Generalization bounds via estimation of tangent sensitivity tensors of ReLU networks and of the H2 norm of neural ODEs and SSMs

Bálint Daróczy
joint works with Dániel Rácz, Mihály Petreczky, Martin Gonzalez, Julien Hendrickx

Institute for Computer Science and Control HUN-REN SZTAKI, Budapest, Hungary

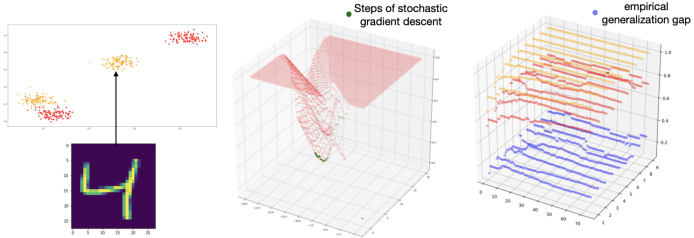
GPU Day 2024, Budapest 05/31/2024

Machine learning and GPU computing

- In early 2010 inference times decreased thus real time inference of fixed models became possible
- In late 2010, early 2020 emerging umber of models under the RL or continual learning paradigm, learning and inference are parallel tasks
- Mid 2020 first time we may compute quality measures besides first order measures (loss, accuracy etc.) about the models on the fly and choosing the proper models based on their true error rate

To identify higher order quality measures let us consider the topological spaces related to learning, focusing on Deep Structured State Space Models (SSM), Neural Ordinary Differential Equations (ODE) and feedforward Rectified Linear Unit (ReLU) networks

Topological spaces related to learning

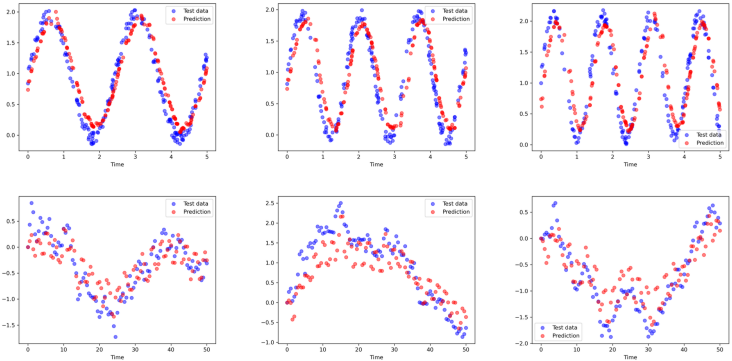


a) Observations b) Error surface c) Hyperparameters

	Observations	Error surface	Hyperparameters
Learning from pairwise comparisons	binary outcomes	e. g. L2, sine	?
Feedforward ReLU nets	data-label pairs?	e.g. Cross-entropy	network structure
Deep SSMs	sequences?	e.g. Cross-entropy	block structure
Neural ODEs	sequences?	?	state dim.?

SSMs and neural ODEs

- New class of neural structures based on a continuous structure usually linear dynamical system combined with nonlinearities, no exact depth!
- SSMs are state-of-the-art for long range benchmarks even beating out transformers
- Neural ODEs are elegant models with adaptive depth close to SOTA on ImageNet



Preliminaries

- Problem: The learning objective is to find $\theta \in \Theta$ such that the true risk $\mathcal{L}(f_\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[l(f_\theta(x), y]$ is small as possible.
  - f_θ: hypothesis e.g. feedforward ReLU NN with L layers, $\theta \in \mathbb{R}^{|\theta|}$.
  - $l(y', y)$: loss function

Preliminaries

- Problem: The learning objective is to find $\theta \in \Theta$ such that the true risk $\mathcal{L}(f_\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[l(f_\theta(x), y]$ is small as possible.
  - f_θ: hypothesis e.g. feedforward ReLU NN with L layers, $\theta \in \mathbb{R}^{|\theta|}$.
  - $l(y', y)$: loss function
- Since we do not know $\mathcal{D}$, we approximate the true risk by the empirical risk $\mathcal{L}_X(f_\theta) = \frac{1}{T}\sum_{i=1}^{T} l(f_\theta(x_i), y_i]$
  - new objective: $\min_{f_\theta \in \mathcal{F}} \mathcal{L}_X(f_\theta) + R(f_\theta)$
  - $X = \{x_1, ..., x_T\} \subset \mathcal{R}^{d_{in}}$: observations
  - $R(f_\theta)$: reg. term, but why we need this term?
  - Question: can we trust our new θ?

Generalization gap vs. identification

How (or what) to measure?

- The divergence between the optimal and the estimated parameters? The surface of a typical ML model is not ideal to assume anything.

Generalization gap vs. identification

How (or what) to measure?

- The divergence between the optimal and the estimated parameters? The surface of a typical ML model is not ideal to assume anything.
- The difference between the true and the estimated loss? **Generalization gap:** $\mathcal{L}(f_\theta) - \mathcal{L}_X(f_\theta)$
  Note, X is not necessary the set we used for training.

Generalization gap vs. identification

How (or what) to measure?

- The divergence between the optimal and the estimated parameters? The surface of a typical ML model is not ideal to assume anything.
- The difference between the true and the estimated loss? **Generalization gap:** $\mathcal{L}(f_\theta) - \mathcal{L}_X(f_\theta)$
  Note, X is not necessary the set we used for training.
- Our main goal is to derive a **Probably Approximately Correct (PAC)** bound

$$Pr(\forall_{f_\theta \in \mathcal{F}} \mathcal{L}(f_\theta) - \mathcal{L}_X(f_\theta) > R(\mathcal{F}, X, \delta)) \leq 1 - \delta$$

  If the bound is low we may trust the model we got from the training that it will perform similarly on unknown sample.

Generalization gap vs. identification

How (or what) to measure?

- The divergence between the optimal and the estimated parameters? The surface of a typical ML model is not ideal to assume anything.
- The difference between the true and the estimated loss? **Generalization gap:** $\mathcal{L}(f_\theta) - \mathcal{L}_X(f_\theta)$
  Note, X is not necessary the set we used for training.
- Our main goal is to derive a **Probably Approximately Correct (PAC)** bound

$$Pr(\forall_{f_\theta \in \mathcal{F}} \mathcal{L}(f_\theta) - \mathcal{L}_X(f_\theta) > R(\mathcal{F}, X, \delta)) \leq 1 - \delta$$

  If the bound is low we may trust the model we got from the training that it will perform similarly on unknown sample.
- Question: what is $R(\mathcal{F}, T, \delta)$?

Generalization gap vs. identification

How (or what) to measure?

- The divergence between the optimal and the estimated parameters? The surface of a typical ML model is not ideal to assume anything.
- The difference between the true and the estimated loss? **Generalization gap:** $\mathcal{L}(f_\theta) - \mathcal{L}_X(f_\theta)$
  Note, X is not necessary the set we used for training.
- Our main goal is to derive a **Probably Approximately Correct (PAC)** bound

$$Pr(\forall_{f_\theta \in \mathcal{F}} \mathcal{L}(f_\theta) - \mathcal{L}_X(f_\theta) > R(\mathcal{F}, X, \delta)) \leq 1 - \delta$$

  If the bound is low we may trust the model we got from the training that it will perform similarly on unknown sample.
- Question: what is $R(\mathcal{F}, T, \delta)$?
- Note, PAC-Bayes bounds: we know some data and/or algorithm dependent prior on the hypothesis set

Rademacher complexity of neural ODEs and deep SSMs

- Typically for a hypothesis set: $R(\mathcal{F}, X, \delta) = O(\mathcal{R}^X(\mathcal{F}) + \frac{\log(4/\delta)}{\sqrt{T}})$ where $\mathcal{R}^X(\mathcal{F})$ is the empirical Rademacher complexity of $\mathcal{F}$ on the set X.

Rademacher complexity of neural ODEs and deep SSMs

- Typically for a hypothesis set: $R(\mathcal{F}, X, \delta) = O(\mathcal{R}^X(\mathcal{F}) + \frac{\log(4/\delta)}{\sqrt{T}})$ where $\mathcal{R}^X(\mathcal{F})$ is the empirical Rademacher complexity of $\mathcal{F}$ on the set X.
- Some recent results:
  - time independent bounded neural ODEs [Marion, 2023]: $\mathcal{R}^X(\mathcal{F}) \leq O(\|\theta\|_{1,\infty}/\sqrt{T})$
  - neural ODEs under stability [Racz et al., 2023]: $\mathcal{R}^X(\mathcal{F}) \leq O(c_\Sigma/\sqrt{T})$ where c_Σ is the upper bound of the λ-weighted $H2$ norm of the dynamical system.

Rademacher complexity of neural ODEs and deep SSMs

- Typically for a hypothesis set: $R(\mathcal{F}, X, \delta) = O(\mathcal{R}^X(\mathcal{F}) + \frac{\log(4/\delta)}{\sqrt{T}})$ where $\mathcal{R}^X(\mathcal{F})$ is the empirical Rademacher complexity of $\mathcal{F}$ on the set X.
- Some recent results:
  - time independent bounded neural ODEs [Marion, 2023]: $\mathcal{R}^X(\mathcal{F}) \leq O(\|\theta\|_{1,\infty}/\sqrt{T})$
  - neural ODEs under stability [Racz et al., 2023]: $\mathcal{R}^X(\mathcal{F}) \leq O(c_\Sigma/\sqrt{T})$ where c_Σ is the upper bound of the λ-weighted $H2$ norm of the dynamical system.
  - some deep SSMs (S4,S5,LRU) w/wo GLU activations under stability [Racz et al., 2024]:
    $\mathcal{R}^X(\mathcal{F}) \leq O((\mu + c)/\sqrt{T})$ where $\mu \leq K_{enc} K_{dec} (\mu_{g_1} c_{H2} + \alpha_1) \prod_{i=2}^{L} (\mu_{g_i} c_{H1} + \alpha_i)$,
    $c \leq K_{dec} \sum_{j=1}^{L} \left[\prod_{i=j+1}^{L} (\mu_{g_i} c_{H1} + \alpha_i) \right] c_{g_j}$ with c_{H2} and c_{H1} are the upper bounds of the $H2$ and $H1$ norm of the SSM.

Rademacher complexity of neural ODEs and deep SSMs

- Typically for a hypothesis set: $R(\mathcal{F}, X, \delta) = O(\mathcal{R}^X(\mathcal{F}) + \frac{\log(4/\delta)}{\sqrt{T}})$ where $\mathcal{R}^X(\mathcal{F})$ is the empirical Rademacher complexity of $\mathcal{F}$ on the set X.
- Some recent results:
  - time independent bounded neural ODEs [Marion, 2023]: $\mathcal{R}^X(\mathcal{F}) \leq O(\|\theta\|_{1,\infty}/\sqrt{T})$
  - neural ODEs under stability [Racz et al., 2023]: $\mathcal{R}^X(\mathcal{F}) \leq O(c_\Sigma/\sqrt{T})$ where c_Σ is the upper bound of the λ-weighted $H2$ norm of the dynamical system.
  - some deep SSMs (S4,S5,LRU) w/wo GLU activations under stability [Racz et al., 2024]:
    $\mathcal{R}^X(\mathcal{F}) \leq O((\mu + c)/\sqrt{T})$ where $\mu \leq K_{enc} K_{dec} (\mu_{g_1} c_{H2} + \alpha_1) \prod_{i=2}^{L} (\mu_{g_i} c_{H1} + \alpha_i)$,
    $c \leq K_{dec} \sum_{j=1}^{L} \left[\prod_{i=j+1}^{L} (\mu_{g_i} c_{H1} + \alpha_i) \right] c_{g_j}$ with c_{H2} and c_{H1} are the upper bounds of the $H2$ and $H1$ norm of the SSM.
- The **norm of the dynamical system** is a key factor to derive meaningful PAC bounds. Under **stability** conditions these norms exist.

Rademacher complexity of ReLU networks

- ff ReLU net with L layers [Truong, 2022]: $\mathcal{R}^X(\mathcal{F}) \leq O((\Pi_{i=1}^{L}\|W_i\|_\infty)/\sqrt{T})$

Rademacher complexity of ReLU networks

- ff ReLU net with L layers [Truong, 2022]: $\mathcal{R}^X(\mathcal{F}) \leq O((\Pi_{i=1}^{L}\|W_i\|_\infty)/\sqrt{T})$
- Under some mild conditions for learning [Racz et al., 2023]: $R^X(\mathcal{F}) \leq O(\sup_{x\in X;\theta\in\mathcal{B}(\theta_0)}\|Sens_{tan}(x;\theta)\|_F)$ where X is the training set, θ_0 is the initial parameter and $Sens_{tan}(x;\theta) = \frac{\nabla_\theta f(x;\theta)|_\theta}{\partial x}\Big|_x = \frac{\partial^2 f(x;\theta)}{\partial\theta\partial x}\Big|_{\theta,x}$

Rademacher complexity of ReLU networks

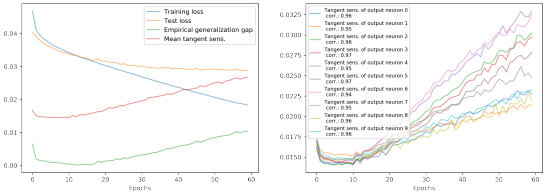
- ff ReLU net with L layers [Truong, 2022]: $\mathcal{R}^X(\mathcal{F}) \leq O((\Pi_{i=1}^{L}\|W_i\|_\infty)/\sqrt{T})$
- Under some mild conditions for learning [Racz et al., 2023]: $R^X(\mathcal{F}) \leq O(\sup_{x\in X;\theta\in\mathcal{B}(\theta_0)} \|Sens_{tan}(x;\theta)\|_F)$ where X is the training set, θ_0 is the initial parameter and $Sens_{tan}(x;\theta) = \frac{\nabla_\theta f(x;\theta)|_\theta}{\partial x}\Big|_x = \frac{\partial^2 f(x;\theta)}{\partial\theta\partial x}\Big|_{\theta,x}$
- Again **norm of the weight matrices** or the **norm of tangent sensitivity** are the key factors delivering generalization bounds.

Rademacher complexity of ReLU networks

- ff ReLU net with L layers [Truong, 2022]: $\mathcal{R}^X(\mathcal{F}) \leq O((\Pi_{i=1}^{L}\|W_i\|_\infty)/\sqrt{T})$
- Under some mild conditions for learning [Racz et al., 2023]: $R^X(\mathcal{F}) \leq O(\sup_{x\in X;\theta\in\mathcal{B}(\theta_0)} \|Sens_{tan}(x;\theta)\|_F)$ where X is the training set, θ_0 is the initial parameter and $Sens_{tan}(x;\theta) = \frac{\nabla_\theta f(x;\theta)|_\theta}{\partial x}\Big|_x = \frac{\partial^2 f(x;\theta)}{\partial\theta\partial x}\Big|_{\theta,x}$
- Again **norm of the weight matrices** or the **norm of tangent sensitivity** are the key factors delivering generalization bounds.
- Question: are these bounds meaningful?

Correlation and nonvacuous bounds for MNIST or CIFAR

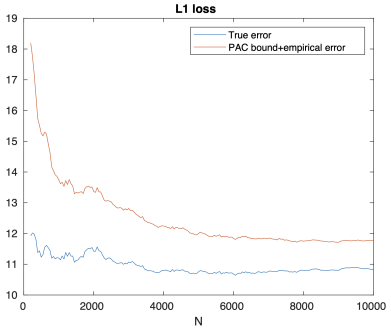
- First meaningful bound for the MNIST [Dziugaite & Roy, 2017]: 0.165 with high prob. Recently 0.022. These are data-dependent PAC-Bayes bounds.
- MNIST with CNN [Truong, 2024]: 0.162 PAC bound with and improved Talagrand-Ledoux concentration bound.
- There exists nonvacuous bounds for CIFAR and even for ImageNet too.
- Tangent sensitivity correlates with the gap



Computation of the norms I.

Still the question remains, how to compute efficiently these norms or just to estimate their upper bounds?

- For neural ODEs and SSMs: still an open problem to even estimate the norms outside of some toy examples is hard, typical solutions are quadratic the least



Computation of the norms II.

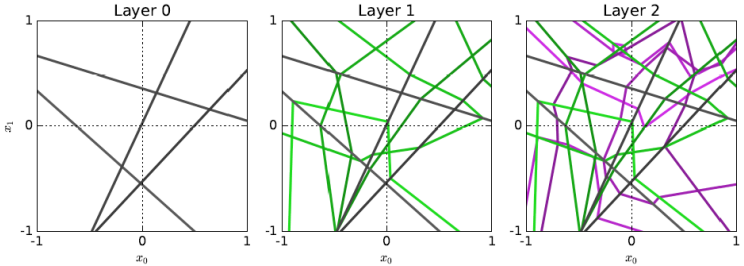
- For ReLU nets: computation of the trivial upper bound is $O(|\Theta|)$
- For ReLU nets: the one based on tangent sensitivity
  - naive implementation: back propagate per sample to get the gradients and differentiate afterwards and collect, lots of issues: memory, computational time etc.

Computation of the norms II.

- For ReLU nets: computation of the trivial upper bound is $O(|\Theta|)$
- For ReLU nets: the one based on tangent sensitivity
  - naive implementation: back propagate per sample to get the gradients and differentiate afterwards and collect, lots of issues: memory, computational time etc.
  - Notice, if we consider fully connected ReLU networks we may take advantage of their sparsity properties, but how?

Activation regions

- a neuron is active for a given input x, if its preactivation at x is positive.
- for a given x the activity of each neuron defines an *activation pattern*, which can be described by a $\{0,1\}^{\#neurons}$ vector
- inputs ($x \in \mathbb{R}^d$) having the same activation patterns define an *activation region* in the input space



Linear regions

- A deep ReLU network is a piecewise linear function.

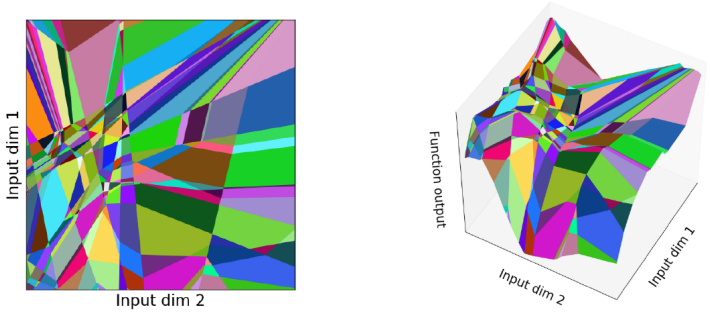


Figure: [Hanin & Rolnick, 2019]

- The more linear regions we have the more complex function we can approximate.

Linear regions

- A deep ReLU network is a piecewise linear function.

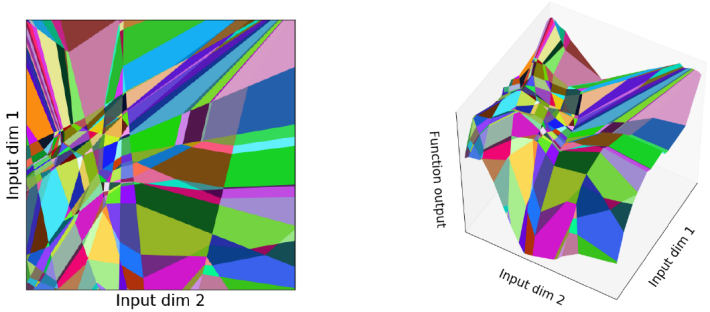


Figure: [Hanin & Rolnick, 2019]

- The more linear regions we have the more complex function we can approximate.
- Number of linear regions $\leq$ Number of (convex!) activation regions

Expressive power of ReLU networks

- number of linear regions a **deep** ReLU networks can realize is exponential in **depth** [Pascanu et al., 2013]
- space folding [Montufar et al., 2014]

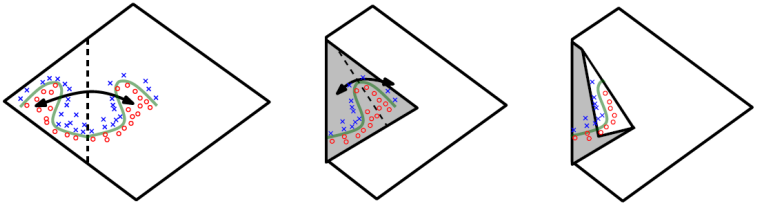


Figure: [Montufar et al., 2014]

Are these extreme configurations stable?

Expressive power

in practice deep networks realize much less regions, even in case of memorization

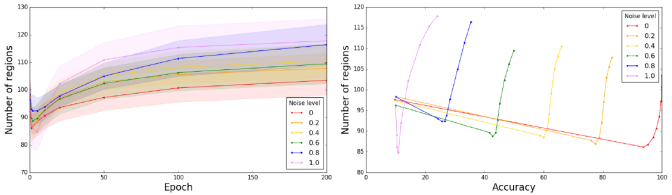


Figure: Depth=3, Width = 32, MNIST with corrupted labels [Hanin & Rolnick, 2019]

Question, what is happening during training?

Idea: focus on the active paths

- Gradient structure forms paths between the input and the output neurons:

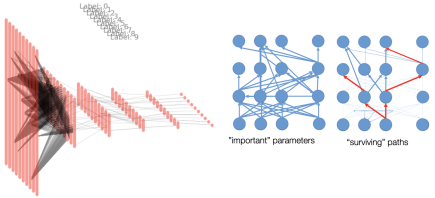


Figure: Black edges are the highest absolute valued elements in the gradient

Idea: focus on the active paths

- Gradient structure forms paths between the input and the output neurons:

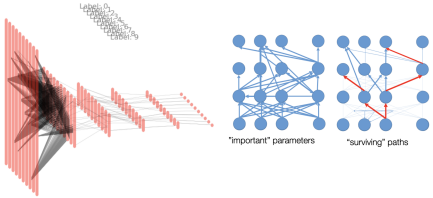


Figure: Black edges are the highest absolute valued elements in the gradient

- Thus $Sens_{tan}(x;\theta)_{i,j} = \sum_{path \in P^{+}_{i,*,j}(x;\theta)} \prod_{w_l \in path, w_l \neq w_j} w_l$ where $\forall x$ $P^{+}_{i,*,j}$ is the set of **corresponding active paths**
  $P^{+}_{i,*,j}(x;\theta) = \cup_{l=\{1,..,d_{out}\}} \{P_{i,l}(x;\theta) | w_j \in P_{i,l}(x;\theta), \forall h_{p_{i,l}}(x;\theta) > 0\}$

Idea: focus on the active paths

- Gradient structure forms paths between the input and the output neurons:

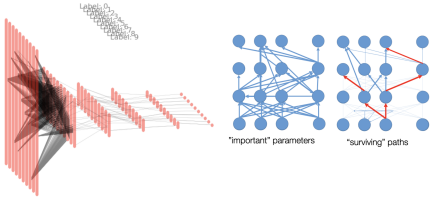


Figure: Black edges are the highest absolute valued elements in the gradient

- Thus $Sens_{tan}(x;\theta)_{i,j} = \sum_{path \in P^{+}_{i,*,j}(x;\theta)} \Pi_{w_l \in path, w_l \neq w_j} w_l$ where $\forall x$ $P^{+}_{i,*,j}$ is the set of **corresponding active paths**
  $P^{+}_{i,*,j}(x;\theta) = \cup_{l=\{1,..,d_{out}\}} \{P_{i,l}(x;\theta) | w_j \in P_{i,l}(x;\theta), \forall h_{p_{i,l}}(x;\theta) > 0\}$
- Based on the regions (paths!) we may avoid non zero batch size and with Jax for the differentiation the gain in computational time is more than 20x over the naive imp.

Conclusions

- Quality monitoring of constantly learning models: **nonvacuous generalization bounds** are useful, high probability upper bounds on the true error aid to decide which models to prefer
- **Overparametrizaton** (double descent) is not an issue as the norms are not directly depend on the number of parameters under additional constraints (stability or regularization)
- It seems **stability** for SSMs and neural ODEs is just as necessary as **explicit or implicit regularization** for classical models
- The bounds first may seem hard to compute however if we know more about the underlying structure we may gain a lot

Thank you!