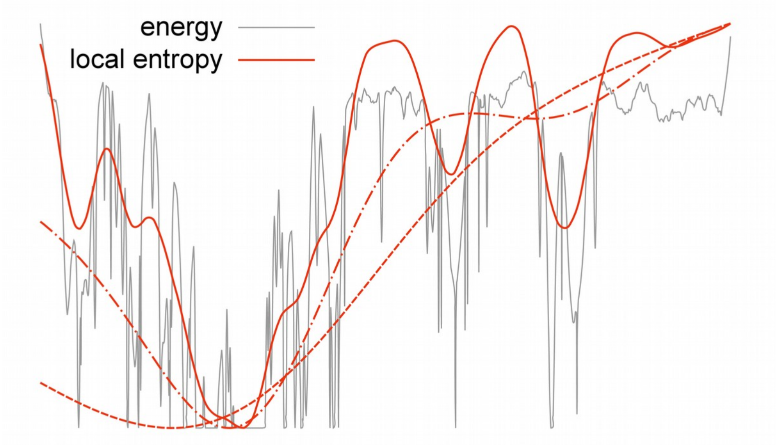


There is a **dense region of solutions,** but it is subdominant in the equilibrium measure (uniform over all solutions).

Local entropy

To uncover the presence of dense regions of solutions we modify the Boltzamnn distribution:

$$Z = \sum_{\tilde{W}} e^{y\, S_{\rm loc}(\gamma, \tilde{W})}$$



smoothing of the landscape

with the **local entropy** [1]

$$S_{\rm loc}(\gamma, \tilde{W}) = \log \sum_{W} e^{-\beta \mathcal{H}(W) - \beta\gamma \|W - \tilde{W}\|^2}$$

Low-lying configurations $\tilde{W}$ surrounded by many others low-lying configurations have more statistical weight than configurations in deep "holes".

The dense region now dominates the measure.

[1] Baldassi, Ingrosso, **Lucibello**, Saglietti, Zecchina (PRL '15)

Local entropy

$$Z = \sum_{\tilde{W}} e^{y\, S_{\rm loc}(\gamma, \tilde{W})}$$

$$S_{\rm loc}(\gamma, \tilde{W}) = \log \sum_{W} e^{-\beta \mathcal{H}(W) - \beta\gamma \|W - \tilde{W}\|^2}$$

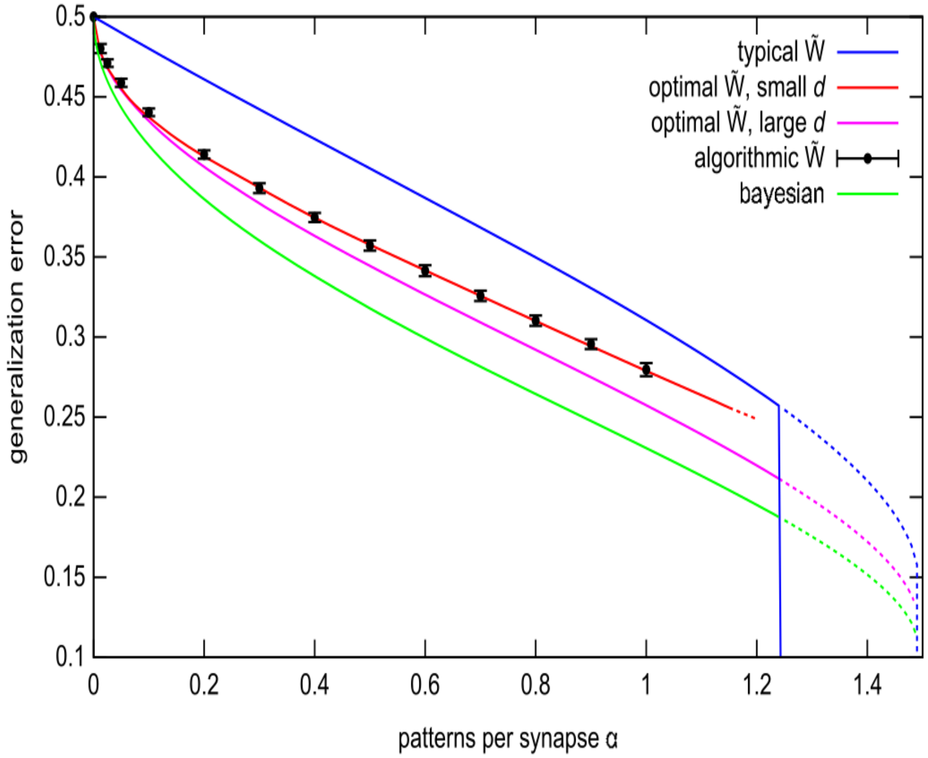
Good conceptual tool, also analitically and algorithmically tractable (with some approximations).

For large β corresponds to the proximal operator.

This new cost function enforces robustness of minimizers.

Gives some theoretical support for good generalization properties of flat minima.

Perceptron T=0 $\alpha = P/N$



[1] Baldassi, Ingrosso, **Lucibello**, Saglietti, Zecchina (PRL ‘15)

Local entropy: algorithms

$$Z = \sum_{\tilde{W}} e^{y\, S_{\mathrm{loc}}(\gamma,\tilde{W})}$$

$$S_{\mathrm{loc}}(\gamma,\tilde{W}) = \log \sum_{W} e^{-\beta\mathcal{H}(W)-\beta\gamma\|W-\tilde{W}\|^2}$$

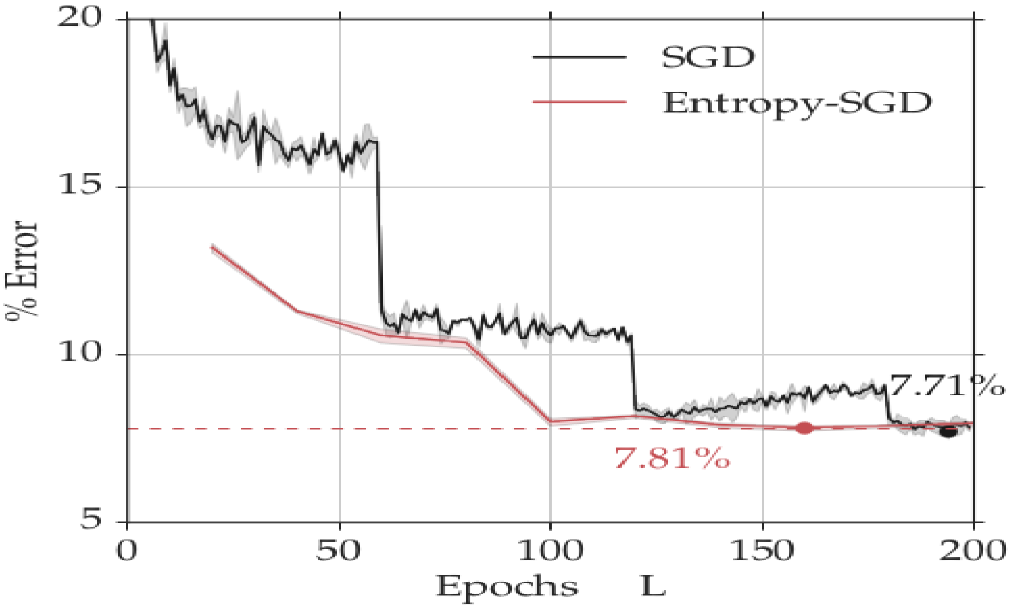
.

For continuous weights we can perform gradient descent approximating the gradient

$$\nabla_{\tilde{W}} S_{\mathrm{loc}}(\gamma,\tilde{W})$$

with a few steps of a Langevin dynamics (Chaudhari et al. '16)

CIFAR-10 test error, 10 layers CNN
(Chaudhari et al. '16)



Local entropy: algorithms

$$Z = \sum_{\tilde{W}} e^{y\, S_{\mathrm{loc}}(\gamma, \tilde{W})}$$

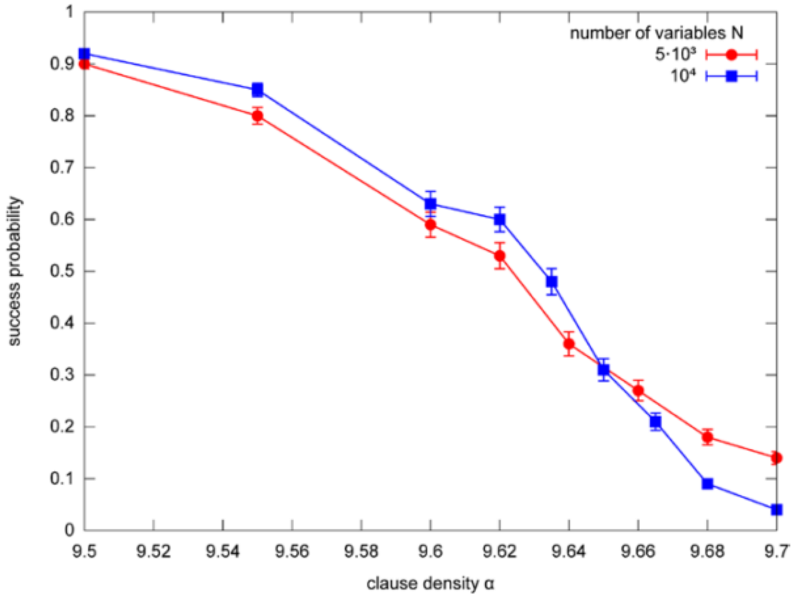
$$S_{\mathrm{loc}}(\gamma, \tilde{W}) = \log \sum_{W} e^{-\beta \mathcal{H}(W) - \beta\gamma \|W - \tilde{W}\|^2}$$

MonteCarlo
+
Belief Propagation to estimate loc. entropy

Success probability in
Random 4SAT Hard Phase

The framework can be applied to any optimization problem.

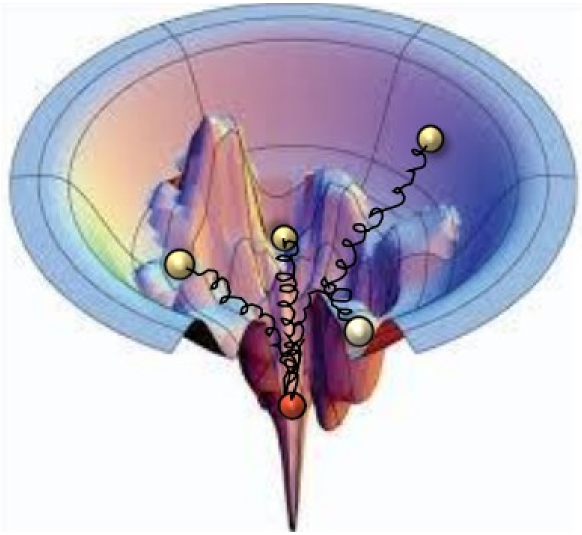
The drawback is the expensive computation of the local entropy.



Replicas: the Robust Ensemble

$$Z = \sum_{\tilde{W}} e^{y\, S_{\mathrm{loc}}(\gamma, \tilde{W})}$$

$$S_{\mathrm{loc}}(\gamma, \tilde{W}) = \log \sum_{W} e^{-\beta \mathcal{H}(W) - \beta\gamma \|W - \tilde{W}\|^2}$$



The local entropy is expensive to compute exactly, therefore has to be approximated. Can we do better?

If we consider **integer y** we can write Z as the partition function of **y+1 coupled systems**!

$$Z = \sum_{\tilde{W}} \sum_{\{W\}_{a=1}^{y}} e^{-\beta \sum_{a=1}^{y} \mathcal{H}(W^a) - \beta\gamma \sum_{a=1}^{y} \|W^a - \tilde{W}\|^2}$$

We call this new ensemble the Robust Ensemble, since it biases the measure towards dense/robust regions.

γ used to tune exploration vs exploitation. Generally increased during training.

[2] Baldassi, Borgs, Chayes, Ingrosso, **Lucibello**, Saglietti, Zecchina (PNAS '16)

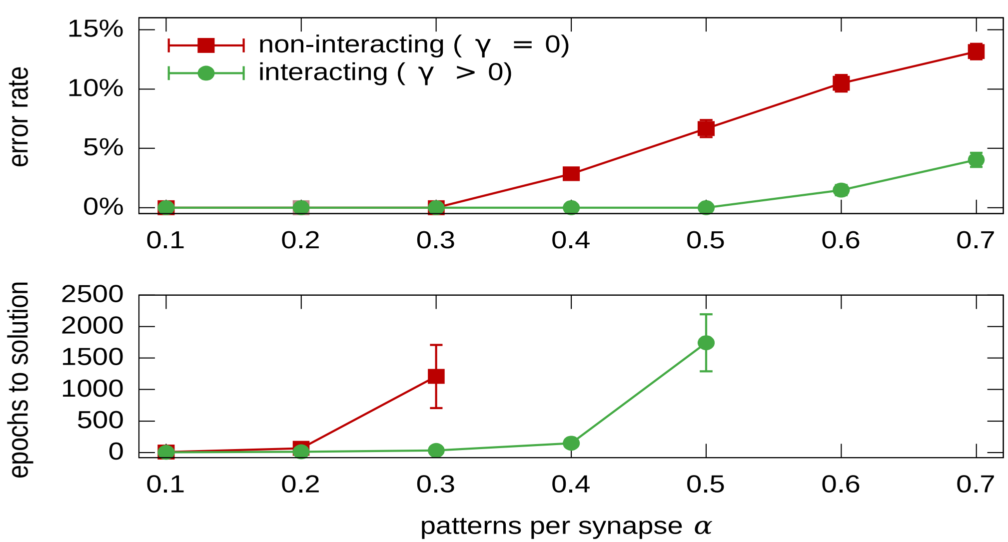
Replicated Algorithms

$$\mathcal{L} = \sum_{a=1}^{y} \mathcal{H}(W^a) + \gamma \sum_{a=1}^{y} \|W^a - \tilde{W}\|^2$$

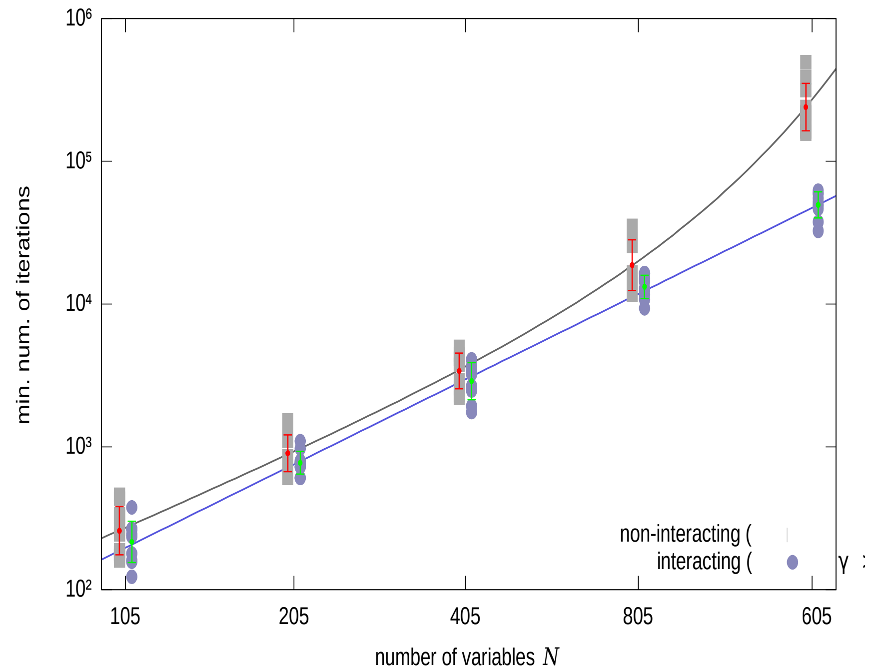
We applied several “replicated” algorithms on 2-layers binary neural networks, with very good results.

It is sufficient to take y = 3,4,5. Here we show the results for continuous and binary 2-layers net.

Replicated Stochastic Gradient Descent



Replicated Simulated Annealing



[2] Baldassi, Borgs, Chayes, Ingrosso, **Lucibello**, Saglietti, Zecchina (PNAS ‘16)

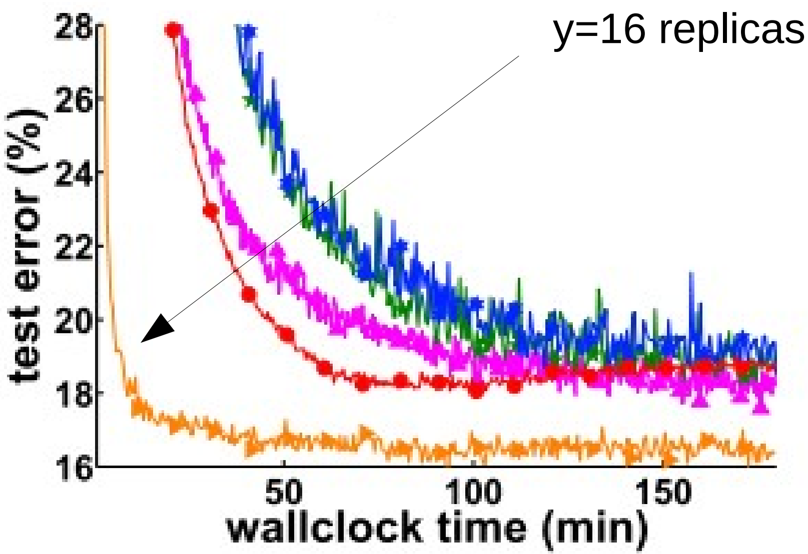
Replicated Algorithms

$$\mathcal{L} = \sum_{a=1}^{y} \mathcal{H}(W^a) + \gamma \sum_{a=1}^{y} \|W^a - \tilde{W}\|^2$$

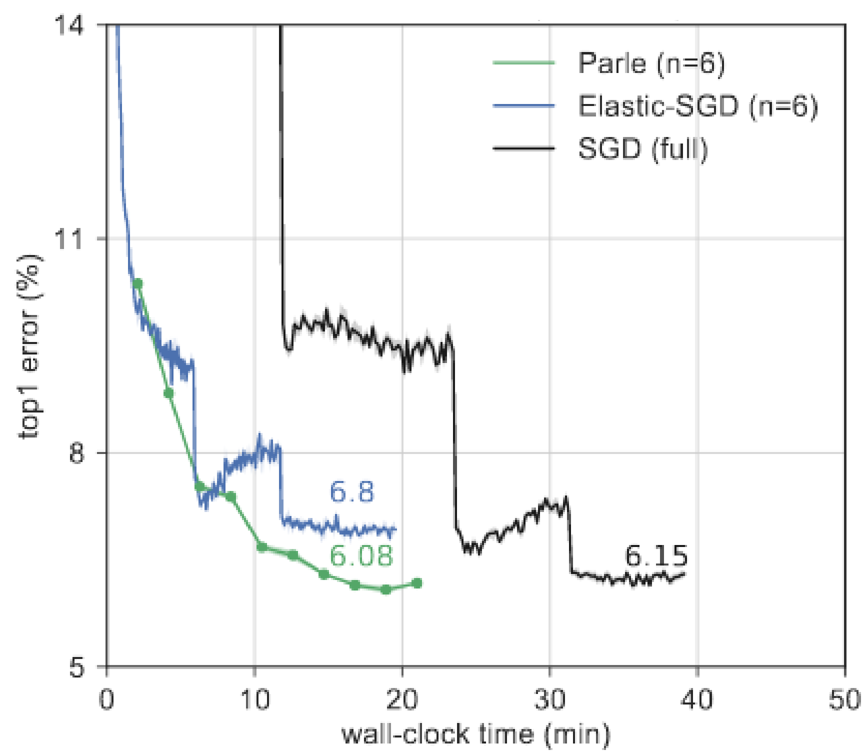
Using replicas is also a way for distributing computation.

One can also partition the training set among the replicas and still get good performance!

CIFAR-10 test error, 7 layers CNN.
(Zhang, LeCun et al ‘15.)



CIFAR-10 test error, 7 layers All-CNN.
(Chaudhari et al ‘17.)



Conclusions

- Deep networks are deeply uncharted. We provided some conceptual tools (**local entropy**) to characterize their loss landscape.
- Some simple algorithms based on **replication** and **coupling** seek comparatively flatter minima (higher local entropy).
- Our findings point towards the relevance of flat minima for robust learning and good generalization. As a by-product this algorithms allow for a high-degree of parallelized computation.
- “Replicated” methods for optimization problems besides learning should be investigated.

Thanks!

[1] Baldassi, Ingrosso, **Lucibello**, Saglietti, Zecchina (PRL ‘15)

[2] Baldassi, Borgs, Chayes, Ingrosso, **Lucibello**, Saglietti, Zecchina (PNAS ‘16)