Nonconvex Optimization by Complexity Progression

Hossein Mobahi
Google Research

EECS Seminar, UC Merced
November 4th 2016
Learning as Optimization

A Typical Learning Scenario

- Training data $\mathcal{X}$.
- Solution space $(\theta_1, \ldots, \theta_n)$.
- Cost criterion $f(\mathcal{X}, \theta_1, \ldots, \theta_n)$.

Learning seeks $(\theta_1, \ldots, \theta_n)$ which minimizes $f(\mathcal{X}, \theta_1, \ldots, \theta_n)$. When $f$ is convex, any local minimum is a global minimum (good for local search algorithms).
Learning Complex Tasks

Progressive Strategy

- Start from what we can solve.
- Gradually increase the complexity of the task.
- Move to the more complex task when the current one is learned.
Examples

- Graduated NonConvexity (GNC).
- Deterministic Annealing.
- Mean Field Approximation.
- Curriculum Learning
- Layerwise Pretraining
Optimization by Continuation

\[ f(x) \triangleq \ldots \]

\[ k_\sigma(x) \triangleq \frac{1}{(\sqrt{2\pi\sigma})^{\dim(x)}} e^{-\frac{||x||^2}{2\sigma^2}} \]

\[ g(x, t) \triangleq tf(x) + (1 - t)(x^2 + y^2) \]

\[ g(x, t) \triangleq [f \ast k^{\frac{1}{t} - 1}](x) \]
Evolution Equation

- A formalism to express changes over time.
- Initial condition $g(x, t = 0) = f(x)$.

Heat Equation
\[
\frac{\partial}{\partial t} g = \Delta g
\]

Schroedinger’s Equation
\[
\frac{\partial}{\partial t} g = i\Delta g
\]
Choice of Simplified Optimization

Convex Envelope

- Any global minimizer of original function is also a global minimizer of convex envelope.
- Computation of convex envelope is generally intractable.
Vese’s Equation

\[
\frac{\partial}{\partial t} h = \sqrt{1 + \|\nabla h\|^2} \min\{0, \lambda_{\min}(\nabla^2 h)\}, \quad \text{s.t. } h(.; 0) = f(.)\]
Heat Equation

Connection to Heat Equation [Mobahi14]

$$\frac{\partial}{\partial t} h = \sqrt{1 + \|\nabla h\|^2} \min\{0, \lambda_{\min}(\nabla^2 h)\} \quad \text{s.t.} \quad h(.); 0 = f(.)$$

$$\approx \frac{\partial}{\partial t} h = \frac{1}{n+1} [\Delta h](x) \quad \text{s.t.} \quad h(.); 0 = f(.)$$

<table>
<thead>
<tr>
<th>$t$</th>
<th>0</th>
<th>0.1</th>
<th>0.5</th>
<th>1</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td><img src="image1" alt="Original" /></td>
<td><img src="image2" alt="Original" /></td>
<td><img src="image3" alt="Original" /></td>
<td><img src="image4" alt="Original" /></td>
<td><img src="image5" alt="Original" /></td>
</tr>
<tr>
<td>Linearized</td>
<td><img src="image1" alt="Linearized" /></td>
<td><img src="image2" alt="Linearized" /></td>
<td><img src="image3" alt="Linearized" /></td>
<td><img src="image4" alt="Linearized" /></td>
<td><img src="image5" alt="Linearized" /></td>
</tr>
</tbody>
</table>
Gaussian Convolution

It has closed form for some interesting function [Mobahi16], e.g. polynomials.

<table>
<thead>
<tr>
<th>Original</th>
<th>Convolved</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$x$</td>
</tr>
<tr>
<td>$x^2$</td>
<td>$\sigma^2 + x^2$</td>
</tr>
<tr>
<td>$x^3$</td>
<td>$3\sigma^2 x + x^3$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$x^{10}$</td>
<td>$945\sigma^{10} + 4725\sigma^8 x^2 + 3150\sigma^6 x^4 + 630\sigma^4 x^6 + 45\sigma^2 x^8 + x^{10}$</td>
</tr>
</tbody>
</table>
Recurrent Neural Networks

\[
\hat{y}_t \triangleq h(n_t) + b \\
n_t \triangleq W h(m_t) \\
m_t \triangleq U x_t + V h(m_{t-1}) + a
\]
Learning

\[
\min_{a,b,U,V,W} \sum_{t=1}^{T} f(\hat{y}_t - y_t)
\]

for \( t = 1, \ldots, T \)

\[
\begin{align*}
\hat{y}_t & \triangleq h(n_t) \\
n_t & \triangleq Wh(m_t) + b \\
m_t & \triangleq Ux_t + Vh(m_{t-1}) + a
\end{align*}
\]
## Diffused Cost

### Has Closed Form Expression

<table>
<thead>
<tr>
<th>Name</th>
<th>Original</th>
<th>Diffused</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign</td>
<td>$\text{sign}(x)$</td>
<td>$\text{erf} \left( \frac{x}{\sqrt{2}\sigma} \right)$</td>
</tr>
<tr>
<td>Error</td>
<td>$\text{erf}(ax)$</td>
<td>$\text{erf} \left( \frac{ax}{\sqrt{1+2(a\sigma)^2}} \right)$</td>
</tr>
<tr>
<td>Tanh</td>
<td>$\tanh(x)$</td>
<td>$\tanh \left( \frac{x}{\sqrt{1+\frac{\pi^2}{2}\sigma^2}} \right)$</td>
</tr>
<tr>
<td>ReLU</td>
<td>$\max(0, x)$</td>
<td>$\frac{\sigma}{\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} + \frac{1}{2} x \left(1 + \text{erf} \left( \frac{x}{\sqrt{2\sigma}} \right)\right)$</td>
</tr>
</tbody>
</table>

![Sign](image1)

![Tanh](image2)

![ReLU](image3)
Popular Training Gadgets

Emerge Spontaneously from Theory

- Annealing Learning Rate
- Careful Initialization
- Noise Injection (SGD, Dropout)
- Saturation Control (Batch Normalization)
- Preventing Exploding/Vanishing Gradient (LSTM)
- Layerwise Pretraining
Popular Training Gadgets

Emerge Spontaneously from Theory

- Annealing Learning Rate
- Careful Initialization
- Noise Injection (SGD, Dropout)
- Saturation Control (Batch Normalization)
- Preventing Exploding/Vanishing Gradient (LSTM)
- Layerwise Pretraining

Taylor’s Remainder Theorem

\[ \hat{f}(w) \triangleq f(w_0) + (w - w_0)^T \nabla f(w_0) \]

\[ |\hat{f}(w) - f(w)| \leq \frac{1}{2} \|w - w_0\|^2 \Lambda \]
Popular Training Gadgets

Emerge Spontaneously from Theory

- Annealing Learning Rate
- Careful Initialization
- Noise Injection (SGD, Dropout)
- Saturation Control (Batch Normalization)
- Preventing Exploding/Vanishing Gradient (LSTM)
- Layerwise Pretraining
Popular Training Gadgets

Emerge Spontaneously from Theory
- Annealing Learning Rate
- Careful Initialization
- Noise Injection (SGD, Dropout)
- Saturation Control (Batch Normalization)
- Preventing Exploding/Vanishing Gradient (LSTM)
- Layerwise Pretraining

MCMC Integration

\[(f \ast k_\sigma)[w] = \int_{\mathbb{R}^n} f(w-t)k_\sigma(t) \approx \frac{1}{N} \sum_{i=1}^{N} f(w - t_i) \text{ s.t. } t_i \sim \mathcal{N}(0, \sigma^2 I)\]
Popular Training Gadgets

Emerge Spontaneously from Theory

- Annealing Learning Rate
- Careful Initialization
- Noise Injection (SGD, Dropout)
- Saturation Control (Batch Normalization)
- Preventing Exploding/Vanishing Gradient (LSTM)
- Layerwise Pretraining

Input Normalization

Example: squashing function \( f(y) \triangleq \text{sign}(y) \):

\[
[\text{sign}(\Box^T \mathbf{x}) \ast k_\sigma(\Box)](\mathbf{w}) = \text{erf}\left(\frac{\mathbf{w}^T \mathbf{x}}{\sqrt{2} \sigma \| \mathbf{x} \|}\right)
\]
Popular Training Gadgets

Emerge Spontaneously from Theory
- Annealing Learning Rate
- Careful Initialization
- Noise Injection (SGD, Dropout)
- Saturation Control (Batch Normalization)
- Preventing Exploding/Vanishing Gradient (LSTM)
- Layerwise Pretraining

High Orders Disappear
Recursive updates... expressions like:

\[ I + V \text{ diag}(\tilde{h}'_\sigma(m_t)) + V^2 \text{ diag}(\tilde{h}'_\sigma(m_t) \cdot \tilde{h}'_\sigma(m_{t-1})) + \ldots \]

When \( \sigma \to \infty \)

\[ I + V \epsilon + V^2 \epsilon^2 + \ldots \]
Popular Training Gadgets

Emerge Spontaneously from Theory

- Annealing Learning Rate
- Careful Initialization
- Noise Injection (SGD, Dropout)
- Saturation Control (Batch Normalization)
- Preventing Exploding/Vanishing Gradient (LSTM)
- Layerwise Pretraining

High Orders Disappear

Recursive updates... expressions like:

\[ I + V \text{ diag } (\tilde{h}'_{\sigma}(m_t)) + V^2 \text{ diag } (\tilde{h}'_{\sigma}(m_t) \cdot \tilde{h}'_{\sigma}(m_{t-1})) + \ldots \]

When \( \sigma \to \infty \)

\[ I + V \epsilon + V^2 \epsilon^2 + \ldots \]

Ignores time dependencies at first, and then gradually reconsiders them.
Example

Learning to Add

<table>
<thead>
<tr>
<th>0.8</th>
<th>0.9</th>
<th>0.3</th>
<th>0.1</th>
<th>0.2</th>
<th>0.2</th>
<th>0.5</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Result

SGD Hyperparameters (Annealed Learning Rate, Mini Batch Size, Momentum Coefficient) carefully searched.
Alignment as Energy Minimization

\[ \theta^* = \arg \min_{\theta} \int_{\mathcal{X}} \left( f_1(\tau(x; \theta)) - f_2(x) \right)^2 dx \]

This is non-convex in variable \( \theta \).
Smoothing the Objective for Alignment

1-d Scale Alignment

$$\min_a \int_{\mathbb{R}} (f_1(ax) - f_2(x))^2 dx$$

Signals

Signal Smoothing

Objective Smoothing
Fovea Effect

Equivalent Blur Operator

\[
[f(Ax + b) \ast k_\sigma](\theta) = \int_{\mathcal{X}} f(y) k(Ax + b - y; \sigma^2(1 + \|x\|^2)) \, dy
\]
Alignment Results [Mobahi12]

- Alignment results with proposed blur, Gaussian blur, and no blur.

Graph showing the progression of alignment accuracy over iterations for different blur techniques.
3D Point Cloud Alignment

Formulation

\[
\min_{A,b,c} \sum_{i,j} c_{i,j} \| A p_i + b - q_j \|^2 \\
\text{s.t.} \sum_i c_{i,j} = 1, \quad c_{i,j} \in \{0, 1\}
\]
3D Point Cloud Alignment

Formulation

\[
\min_{A, b, c} \sum_{i, j} c_{i,j} \| A p_i + b - q_j \|^2 \\
\text{s.t.} \sum_i c_{i,j} = 1, c_{i,j}(1 - c_{i,j}) = 0
\]
3D Point Cloud Alignment

Optimization Landscape of Smoothed Objective

\[ \sigma = \frac{0}{88}, \quad \sigma = \frac{3}{88}, \quad \sigma = \frac{6}{88}, \quad \sigma = \frac{9}{88}, \quad \sigma = \frac{12}{88}, \quad \sigma = \frac{15}{88}, \quad \sigma = \frac{27}{88}, \quad \sigma = \frac{30}{88}, \quad \sigma = \frac{33}{88}, \quad \sigma = \frac{18}{88}, \quad \sigma = \frac{21}{88}, \quad \sigma = \frac{24}{88} \]
Transforming $P$ to $Q$

$P$  

$Q$  

ICP  

Ours
New theory for optimization by continuation presented.

Efficient algorithms based on closed form expressions were discussed.

Future Directions:

- Investigating other applications (NLP, Robotics, Etc.).
- Improving approximations used in deep learning application.
- Improving the approximation quality of Vese’s equation (E.g. by nonlinear PDEs with closed form, combination of analytical and numerical methods).
Acknowledgment

- John W. Fisher III (MIT)
- William T. Freeman (MIT)
- Yi Ma (UIUC/ShanghaiTech)
- Stefano Soatto (UCLA)

- Anima Anandkumar (UCI)
- Alan L. Yuille (UCLA)
- Rong Ge (Duke)
- John Wright (Columbia)