Foundations of Deep Learning: SGD, Overparametrization, and Generalization

Jason D. Lee

University of Southern California

November 13, 2018
Deep Learning

**Single Neuron**

\[ x \rightarrow \sigma(\langle w, x \rangle) \]

ReLU: \( \sigma(z) = [z]_+ \)

**Figure:** ReLU and Sigmoid

**Feedforward Network**

\[ x \rightarrow W_L \sigma(\ldots \sigma(W_1 x)) \]

**Depth** is \( L \) and **width** is the maximal dimension of \( W_l \).

**Figure:** Feedforward Neural Net

Credit: Ohad Shamir
Deep Learning
Successes of Deep Learning

- Game-playing (AlphaGo, DOTA, King of Glory)
- Computer Vision (Classification, Detection, Reasoning.)
- Automatic Speech Recognition
- Natural Language Processing (Machine Translation, Chatbots)
Successes of Deep Learning

- Game-playing (AlphaGo, DOTA, King of Glory)
- Computer Vision (Classification, Detection, Reasoning.)
- Automatic Speech Recognition
- Natural Language Processing (Machine Translation, Chatbots)

Today’s Talk

Goal: Develop theoretical understanding of **Optimization** and **Generalization** in Deep Learning.
1. Challenges

2. Saddlepoints and SGD

3. Landscape Design via Overparametrization

4. ReLU Networks via Margin Theory

5. Algorithmic/Implicit Regularization
Theoretical Challenges: Two Major Hurdles

1. **Optimization**
   - Non-convex and non-smooth with exponentially many critical points.

2. **Statistical**
   - Successful Deep Networks are huge with more parameters than samples (overparametrization).
Theoretical Challenges: Two Major Hurdles

1. Optimization
   - Non-convex and non-smooth with exponentially many critical points.

2. Statistical
   - Successful Deep Networks are huge with more parameters than samples (overparametrization).

Two Challenges are Intertwined

Learning = Optimization Error + Statistical Error.

But Optimization and Statistics Cannot Be Decoupled.

- The choice of optimization algorithm affects the statistical performance (generalization error).
- Improving statistical performance (e.g. using regularizers, dropout . . . ) changes the algorithm dynamics and landscape.
Practical observation: Empirically, non-convexity is not an issue. Gradient methods find high quality solutions.
Practical observation: Empirically, non-convexity is not an issue. Gradient methods find high quality solutions.

Theoretical Side: NP-hard in many cases. e.g. Finding a local minimum in a smooth function is NP-Hard. Finding a 2nd order optimal point in a non-smooth function is NP-Hard.
Practical observation: Empirically, non-convexity is not an issue. Gradient methods find high quality solutions.

Theoretical Side: NP-hard in many cases. e.g. Finding a local minimum in a smooth function is NP-Hard. Finding a 2nd order optimal point in a non-smooth function is NP-Hard.

Follow the Gradient Principle: No known convergence results for even back-propagation to stationary points!
Practical observation: Empirically, non-convexity is not an issue. Gradient methods find high quality solutions.

Theoretical Side: NP-hard in many cases. e.g. Finding a local minimum in a smooth function is NP-Hard. Finding a 2nd order optimal point in a non-smooth function is NP-Hard.

Follow the Gradient Principle: No known convergence results for even back-propagation to stationary points!

Question

1. Why is (stochastic) gradient descent (GD) successful? Or is it just “alchemy”? 
(Sub)-Gradient Descent

Gradient Descent algorithm:

\[ x_{k+1} = x_k - \alpha_k \partial f(x_k). \]

Non-smoothness

Deep Learning Loss Functions are not smooth! (e.g. ReLU, max-pooling, batch-norm)
Theorem (Davis, Drusvyatskiy, Kakade, and Lee)

Let \( x_k \) be the iterates of the stochastic sub-gradient method. Assume that \( f \) is locally Lipschitz, then every limit point \( x^* \) is critical:

\[
0 \in \partial f(x^*).
\]

- Previously, convergence of sub-gradient method to stationary points is only known for weakly-convex functions \((f(x) + \frac{\lambda}{2} \|x\|^2\) convex \). \((1 - \text{ReLU}(x))^2\) is not weakly convex.
- Convergence rate is polynomial in \( \frac{\sqrt{d}}{\epsilon^4} \), to \( \epsilon \)-subgradient for a smoothing SGD variant.
Can subgradients be efficiently computed?

Automatic Differentiation a.k.a Backpropagation

Automatic Differentiation uses the chain rule with dynamic programming to compute gradients in time $5x$ of function evaluation.

However, there is no chain rule for subgradients!

$$x = \sigma(x) - \sigma(-x),$$

TensorFlow/Pytorch will give the wrong answer.
Can subgradients be efficiently computed?

**Automatic Differentiation a.k.a Backpropagation**

Automatic Differentiation uses the chain rule with dynamic programming to compute gradients in time $5x$ of function evaluation.

However, there is no chain rule for subgradients!

$$x = \sigma(x) - \sigma(-x),$$

TensorFlow/Pytorch will give the wrong answer.

**Theorem (Kakade and Lee 2018)**

There is a chain rule for directional gradients. Using this chain rule with randomization, Automatic Differentiation can compute a subgradient in time $6x$ of function evaluation.
Theorem (Lee et al., COLT 2016)

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice continuously differentiable function with the strict saddle property, then gradient descent with a random initialization converges to a local minimizer or negative infinity.

- Theorem applies for many optimization algorithms including coordinate descent, mirror descent, manifold gradient descent, and ADMM (Lee et al. 2017 and Hong et al. 2018)
- Stochastic optimization with injected isotropic noise finds local minimizers in polynomial time (Pemantle 1992; Ge et al. 2015, Jin et al. 2017)
Why are local minimizers interesting?

All local minimizers are global and SGD/GD find the global min:

1. Overparametrized Networks with Quadratic Activation (Du-Lee 2018)
2. ReLU networks via landscape design (GLM18)
3. Matrix Completion (GLM16, GJZ17, . . . )
4. Rank $k$ approximation (Baldi-Hornik 89)
5. Matrix Sensing (BNS16)
6. Phase Retrieval (SQW16)
7. Orthogonal Tensor Decomposition (AGHKT12, GHJY15)
8. Dictionary Learning (SQW15)
9. Max-cut via Burer Monteiro (BBV16, Montanari 16)
Designing the Landscape

Goal: Design the Loss Function so that gradient decent finds good solutions (e.g. no spurious local minimizers)\(^a\).

\(^a\)Janzamin-Anandkumar, Ge-Lee-Ma, Du-Lee
Figure: Data is generated from network with $k_0 = 50$ neurons. Overparametrized network has $k = 100$ neurons.

Without some modification of the loss, SGD will get trapped.
Conventional Wisdom on Overparametrization
If SGD is not finding a low training error solution, then fit a more expressive model until the training error is near zero.

Problem
How much over-parametrization do we need to efficiently optimize and generalize?

- Adding parameters increases computational and memory cost.
- Too many parameters may lead to overfitting (???).
Previous Work on Overparametrization

Folklore
Optimization is “easy” when parameters > sample size.

- Soudry and Carmon 2016 justified this for ReLU networks.
- Livni et al. empirically demonstrated that over-parametrization is necessary for SGD to work.
- Super Wide Layers: When poly(# neurons per layer) > sample size, then gradient descent can optimize unregularized train loss (Li-Liang 18, Du et al. 18, Du-Lee-Li-Wang-Zhai 18). Generalization is roughly same as kernel methods.
Previous Work on Overparametrization

Folklore

Optimization is “easy” when parameters $> \text{sample size}$. 

- Soudry and Carmon 2016 justified this for ReLU networks.
- Livni et al. empirically demonstrated that over-parametrization is necessary for SGD to work.
- Super Wide Layers: When $\text{poly}(\# \text{neurons per layer}) > \text{sample size}$, then gradient descent can optimize unregularized train loss (Li-Liang 18, Du et al. 18, Du-Lee-Li-Wang-Zhai 18). Generalization is roughly same as kernel methods.

Summary

- Require extremely wide layers, so convex optimization could be used (e.g. train last layer).
- Results frequently only hold for unregularized loss, and thus can only analyze training loss.
Case Study: Quadratic Activation Networks

\[ f(x; W) = \sum_{i=1}^{k} \phi(w_i^T x) = x^T W^T W x, \]

where \( \phi(z) = z^2. \)

These can be formulated as matrix sensing with \( X_i = x_i x_i^T. \)

Regularized Loss

\[ \min_{W} \sum_{i} \ell(f(x_i; W), y_i) + \frac{\lambda}{2} \|W\|_F^2. \]
Smoothed Analysis

Random Regularization

\[ L_C(W) = \sum_i \ell(f(x_i; W), y_i) + \frac{\lambda}{2} \|W\|_F^2 + \langle C, W^T W \rangle, \]

where \( C \) is a random matrix of norm \( \delta \).
Let $\ell$ be a convex loss function, $\lambda > 0$, and $\delta > 0$. If $k \geq \sqrt{2n}$, then almost surely all local min are global minima.

- Applies for arbitrarily small perturbation $\delta$, so for small $\delta$ we can closely approximate the solution of the unperturbed objective $^1$.

- Motivated by work on SDP (Burer & Monteiro, Boumal-Voroninski-Bandeira) which show that $k \geq \sqrt{2n}$ all non-degenerate local minima are global. Smoothing allows us to remove the degenerate local minima.

- Surprisingly, the same smoothing works even though our objective is not SDP-representable.

---

$^1$Runtime depends on $\frac{1}{\delta}$
How about Generalization?

Generalization

$f_W(x) = x^T W^T W x$, so regularizer $\|W\|_F^2$ corresponds to $\|W^T W\|_*$. 

Small nuclear norm promotes networks with few active neurons, which leads to generalization!
How about Generalization?

**Generalization**

\[ f_W(x) = x^T W^T W x, \] so regularizer \( \|W\|_F^2 \) corresponds to \( \|W^T W\|_* \).

Small nuclear norm promotes networks with few active neurons, which leads to generalization!

**Corollary**

Assume that \( y = \sum_{i=1}^{k_0} \sigma(w_i^T x) \). Then for \( n \gtrsim \frac{d k_0}{\epsilon^2} \) and \( t \geq \text{poly}(\frac{1}{\epsilon}, d, k) \), SGD finds a solution

\[ L_{te}(W_t) \lesssim \epsilon \]

The sample complexity is independent of \( k \), the number of neurons.
Training Error: Over-parametrization makes the optimization easy, since all local are global.

Test Error: The generalization is not hurt by over-parametrization. The sample complexity only depends on $k_0$, the number of effective neurons, and not $k$, the number of neurons in the model.
**Summary**

**Quadratic Activation Network**

1. **Training Error**: Over-parametrization makes the optimization easy, since all local are global.

2. **Test Error**: The generalization is not hurt by over-parametrization. The sample complexity only depends on $k_0$, the number of effective neurons, and not $k$, the number of neurons in the model.

What about ReLU activations and deeper networks?
Margin Theory

Margin $\gamma(W) = \min_{i} y_i f\left(\frac{W}{\|W\|_2}, x_i\right)$. When $\gamma$ is large, the network predicts the correct label with high confidence.

- Large margin potentially guarantees parameter-free generalization bounds (Bartlett & Mendelson, Koltchinskii)

Large margin

Do we obtain large margin classifiers in Deep Learning?
Regularized Loss

Neural networks are train via minimizing the regularized cross-entropy loss:

\[ \ell(f(W; x)) + \lambda \| W \|. \]

Rosset-Zhu-Hastie showed that when \( f(w; x) = \langle w, x \rangle \) minimizing the weakly regularized loss attains the optimal max-margin.
Regularized Loss

Neural networks are trained via minimizing the regularized cross-entropy loss:

\[ \ell(f(W; x)) + \lambda ||W||. \]

- Rosset-Zhu-Hastie showed that when \( f(w; x) = \langle w, x \rangle \) minimizing the weakly regularized loss attains the optimal max-margin.

Theorem (Wei-Lee-Liu-Ma 2018 extending Rosset-Zhu-Hastie)

Let \( f \) be a positive homogeneous network and \( \gamma^* = \max_{\|W\| \leq 1} \min_{i \in [n]} y_i f(W; x_i) \) be the optimal normalized margin.

- Minimizing cross-entropy loss is max-margin: \( \gamma(W_\lambda) \to \gamma^* \).
- The optimal margin is an increasing function of network size.
Does large margin lead to parameter-independent generalization in Neural Networks?
Does large margin lead to parameter-independent generalization in Neural Networks?

Parameter-independent Generalization Bounds (Neyshabur et al.)

Let $f(W; x) = W_2\sigma(W_1x)$.

$$\Pr \left( yf(W; x) < 0 \right) \lesssim \frac{1}{\gamma\sqrt{n}}.$$  

- Completely independent of the number of parameters.
- Generalization to the deep case is also parameter-independent (Golowich-Rakhlin-Shamir)$^a$

$$^a\Pr \left( yf(W; x) < 0 \right) \lesssim \frac{\|W\|_2}{\tilde{\gamma}\sqrt{n}}$$ and $\tilde{\gamma}$ is un-normalized margin.
Training Loss

Let \( f(x; W) = \sum_{j=1}^{m} w_{1,j} \sigma(\langle w_{2,j}, x \rangle) \) with \( \sigma = \text{ReLU} \).

\[
\min_{W} \sum_{i} \ell(f(x_i; W), y_i) + \frac{\lambda}{2} \sum_{j=1}^{m} (w_{1,j}^2 + \|w_{2,j}\|_2^2).
\]
Does GD Minimize Regularized Loss?

Training Loss

Let $f(x; W) = \sum_{j=1}^{m} w_{1,j} \sigma(\langle w_{2,j}, x \rangle)$ with $\sigma = \text{ReLU}$.

$$\min_W \sum_i \ell(f(x_i; W), y_i) + \frac{\lambda}{2} \sum_{j=1}^{m} (w_{1,j}^2 + \|w_{2,j}\|_2^2).$$

Theorem (Very Informal, see arXiv)

For a two-layer network that is infinitely wide (or $\exp(d)$ wide), gradient descent with noise converges to a global minimum of the regularized training loss in polynomial time.

- Overparametrization helps gradient descent find solutions of low train loss$^2$

---

$^2$see also Chizat-Bach, Mei-Montanari-Nguyen
Modern networks are over-parametrized meaning $p \gg n$ ($\frac{p}{n} \in (10, 200)$).

Over-parametrization allows SGD to drive the training error to 0. But shouldn’t the test error be huge due to overfitting?
Experiment

$p \gg n$, no regularization, no early stopping, and yet we do not overfit.

In fact, test error decreases even after the train error is zero.

Weight decay helps a little bit ($< 2\%$), but generalization is already good without any regularization.

Figure: Credit: Neyshabur et al. See also Zhang et al.
Experiment

Figure: Credit: Neyshabur et al. See also Zhang et al.

Problem

Why does SGD not overfit?
Today’s Setting

**Definition (Separable Data)**

We will assume that \( y_i(x_i^T w) > 0 \) for some \( w \).

- Equivalent of the over-parametrized regime in linear models. If \( p \gg n \), this holds for almost all \( \{x_i\} \).
- When the data is separable, there are infinitely many linear separators.
Warm-up: Logistic Regression with separable data

Gradient descent with any initial point \( w_0 \) on

\[
\mathcal{L}(w) = \sum_i \log(1 + \exp(-y_i x_i^T w))
\]

converges in direction to the \( \ell_2 \)-SVM solution. In equations,

\[
\frac{w(t)}{\|w(t)\|} \to C \arg \min_{y_i w^T x_i \geq 1} \|w\|_2.
\]

(Soudry-Hoffer-Srebro, Ji-Telgarsky, Gunasekar-Lee-Soudry-Srebro)
Warm-up: Logistic Regression with separable data

Gradient descent with any initial point $w_0$ on

$$\mathcal{L}(w) = \sum_i \log(1 + \exp(-y_i x_i^T w))$$

converges in direction to the $\ell_2$-SVM solution. In equations,

$$\frac{w(t)}{\|w(t)\|} \to C \arg\min_{y_i w^T x_i \geq 1} \|w\|_2.$$

(Soudry-Hoffer-Srebro, Ji-Telgarsky, Gunasekar-Lee-Soudry-Srebro)

This means that if the data is separable with a large margin, then GD+Logistic Regression generalizes as well as SVM.
Steepest Descent

\[ w(t + 1) = w(t) + \alpha \Delta w(t) \]
\[ \Delta w(t) = \arg \min_{\|v\| \leq 1} v^T \nabla L(w(t)). \]

Coordinate descent is steepest descent wrt \( \| \cdot \|_1 \) and signed gradient method is steepest descent wrt \( \| \cdot \|_\infty \).
Theorem (Gunasekar, Lee, Soudry, and Srebro)

On separable data, steepest descent converges in direction to the $\| \cdot \|_S$-SVM solution, meaning

$$\frac{w(t)}{\|w(t)\|} \rightarrow C \arg \min_{y_i w^T x_i \geq 1} \|w\|.$$

- Solution depends on the choice of algorithm.
- For coordinate descent, it is already known from the boosting literature that AdaBoost achieves the minimum $\ell_1$ norm solution (Ratsch et al. 2004, Zhang-Yu, Telgarsky). Also related to the study of LARS algorithms.
- For $\ell_2$ norm, this recovers the theorem before.
Theorem (Gunasekar, Lee, Soudry and Srebro 2018)

For any homogeneous polynomial $p$, GD on

$$\sum_i \exp(-y_i \langle p(W), x_i \rangle)$$

converges\(^a\) to a first-order optimal point of the non-linear SVM:

$$\min \|W\|_2 \quad \text{st} \quad y_i \langle p(W), x_i \rangle \geq 1.$$  

\(^a\)Technical assumptions on limits existing is needed.

GD is implicitly regularizing $\ell_2$-norm of parameters.
1 Quadratic Activation Network\(^3\): \( p(W) = WW^T \) leads to an implicit nuclear norm regularizer, and thus a preference for networks with a small number of neurons.

2 Linear Network\(^4\): \( p(W) = W_L \ldots W_1 \) leads to an Schatten quasi-norm regularizer \( \|p(W)\|_{2/L} \).

3 Linear Convolutional Network: Sparsity regularizer \( \| \cdot \|_{2/L} \) in the Fourier domain.

---

\(^3\)see also Gunasekar et al. 2017, Li et al. 2017

\(^4\)see also Ji-Telgarsky
Implicit Regularization

1. Overparametrize to make training easy, but there are infinitely many possible global minimum.
2. The choice of algorithm and parametrization determine the global minimum.
3. Generalization is possible in the over-parametrized regime with no regularization by choosing the right algorithm.
4. We understand only very simple problems and algorithms.
Acknowledgements: Joint work with the co-authors below.

5. Davis, Drusvyatskiy, Sham Kakade, and Jason D. Lee, *Stochastic subgradient method converges on tame functions*.