SnapStar Algorithm: a new way to ensemble Neural Networks

Sergey Zinchenko[1], Dmitry Lishudi[2]

[1] NSU zinch.s.e@gmail.com, [2] HSE dlishudi@hse.ru

Introduction We have a $S_n = (\mathbf{X}_i, Y_i)_{i=1}^n$ sample of i.i.d. input-output pairs $(\mathbf{X}_i, Y_i) \in \mathcal{X} \times \mathcal{Y}$ distributed according to some unknown distribution $\mathcal{P}$. We also chose a certain family of predictors $\mathcal{F}$. Our goal is to build a new predictor $\widehat{f}$ (which may not lie in $\mathcal{F}$) minimizing the excess risk $\mathcal{E}$, but in practice we can only calculate the empirical risk r (which depends on the sample S_n) :

$$\mathcal{E}(\widehat{g}) := \mathbb{E}(\widehat{g} - Y)^2 - \inf_{f \in \mathcal{F}} \mathbb{E}(f - Y)^2, \quad r(\widehat{g}) = \frac{1}{n}\sum_{i=1}^{n}(\widehat{g}(\mathbf{X}_i) - Y_i)^2. \qquad (1)$$

To solve this problem, we propose the following $Star_d$ procedure:

1) Get d empirical risk minimizers $\{\widehat{g}_i\}_{i=1}^d$ using snapshot technique [2].
2) Find empirical risk minimizer $\widehat{f}$ on set $Star_d(\widehat{g}_1 \ldots \widehat{g}_d)$:

$$Star_d(\widehat{g}_1 \ldots \widehat{g}_d) = \bigcup_{f \in \mathcal{F}} \mathrm{Conv}(\widehat{g}_1 \ldots \widehat{g}_d, f)$$

The model built in this way combines **the fast order** of the Audibert star procedure [1], **the power of the ensemble** of models, and **the budget construction** of the snapshot technique. We also take into account that minimization is performed inaccurately in practice. Errors from the first and second steps we denote as Δ_1 and Δ_2 respectively.

Theoretical results We focused on the class of sparse fully connected neural networks $\mathcal{F}(L, \mathbf{p}, s)$ defined in [4]. Continuing the technique of Liang et al. [3] we have obtained the fast order for excess risk both in the sense of expectation and in the sense of deviation.

Definition 1 (Lower Isometry Bound) *Class $\mathcal{F}$ satisfies the lower isometry bound with some parameters $0 < \eta < 1$ and $0 < \delta < 1$ if*

$$\mathbb{P}\left(\inf_{f \in \mathcal{F}\setminus\{0\}} \frac{1}{n}\sum_{i=1}^{n}\frac{f^2(\mathbf{X}_i)}{\mathbb{E} f^2} \geq 1 - \eta\right) \geq 1 - \delta$$

for all $n \geq n_0(\mathcal{F}, \delta, \eta)$, where $n_0(\mathcal{F}, \delta, \eta)$ depends on the complexity of the class of functions $\mathcal{F}$.

$$Hull_d(\mathcal{F}) := \left\{ \sum_{i=1}^{d} \lambda_i (g_i - f) \,\middle|\, \lambda_i \in [0,1]; \ \sum_{i=1}^{d}\lambda_i \leq 1; \ f, g_1 \ldots g_d \in \mathcal{F} \right\}, \qquad (2)$$

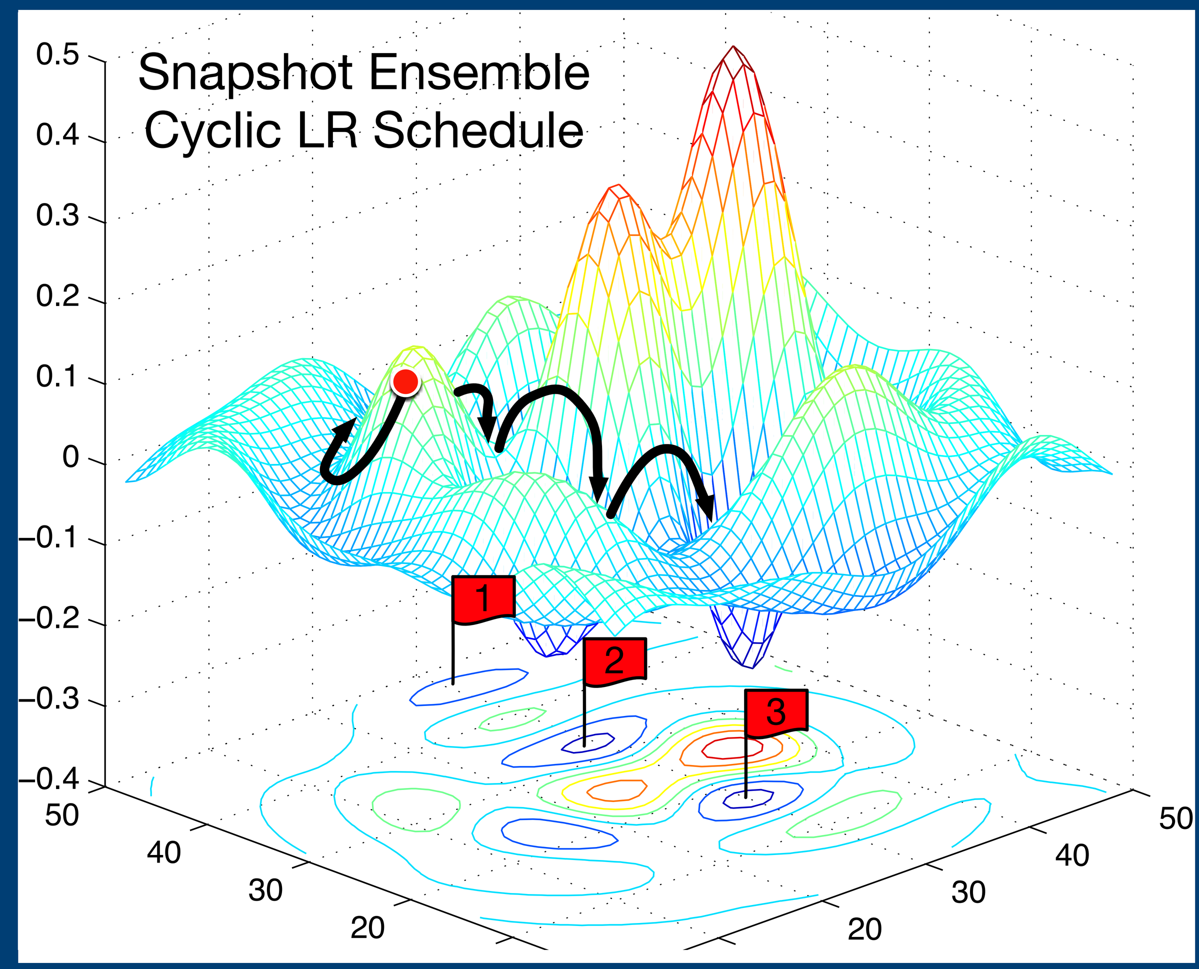
Theorem 2 *Let $\xi_i = Y_i - f^*(\mathbf{X}_i)$ and $f^* := \arg\min_{f \in \mathcal{F}} \mathbb{E}(f(\mathbf{X}) - Y)^2$. If for $Hull_d$ the bound 1 holds with $\eta_{lib} = \frac{1}{144}$ and some $\delta_{lib} < 1$ then there exist constant $C_2 = C_2(K, M, A, B)$ and absolute constants $\tilde{c}, c', c$ such that*

$$\mathbb{P}\left(\mathcal{E}(\widehat{f}) > C_2\left[\frac{d \log n/\delta}{n} + \Delta_1 + \Delta_2\right]\right) \leq 4(\delta_{lib} + \delta)$$

as long as $n > \frac{16(1-c')^2 A}{c'^2} \vee n_0(\mathcal{H}, \delta_{lib}, c/4)$, where $B := \sup_{\mathbf{X},Y} \mathbb{E}\,\xi^4$ and

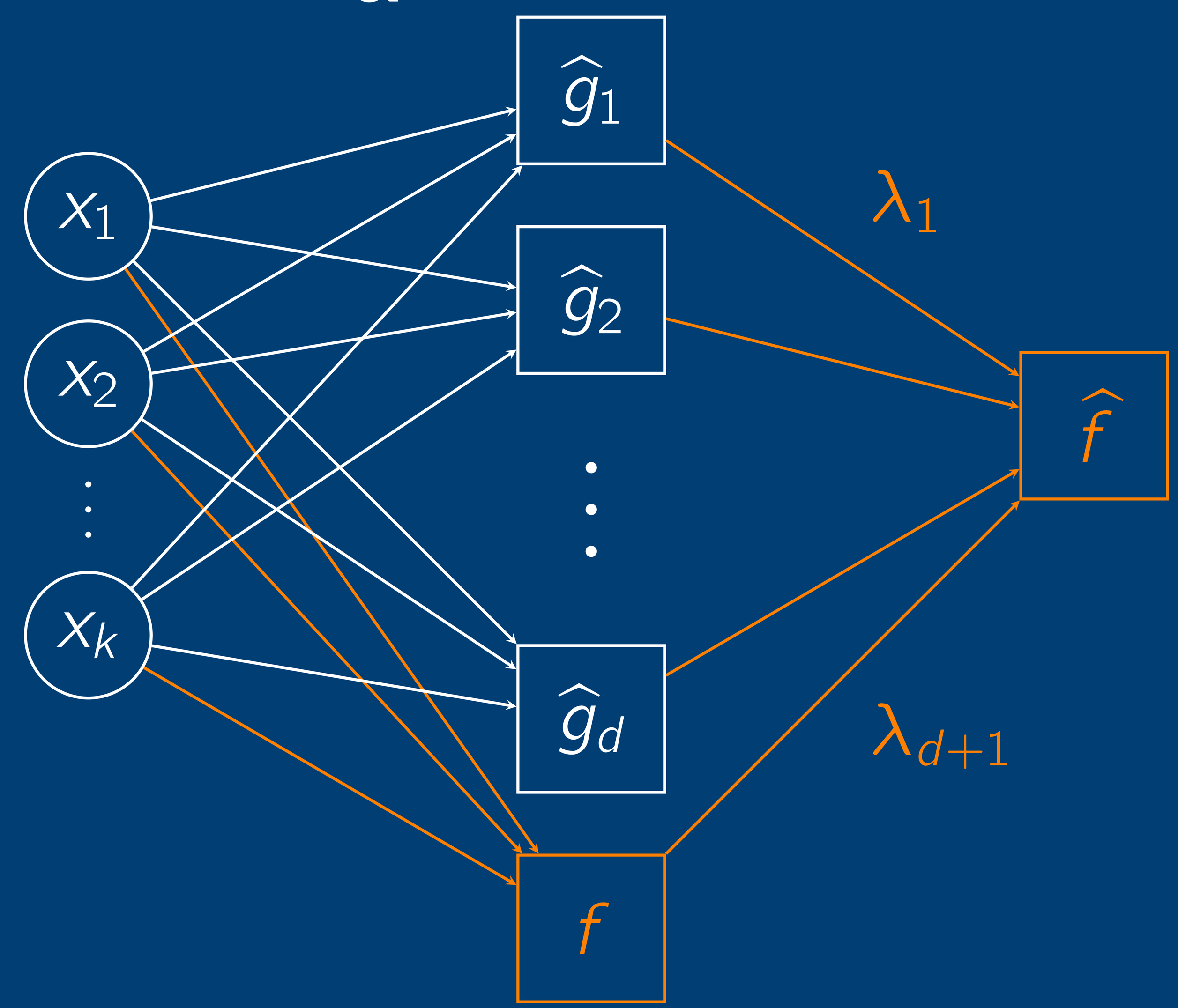
$$A := \sup_{h \in \mathcal{H}} \frac{\mathbb{E}\, h^4}{(\mathbb{E}\, h^2)^2}, \quad K := \left(\sqrt{\sum_{i=1}^{n}\xi_i^2/n} + 2\tilde{c}\right), \quad M := \sup_{h \in \mathcal{H}\setminus\{0\}} \frac{\sum_{i=1}^{n} h(\mathbf{X}_i)^2 \xi_i^2}{\tilde{c}\sum_{i=1}^{n} h(\mathbf{X}_i)^2}.$$

Snapshot Technique



The picture is taken from paper [2]. The method consists in changing lr cyclically and getting into several local optima during the entire training, the weights of which are saved for further construction of the ensemble.

$Star_d$ procedure



Using the snapshot technique, we consecutively get d models. Optimization on set $Star_d$ is performed by adding one more neural network and optimizing its parameters along with convex weights $\lambda_1 \ldots \lambda_{d+1}$.

Arxiv GitHub

Theorem 3 *Let $\widehat{f}$ is $Star_d$ estimator for $\mathcal{F} = \mathcal{F}(L, \mathbf{p}, s)$. The following expectation bound on excess loss holds:*

$$\mathbb{E}\,\mathcal{E}(\widehat{f}) \leq C_3\left(\frac{d \log n}{n} + \Delta_1 + \Delta_2\right),$$

where C_3 depends only on the complexity of the class of neural networks $\mathcal{F}$.

Experiments Description of competitors: training one large neural network of $d+1$ blocks (**Big NN**), learning $d+1$ blocks independently and averaging (**Ensemble**), learning blocks sequentially using the snapshot technique with subsequent averaging (**Snap Ensemble**).

Name	d	MSE	MAE	R^2
Snap Star	5	**10.881±0.575**	**2.229**	**0.869**
Snap Ensemble	5	11.862±0.616	2.306	0.858
Ensemble	5	12.568±0.878	2.399	0.849
Big NN	5	12.068±0.860	2.411	0.855
Snap Star	4	**11.276±0.582**	**2.269**	**0.865**
Snap Ensemble	4	11.819±0.341	2.316	0.858
Ensemble	4	12.059±0.614	2.365	0.855
Big NN	4	12.556±0.904	2.383	0.849

Table 1: Boston Housing Dataset (30 epochs)

Name	d	accuracy	entropy
Snap Star	3	**0.900±0.002**	**0.284±0.008**
Snap Ensemble	3	0.897±0.003	0.290±0.009
Ensemble	3	0.887±0.001	0.310±0.005
Big NN	3	0.890±0.010	0.299±0.022
Snap Star	2	**0.894±0.007**	**0.294±0.020**
Snap Ensemble	2	0.891±0.006	0.302±0.021
Ensemble	2	0.886±0.004	0.313±0.008
Big NN	2	0.892±0.003	0.304±0.007

Table 2: Fashion Mnist Dataset (5 epochs)

Conclusion We have proved the optimality and stability of $Star_d$ procedure for MSE minimization in a class of sparse neural networks. In practice, we were convinced of its performance for other tasks and types of neural networks.

References

[1] J.-Y. Audibert, *Progressive mixture rules are deviation suboptimal*, NeurIPS, (2007).

[2] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, *Snapshot ensembles: Train 1, get m for free*, 2017.

[3] T. Liang, A. Rakhlin, and K. Sridharan, *Learning with square loss: Localization through offset rademacher complexity*, in Conference on Learning Theory, PMLR, 2015, pp. 1260–1285.

[4] J. Schmidt-Hieber, *Nonparametric regression using deep neural networks with relu activation function*, The Annals of Statistics, 48 (2020), pp. 1875–1897.