Optimization Technique in Training Deep Models

Chapter 8
Outline

• Problem definition

• First order and second order methods

• SGD and momentum techniques in deep models

• Function properties in optimization

• Discussions
Background

• Machine learning problems

\[ J(\theta) = \mathbb{E}_{(x, y) \sim \hat{p}_{\text{data}}} L(f(x; \theta), y), \]

• Empirical risk minimization with independence

\[ \min \mathbb{E}_{x, y \sim \hat{p}_{\text{data}}}(x, y) [L(f(x; \theta), y)] = \frac{1}{m} \sum_{i=1}^{m} L(f(x^{(i)}; \theta), y^{(i)}), \]

What if the data are not independent?
Risk Minimization

- True risk in machine learning

\[
\min \mathbb{E}_{x,y \sim \hat{p}_{\text{data}}(x,y)}[L(f(x; \theta), y)]
\]  

(1)

- Empirical risk minimization

\[
\min \frac{1}{m} \sum_{i=1}^{m} L(f(x^{(i)}; \theta), y^{(i)})
\]  

(2)

- Generalization error

\[
P[(1) - (2) \geq t] \leq \exp(-t), \forall t \geq 0
\]

Risk Minimization

- True risk in machine learning

\[
\min \mathbb{E}_{x,y \sim \hat{p}_{\text{data}}(x,y)}[L(f(x; \theta), y)]
\]  
(1)

- Empirical risk minimization

\[
\min \frac{1}{m} \sum_{i=1}^{m} L(f(x^{(i)}; \theta), y^{(i)})
\]  
(2)

- Generalization error

\[
P[(1) - (2) \geq t] \leq \exp(-t), \forall t \geq 0
\]

What is the effect of \( m \)?

Problem Definition

• Empirical risk minimization (ERM)

\[
\min \frac{1}{m} \sum_{i=1}^{m} L(f(x^{(i)}; \theta), y^{(i)})
\]

• Solution

\[
\theta_{ML} = \arg \max_{\theta} \sum_{i=1}^{m} \log p_{\text{model}}(x^{(i)}, y^{(i)}; \theta)
\]

• Square loss
  – Gaussian distribution in model errors

• Tools
  – SVM
  – Neural networks
From ERM to Deep Learning

- Model of feedforward neural network

\[
Y(\theta, X) = \theta_h \times \theta_{h-1} \times \theta_1 \times X
\]

- Batch learning

\[
\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} L(f(x^{(i)}; \theta), y^{(i)})
\]

- First order and second order in optimization
First Order Method

• Gradient (a vector)

\[ g = \nabla_{\theta} J^*(\theta) = \sum_{x} \sum_{y} P_{\text{data}}(x, y) \nabla_{\theta} L(f(x; \theta), y). \]

• Cons
  – Time consuming for each iteration
  – Not linear convergence rate
    • Convex case: \( O \left( \frac{1}{T} \right) \) with \( T \) being the total iteration
    • Acceleration case: \( O \left( \frac{1}{T^2} \right) \)
  – How to set the learning rate

• Pros
  – Exact gradient information

Second Order Method

• Hessian matrix (a square matrix)

\[ H_{i,j} = \frac{\partial g_i}{\partial \theta_j} \]

• Cons
  – Ill-conditioning of matrix (zero eigenvalue)
  – Time consuming in each iteration, or even failure, in calculating the inverse of Hessian matrix

• Pros
  – linear convergence rate
    • Strongly convex case: \( O(\rho^T) \) with \( 0 < \rho < 1 \)
      \[ T = O(\ln(\frac{1}{\epsilon})) \] with \( \epsilon \) being the accuracy
Update Rules

• First order method

\[ \theta_{t+1} = \theta_t - \eta g_t \]

• Second order method

\[ \theta_{t+1} = \theta_t - H_t^{-1} g_t \]

1. Optimal learning rate
2. The inverse is not easily to solve
3. Estimation error of Hessian leads to large deviation in training of deep models
Taylor Series Approximation

• Function approximation

\[ f(x) = f(x_0) + (x-x_0)f'(x) + \frac{1}{2}(x-x_0)^2f''(x) + \ldots \]

• ERM problem

\[ J(\theta) = J(\theta_0) + (\theta-\theta_0)^\top g + \frac{1}{2}(\theta-\theta_0)^\top H(\theta-\theta_0) + \ldots \]
SGD in Deep Learning

• Stochastic gradient descent (SGD)

\[ \theta_{t+1} = \theta_t - \eta \hat{g}_t \]

\( \hat{g}_t \) is calculated based on: 1) one sample (online learning); 2) a small subset of samples (mini batch)

• Pros
  – Improve the efficiency in each iteration

• Cons
  – Noise in the estimation of gradient

How to deal with this issue?
An Ideal Assumption

- Gradient in (conditional) expectation

\[ g_t = \mathbb{E}[\hat{g}_t] \quad \Rightarrow \quad g_t = \mathbb{E}[\hat{g}_t | F_{t-1}] \]

Why this assumption works?
An Ideal Assumption

- Gradient in (conditional) expectation

\[ g_t = \mathbb{E}[\hat{g}_t] \quad \Rightarrow \quad g_t = \mathbb{E}[\hat{g}_t | F_{t-1}] \]

- Stochastic (convex) optimization

\[
\min F(\theta) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{E} [f_t(\theta)]
\]

  - Convex convergence rate: \( O\left(\frac{1}{\sqrt{T}}\right) \)

  - Acceleration rate: \( O\left(\frac{1}{T}\right) \) for strongly convex and smooth function


A Realistic Assumption

- Gradient in (conditional) expectation
  \[ g_t = \mathbb{E}[\hat{g}_t] \rightarrow g_t = \mathbb{E}[\hat{g}_t|F_{t-1}] \]

- Stochastic (convex) optimization
  \[
  \min F(\theta) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{E} [f_t(\theta)]
  \]
  - Convex convergence rate: \( O\left(\frac{1}{\sqrt{T}}\right) \)
  - Acceleration rate: \( O\left(\frac{1}{T}\right) \) for strongly convex and smooth function


An Illustration

• A Comparison of GD and SGD

https://am207.github.io/2017/wiki/gradientdescent.html
An Illustration

- A Comparison of GD and SGD

https://am207.github.io/2017/wiki/gradientdescent.html
Ill-Conditioning of Objectives

• Ill-conditioning is general in deep models
  – Metric: condition number
    \[ r = \frac{\lambda_{\text{max}}(H)}{\lambda_{\text{min}}(H)} \]
  – Large \( r \) means ill-condition

https://distill.pub/2017/momentum/
Momentum

- Classical method

\[ v_{t+1} = \mu v_t - \eta g_t(\theta_t), \mu \in [0,1] \]
\[ \theta_{t+1} = \theta_t + v_{t+1} \]

https://distill.pub/2017/momentum/

Momentum

• Classical method

\[ v_{t+1} = \mu v_t - \eta g_t(\theta_t), \mu \in [0,1] \]
\[ \theta_{t+1} = \theta_t + v_{t+1} \]

https://distill.pub/2017/momentum/

Momentum Intuition

- Classical method

\[\nu_{t+1} = \mu \nu_t - \eta g_t(\theta_t), \mu \in [0,1]\]
\[\theta_{t+1} = \theta_t + \nu_{t+1}\]

Momentum Intuition

• Classical method

\[ v_{t+1} = \mu v_t - \eta g_t(\theta_t), \mu \in [0,1] \]

\[ \theta_{t+1} = \theta_t + v_{t+1} \]

• Pros
  – Partially solve ill-conditioning
  – Help to adjust the learning rate
  – Faster convergence, and less oscillation
  – Set \( \mu = 0 \), we have GD/SGD

• Cons
  – A new parameter
Nesterov Accelerated Gradient

• Update rules

\[ \nu_{t+1} = \mu \nu_t - \eta g_t(\theta_t + \mu \nu_t), \mu \in [0,1] \]

\[ \theta_{t+1} = \theta_t + \nu_{t+1} \]

• Convergence rate in convex case

\[ -O\left(1/T^2\right) \]

Momentum in Deep Learning

• Some deep models
  – SGD cannot obtain good performance
  – Try momentum technique

• Random initialization
  – Good performance in FNN and RNN
  – Constant initialization leads to failure of training

Function Properties in Optimization

• Revisit SGD

\[ \theta_{t+1} = \theta_t - \eta \hat{g}_t \]
Function Properties in Optimization

- Revisit SGD
  \[ \theta_{t+1} = \theta_t - \eta \hat{g}_t \]

- Bernstein condition
  \[ \mathbb{E} \left[ L(x^i, y^i, \theta_t)^2 \right] \leq B \left( \mathbb{E} L(x^i, y^i, \theta_t) - L(x^i, y^i, \theta^*) \right)^\gamma \]

- Convergence rate
  \[ - O \left( T^{2-\gamma} \right) \text{ with } \gamma \in [0,1] \]

Holderian Error Bound

- Local Holderian error bound

\[ \| \theta_t - \theta^* \| \leq C (\mathbb{E}[L(x^i, y^i, \theta_t)] - \mathbb{E}[L(x^i, y^i, \theta^*)])^{\gamma} \]

- Convergence rate

\[ - O(T^{\frac{1-\gamma}{2-\gamma}}) \text{ with } \gamma \in [0,1] \]

Can one design deep models to have this property?

Discussions

• ERM problem in deep models
• Optimization to solve ERM problem
• SGD and momentum
• Function properties help to solve optimization