Analyzing Stochastic Gradient Descent for Some Non-Convex Problems

Christopher De Sa
Soon at Cornell University
cdesa@stanford.edu
stanford.edu/~cdesa

Kunle Olukotun
Stanford University

Christopher Ré
Same motivation as plenary talk:

Large Scale Machine Learning

• Machine learning problems are big
  • Larger datasets & more sophisticated models → better ML services

• Today, all production machine learning algorithms run in parallel
  • Because modern architectures are parallel

• Most work is done with stochastic iterative algorithms
  • Because of the scale of the datasets and models
Background:
Learning

• Perhaps the biggest class of machine learning applications
  • Goal: learn the model that best accomplishes a task
  • Formally: find model that minimizes some loss function for a dataset

\[
\min_x \sum_{i=1}^N f(x; y_i)
\]

• Applications ranging from linear regression to deep learning

• De facto algorithm used at scale: **Stochastic gradient descent (SGD)**
What is stochastic gradient descent?

Repeat:

• Pick a training example \( y_{i_t} \) uniformly at random

• Update the model \( x_t \) using a gradient estimate

\[
x_{t+1} = x_t - \alpha \nabla f(x_t; y_{i_t})
\]

• Iterate

This is not generally guaranteed to converge to the global optimum for non-convex problems.
No General Guarantees

• Because of local minima

• Example: for this function, even gradient descent won’t get to the global optimum from everywhere
How to proceed?

• Show only convergence to a critical point
  • Straightforward to do
Showing we get close to critical points

For any $x$, let $g$ be the gradient estimate at $x$. Then,

$$
E \left[ f(x - \alpha \tilde{g}) \right] = E \left[ f(x) - \alpha \tilde{g}^T \nabla f(x) + \alpha^2 \tilde{g}^T \nabla^2 f(z) \tilde{g} \right]
\leq f(x) - \alpha \| \nabla f(x) \|^2 + \alpha^2 M
$$

Summing up across iterations, for square-summable but not summable step sizes

$$
\sum_{t=0}^{T-1} \alpha_t E \| \nabla f(x_t) \|^2 \leq E \left[ f(x_0) - f(x_T) \right] + \sum_{t=0}^{T-1} \alpha_t^2 M \leq \infty
$$
How to proceed?

• Show only convergence to a critical point
  • Straightforward to do
  • Results are weak — usually convergence to a critical point is not good enough

• Show only local convergence, given “close enough” initialization
  • Also straightforward to do
Showing local convergence

Assume that the second derivative matrix is nonsingular at the optimum, 
\[ \nabla^2 f(x^*) \succeq 0. \]

Then there exists a ball around the optimum where \( f \) is convex.

- In fact, strongly convex.

If we initialize far enough into the interior of this ball, any standard analysis of convex SGD will work.

- With minor tweaks to make sure we don’t escape the ball.
How to proceed?

• Show only convergence to a critical point
  • Straightforward to do
  • Results are weak — often convergence to a critical point is not good enough

• Show only local convergence, given “close enough” initialization
  • Also straightforward to do
  • Hard bit is finding explicit initialization algorithm that is close enough

• Show global convergence from almost all initializations
  • For a restricted class of problems
Guarantees for Non-Convex SGD

• Main question: **when can we show that SGD converges**
  • globally (from almost all initializations)
  • for a non-convex problem
  • with an explicit rate of convergence
  • that is robust to additional sources of noise (such as asynchrony)?

• Problem for this talk: **matrix completion**
  • Global Convergence of SGD for Some Nonconvex Matrix Problems [De Sa et al, ICML 2015]
  • Taming the Wild: A Unified Analysis of Hogwild!-Style Algorithms [De Sa et al, NIPS 2015]
Outline

1. A nonconvex problem: matrix completion

2. SGD for matrix completion

3. Convergence guarantees

4. Robustness to additional noise
Outline

1. A nonconvex problem: matrix completion
2. SGD for matrix completion
3. Convergence guarantees
4. Robustness to additional noise
Matrix Completion Problem

• Goal is to fit a low-rank matrix $X$ to samples

$$\text{minimize} \quad \mathbb{E} \left[ \| \tilde{A} - X \|_F^2 \right]$$
$$\text{subject to} \quad X \in \mathbb{R}^{n \times n}, \text{rank}(X) \leq 1, X \succeq 0.$$

• Apply quadratic substitution $X = yy^T$ (Burer-Monteiro)

$$\text{minimize} \quad \mathbb{E} \left[ \| \tilde{A} - yy^T \|_F^2 \right]$$
$$\text{subject to} \quad y \in \mathbb{R}^n.$$
Many Applications

- Standard matrix completion
- Matrix sensing
- Subspace tracking
- Principle component analysis

How to represent many applications?

- Same optimization problem
- Different application → different noise model
Key Insight: Use Weak Noise Model

• Using only weak assumptions about the sample distribution lets us handle many application cases with a unified analysis.

• How much can we prove by assuming only
  • that the estimates are unbiased, and

\[
E \left[ \hat{A} \right] = A.
\]

• that their variance is bounded?

\[
\forall y, z : \ E \left[ \left( y^T \hat{A} z \right)^2 \right] \leq \sigma^2 \|y\|^2 \|z\|^2
\]
Simple Gradient Flow of 2D Case

\[
\begin{align*}
\dot{x} &= 4x - (x^2 + y^2)x \\
\dot{y} &= y - (x^2 + y^2)y
\end{align*}
\]

Consequences of non-convexity:
• We get pushed in multiple directions
• There are multiple unstable fixed points

What’s nice about this problem:
• Symmetry
• No non-optimal local minima
If we initialize on a bad trajectory, **could take forever to escape.**

Using weak noise model:

- **Can’t show convergence from everywhere** in reasonable time
  - can’t show convergence from initial points near bad trajectory
  - algorithm can always “jump” onto bad trajectory then stay there for an arbitrarily long time
Outline

1. A nonconvex problem: matrix completion

2. SGD for matrix completion

3. Convergence guarantees

4. Robustness to additional noise
SGD for Matrix Completion

- Standard SGD gives the update step

\[ y_{k+1} = y_k - 4\alpha_k \left( y_k y_k^T y_k - \tilde{A}_k y_k \right). \]

- By using a variable step size scheme (SGD on a manifold)

\[ y_{k+1} = \left( I + \eta \tilde{A}_k \right) y_k \left( 1 + \eta \| y_k \|^2 \right)^{-1}. \]

- To just get the direction

\[ y_{k+1} = \left( I + \eta \tilde{A}_k \right) y_k. \]
Many Related Algorithms

\[ y_{k+1} = \left( I + \eta \tilde{A}_k \right) y_k. \]

• Many algorithms use this or a similar update rule
  • Alecton [De Sa et al, ICML 2015]
  • Oja’s algorithm [Oja, Journal of mathematical biology, 1982]
  • Stochastic power iteration

• Need to periodically normalize: \[ y := y / \| y \| \]
  • Doesn’t affect direction
Outline

1. A nonconvex problem: matrix completion
2. SGD for matrix completion
3. Convergence guarantees
4. Robustness to additional noise
Analyze Non-Convex SGD using Martingales

• Using a standard Lyapunov-function approach won't work.
  • This approach shows convergence from everywhere, which we've shown doesn't happen

• Martingale approach:
  • Handles processes which can fail with some probability
  • Bounds the probability of failure
  • Bonus: lets us combine noise from different sources with a unified analysis

\[ E \left[ x_{t+1} | x_t \right] = x_t \]
Measuring Convergence

• Success condition: angular closeness to global optimum \( u_1 \)

\[
\rho_k = \frac{(u_1^T y_k)^2}{\|y_k\|^2} \geq 1 - \epsilon
\]

• We let \( F_t \), the failure event, denote the event that the algorithm hasn’t succeeded after \( t \) iterations.
Theorem

For any $\chi > 0$, if we run for $t$ iterations where

$$t \geq \frac{n \log n}{\epsilon} \cdot \frac{\sigma^2}{\Delta^2} \cdot G(\chi),$$

then the probability of failure is bounded by

$$\Pr(F_t) \leq \chi,$$

where $G$ is some fixed function of $\chi$. 
Technique

• Analyze the quantity

\[ \tau_t = \frac{(u_1^T y_t)^2}{y_t^T (\gamma I + (1 - \gamma)u_1 u_1^T) y_t} \]

• Can prove that if success hasn’t occurred, for some \( R \)

\[ \mathbb{E} [\tau_{t+1} | \mathcal{F}_t] \geq \tau_t (1 + R (1 - \tau_t)) \]

• Careful martingale analysis of this recurrence proves the theorem
Outline

1. A nonconvex problem: matrix completion
2. SGD for matrix completion
3. Convergence guarantees
4. Robustness to additional noise
Techniques for Fast Parallel SGD

• **Asynchronous execution** *(HOGWILD!)*
  - run multiple threads of SGD **without locks**
  - exposes **multi-core** parallelism
  - may have race conditions, but theory says: **it’s okay**

• **Low-precision computation** *(BUCKWILD!)*
  - use low-precision **fixed-point arithmetic** *(e.g. 8-bit)*
  - exposes **SIMD** parallelism
Low-Cost Parallelism With HOGWILD!

Multiple parallel workers

- Pick a training example $y_{i_t}$ uniformly at random
- Update the model $x_t$ using a gradient estimate
  $x_{t+1} = x_t - \alpha \nabla f(x_t; y_{i_t})$
- Iterate

Asynchronous parallel updates (no locks) to a single shared model

$x_t$
Recent Hardware Trend: SIMD

- Single data-parallel instruction can process **multiple values at once**.

- Source of parallelism independent of threading.
Low Precision and SIMD Parallelism

- Major benefit of low-precision: use **SIMD instructions** to get more parallelism on CPU

**SIMD Precision**

- 64-bit float vector
- 32-bit float vector
- 16-bit int vector
- 8-bit int vector

**SIMD Parallelism**

- 4 multiplies/cycle (vmulpd instruction)
- 8 multiplies/cycle (vmlps instruction)
- 16 multiplies/cycle (vpmaddwd instruction)
- 32 multiplies/cycle (vpmaddubsw instruction)
Low Precision and Memory

• Major benefit of low-precision: puts **less pressure** on memory and caches

### Precision in DRAM

- **64-bit float vector**
- **32-bit float vector**
- **16-bit int vector**
- **8-bit int vector**

### Memory Throughput

- 64-bit float vector: 5 numbers/ns (assuming ~40 GB/sec memory bandwidth)
- 32-bit float vector: 10 numbers/ns
- 16-bit int vector: 20 numbers/ns
- 8-bit int vector: 40 numbers/ns
Taming the Wild

• Can we also prove convergence results that give rates for nonconvex SGD using these techniques?

• Taming the Wild [De Sa et al, NIPS 2015])
  • We show a principled way to give a rate for a HOGWILD! version of an algorithm, given a martingale-based rate for its sequential version.
  • This applies to the non-convex matrix completion results.
  • First convergence rate for asynchronous SGD on a non-convex problem.

• Same results also apply to low-precision methods!
Result Outline

• Martingale proof uses a martingale $W$ such that

$$W_t(x_t, x_{t-1}, \ldots, x_0) \geq t \cdot 1 \text{ if algorithm hasn’t succeeded}$$

• Just need simple continuity bounds:
  • the martingale $W$ is Lipschitz continuous with parameter $H$
  • the gradients are Lipschitz continuous with parameter $R$
  • the expected magnitude of an update is bounded by $\xi$
  • the expected staleness due to asynchrony is bounded by $\tau$

$$\frac{P(\text{failure}_{\text{HOGWILD}})}{P(\text{failure}_{\text{sequential}})} \leq \frac{1}{1 - HR \xi \tau}$$
Intuition

• If the algorithm is continuous, noisy perturbations (due to race conditions, round-off error, etc.) just propagate and sum up to be roughly Gaussian noise at the output.

• Can use Lipschitz continuity to formalize this intuition.

• Practical upshot: **works for almost all types of extra noise!**
  • As long as the original proof uses a weak enough noise model.
  • Even for non-convex problems like matrix completion.
Follow-Up Work

• Systems analysis of low-precision asynchronous SGD
  • Understanding and Optimizing Asynchronous Low-Precision Stochastic Gradient Descent [De Sa et al, ISCA 2017]
  • How can we change the hardware to better support these algorithms?

• Accelerating the convergence of SGD for PCA/matrix completion
  • A general analysis of momentum + stochastic power iteration
  • Using a weak noise model
  • Great previous work here (especially for PCA application)
Conclusion

• Main question: **when can we show that nonconvex stochastic gradient descent converges?**

• Approach in this talk: study matrix completion/PCA
  • SGD converges globally
  • Using very weak noise models

• Theory makes analyzing techniques easy
  • Asynchronous execution
  • Low-precision computation

Thank you!

cdesa@stanford.edu
stanford.edu/~cdesa