Data-dependent Regularization and Generalization Bounds of Neural Networks

Tengyu Ma
Stanford University
Two Slides of Ads: Theory for Model-based Deep RL

Better sample-efficiency by learning the dynamical model from data?

Repeat:
1. Sample trajectories from the environment using current policy
2. Learn a dynamical model $((s, a) \rightarrow s')$ using existing trajectories
3. Optimizing policy with the estimated model as the environment

Theoretical challenges: analysis for tabular cases requires confidence interval bounds, which are difficult for continuous case with neural networks models

Empirical challenges: model are not accurate/robust with good uncertainty quantification
A Meta-Algorithm with Theoretical Guarantees

Repeat:
1. Sample trajectories from the environment using current policy
2. Build a lower bound of the payoff using the estimated model and data
   \[ V^\pi \geq L_M(\pi) \]
3. Maximize \( L_M(\pi) \) over both \( M, \pi \)

- Theory: provably converges to a local maximum with poly samples
  (assuming the lower bound is built, and the optimization is solved)
- Theory-inspired heuristic: model learning loss needs to be compatible
  with reward (e.g., \( \ell_2 \) is better than \( \ell_2^2 \)); stochasticity in the model
- State-of-the-art performance when 1M (or fewer) samples are
  permitted (on MuJoCo tasks of Cheetah, Ant, Walker, Humanoid)

[Luo-Xu-Li-Tian-Darrell-M. ICLR’19]
Demo: learning to walk to the right as fast as possible

- Trajectories in estimated environments (neural nets)
- Trajectories in real environments (MuJoCo)

Iteration 10

[Luo-Xu-Li-Tian-Darrell-M. ICLR’19]
Demo: learning to walk to the right as fast as possible

- Trajectories in estimated environments (neural nets)
- Trajectories in real environments (MuJoCo)

Iteration 20

[Luo-Xu-Li-Tian-Darrell-M. ICLR’19]
Demo: learning to walk to the right as fast as possible

- Trajectories in estimated environments (neural nets)
- Trajectories in real environments (MuJoCo)

Iteration 210

[Luo-Xu-Li-Tian-Darrell-M. ICLR’19]
Data-dependent Regularization and Generalization Bounds of Neural Networks

Some recent/new angles to the old notions of regularization and generalization
My starting point:

“how do we design faster optimizers for deep learning?"

Faster training is not that difficult: use smaller learning rate!

![Graph showing error vs. epoch for different learning rates]

Algorithms can regularize!

The lack of understanding of the generalization hampers the study of optimization!

(We have an explanation for this but complicated [Li-Wei-M.’19])

[Keskar et al’17, Hoffer et al’18]
Algorithmic/Implicit Regularization

- Over-parametrized regime (# parameters > # samples):
  - $\exists$ multiple solutions that can perfectly fit to training data

- [Zhang et al’17]: the same architecture and the same algorithm can fit perfectly
  - training data with real labels
  - training data with random labels

Hypothesis:
Stochastic gradient descent, with proper initialization and learning rate, prefers to converging to a low-complexity solution, when it exists

- Intrinsic complexity of the data matters
- Algorithms matter
- No low-complexity solution exists
Algorithmic/Implicit Regularization (Cont’d)

Example 1

- Linear regression: stochastic gradient descent with zero initialization converges to the minimum $\ell_2$ solution
- the parameter stays in the span of the data

Example 2

- Logistic regression: gradient descent converges to the max-margin solution [Soudry-Hoffer-Nacson-Gunasekar-Srebro’17, Ji-Telgarsky’18]
Example 3

- Neural networks with quadratic activations (or matrix sensing)

For any $m \geq r$, gradient descent, with small initialization, converges to the minimum Frobenius norm solution, when $U^*$ exists and # samples $\gtrsim dr^2$. [Li-M.-Zhang’18]
Algorithmic/Implicit Regularization (Cont’d)

Example 4

- Neural networks with “large” initialization

   For any large enough width $m$, gradient descent with a certain initialization converges to a function that fits the data perfectly with minimum norm in the RKHS defined by the neural tangent kernel. [Li-Liang’18, Allen-Zhu et al’18, Du et al’18, Zou et al’18, Arora et al’19]

- Initialization is chosen so that the initial output is on the order of 1

- My view: this is not the best that one can achieve with neural nets
  - Either smaller initialization or some weak regularization helps, empirically and theoretically (more on this topic later)
Hypothesis:
Stochastic gradient descent, with proper initialization and learning rate, prefers to converging to a low-complexity solution, when it exists.

Two key components to fill in:

1. The right definition of the complexity measure, so that low-complexity implies generalization

2. A proof that shows the algorithms’ preferences over low-complexity solution

- For Example 1, 2, 4, convex optimization based analysis
- For Example 3 (quadratic NNs/matrix sensing), eigen-decomposition based analysis. (Very involved IMO)
Algorithmic/Implicit Regularization =
Algorithm Analysis + Generalization Theory

- Intractability?
- Require strong assumptions on the data distribution for non-convex cases
- Upon achieving understanding (explaining existing phenomenon), can we manipulate the algorithm to get better regularization? 😐
Why Not Only Do

\[ \text{Algorithmic/Implicit Regularization} = \text{Algorithm Analysis + Generalization Theory?} \]

- Anyway necessary (in the current framework)
- Perhaps sufficient: add the complexity measure as regularizers
- Can be proactive by manipulating the regularizers
- Separation of concerns: train regularized objective without worrying generalization and implicit bias of algorithms
- Need to assume the training is successful
Generalization Bounds of Deep Neural Networks

The rest of the talk:

- Norm-based complexity measure/regularization
- Data-dependent complexity measure/regularization

\[
\text{generalization} \leq \sqrt{\frac{\text{complexity}}{\text{margin}}} \cdot \frac{1}{\sqrt{n}}
\]
Recall that no regularization + certain initialization $\iff$ minimum norm solution of the neural tangent kernel (NTK)

Theorem [Wei-Lee-Liu-M.’18]:
- Using 2-layer neural net with cross entropy loss and $\ell_2$ regularization, assuming optimization succeeds; sample complexity $= \tilde{O}(d)$
- With the NTK that corresponds to 2-layer neural nets, sample complexity $\geq \Omega(d^2)$

- Gap empirically observed on both synthetic and real data
- Optimization can be provably solved in poly iteration if width $\to \infty$ [Wei-Lee-Liu-M.’18]
Norm-Based Generalization Bounds (for Deep Models)

- Let $r$ be the depth

- [Golowich et al’18]:

  $$\text{generalization} \lesssim \sqrt{\frac{\text{product of } \ell_2\text{-norms of weights}}{\text{margin}}} \cdot \frac{1}{\sqrt{n}} \lesssim \sqrt{\frac{(\ell_2\text{-norm}/\sqrt{r})^r}{\text{margin}}} \cdot \frac{1}{\sqrt{n}}$$

- [Bartlett et al’17, Neyshabur et al.’17]:

  $$\text{generalization} \lesssim \sqrt{\frac{\text{product of spectral norms of weights}}{\text{margin}}} \cdot \frac{1}{\sqrt{n}} \cdot \text{“sum of norms”}$$

- The products of norms comes from the worst case Lipschitzness of the model (over all possible inputs)
Existing analyzable regularization/complexity measures depend only on the parameters.

Teaser: what happens if we regularize the norm of the hidden activations (which is a function of both data and parameters).

Many empirically successful regularization depends on both the parameters and the data:
- dropout
- batch normalization/layer normalization
- mix-up [Zhang et al’18]
**Batchnorm Regularizes**

ImageNet, ResNet-50

- Fixup [Zhang-Dauphin-M.’19]: a certain initialization scheme (no BN)
- Training is great but test is not good; needs additional regularization
Goals and Results

- Goal 1: new mathematical tools to handle data-dependent bounds
  
  We propose a somewhat generic approach

- Goal 2: better generalization bounds with data-dependent complexity measures
  
  We remove the exponential dependency on the depth

- Goal 3: empirically regularize the complexity measure for better accuracy (or just reduce the influence of the algorithms)
  
  Some preliminary improvements in various settings
Data-Dependent Generalization Bounds of Neural Nets

- $t$ is the maximum $\ell_2$ norm of any hidden layer of training data
  $$t = \max_{\text{training data}} \max_i ||h_i||$$

- $\sigma$ is the maximum operator norm of any interlayer Jacobian
  $$\sigma = \max_{\text{training data}} \max_{i>j} \left| \left| \frac{\partial h_i}{\partial h_j} \right|\right|_{\text{op}}$$

- $\gamma > 0$ is the margin (the output for the correct label – second largest output)

- we assume that the training error $= 0$ (because of over-parameterization)
Theorem (informal) [Wei-M.’19]: W.h.p over the randomness of the data

\[
\text{test error} \lesssim \frac{1}{\sqrt{n}} \left( \frac{\sigma t}{\gamma} + r^3 \sigma^2 t + r^2 \sigma \right) \left( \sum_i |W_i|^{2/3} \right)^{3/2}
\]

- \( t \) is the maximum \( \ell_2 \) norm of any hidden layer of training data:
  \[
t = \max_{\text{training data}} \max_i \max_{||x|| \leq 1} ||h_i|| \leq \max \max_{||x|| \leq 1} ||h_i||
\]
- \( \sigma \) is the maximum operator norm of any interlayer Jacobian:
  \[
  \sigma = \max_{\text{training data}} \max_{i > j} \left\| \frac{\partial h_i}{\partial h_j} \right\|_{\text{op}} \leq \max \max_{||x|| \leq 1} \left\| \frac{\partial h_i}{\partial h_j} \right\|_{\text{op}}
\]

\[\prod \|W_i\|_{\text{op}} \text{ in previous bound [Bartlett et al’17]}\]

- Note: the bound is heavily simplified
Theorem (informal) [Wei-M.’19]: W.h.p over the randomness of the data

test error $\lesssim \frac{1}{\sqrt{n}} \left( \frac{\sigma t}{\gamma} + r^3 \sigma^2 t + r^2 \sigma \right) \left( \sum_i |W_i|_1^{2/3} \right)^{3/2}$

$\prod \|W_i\|_{op}$ in previous bound [Bartlett et al’17]

- $t$ is the maximum $\ell_2$ norm of any hidden layer of training data
  $t = \max_{\text{training data}} \max_i ||h_i|| \leq \max_{||x|| \leq 1} \max_i ||h_i|| \leq \prod \|W_i\|_{op}$

- If no nonlinear activations
  $\max_{x \in \text{training data}} \max_i ||W_iW_{i-1} \cdots W_1 x|| \quad \text{v.s.} \quad \prod \|W_i\|_{op}$
Comparison Against Existing Bounds

- Bounds in [Bartlett et. al’17, Neyshabur et. al’17, Golowich et. al’18] depend exponentially in depth via product of parameter norms.

- We depend on max Jacobian and hidden layer norm on training data (much smaller in practice).

- Bound by [Nagarajan, Kolter’19] depends on Jacobian and hidden layer norms, but also inverse pre-activations (large in practice).

- [Arora et. al ‘18] – data-dependent bounds, but only holds for compressed network (generalization follows via parameter counting).
Correlation Test

- spectral --- approx. of Bartlett’s complexity measure
- ours --- approx. of our complexity measure
## Regularizing the Complexity Measure

- Hidden layer norm is already regularized by normalization layers
- Add interlayer Jacobian norms \( \left\| \frac{\partial h_i}{\partial h_j} \right\| \) as a regularizer

<table>
<thead>
<tr>
<th>Setting</th>
<th>Normalization</th>
<th>Jacobian Reg</th>
<th>Test Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>BatchNorm</td>
<td>×</td>
<td>4.43%</td>
</tr>
<tr>
<td></td>
<td>BatchNorm</td>
<td>✓</td>
<td>3.99%</td>
</tr>
<tr>
<td>Low learning rate (0.01)</td>
<td>BatchNorm</td>
<td>×</td>
<td>5.98%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓</td>
<td>5.46%</td>
</tr>
<tr>
<td>No data augmentation</td>
<td>BatchNorm</td>
<td>×</td>
<td>10.44%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓</td>
<td>8.25%</td>
</tr>
<tr>
<td>No BatchNorm</td>
<td>None</td>
<td>×</td>
<td>6.65%</td>
</tr>
<tr>
<td></td>
<td>LayerNorm</td>
<td>×</td>
<td>6.20%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓</td>
<td>5.57%</td>
</tr>
</tbody>
</table>
The complexity measure and hypothesis class depends on the data
\( \mathcal{F}_X = \{ \text{all neural nets with small hidden layer and Jacobian on the training data} \} \)

For VC dim or Radamacher complexity, hypothesis class \( \mathcal{F} \) is pre-defined before drawing examples \( X \)

All the existing tools breaks when hypothesis class is a random variable depending on the training data
Recall standard workflow to use Rademacher complexity:

\[
\text{complexity}(\mathcal{F}) \rightarrow \text{complexity(classes of losses } \ell \circ \mathcal{F}) \rightarrow \text{generalization}
\]

\[\ell = \text{margin loss}\]

Our approach: fold more data-dependent properties in the loss

- Set of properties \(\{P_i\}\) with desired bounds \(\{\gamma_i\}\)

Augmented loss: \(l_{aug} = (l_{old} - 1) \prod_i 1(P_i \leq \gamma_i) + 1\)

- \(l_{aug} \geq l_{old}\), and \(l_{aug} = l_{old}\) if properties hold

Suffices to bound generalization error of augmented class \(\mathcal{L}_{aug}\)
Data-dependent Loss Augmentation

The data-dependent region that we care about

\[ l_{\text{aug}} = l_{\text{old}} \] in this region

\[ l_{\text{old}} \]

\[ \theta \]

\[ \text{test} \]

\[ \text{train} \]

augmentation
Rademacher Complexity of The Augmented Loss

- $l_{aug} = (l_{old} - 1) \prod 1(\text{hidden layer sizes small}) \prod 1(\text{Jacobians small}) + 1$

- A methods for bounding Rademacher complexity of computational graphs

- Can also be applied to recurrent neural networks
Data-dependent explicit regularization is (still) important

- more powerful than neural tangent kernel
- may replace the effect of algorithmic regularization

Open Questions: other data-dependent regularizations?

- dropout
- data augmentation (mixup, cutout, ...)
- new ones?

Thank you!